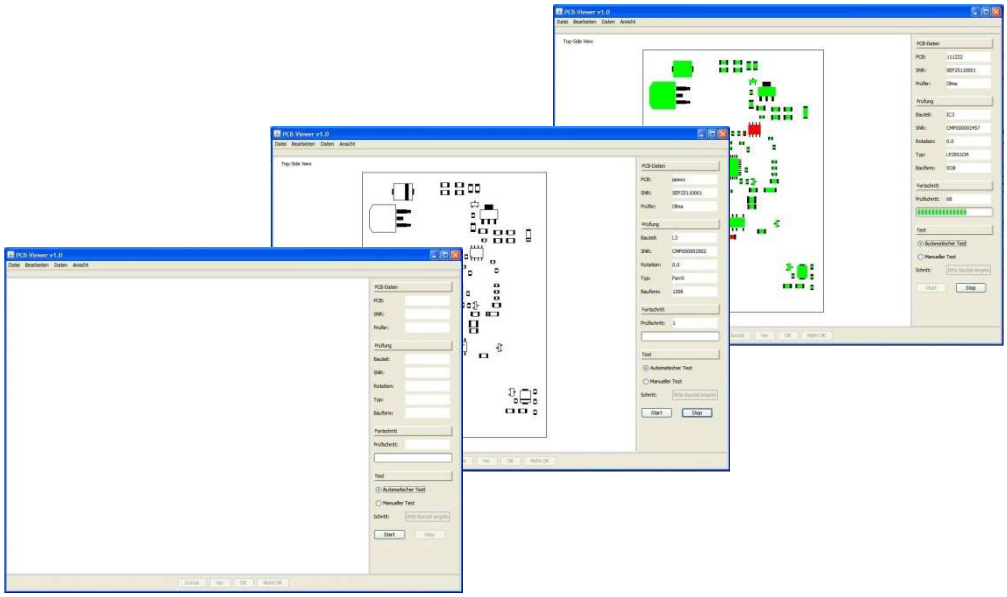
Projektdokumentation

PCB-Viewer

Software zur automatisierten Prüfung elektronischer Leiterplatten



Team: Patrick Olma

Betreuer: Prof. M. Goik

Inhaltsverzeichnis

1. Vorwort

Die Idee des Projektes PCB-Viewer entstand in Zusammenarbeit mit der Firma Seprotronic GmbH in Stuttgart.
Die Seprotronic GmbH bietet Dienstleistungen im Bereich der Entwicklung und Produktion von elektronischen Produkten und Systemen an.

Auf Grund der Dienstleistungen und des Einsatzszenarios der Software entstand letztendlich auch der Name des Projektes: **PCB-Viewer**
Der Name PCB steht für das Printet Curcuit Board und beschreibt die el. Leiterplatte. Viewer steht in diesem Kontext für das „anschauen“ einer solchen Leiterplatte.

Speziell im Bereich der Produktion steigt der Anspruch der Kunden bezüglich Qualität stetig an. Die Prüfung ist daher eines der Hauptbestandteile einer wirksamen Qualitätssicherung. Sie stellt nachhaltig sicher, dass das hergestellte Produkt vollständig den Kundenanforderungen entspricht.

Basierend auf den Prozessen, welche innerhalb der Produktion der Firma momentan abgearbeitet werden, wurde das Projekt PCB-Viewer in enger Zusammenarbeit mit Mitarbeitern der Firma erstellt, mit dem Ziel die Prozesse der Firma zu unterstützen und nachhaltig zu verbessern.

Der Inhalt dieser Projektdokumentation gliedert sich wie folgt:
Anfangs beschreibt diese Dokumentation die einzelnen Phasen der Ist- bzw. der Soll-Analyse, d.h. vor der Umsetzung der Software wurde im 1. Schritt festgestellt, wie der momentane Prozess während der Produktion abläuft und welche Verbesserungen sich auf Grund dieser Analyse mit Hilfe dieser Software realisieren lassen können.
Im 2. Schritt befasst sich die Dokumentation mit einer detaillierten Beschreibung der einzelnen Module, welche innerhalb der Software realisiert wurden und welche generellen Aufgaben diese Module beinhalten.
Im letzen Teil der Dokumentation wird letztendlich auf die Probleme eingegangen, welche innerhalb der jeweiligen Projektphasen auftraten und was die zukünftigen Aussichten des Projektes sind.

2. Die Ausgangslage (Ist-Analyse)

Stand heute erfolgt nach der Produktion eine manuelle, visuelle Sichtprüfung der elektronischen Baugruppen. Basierend auf einem Referenzdokument, auf dem die Anordnung aller enthaltenen elektronischen Komponenten eindeutig dargestellt ist, wird mit Hilfe eines Mikroskops die Übereinstimmung der fertig bestückten Leiterplatte mit dem Referenzdokument überprüft.

Dabei steht der Prüfer vor folgenden Problemen:

- **Manuelle Suche der zu prüfenden Bauteile auf dem Dokument bzw. auf der Baugruppe:**
  Wie oben bereits beschrieben steht der Prüfer vor der Aufgabe, das zu prüfende Bauteil im 1. Schritt auf dem Referenzdokument zu finden und danach das entsprechende Bauteil wiederum auf der Leiterplatte zu identifizieren und anschließend zu prüfen. Bei einer Baugruppe mit mehreren hundert verschiedenen Bauteilen ist dieser Schritt enorm langwierig und reduziert damit die Anzahl der fertig geprüften Leiterplatten innerhalb eines bestimmten Zeitintervalls.

- **Sehr aufwendiger Prüfprozess:**
  Durch die eben beschriebene Art und Weise der Prüfung wird dieser Prozess sehr aufwendig. Effekt dieses aufwendigen Prüfprozesses ist eine Reduzierung der Ergebnisse, sowohl im qualitativen, als auch im quantitativen Bereich.

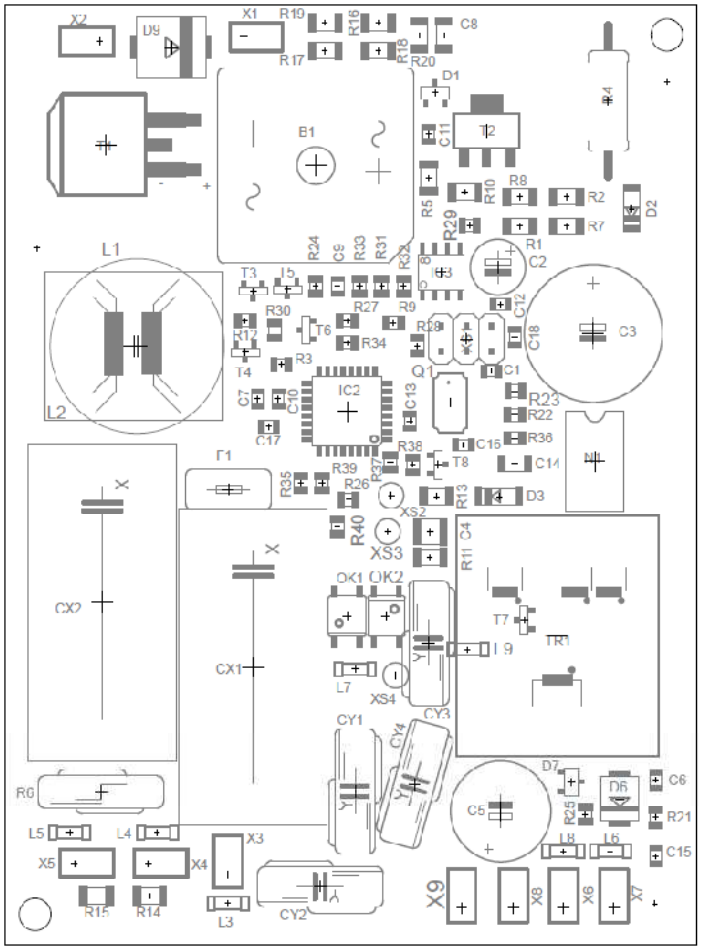
- **Lang andauernde Konzentrationsphase:**
  Je länger dieser Prüfprozess andauert, desto länger muss der Prüfer seine Konzentration aufrecht erhalten. Analysen ergaben, dass bereits nach der 1. Pause Schwierigkeiten auftraten. Während der Nachtschicht trat dieser Effekt nochmals erhöht auf.

- **Hohes Risiko, dass vorhandene Fehler nicht entdeckt werden:**
  Auf Grund der eben beschriebenen Konzentrationsphase bzw. des Problems, diese auf Dauer aufrecht zu erhalten, erhöht sich im Laufe der Zeit das Risiko, dass eventuell auftretende Fehler auf der Leiterplatte durch den Prüfer nicht mehr richtig erkannt werden können. Eine wiederholte Prüfung einer Leiterplatte bzw. ein Versandt von defekten Leiterplatten muss unter allen Umständen vermieden werden.

- **Manuelle schriftliche Festhaltung aller Prüfschritte und deren Ergebnisse:**
  Alle Ergebnisse, welche während dieser Prüfung auftreten, müssen vom Prüfer in eigens dafür vorgefertigten Dokumenten festgehalten werden. Pro Dokument können momentan 20 verschiedene Prüfschritte festgehalten werden. Schon bei einer Anzahl von 100 Bauteilen, benötigen wir also 5 Blätter. Alle diese Blätter müssen nach Vollendung der Prüfung archiviert werden, was zu hohen Kosten und zu viel Lagerplatz führt.

- **Manuelle Implementierung der Prüfergebnisse in das vorhandene QMS:**
  Die eben bereits beschriebenen Dokumente müssen nun von einem weiteren Mitarbeiter in ein QMS (Qualitätsmanagementsystem) implementiert werden, um auch zu späteren Zeitpunkten eindeutig nachvollziehen zu können, welche Ergebnisse bei der Prüfung entsprechender Leiterplatten aufgetreten sind. Eine Analyse ergab, dass im Schnitt ein Mitarbeiter zu 100% damit ausgelastet ist, diesen Vorgang durchzuführen. D.h. man verliert mit diesem Prozess eine Ressource, welche lediglich dafür zuständig ist, bereits dokumentierte Ergebnisse ein weiteres Mal zu dokumentieren.

Da in der heutigen Zeit eine nahezu vollständig manuell fortlaufende Prüfung weder effizient noch effektiv ist, wird eine automatisierte Lösung angestrebt.



Beispiel eines Referenzdokumentes

3. Die Anforderungen (Soll-Analyse)

Mit Hilfe eines Software-Tools soll diese Prüfung nun nahezu automatisiert werden.
Ziel ist es also, eine Zeitersparnis zu erlangen, den Prüfenden von der andauernd hohen Konzentrationsphase bei der visuellen Prüfung zu entlasten und die Schritte, welche während der Prüfung auftreten, automatisch in ein vorhandenes QMS zu implementieren, um eine weitere Ressource einzusparen.
Dadurch kann eine wesentlich höhere Effizienz und Wirksamkeit bei dieser Art von Prüfung erreicht werden.

Dabei werden folgende Anforderungen an die Applikation gestellt:

- **Digitale Visualisierung des Layouts:**
  Mit Hilfe eines Bestückprogramms in Form einer Text-Datei, welche Informationen wie z.B. Bauteil-Sachnummer, Bauteil-Bezeichnung, Position auf der Baugruppe, Rotation etc. beinhaltet, werden Informationen über die Baugruppe von der Entwicklung an die Produktion geleitet. Mit Hilfe der Software soll diese Baugruppe nun anhand der vorhandenen Daten visualisiert werden und auf einer GUI angezeigt werden.

- **Prüfung der Baugruppe mit Hilfe der Applikation:**
  Im nächsten Schritt soll die Software nun Schritt für Schritt die zu prüfenden Bauteile farbig hervorheben. So spart sich der Prüfer die langwierige Suche des Bauteils auf dem Referenzdokument, denn das Bauteil kann nun sofort erkannt und dessen Position festgestellt werden. Im Falle einer korrekten Prüfung sollten diese Bauteile grün markiert werden, im anderen Falle rot. Somit ist schon auf den ersten Blick ersichtlich, welche Bauteile fehlerhaft sind. Dies ist z.B. dann wichtig, wenn die Leiterplatte in den Reparaturprozess geht. Jeder Stand der Prüfung sollte im Falle einer Unterbrechung der Prüfung wiederherstellbar sein, d.h. die Software muss die Funktionen besitzen, Prüfungen zu speichern und diese zu einem späteren Zeitpunkt wieder zu öffnen.

- **Definition der einzelnen Bauteile:**
  Einzelne Bauteile sollten innerhalb des Programmes bzgl. Geometrie, Abmessungen etc. definiert werden können. Dies ist wichtig, da bsp. verschiedene Bauformen existieren bzw. bei vielen Bauteilen auf besondere Polung geachtet werden sollte etc.
  Diese definierten Bauteile sollten automatisch mit einer vorhandenen SQL-Datenbank verglichen werden, da viele Bauteile schon einmal verwendet wurden. Dadurch erspart man sich die ständige Neudefinition eines schon einmal verwendeten Bauteils.

- **Festhalten und Implementierung der entstandenen Daten in vorhandene Datenbanken:**
  Um eine sichere und gute Qualität zu gewährleisten, ist es sehr wichtig, dass zu jedem Zeitpunkt eindeutig nachvollziehbar ist, wer was und wann geprüft hat.
  Daher müssen alle Schritte, welche innerhalb der Prüfung aufgetreten sind, in eine vorhandene SQL-Datenbank des QMS eingepflegt werden.

In diese Datenbank werden also folgende Daten automatisch nach jedem Prüfschritt integriert:

- Name des Prüfers
- Zeitpunkt der Prüfung
- Seriennummer der Leiterplatte
- Sachnummer der Baugruppe
- Sachnummer/Chargennummer des geprüften Bauteils
- Ergebnis der Prüfung
- Falls vorhanden, Art des Fehlers

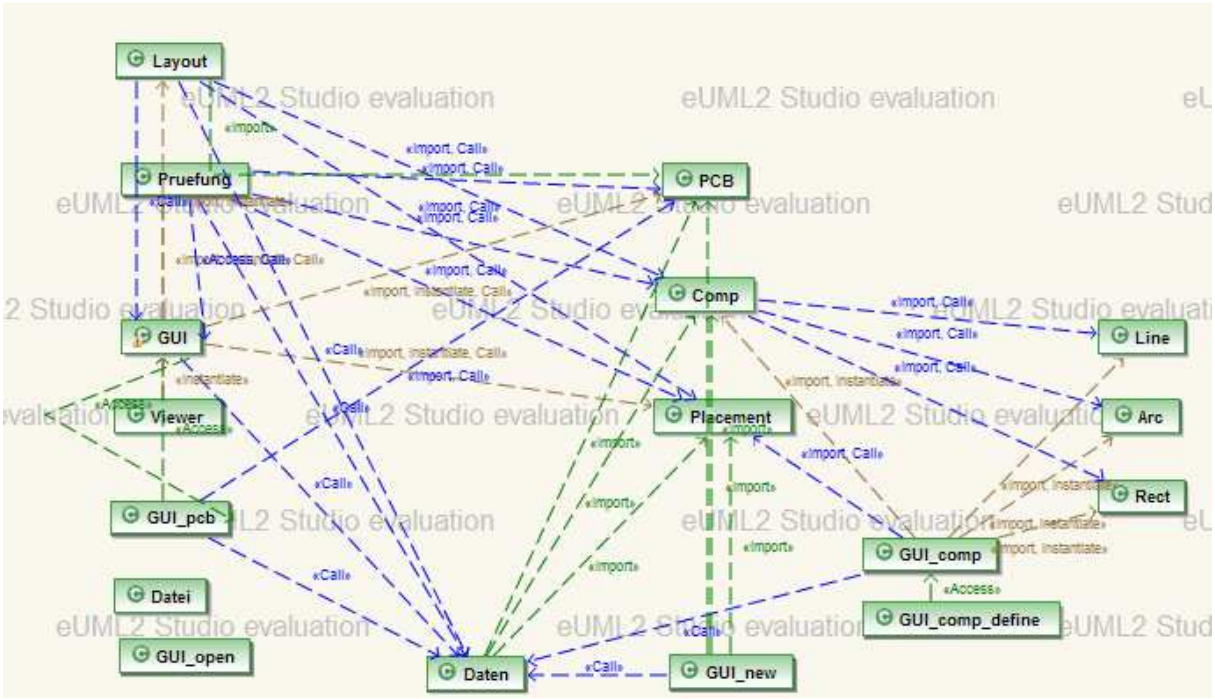
Zusätzlich sollten die definierten Daten der Bauteile, wie schon beschrieben, in eine separate SQL-Datenbank eingepflegt werden.
Dadurch erspart man sich zum einen ein erneutes Definieren eines Bauteils, welches bereits verwendet wurde, zum anderen können diese Daten zu einem späteren Zeitpunkt von Maschinen innerhalb der Produktion wieder verwendet werden. Diese Daten beinhalten folgendes:

- Bezeichnung des Bauteils
- Sachnummer des Bauteils
- Bauform
- Abmessungen
- Rotation
- Typ
- Geometrie-Library (Textdatei, welche die Geometrie-Daten beinhaltet)

Ein weiteres Ziel der Applikation ist also, die teilweise großen Datenbestände sinnvoll in bestehende Datenbanken zu integrieren, um diese Daten zu einem späteren Zeitpunkt sinnvoll auswerten zu können.

4. Die Software

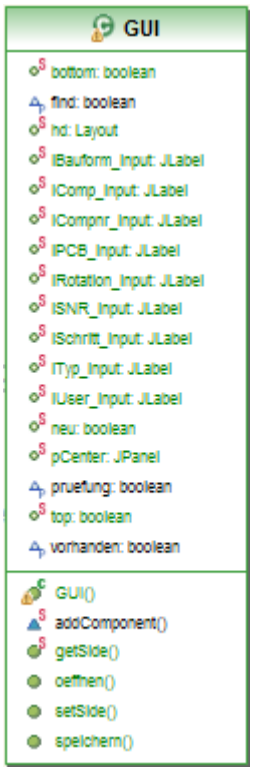
Dieser Teil der Dokumentation befasst sich nun mit der Beschreibung der einzelnen Module der Software und den generellen Aufgaben, welche diese Module realisieren. Im folgenden Schaubild erhält man einen groben Überblick über die verschiedenen Module und deren Beziehungen zueinander.



Wie leicht erkennbar ist, wirkt dieses Schaubild zu Beginn etwas verwirrend. Es ist jedoch gut zu sehen, dass die Software über verschiedene Module verfügt, wobei jedes Modul eine eigene Aufgabe besitzt. Alle Module sind letztendlich in Beziehung zueinander und bilden somit die komplette Software.

Um dieses noch etwas verwirrende Schaubild nun zu verdeutlichen, werden nachfolgend alle Module genauer beschrieben.

4.1 GUI.java

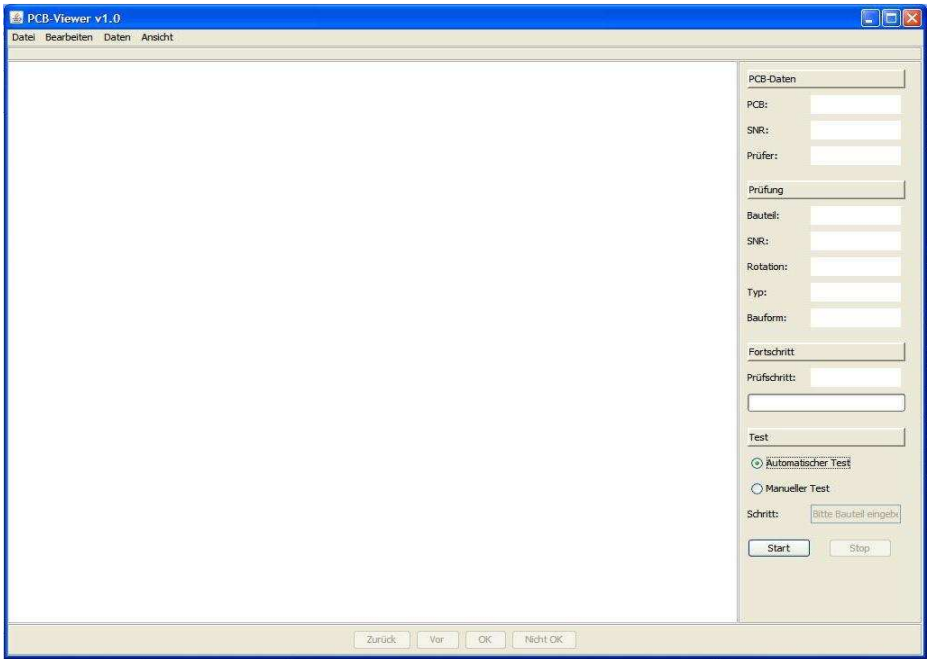


Diese Klasse bildet sozusagen das Fundament der ganzen Software. Aufgabe dieses Moduls ist zum einen die Erstellung des Hauptfensters, also jener GUI, in welchem die ganze Software abläuft, zum anderen hat diese Klasse die Aufgabe, alle anfallenden Eingaben zu steuern, d.h. je nach Eingabe des Anwenders Funktionen bereitzustellen, um bsp. Prüfungen zu starten, Stände zu speichern, Ansichten zu wechseln, Daten anzulegen und diese Eingaben an die richtigen Module weiterzuleiten, in welchen die Funktionen dann realisiert werden können.

Wie oben bereits beschrieben, ist eine der Hauptaufgaben dieser Klasse, die benötigte GUI zu realisieren.

4.1.1 Die GUI

Um die Komplexität der Software für den Prüfer so gering wie möglich zu halten, musste eine GUI realisiert werden, welche zum einen sehr bedienerfreundlich ist, zum anderen jedoch genügend Informationen zur laufenden Prüfung bereitstellt und den aktuellen Stand der Prüfung deutlich und gut sichtbar wiedergibt.
Auf Grund dieser Anforderungen wurde nun folgende GUI realisiert, deren Teilbereiche im Folgenden näher beschrieben werden. Grundlage der GUI ist ein GridBagLayout, welches in 4 Teilbereiche aufgeteilt wurde.



- **Navigationsbereich**
  Im oberen Teil der GUI ist ein traditioneller Navigationsbereich zu finden, welcher in 4 Teilbereiche gegliedert wurde.
  - **Datei**
    Dieser Teilbereich ist für generelle Funktionen rund um die Dateiverwaltung zuständig. Folgende Funktionen werden in diesem Navigationsteil realisiert.
    - Neu:
      Dieser Menüpunkt realisiert die Funktionen, eine neue Leiterplatte anhand dessen Bestückfile anzulegen. Dabei wird ein Bestückfile zeilenweise eingelesen und die daraus entstehenden Daten auf der GUI dargestellt.

    - Öffnen:
      Diese Funktion realisiert das Öffnen einer bereits gespeicherten Leiterplatte inkl. deren Prüfdaten. Dabei werden die gespeicherten Daten aus einer Datenbank ausgelesen.

    - Schließen:
      Dieser Menüpunkt ist wie der Name schon sagt für das Schließen der aktuell geöffneten Leiterplatte zuständig. Dabei werden alle Daten zurückgesetzt und das Layout von der GUI entfernt.

    - Speichern:
      Als Gegensatz zum Öffnen können mit diesem Menüpunkt die aktuellen Daten der Leiterplatte inkl. deren Prüfdaten gespeichert werden. Dabei werden alle anfallenden Daten in vorhandene Datenbanken geschrieben.

    - Beenden:
      Dieser Menüpunkt beendet die Software.

  - **Bearbeiten**
    Der aktuelle Teilbereich der Navigation bietet generelle Funktionen rund um die Leiterplatte an sich an.
    - Suche:
      Mit Hilfe der Suche kann der Anwender einzelne Bauteile anhand dessen Namen direkt auf der Leiterplatte suchen. Dabei werden alle Bauteilnamen mit dem zu suchenden Namen verglichen. Sollte eine Übereinstimmung gefunden werden, so wird jenes Bauteil gelb eingefärbt.

    - Zoom:
      Mit Hilfe dieses Menüpunktes kann man das Layout der Leiterplatte vergrößern bzw. verkleinern. Dabei wird der entsprechenden Funktion,

welche das Layout zeichnet, ein sog. Faktor übergeben, um welchen die Darstellung vergrößert bzw. verkleinert wird.

- Drehung:
  Diese Funktion realisiert die Drehung des Layouts. Dabei wird je nach gewünschter Drehung ein Rotations-Wert an die entsprechende Funktion, welche das Layout zeichnet, übergeben und diese dreht das Layout über die rotate()-Funktion.

- **Daten**
  Dieser Teilbereich der Navigation behandelt die Daten-Definition der Leiterplatte und der jeweiligen Bauteile.
  - PCB:
    Mit Hilfe dieses Menüpunktes kann man die generellen Daten der Leiterplatte an sich definieren. Daten die hierbei anfallen sind die Abmessungen der Leiterplatte (Breite, Höhe, Dicke), die Sachnummer der Leiterplatte und deren Bezeichnung.

  - Bauteil:
    Diese Funktion realisiert die Definition der einzelnen Bauteildaten. Daten, welche hierbei anfallen sind die generellen Abmessungen des Bauteils (Breite, Länge, Höhe), die Bauform, der Typ und der Pfad zur Library, welche die Geometrie-Daten der Bauteile beinhaltet.

- **Ansicht**
  Der letzte Bereich der Navigation befasst sich mit Aufgaben, welche generelle Funktionen zur Ansichtsdarstellung bereitstellen, denn jede Leiterplatte besteht aus 2 Seiten, einer Top- und einer Bottom-Seite.
  - Top:
    Mit dieser Funktion wird die Top-Seite der Leiterplatte dargestellt. Dabei werden nur entsprechende Objekte der Top-Seite, welche in einer jeweiligen ArrayList gespeichert sind gezeichnet.

  - Bottom:
    Im Gegensatz zur Top-Seite, werden nur jene Objekte der Bottom-Seite gezeichnet.

- **Informationsbereich**
  Im rechten Teil der GUI ist ein Informationsbereich realisiert worden. Sinn und Zweck dieses Bereiches ist zum einen die Darstellung von Informationen zur aktuellen Leiterplatte. Zum anderen werden in diesem Bereich Informationen zur Prüfung des aktuellen Bauteils bzw. zum Fortschritt der Prüfung angezeigt. Abschließend hat man in diesem Bereich letztlich die Möglichkeit über jeweilige Buttons die Prüfung zu starten bzw. zu stoppen.

- **Hauptbereich**
  Im Zentrum der GUI befindet sich der Hauptbereich. Im größten Teil aller GUI-Bereiche wird das Layout gezeichnet, d.h. hier läuft die Software hauptsächlich ab. Nach jedem Prüfschritt wird das Layout neu gezeichnet und auftretende Prüfergebnisse farblich und gut ersichtlich dargestellt.

- **Steuerungsbereich**
  Über den Steuerungsbereich im unteren Teil der GUI besteht für den Prüfer die Möglichkeit, die komplette Prüfung zu steuern. Ihm stehen innerhalb dieses Bereiches folgende Funktionen zur Verfügung:
    - Vor:
      Mit diesem Button wird der Zähler während der Prüfung erhöht und die Software springt zum nächsten Bauteil. Dieses nächste Bauteil wird gelb hervorgehoben und kann nun geprüft werden.

    - Zurück:
      Ähnlich wie beim Button „Vor“ wird bei diesem Button der Zähler der Prüfung um 1 verringert und die Software springt zum vorherigen Bauteil. Dieses Bauteil wird wiederum gelb hervorgehoben und kann nun geprüft werden.

    - Ok:
      Mit diesem Button identifiziert der Prüfer das Bauteil als korrekt. In diesem Fall wird das Bauteil grün eingefärbt. Anschließend wird der Zähler der Prüfung um 1 erhöht und der Prüfer kann das nächste, nun gelb eingefärbte Bauteil, prüfen.

    - Nicht Ok:
      Drückt der Prüfer diesen Button, so teilt er der Software mit, dass jenes geprüfte Bauteil fehlerhaft war. Die Software färbt das Bauteil nun rot ein und der Prüfer muss über ein Dialogfeld eine Fehlerbeschreibung eingeben. Anschließend wird der Zähler der Prüfung um 1 erhöht und der Prüfer kann wiederum das nächste Bauteil prüfen.

4.1.2 Die Funktion öffnen()

Die Funktion öffnen realisiert wie der Name schon sagt das Öffnen eines bereits gespeicherten Prüfungsstandes.
Dabei werden je nach eingegebener eindeutiger Seriennummer einer Leiterplatte die entsprechenden Daten aus der Datenbank selektiert. Anschließend werden Objekte der Klasse PCB, welche die Leiterplatte an sich repräsentieren, angelegt, Objekte der Klasse Placement, für jedes Bauteil, welches auf der Leiterplatte verwendet wird und Objekte der Klasse Comp für die verschiedenen Geometrien der Bauteile.
Anschließend werden alle Objekte in entsprechende ArrayListen gespeichert um die Objekte zentral verfügbar zu machen. Anschließend kann das Layout dann anhand der verschiedenen Eigenschaften der Objekte wie z.B. Position, Rotation, Status etc. auf die GUI gezeichnet werden.

4.1.3 Die Funktion speichern()

Diese Funktion realisiert im Gegensatz zum Öffnen das Speichern eines aktuellen Prüfungsstandes.
Dabei wird im 1. Schritt eine Verbindung mit der Datenbank aufgenommen. Anschließend werden alle ArrayListen, welche die vorher bereits angesprochenen Objekte speichern, komplett durchlaufen und für jedes Objekt werden über deren get-Methoden die Eigenschaften ausgelesen und in die entsprechenden Tabellen der Datenbank geschrieben.
Hierbei ist zu beachten, dass die Funktion des Speicherns auf 2 verschiedene Art und Weisen stattfinden muss:

- **Leiterplatte neu angelegt:**
  Sofern eine Leiterplatte neu angelegt wurde, d.h. noch nicht in der Datenbank vorhanden war, müssen die vorhandenen Daten über die „insert * into…“ in die Datenbank geschrieben werden.

- **Leiterplatte geöffnet:**
  Sollte die Leiterplatte im Vorfeld über die öffnen()-Funktion eingelesen worden sein, d.h. die Leiterplatte ist in der Datenbank bereits vorhanden, müssen alle Daten über die „update…“ Anweisung in der Datenbank aktualisiert werden.

4.2 GUI_new.java

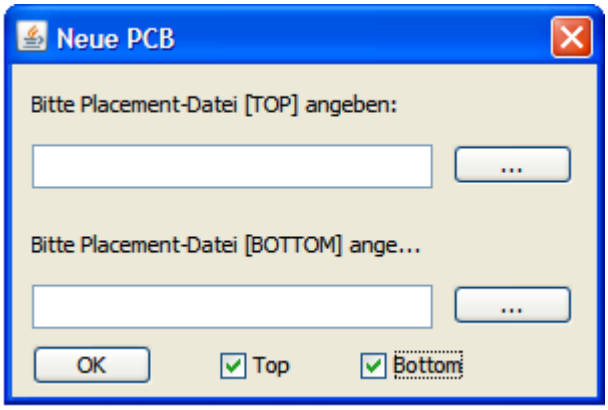
Diese Klasse wird innerhalb der Klasse GUI über den Menüpunkt Datei → Neu aufgerufen und ist dafür zuständig, neue Leiterplatten anzulegen.
Hierzu wurde ein Dialog erstellt, welcher 2 Textfelder, jeweils für die Top- bzw. die Bottom-Seite, beinhaltet. Über einen Öffnen-Dialog, welcher über die Klasse Datei aufgerufen wird, kann der Anwender nun den Pfad zum gewünschten Bestückfile angeben.
Sofern die jeweiligen Bestückfiles angegeben wurden, werden diese nun zeilenweise eingelesen. Ein Bestückfile besteht dabei in Form einer Textdatei und beinhaltet verschiedene Daten für die jeweiligen Bauteile:

- Sachnummer des Bauteils
- Position des Bauteils auf der Leiterplatte in Form von x- und y-Koordinaten
- Rotation des Bauteils auf der Leiterplatte
- Name des Bauteils auf der Leiterplatte

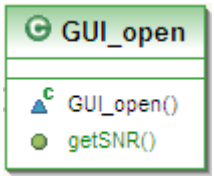
Diese Daten sind über das Trennzeichen „;“ getrennt, d.h. pro eingelesener Zeile wird ein Objekt der Klasse Placement erstellt, welches das Bauteil repräsentiert. Danach wird die Zeile über das Trennzeichen aufgeteilt und die einzelnen Daten werden den jeweiligen Klassenvariablen der Objekte zugewiesen. Anschließend wird jedes Objekt, welches erstellt wurde, einer vorhandenen ArrayList hinzugefügt, um alle Objekte zentral zu speichern und auf diese an jeder Stelle der Software zugreifen zu können.
Abschließend kann das Layout anhand dieser eingelesenen Eigenschaften für jedes Bauteil gezeichnet werden.

```
....
CMP000002502;22.54;4.13;0.0;L3
CMP000002502;15.40;11.27;0.0;L4
CMP000002502;6.51;11.27;90.0;L5
....
```

Auszug einer Bestückdatei

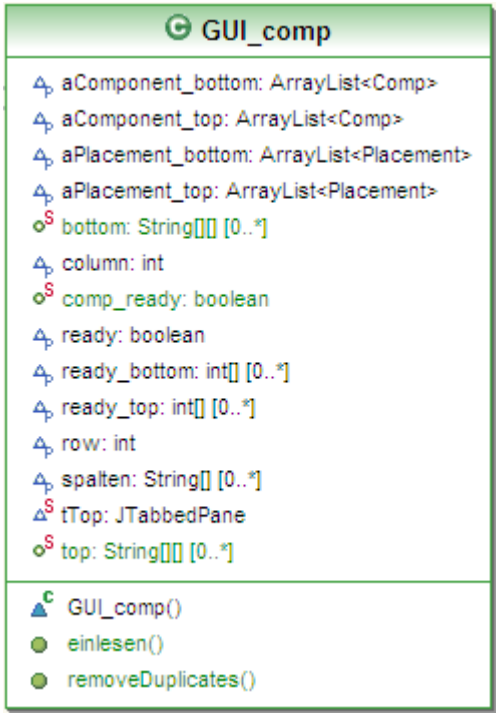


4.3 GUI_open.java



Die Klasse GUI_open.java tritt in Erscheinung sobald der Anwender über die Menüführung Datei → Öffnen einen bereits gespeicherten Stand öffnen will. Dazu wurde eine GUI erstellt, welche lediglich ein Textfeld beinhaltet. In dieses Textfeld wird die gewünschte eindeutige Seriennummer der Leiterplatte eingegeben. Anschließend wird dieser String in einer Variablen gespeichert und über deren get-Methode zurückgeliefert. Dieser String wird innerhalb der Klasse GUI.java von der Methode oeffnen(), welche bereits beschrieben wurde verwendet, um die gewünschten Daten aus der Datenbank zu lesen.

4.4 GUI_comp.java



Die Klasse GUI_comp.java wird mehrmals innerhalb der Software aufgerufen. Zum einen wird sie automatisch aufgerufen, wenn Leiterplatten neu angelegt bzw. geöffnet werden, des Weiteren wird diese Klasse innerhalb der Klasse GUI.java aufgerufen indem man über den Menüpunkt Daten den Punkt Bauteil wählt. Hauptaufgabe dieser Klasse ist die Definition der Bauteile bzgl. deren Abmessungen (Länge, Breite, Höhe), deren Typ, deren Bauform und der Pfad zur Geometrie-Library, welche die Daten der geometrischen Form des Bauteils beinhaltet.

Die entstehenden Daten aller Bauteile werden innerhalb dieser GUI zur besseren Übersicht tabellarisch dargestellt. Hierzu wird im 1. Schritt eine Tabelle erstellt, welche anschließend gefüllt wird. Dazu wird zunächst ein Array erstellt, welches die Größe der Placement-ArrayList besitzt, also jene ArrayList, in welchem in vorheriger Klasse alle Objekte (Bauteile) gespeichert wurden. Anschließend wird dieses Array mit den Sachnummern aller Bauteile gefüllt und von Duplikaten entfernt. Grund ist folgender: Bauteile können auf einer Leiterplatte mehrfach vorkommen, z.B. ein Widerstand von 100Ohm der an verschiedenen Stellen der Leiterplatte vorhanden ist. Diese Bauteile müssen zwar in der Placement-ArrayList komplett gespeichert werden, müssen jedoch nur einmal in aktueller Klasse definiert werden.

Anschließend wird für jedes Bauteil des Arrays geprüft, ob die Sachnummer des gespeicherten Bauteils bereits in der Datenbank vorhanden ist. Ist dies der Fall, so werden die ausgelesenen Daten

aus der Datenbank in ein weiteres Array geschrieben. Ist dies nicht der Fall, wird lediglich die Sachnummer des Bauteils in das weitere Array geschrieben.

Anschließend wird der Inhalt des zusätzlich gefüllten Arrays in die Tabelle geschrieben und alle Daten sind nun tabellarisch und übersichtlich aufgelistet.

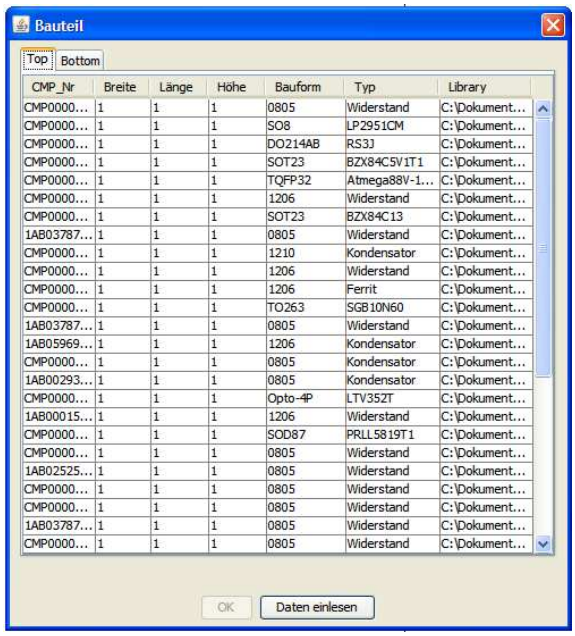
Über einen Klick auf die jeweilige Tabellen-Zeile können nun die Daten definiert werden, sofern sie nicht schon vorhanden sind. Dabei wird die Klasse GUI_comp_define aufgerufen, welche später näher beschrieben wird.

Sobald der Dialog wieder geschlossen wird, werden die eingegebenen Daten in die Datenbank geschrieben, sofern sie neu angegeben wurden, bzw. aktualisiert, sofern die Daten bereits vorhanden waren und lediglich geändert wurden.

Ein Klick auf den Button Daten einlesen liest nun alle relevanten Daten über die Funktion einlesen() ein.

Nun kann also jedes Bauteil, dessen Eigenschaften schon über die Klasse Placement bekannt sind, auf dem Layout anhand der nun eingelesenen Geometrie-Daten gezeichnet werden, denn erst jetzt kennt die Software die eigentliche Form bzw. Geometrie des Bauteils.

4.4.1 Die Funktion einlesen()



Die Funktion einlesen() ist eine der zentralsten Funktionen dieser Software, denn mit Hilfe dieser Funktion geben wir der Software bekannt, welche Form, d.h. Geometrie das jeweilige Bauteil hat. Hierzu werden sog. Geometrie-Libraries in Form von Text-Dateien verwendet, welche zentral auf dem Server gespeichert sind und wie schon angesprochen die Geometrie-Daten beinhalten. Eine Geometrie-Library besteht aus 3 Grundgeometrien, welche zeilenweise in jeweiliger Textdatei vorhanden sind:

- ElementArc:
  Diese Geometrie beschreibt Kreisbögen, welche beschrieben werden durch einen Anfangs- und einen Endpunkt, einen Durchmesser, sowie den Öffnungs- und den Schließungswinkel.

- ElementLin
  Diese Geometrie beschreibt Geraden, repräsentiert durch einen Anfangs- und einen Endpunkt

- ElementRec
  Diese Geometrie beschreibt Rechtecke, beschrieben durch einen Anfangs- und einen Endpunkt

```
....
ElementLin;-89;-75;89;-75;4
ElementLin;89;-75;89;75;4
ElementRec;-110;43;-89;-43
....
```

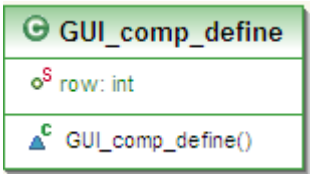
Auszug einer Geometrie-Library

Die Funktion einlesen erstellt im 1. Schritt nun für jede dieser Geometrien eine eigene ArrayList, abhängig von der jeweiligen Klasse der Geometrie (Arc, Rect, Line). Anschließend wird die Library zeilenweise eingelesen, dabei wird zuerst geprüft, um welche Geometrie es sich handelt. Abhängig von der erkannten Geometrie-Art wird nun das jeweilige Objekt der Klasse (Arc, Rect, Line) erstellt und anschließend wird dieses Objekt der jeweiligen ArrayList hinzugefügt. Im letzten Schritt wird für jedes Bauteil ein Objekt der Klasse Comp erstellt, welches also dessen Geometrie repräsentiert und die Daten der Tabelle bzw. der eingelesenen Geometrie-Daten werden den jeweiligen Klassenvariablen zugewiesen. Zuletzt wird dieses erstellte Objekt der zentralen Comp-ArrayList hinzugefügt. Dadurch kann jetzt also jedes Bauteil anhand seiner Placement- und seiner Comp-Daten auf die GUI gezeichnet werden.

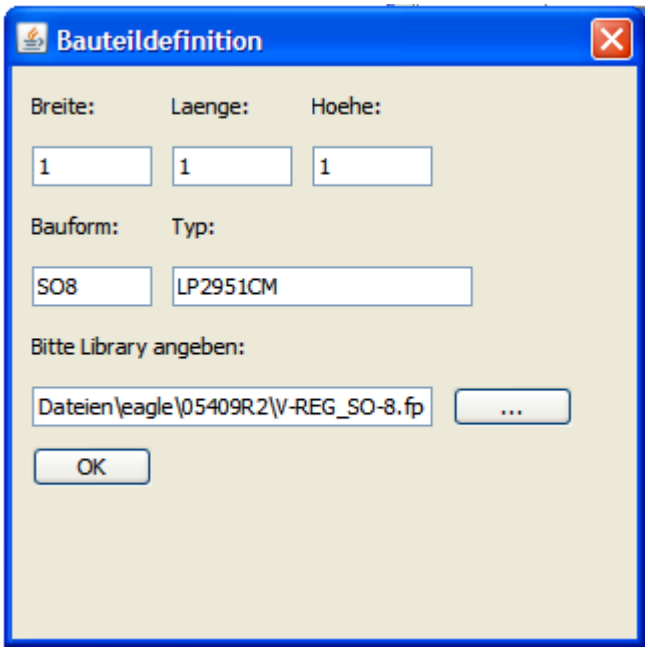
4.4.2 Die Funktion removeDuplicates()

Wie schon erwähnt, können Bauteile zwar mehrfach auf einer Leiterplatte vorkommen, müssen jedoch lediglich einmal definiert werden. Hierzu dient die Funktion removeDuplicates(). Im 1. Schritt dieser Funktion wird ein übergebenes Array mit allen Sachnummern der Bauteile gefüllt. Anschließend wird ein HashSet erstellt. Vorteil eines HashSet ist, dass er keine Duplikate erlaubt! In diesen HashSet werden nun über eine for-Schleife alle Sachnummern aus dem Array hinzugefügt, mit dem Vorteil, dass bereits vorhandene Sachnummern nicht hinzugefügt werden. Im letzten Schritt wandelt die Funktion diesen HashSet wiederum in ein Array um und gibt dieses zurück. Als Ergebnis bekommen wir also das von Duplikaten entfernte Array zurück.

4.5 GUI_comp_define.java



Die Klasse GUI_comp_define ist prinzipiell eine Erweiterung der im Vorfeld vorgestellten Klasse GUI_comp. Aktuelle Klasse wird innerhalb dieser Klasse als ActionListener der Tabelle implementiert. Sobald man in der tabellarischen Übersicht der Bauteile auf die jeweilige Zeile klickt, so instanziiert man ein Objekt der Klasse GUI_comp_define. Diese Klasse hat lediglich die Aufgabe, jene Daten einzugeben, welche danach in der Tabelle angezeigt und danach eingelesen werden.



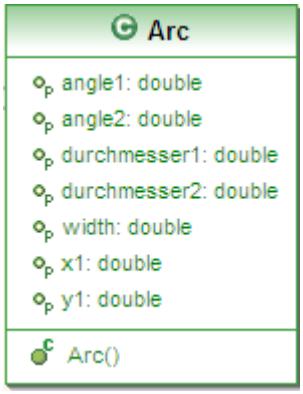
Hierzu wurde innerhalb dieser Klasse eine GUI erstellt, welche die geforderten Textfelder enthält. Über diese Textfelder, welche zu Beginn mit den aktuellen Werten der Tabelle gefüllt sind, können letztendlich die gewünschten Daten eingegeben werden. Daten sind innerhalb dieser Klasse die Abmessungen des Bauteils (Länge, Breite, Höhe), die Bauform des Bauteils, der Typ und der Pfad zur Library. Letzterer Pfad wird über den Öffnen-Dialog der Klasse Datei.java, welche später näher beschrieben wird, ausgewählt.

Nachdem alle Daten korrekt angegeben wurden, werden diese in die Tabelle, aus welcher diese Klasse gestartet wurde, übernommen und können über die Klasse GUI_comp.java eingelesen werden.

4.6 Arc.java / Rect.java / Line.java

Bereits in der Klasse GUI_comp.java wurden die 3 Grundgeometrien vorgestellt, aus denen sich die Form, also die Geometrie eines Bauteils ergibt.
Wie bereits erwähnt, werden diese 3 Geometrien innerhalb der Klasse GUI_comp.java eingelesen, indem eine Geometrie-Library in Form einer Textdatei zeilenweise eingelesen wird und für jede erkannte Geometrie ein Objekt der folgenden Klassen erstellt wird.
In diesem Abschnitt der Dokumentation sollen diese 3 Klassen, welche jeweils eine der 3 Geometrien repräsentieren, vorgestellt werden.

4.6.1 Arc.java

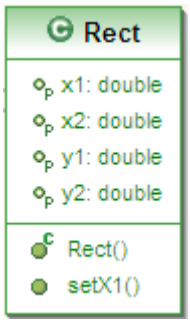


Objekte der Klasse Arc.java repräsentieren die Geometrie des Kreisbogens, d.h. auch komplette Kreise. Diese Objekte werden über folgende Klassenvariablen beschrieben:

- **angle1:**
  Steht für den Anfangswinkel des Kreisbogens
- **angle2:**
  Bezeichnet den Öffnungswinkel des Kreisbogens
- **durchmesser1:**
  Steht für den Durchmesser eines Kreises
- **durchmesser2:**
  siehe Durchmesser1
- **width:**
  Steht für den horizontalen Durchmesser, sofern Ellipsenbögen erstellt werden
- **x1, y1:**
  Beschreibt die obere linke Ecke, des die Ellipse oder des Kreises umfassenden Rechtecks

Für jede Zeile einer Library, welche also diese Geometrie betrifft, werden die Daten eingelesen und über einen Konstruktor, dem diese Daten übergeben werden, werden die Objekte angelegt und einer ArrayList dieser Klasse hinzugefügt.

4.6.2 Rect.java



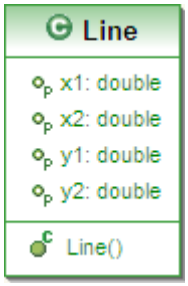
Diese Klasse, bzw. Objekte dieser Klasse stehen für die Geometrie des Rechteckes. Ein Rechteck dieser Klasse wird über folgende Klassenvariablen beschrieben:

- **x1, y1:**
  Die obere, linke Ecke des Rechteckes.

- **x2, y2:**
  Die untere, rechte Ecke des Rechteckes

Ähnlich wie in der Klasse Arc.java, werden diese Geometrien-Daten ebenfalls aus der Geometrie-Library ausgelesen und die Objekte werden über einen Konstruktor erstellt. Anschließend werden auch hier alle Objekte einer vorhandenen ArrayList der Klasse Rect hinzugefügt.

4.6.3 Line.java



Die letzte Klasse der 3 Geometrien repräsentiert einfache Geraden. Solche Geraden werden durch folgende Klassenvariablen beschrieben:

- **x1, y1:**
  Anfangspunkt der Geraden

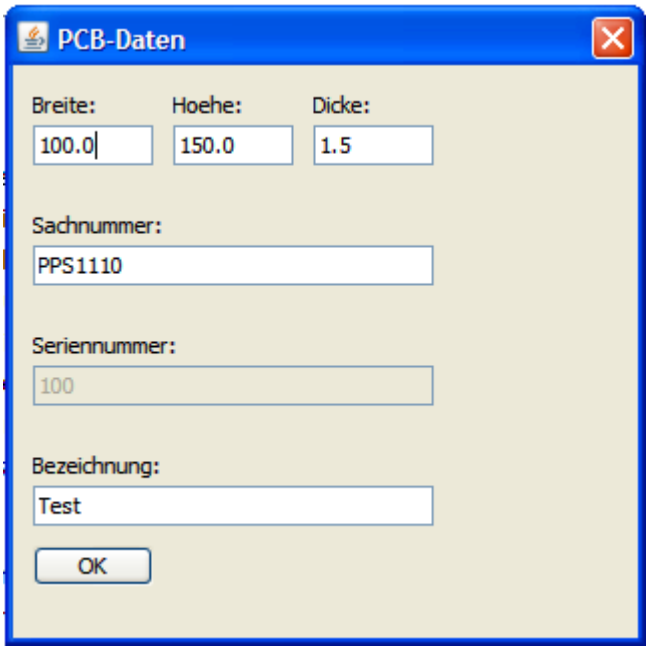
- **x2, y2**
  Endpunkt der Geraden

Genau wie bei den anderen 2 Klassen auch, werden diese Daten aus der jeweiligen Zeile der Geometrie-Library ausgelesen und wiederum werden über den Konstruktor Objekte erstellt, welche der vorhandenen ArrayList hinzugefügt werden.

Abschließend lässt sich also beim Ablauf der Geometrie folgendes zusammenfassen:
Alle Daten der Geometrien liegen zentral auf dem Server als Library in Form einer Textdatei, welche über die Klasse GUI_com.java zeilenweise eingelesen werden.
Basierend auf der Art der Geometrie werden für jede Zeile Objekte der Klassen Arc, Rect oder Line erstellt, um die vorliegende Art der Geometrie zu repräsentieren.
Zudem existieren für jede Klasse ArrayListen, welche den Vorteil bieten, Objekte der jeweiligen Klassen zu speichern. Diese ArrayListen werden jedem Objekt der Klasse Comp.java, welche die Abmessungen inkl. der Geometrien repräsentieren, bei der Instanziierung übergeben. Auf diese Art und Weise sind alle Daten der Bauteile zentral vorhanden.

4.7 GUI_pcb.java

In den letzten Abschnitten wurde beschrieben, wie Daten der Bauteile definiert werden bzw. wie Geometrie-Daten eingelesen werden. Die Klasse GUI_pcb.java ist nun die letzte Klasse, mit welcher Daten definiert werden können, nämlich solche für die Leiterplatte selbst. Diese Klasse wird aufgerufen einerseits beim Erstellen einer neuen Leiterplatte und andererseits innerhalb der Klasse GUI.java über das Menü Daten → PCB. Bei jedem Anlegen einer neuen Leiterplatte wird ein Objekt der Klasse PCB erstellt, welch die Leiterplatte selbst repräsentiert. Über die aktuelle Klasse können die Daten dieses Objektes definiert werden.



Hierzu wurde eine GUI erstellt, die entsprechende Textfelder enthält. Über diese GUI können nun entsprechende Daten eingegeben werden. Zu den Daten einer Leiterplatte gehören zum einen deren Abmessungen (Breite, Höhe, Dicke), zum anderen die Sachnummer, deren Seriennummer und die Klartext-Beschreibung. Sobald alle Daten eingegeben wurden, werden die Klassenvariablen der Objekte gesetzt und wie auch bei allen anderen Objekten einer entsprechenden ArrayList hinzugefügt.

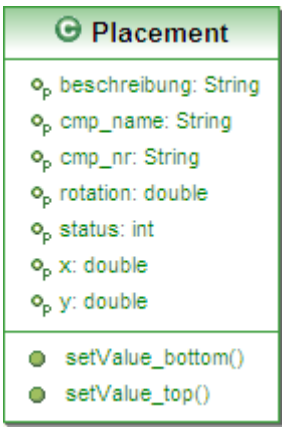
Nachdem wir diese Daten eingegeben haben, sind die Definitionen aller Bauteile und Leiterplatten abgeschlossen. Die Software beinhaltet nun 3 ArrayListen mit entsprechenden Objekten:

- ArrayList<Placement>:
  Objekte repräsentieren jedes Bauteil, welches auf der Leiterplatte verwendet wird. Eigenschaften dieser Objekte sind zum einen die Position der Bauteile auf der Leiterplatte, deren Rotation, die Sachnummer und der Name

- ArrayList<Comp>:
  Objekte dieser ArrayList stehen für die Geometrie der Bauteile, d.h. deren Abmessungen, Bauformen, Typen und Formen in Form von Kreisbögen, Geraden und Rechtecke

- ArrayList<PCB>:
  Diese Objekte repräsentieren die Leiterplatte selbst über deren Eigenschaften wie Abmessungen, Sach- und Seriennummer und deren Bezeichnung

4.8 Placement.java



Wie bereits beschrieben repräsentieren Objekte der Klasse Placement jedes Bauteil, welches auf der Leiterplatte verwendet wird. D.h. für jede Zeile der Bestückdatei, welche über die Klasse GUI_new.java eingelesen wurde, wird ein solches Objekt dieser Klasse erstellt. Diese Objekte werden dabei über folgende Klassenvariablen beschrieben:

- **x,y:**
  Diese Variablen stehen für die Position des Bauteils auf der Leiterplatte in Form von x- und y-Koordinaten

- **rotation:**
  Die Rotations-Variable beschreibt wie der Name schon sagt die Rotation eines Bauteils auf der Leiterplatte. Dies ist besonders dann enorm wichtig, wenn es sich um gepolte Bauteile handelt, wie z.B. Dioden oder ICs etc.

- **cmp_nr:**
  Beschreibt die eindeutige Sachnummer eines Bauteils. Über diese eindeutige Sachnummer kann bsp. das Bauteil innerhalb einer Datenbank oder anderen Programmen eindeutig referenziert werden.

- **cmp_name:**
  Steht für den Namen des Bauteils auf der Leiterplatte. Diese Namen werden im Grunde zur besseren Übersichtlichkeit der Bestückung verwendet, so werden z.B. Widerstände (eng. Resistor) als R1 – Rx bezeichnet usw.

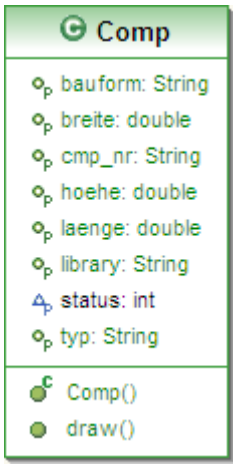
- **status:**
  Diese Variable wird während der Prüfung verwendet. Es gibt innerhalb der Prüfung 3 Stadien:
  - **0:** Bauteil wurde noch nicht geprüft
  - **1:** Bauteil wird aktuell geprüft (gelb eingefärbt)
  - **2:** Bauteil ist korrekt (grün eingefärbt)
  - **3:** Bauteil ist fehlerhaft (rot eingefärbt)

- **beschreibung:**
  Auch diese Variable wird während der Prüfung verwendet. In diese String-Variable werden Fehlerbeschreibungen geschrieben, sofern ein Bauteil als fehlerhaft identifiziert wurde.

4.8.1 Die Funktion setValue()

Die Funktion setValue() wird innerhalb dieser Klasse verwendet, um die eingelesenen Zeilen des Bestückfiles in kleinere Teilbereiche aufzuteilen. Dabei wird der Funktion die komplette Zeile als String übergeben. Anhand des Trennzeichens werden nun Sub-Strings dieser Zeile, also Teilbereiche, erstellt, welche Daten für die entsprechenden Klassenvariablen beinhalten. Anschließend wird jeder dieser Sub-Strings diesen Variablen zugewiesen.

4.9 Comp.java



Die Klasse Comp.java beinhaltet zum einen Objekte, welche wie bereits erwähnt, die Geometrien der einzelnen Bauteile repräsentieren.
Diese Objekte werden wie bereits erwähnt über die Klasse GUI_comp.java für jede Art von Bauteil einmalig erstellt.

Zum anderen ist diese Klasse auch dafür zuständig diese Bauteile über die Methode draw(), welche innerhalb der Klasse Layout aufgerufen wird, zu zeichnen. Objekte dieser Klasse werden dabei über folgende Klassenvariablen beschrieben:

- **breite:**
  Die Breite des Bauteils in mm
- **hoehe:**
  Die Höhe des Bauteils in mm
- **laenge:**
  Die Länge des Bauteils in mm
- **bauform:**
  Die Bauform des Bauteils, z.B. 0805, PLCC, BGA usw.
- **typ:**
  Der Typ des Bauteils, z.B. Widerstand, Diode, Kondensator usw.
- **library:**
  Der Pfad zur Geometrie-Library, welcher die Geometrie-Daten des Bauteils beinhaltet

- **status:**
  Der Status des Bauteils während der Prüfung. Dieser Status wurde bereits in vorherigen Abschnitt innerhalb der Klasse Placement.java beschrieben.

- **aRect:**
  ArrayList, welche Objekte der Klasse Rect beinhaltet. Diese Objekte stehen für Geometrien der Form Rechteck.

- **aArc:**
  ArrayList, welche Objekte der Klasse Arc beinhaltet. Diese Objekte stehen für Geometrien der Form Kreis- und Ellipsebögen bzw. Kreise.

- ArrayList, welche Objekte der Klasse Line beinhaltet. Diese Objekte stehen für die Geometrien der Form Gerade.

Letztere 3 ArrayListen wurden bereits beschrieben und werden jedem Objekt der Klasse Comp beim Erstellen über dessen Konstruktor übergeben.

4.9.1 Die Funktion draw()

Wie bereits zu Beginn dieses Abschnittes erwähnt hat die Klasse Comp.java zudem die Funktion, die Bauteile anhand deren gespeicherter Geometrie-Daten auf die GUI zu zeichnen. Dazu wird die Funktion draw() verwendet, welche innerhalb der Klasse Layout aufgerufen wird. Diese Funktion bekommt folgende Übergabeparameter beim Aufruf:

- **g:**
  Das Graphics Objekt g, welches zum Zeichnen verwendet wird

- **x, y:**
  Die Position des Bauteils auf der Leiterplatte

- **faktor:**
  Der Wert der Vergrößerung bzw. Verkleinerung. Über diesen Faktor wird die ursprüngliche Größe des Bauteils erhöht bzw. verringert.

- **nr:**
  Die eindeutige Sachnummer des Bauteils.

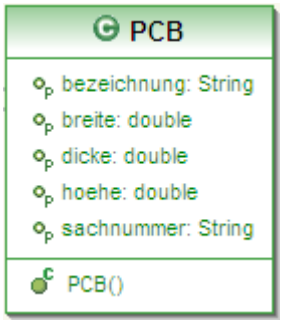
- **rotation:**
  Die Rotation des Bauteils auf der Leiterplatte in der Einheit Grad.

- **name:**
  Der Name des Bauteils auf der Leiterplatte, z.B. R1.

- **status:**
  Der Status der Prüfung, welcher schon beschrieben wurde.

Anhand dieser übergebenen Parameter ist diese Funktion nun in der Lage, das jeweilige Bauteil an die richtige Stelle der GUI zu zeichnen.
Hierzu werden im 1. Schritt 3 Affine Transformationen erstellt, jeweils für die Verschiebung des Bauteils an die richtige Koordinate (x und y), für die Skalierung des Bauteils anhand des übergebenen Faktors und für die Rotation des Bauteils auf der Leiterplatte anhand des übergebenen Rotations-Wertes. Im 2. Schritt wird nun für jede der 3 Grund-Geometrien (Arc, Rect, Line) ein sog. Shape erzeugt, welches im weiteren Verlauf mit den Geometrie-Objekten gefüllt wird.
Anschließend wird nun für jede ArrayList der 3 Geometrien geprüft, ob diese Objekte beinhalten. Sofern dies der Fall ist, so werden diese Objekte anhand ihrer Klassenvariablen (Anfangspunk, Endpunkt etc.) in die jeweiligen Shapes gezeichnet. Diese Shapes werden dann mit den Affinen Transformationen verknüpft, um die Bauteile an die richtige Position der GUI, mit der richtigen Skalierung und der richtigen Rotation zu zeichnen.
Diese Zeichnung geschieht im letzten Schritt, so wird für jedes Objekt der jeweilige übergebene Status abgefragt und dementsprechend die Farben (gelb, grün, rot) zugewiesen. Anschließend werden die Bauteile nun auf die GUI gezeichnet.

4.10 PCB.java



Die Klasse PCB.java referenziert Objekte, welche die Leiterplatte an sich repräsentieren.
Diese Objekte werden erstellt, sobald eine neue Leiterplatte angelegt wird.
Dabei werden die Objekte über folgende Eigenschaften in Form von Klassenvariablen beschrieben:

- **breite:**
  Die Breite der Leiterplatte in mm

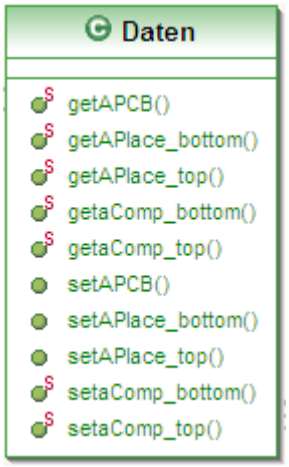
- **hoehe:**
  Die Höhe der Leiterplatte in mm.

- **dicke:**
  Die Dicke der Leiterplatte in mm.

- **bezeichnung:**
  Die Klartext-Bezeichnung der Leiterplatte.

- **sachnummer:**
  Die eindeutige Sachnummer der Leiterplatte um diese z.B. in Datenbanken oder weiteren Programmen eindeutig referenzieren zu können.

4.11 Daten.java



Die Klasse Daten.java wurde realisiert, um die bereits angesprochenen ArrayListen zentral in der Software abzulegen und auf die in ihnen enthaltenen Objekte zugreifen zu können. Klassenvariablen dieser Klasse sind also die ArrayListen selbst. Folgende 3 ArrayListen werden verwendet:

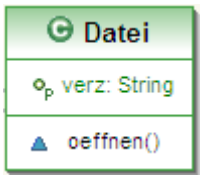
- **aPCB:**
  ArrayList der Klasse PCB. In ihr werden Objekte der Klasse PCB, welche die Daten der Leiterplatte selbst repräsentieren, gespeichert.

- **aPlace_top / aPlace_bottom**
  ArrayList der Klasse Placement. Hier warden Objekte der Klasse Placement gespeichert. Diese Objekte repräsentieren alle auf der Leiterplatte vorkommenden Bauteile anhand ihrer Eigenschaften wie Position, Rotation etc.

- **aComp_top / aComp_bottom**
  ArrayList der Klasse Comp. Diese ArrayList sichert Objekte der Klasse Component. Diese Objekte repräsentieren die geometrischen Daten der Bauteile.

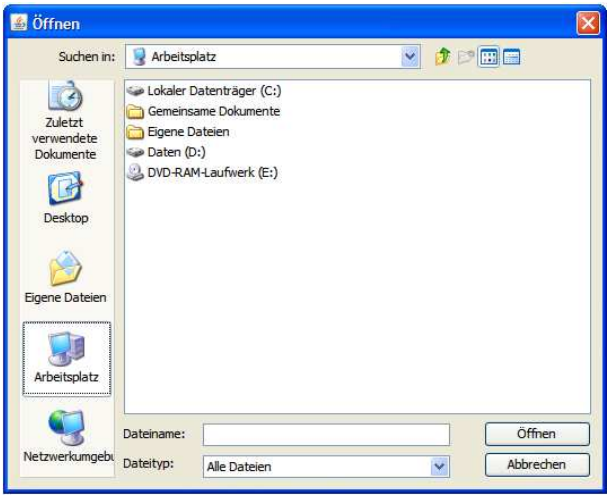
Mit Hilfe der get- bzw. set-Methoden der jeweiligen ArrayListen können diese nun zentral gefüllt werden bzw. zurückgegeben werden. Dadurch können alle Daten zentral von der Software gespeichert werden und an jeder beliebigen Stelle wieder abgerufen werden.

4.12 Datei.java



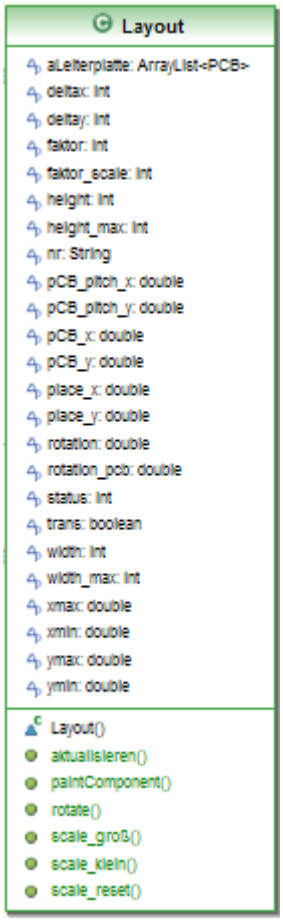
Die Klasse Datei.java realisiert den Öffnen-Dialog, welcher innerhalb der Klassen GUI_new.java und GUI_comp_define.java aufgerufen wird. Dabei wird ein Öffnen-Dialog angezeigt, in dem man den Pfad zur gewünschten Datei angeben kann. Anschließend wird dieser Pfad zurückgegeben.

4.12.1 Die Funktion oeffnen()



Zentraler Teil dieser Klasse ist die Funktion oeffnen(), welche den bereits angesprochenen Dialog realisiert. Dazu wird ein sog. JFileChooser instanziiert, welcher den Dialog öffnet. Einstiegspunkt in die Verzeichnisstruktur ist dabei das Home-Verzeichnis des Anwenders. Anschließend kann der Anwender nun den gewünschten Pfad auswählen, welcher in die Klassenvariable „verz“ als String geschrieben wird. Diese Variable beinhaltet eine get-Methode, über welche der String des Pfades nun zurückgegeben werden kann.

4.13 Layout.java



Hauptaufgabe dieser Klasse ist das Zeichnen des Layouts der Leiterplatte auf die GUI. Also das digitale Visualisieren des Layouts von der ursprünglichen Papier-Form, wie es in den Anforderungen verlangt wurde. Dieser Prozess wird durch mehrere Funktionen realisiert, welche im Folgenden näher beschrieben werden.

4.13.1 Die Funktion zeichneLayout()

Wie eben bereits angedeutet, zeichnet diese Klasse das Layout auf die GUI. Hierzu werden vorhandene ArrayListen abhängig von der entsprechenden Seite der Leiterplatte mit den benötigten Objekten gefüllt. Wie bereits beschrieben, sind diese Objekte zentral in ArrayListen der Klasse Daten.java gespeichert und können nun über deren get-Methoden ausgelesen werden.

Anschließend werden die Positionskoordinaten aller Bauteile auf der Leiterplatte sortiert, um festzustellen, wie breit die eigentliche Leiterplatte ist. Um dies zu realisieren werden 2 temporäre Arrays, jeweils für die x- und die y-Koordinaten, erstellt und mit den dementsprechenden Werten aller Bauteile gefüllt. Anschließend werden diese aufsteigend sortiert. Wir erhalten also 4 Koordinaten:

- **xmin:**
  Koordinate des Bauteils mit dem geringsten Wert in horizontaler Richtung, also ganz links.

- **xmax:**
  Koordinate des Bauteils mit dem maximalsten Wert in horizontaler Richtung, also ganz rechts.

- **ymin:**
  Koordinate des Bauteils mit dem geringsten Wert in vertikaler Richtung, also ganz oben.

- **ymax:**
  Koordinate des Bauteils mit dem maximalsten Wert in vertikaler Richtung, also ganz unten.

Sobald diese 4 Werte berechnet wurden, können wir die Höhe bzw. die Breite der Leiterplatte berechnen, indem wir lediglich die Differenz zwischen maximalem und minimalem Wert, sowohl in horizontaler als auch in vertikaler Richtung berechnen.
Hierbei sei noch erwähnt, dass es sich bei dieser Berechnung nicht um die realen Abmessungen der Leiterplatte handelt, diese werden über die Klasse GUI_pcb.java definiert. Diese Abmessungen

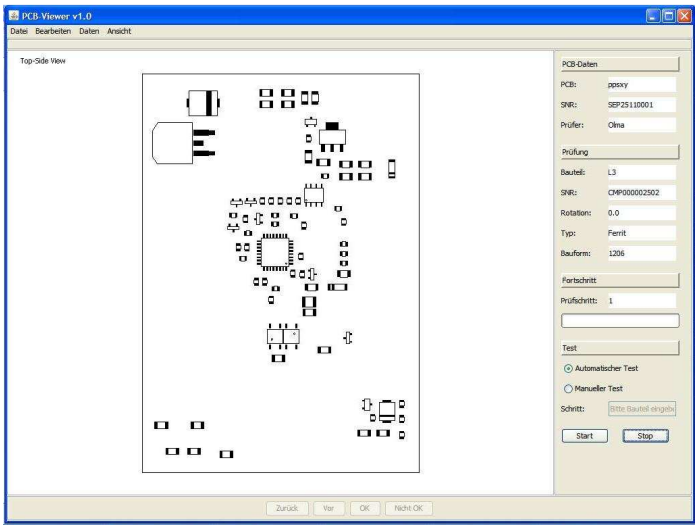
dienen lediglich dazu, die groben Umrisse der Leiterplatte und die dementsprechende Größe auf der GUI zu berechnen, um die komplette Leiterplatte auf dieser GUI darstellen zu können.

Im nächsten Schritt wird ausgehend von der momentan berechneten Breite bzw. Höhe und einer fest eingestellten Größe, nämlich der Größe des Panels auf der GUI, ein Vergrößerungs- bzw. Verkleinerungsfaktor berechnet, welcher dafür zuständig ist, die komplette Leiterplatte in der größtmöglichen Ansicht darzustellen. Hierzu wird lediglich abgefragt, ob die Ansicht, multipliziert mit dem Faktor, die maximale Größe überschreitet. Sofern dies der Fall ist, wird der Faktor verringert, bis wir den idealen Wert gefunden haben.

Anschließend wird in die GUI ein Rechteck, ausgehend von der berechneten Breite und Höhe und dem Faktor in die GUI gezeichnet, um den Umriss der Leiterplatte darzustellen.

Im nächsten Schritt, wird nun ein sog. Pitch in x- und in y-Richtung berechnet. Dieser Pitch ist der Wert vom Mittelpunkt der Leiterplatte bis zur oberen, linken Ecke der Leiterplatte. Dieser Pitch wird benötigt, um einen definierten Nullpunkt zu erlangen, von dem aus die Koordinaten für die Bauteile berechnet werden können. Denn die Koordinaten, welche über das Bestückfile eingelesen werden, gehen von einem Nullpunkt aus, welcher irgendwo im Raum liegen kann. D.h. wir nehmen das komplette Layout aus dem Bestückfile und verschieben dieses komplette Layout in die Mitte der GUI.

Im letzten Schritt dieser Funktion werden jetzt für jedes Objekt der Klasse Placement, deren Koordinaten, Rotation und weitere Daten ausgelesen. Anschließend wird anhand der Sachnummer für jedes Bauteil abgefragt, ob dieses ein vergleichbares Objekt der Klasse Component hat. Sofern diese Abfrage zutrifft, kann jetzt die entsprechende Methode draw() des Comp-Objektes aufgerufen werden, um das entsprechende Bauteil an die entsprechende Stelle der GUI zu zeichnen. Die Methode draw() wurde innerhalb der Klasse Comp.java bereits beschrieben.



4.13.2 Die Funktion paintComponent()

Das eigentliche Zeichnen (auf einem JPanel) übernimmt die Methode *paintComponent()*. Diese Methode wird immer dann vom Betriebssystem / der Java Virtual Machine aufgerufen, wenn sich das Windowsfenster neu zeichnen muss, wenn es z. B. von einem anderen Fenster verdeckt war. Für das Betriebssystem stellt der gesamte Bildschirm eine einzige Grafikfläche dar, für uns sieht es jedoch so aus, als gebe es Fenster, Buttons usw. Wenn dann also die JVM die *paintComponent()* auruft, wird dort ein Graphics-Objekt übergeben, das praktisch den Bereich auf dem Bildschirm repräsentiert, wohin gezeichnet werden kann.

Im 1. Schritt dieser Funktion wird abhängig von der gewählten Ansicht (Top- oder Bottom) ein String in die GUI gezeichnet, um dem Anwender klar sichtbar zu machen, auf welcher Seite der Leiterplatte er sich momentan befindet.

Anschließend wird abhängig vom Rotationswinkel das komplette Layout gedreht. Dieser Rotationswinkel ist anfangs mit dem Wert 0 initialisiert, kann aber über die Menüführung Bearbeiten → Drehen entsprechend verändert werden, was an dieser Stelle zu einer entsprechenden Drehung des Layouts führt. Zusätzlich wird das komplette Layout an dieser Stelle um einen bestimmten Wert in x- und in y-Richtung verschoben. Auch diese Werte sind Anfangs mit 0 initialisiert, werden aber dann verändert, wenn beispielsweise die Menüführung Bearbeiten → Drehen angewandt wird. Dann nämlich wird das aktuell zu vergrößernde oder zu verkleinernde Bauteil ins Zentrum der GUI verschoben.

Im letzen Schritt wird nun die eben bereits beschriebene Funktion zeichneLayout() aufgerufen, um das Layout auf die GUI zu zeichnen.

4.13.3 Die Funktion aktualisieren()

Die Methode aktualisieren() realisiert das Neuzeichnen des Layouts, indem sie die Methode repaint() aufruft. Mit der Methode *repaint()* haben wir die Möglichkeit an jeder beliebigen Stelle im Hauptprogramm ein Neuzeichnen zu erzwingen, denn diese Methode ruft wiederum die eben bereits vorgestellte Methode paintComponent() auf.

4.13.4 Die Funktion rotate()

Die Funktion rotate() hat die Aufgabe, den im Vorfeld bereits erwähnten Wert des Rotationswinkels zu setzen. Diese Funktion bekommt einen übergebenen Wert und weist diesen Wert der entsprechenden Klassenvariablen zu, so dass innerhalb der Funktion paintComponent() das Layout entsprechend des Wertes gedreht werden kann.

4.13.5 Die Funktion scale_groß()

Aufgabe dieser Funktion ist das Vergrößern der Ansicht bzw. des aktuell zu prüfenden Bauteils. Dazu wird im 1. Schritt der Abstand des Bauteils zum Mittelpunkt der Leiterplatte in horizontaler und vertikaler Richtung berechnet. Anschließend werden diese Werte den entsprechenden Klassenvariablen zugewiesen. Dadurch kann das Bauteil, wie bereits innerhalb der Funktion paintComponent() erwähnt, in das Zentrum der GUI verschoben werden und dort gezeichnet werden. Des Weiteren vergrößert diese Funktion den globalen Vergrößerungsfaktor um einen bestimmten Wert, so dass die komplette Ansicht des Layouts nun vergrößert dargestellt wird mit dem aktuell zu prüfenden Bauteil im Zentrum.

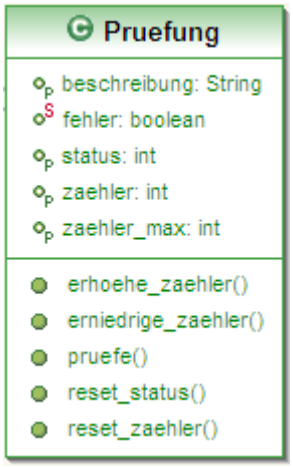
4.13.6 Die Funktion scale_klein()

Analog zu Funktion scale_groß() ist diese Funktion für das Verkleinern der Ansicht bzw. des aktuell zu prüfenden Bauteils zuständig.

4.13.7 Die Funktion scale_reset()

Diese Funktion macht im Grunde nichts anderes, als die geänderten Werte der Funktionen scale_groß() bzw. scale_klein() zurückzusetzen. D.h. die Verschiebungs-Variablen werden wieder auf 0 gesetzt und der globale Vergrößerungsfaktor wird wieder auf den Ursprungszustand gesetzt. So ist nun wieder das komplette Layout wie zuvor sichtbar.

4.14 Pruefung.java



Diese Klasse übernimmt die Steuerung der Prüfung, d.h. füllt den Informationsbereich der GUI mit den jeweiligen Information über das aktuell zu prüfende Bauteil, setzt die Stadien der Bauteile, je nachdem was der Anwender über den Steuerungsbereich der GUI angegeben hat und springt zwischen den Bauteilen vor bzw. zurück. Diese Prozesse werden über nachfolgend näher beschriebene Funktionen realisiert.

4.14.1 Die Funktion pruefe()

Zentrale Variable innerhalb dieser Funktion ist die Variable „zaehler“. Diese Variable zeigt auf das aktuelle Objekt, also Bauteil innerhalb der ArrayList. Beim ersten Aufruf, also beim Start der Prüfung, ist diese Variable auf 0 gesetzt und zeigt somit auf das 1. Bauteil bzw. Objekt in der ArrayList. Diese ArrayListen werden im 1. Schritt dieser Funktion abhängig von der Seite der Leiterplatte (Top oder Bottom) mit den jeweiligen Objekten gefüllt, welche zentral in der Klasse Daten.java abgelegt wurden.

Basierend auf dem aktuellen Zähler werden danach die jeweiligen Daten der Objekte über deren get-Methoden ausgelesen und die Textfelder im Informationsbereich der GUI gefüllt. Anschließend wird basierend auf der Eingabe des Anwenders der dementsprechende Status des aktuellen Objektes über die set-Methode gesetzt. Sollte während der Prüfung ein Fehler aufgetreten sein, wird auch die daraus folgende Beschreibung des Fehlers dem aktuellen Objekt über dessen set-Methode zugewiesen.

Nachdem die Prüfung also komplett durchlaufen wurde, hat jedes Objekt der ArrayList, also jedes Bauteil seinen dementsprechenden Status bzw. die eventuell auftretende Fehlerbeschreibung zugewiesen bekommen.

4.14.2 Die Funktion erhoehe_zaehler()

Jedes Mal wenn der Anwender über den Steuerungsbereich ein Bauteil geprüft hat, wird diese Funktion aufgerufen. Sie erhöht den Zähler, welcher innerhalb der Funktion pruefe() verwendet wird. D.h. nachdem aktuelle Funktion aufgerufen wird, wird der Zähler um eins erhöht und die Prüfung gelangt zum nächsten Objekt bzw. Bauteil.

4.14.3 Die Funktion erniedrige_zaehler()

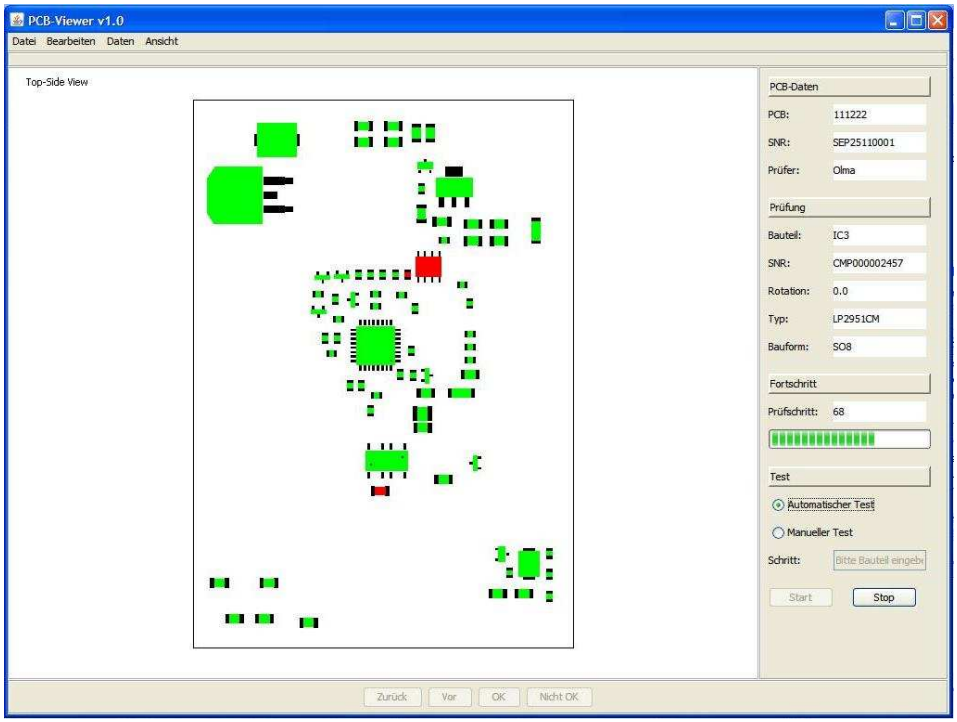
Analog zu Funktion erhoehe_zaehler() verringert diese Funktion den Zähler um 1.

4.14.4 Die Funktion reset_zaehler()

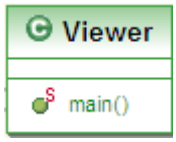
Diese Funktion setzt den Zähler auf 0 zurück. D.h. die Prüfung beginnt jetzt wieder beim ersten Objekt bzw. dem ersten Bauteil. Diese Funktion wird beispielsweise angewandt, sofern der Anwender die Prüfung stoppt, bzw. neu startet.

4.14.5 Die Funktion reset_status()

Die Funktion reset_status() setzt den Status aller Objekte auf 0 zurück, d.h. noch nicht geprüft. Dafür wird die komplette ArrayList durchlaufen und dementsprechend jedem Objekt über dessen set-Methode der Status 0 zugeteilt. Die Leiterplatte ist somit also wieder um Ursprungszustand, d.h. noch nicht geprüft.



4.15 Viewer.java



Die Klasse Viewer.java ist sozusagen die Anfangsklasse der ganzen Software. Diese Klasse beinhaltet die main()-Funktion, von der aus das Programm gestartet wird. Hierzu wird zuerst das Look-and-Feel der Software gesetzt und anschließend die Klasse GUI.java aufgerufen. Ausgehend von der Klasse GUI.java werden dann alle in den vorhergegangenen Abschnitten beschriebenen Klassen und Funktionen aufgerufen und instanziiert.

4.16 export-library.ulp

Schon des Öfteren wurde während dieser Dokumentation von den sog. Geometrie-Libraries gesprochen, welche zentral auf dem Server abgelegt wurden und die Geometrie-Daten der einzelnen Bauteile in Form von Textdateien enthalten. Um diese Textdateien zu erstellen, ist das Skript export-library.ulp zuständig.

Ausgangspunkt hierfür ist das Layout-System Eagle, welches innerhalb der Firma verwendet wird.

4.16.1 Was ist ein ULP?

Die Layout-Software Eagle ist dazu in der Lage Programme ausführen, die in einer C-ähnlichen Programmiersprache geschrieben sind, der sog. Eagle-User-Language. Diese Programme liegen immer als lesbare Textdatei vor. ULPs sind äußerst flexible Werkzeuge wenn es darum geht, interne Daten aus Eagle zu bearbeiten oder für andere Zwecke aufzubereiten. (Quelle: www.cadsoft.de)

4.16.2 Die Funktionsweise des Skripts

Das Skript wandelt also, wie bereits erwähnt die Bauteile, welche innerhalb der Software Eagle für das Layout der aktuellen Leiterplatte verwendet werden in sog. Footprints um, welche als Text-Dateien erstellt werden.
Hierzu geht das Skript ein Bauteil Element für Element durch und prüft dabei aus welcher Geometrie dieses Element besteht. Für jedes gefundene Element, welches aus einer entsprechenden Geometrie besteht, wird dieses als Zeile in die Textdatei geschrieben. Eine Zeile besteht also zum einen aus der Bezeichnung der Geometrie, welche in unserem Fall folgende sind:

- ElementLin für Geraden
- ElementRec für Rechtecke
- ElementArc für Kreisbögen, Ellipsebögen und Kreise

Zum anderen aus den Daten, welche die Geometrie bilden, z.B. Anfangs- und Endpunkt.

Wenn das Skript jedes Element durchlaufen hat und jede Zeile in die Textdatei geschrieben hat erhalten wir also eine Textdatei, welche die kompletten geometrischen Daten eines Bauteils wiederspiegelt. Diese Textdateien bezeichnen wir als Geometrie-Libraries, welche zentral abgelegt werden und bsp. von dieser, aber auch von anderen Softwares wieder eingelesen werden können.

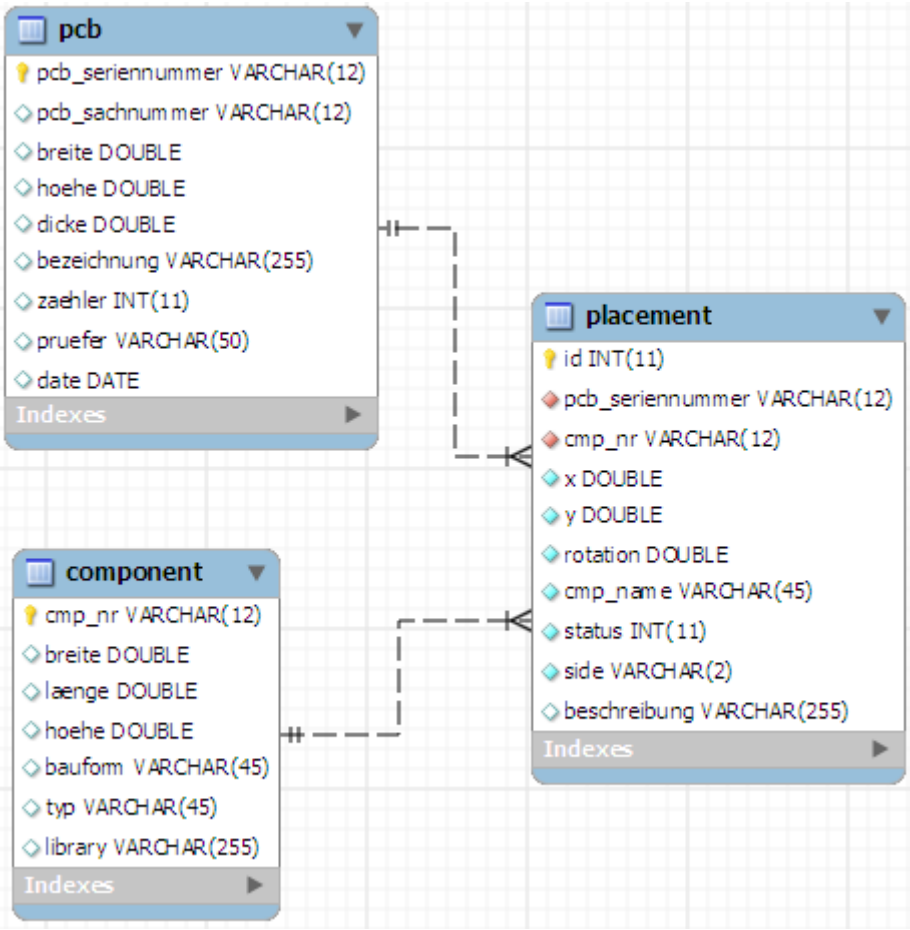
```
....
ElementLin;-89;-75;89;-75;4
ElementLin;89;-75;89;75;4
ElementRec;-110;43;-89;-43
....
```

Auszug aus einer Geometrie-Library

5. Die Datenbank

Im Abschnitt 4 wurde detailliert auf die verschiedenen Module und Funktionen der Software eingegangen. Dabei wurde mehrmals angesprochen, dass Daten gespeichert und zu späteren Zeitpunkten wieder geöffnet werden können. Dies geschieht innerhalb der Klasse GUI.java über die Methoden speichern() und öffnen(), welche die entstandenen Daten in die Datenbank schreiben bzw. wieder auslesen.

Auf diese Datenbank soll nun im aktuellen Abschnitt näher eingegangen werden.
Grundlage für die Tabellen, welche von dieser Software verwendet werden ist eine bereits bestehende SQL-Datenbank. Dort werden nahezu alle Daten, die während verschiedenen Prozessen innerhalb des Unternehmens vorkommen, abgelegt, wie z.B. Kundendaten, Auftragsdaten, Stammdaten, Qualitätsdaten.
Bereits vorhandene Datenbank, welche bereits als Grundlage eines Qualitätsmanagementsystems dient, musste nun also durch 3 Tabellen erweitert werden, um auch die anfallenden Daten dieser Prüfung nahtlos in das QMS einbinden zu können.



5.1 Die Tabelle pcb

In der Tabelle pcb werden zum einen generelle Daten über die Leiterplatte selbst abgelegt, zudem werden zusätzlich der Name des Prüfers und der aktuelle Zeitpunkt der Prüfung gespeichert. Dies wird über folgende Felder realisiert:

- **pcb_seriennummer:**
  Dieses Feld dient zur eindeutigen Identifikation der Leiterplatte und ist daher der PRIMARY KEY dieser Tabelle. Die Seriennummer ist für jede produzierte Leiterplatte eindeutig und wird ständig hochgezählt, d.h. es kann keine Seriennummer doppelt vorkommen.

- **pcb_sachnummer:**
  Die Sachnummer identifiziert die Art der Leiterplatte. Jede Leiterplatte der gleichen Art hat somit die gleiche Sachnummer.

- **breite, hoehe, dicke:**
  Diese Felder beschreiben die Abmessungen der Leiterplatte.

- **bezeichnung:**
  Durch dieses Feld kann die Klartextbeschreibung der Leiterplatte angegeben werden.

- **zaehler:**
  Über dieses Feld kann der aktuelle Fortschritt der Prüfung beim Speichern gespeichert werden. Sollte diese Prüfung zu einem späteren Zeitpunkt wieder geöffnet werden, so kann durch diesen Zähler festgestellt werden, an welcher Stelle die Prüfung beendet wurde.

- **pruefer:**
  In diesem Feld wird der Name des Prüfers gespeichert

- **date:**
  Dort wird der aktuelle Timestamp beim Speichern der Prüfung gespeichert. So kann direkt nachvollzogen werden, wann die Prüfung stattfand.

5.2 Die Tabelle component

Diese Tabelle speichert alle Daten, welche Auskunft über die Abmessungen bzw. die Geometrien der Bauteile geben. Sie beinhaltet folgende Felder:

- **cmp_nr:**
  Die eindeutige Seriennummer des Bauteils. Da diese Seriennummer für verschiedene Bauteile unterschiedlich ist, dient dieses Feld auch als PRIMARY KEY der Tabelle.

- **breite, laenge, hoehe:**
  Diese Felder dienen für die Abmessungen der Bauteile.

- **bauform:**
  In diesem Feld wird die Bauform des Bauteils gespeichert, z.B. 0805, PLCC, QFP, BGA etc.

- **typ:**
  Dort wird der eigentliche Typ des Bauteils gespeichert, z.B. Widerstand, Kondensator etc.

- **library:**
  In diesem Feld wird der Pfad zur Geometrie-Library gespeichert, welche zentral auf dem Server abgelegt ist.

In dieser Tabelle wird schnell ersichtlich, dass Daten gespeichert werden, welche für das eigentliche QMS wenig Sinn machen. Grund dafür ist die einmalige Definition eines Bauteils und die mehrmalige Wiederverwendung in anderen Prozessen der Firma.
So kann z.B. ein Bauteil, dessen Daten in aktueller Software definiert wurde, durch diese Datenbank bsp. in der Entwicklung aber auch innerhalb der Produktionsmaschinen wieder verwendet werden.

5.3 Die Tabelle Placement

Diese Tabelle dient letztendlich zur Speicherung aller Daten, welche die Bauteile in Verbindung mit der Leiterplatte haben, wie z.B. Position, Rotation etc. Realisiert wird dies über folgende Felder:

- **id:**
  Jedes Bauteil erhält eine eindeutige ID, daher dient dieses Feld als PRIMARY KEY.

- **pcb_seriennummer:**
  Dieses Feld speichert die eindeutige Seriennummer der Leiterplatte und dient als FOREIGN KEY zur Tabelle pcb. Wir verknüpfen hier also aktuelle Tabelle mit der Tabelle pcb um darzustellen, welche Bauteile auf welcher Leiterplatte vorkommen.

- **cmp_nr:**
  In diesem Feld wird die eindeutige Seriennummer des Bauteils gespeichert, es dient daher als FOREIGN KEY zur Tabelle component. Auch hier verknüpfen wir aktuelle Tabelle, dieses Mal mit der Tabelle component. Dadurch können wir eine Beziehung vom aktuellen Bauteil zu dessen Geometrie herstellen.

- **x,y:**
  Diese 2 Felder speichern die Position des Bauteils auf der Leiterplatte in Form von x- und y-Koordinaten.

- **rotation:**
  Wie der Name schon sagt, wird in diesem Feld die Rotation des Bauteils auf der Leiterplatte gespeichert.

- **cmp_name:**
  Hier wird der Name des Bauteils auf der Leiterplatte gespeichert. Dieser Name dient lediglich zur besseren Übersicht, so werden verschiedene Widerstände bsp. als R1 – Rx bezeichnet.

- **status:**
  In diesem Feld wird der Status der Prüfung beim Stand der Speicherung abgelegt. Bei einem erneuten Einlesen der Daten durch die öffnen()-Funktion können diese Stadien wiederhergestellt werden.

- **side:**
  Dieses Feld beschreibt, auf welcher Seite der Leiterplatte (Top oder Bottom) sich das Bauteil befindet.

- **beschreibung:**
  In diesem Feld wird eine Fehlerbeschreibung des Bauteils gespeichert, sofern der Prüfer dieses als fehlerhaft identifiziert.

Wir sehen also, dass für alle Daten, welche während der Prüfung innerhalb der Software anfallen, entsprechende Tabellen vorhanden sind um diese dort abzulegen bzw. wieder auszulesen. Durch intelligente Verknüpfung ist dies ohne weitere Probleme möglich.
Wir sind dadurch also in der Lage einen Stand der Prüfung zu speichern und später wieder zu öffnen. Zudem entfällt durch diese Tabellen das zu Beginn der Dokumentation angesprochene Problem, in welchem alle Daten der Prüfung ursprünglich manuell in Dokumente eingetragen wurden. Diese mussten in der Historie danach manuell von einem weiteren Mitarbeiter in das QMS implementiert werden, was durch diese Erweiterung der Datenbank und die automatische Speicherung der Daten durch die Software entfällt.

6. Probleme

Wie in nahezu jeder anderen Softwareentwicklung auch, traten natürlich auch in diesem Projekt Probleme auf, welche in diesem Abschnitt kurz beschrieben werden sollen:

- **Speicherung der Objekte:**
  Schon in einem sehr frühen Stadium des Projekts wurde schnell klar, das innerhalb der Software einige Daten auftreten, welche in geeigneter Weise zentral gespeichert werden sollten, um von jedem Punkt der Software Zugriff darauf zu haben. Was anfangs mit Hilfe von einfachen Arrays realisiert wurde, stellte sich schnell als nicht geeignet heraus.
  Nimmt man als Beispiel die Zahl der Bauteile, welche für jede Art einer Leiterplatte stark variiert, so stellt man schnell fest, dass schon im Vorfeld eine feste Größe des Arrays definiert werden müsste, welche groß genug für alle Arten von Leiterplatten und den damit variierenden Zahlen an Bauteilen ist. Die Wahl musste also auf eine Art dynamisches Array fallen. Die Wahl fiel hierfür auf eine ArrayList, jedes Exemplar der Klasse ArrayList vertritt ein Array mit variabler Länge. Der Zugriff auf die Elemente erfolgt über Indizes. Da in einer ArrayList jedes Exemplar einer von Object abgeleiteten Klasse Platz findet, ist eine ArrayList nicht auf bestimmte Datentypen fixiert, doch Generics spezifizieren diese Typen genauer.

- **Plattenzugriff durch die Geometrie-Libraries:**
  Anfangs wurde bei jedem Neuzeichnen des Layouts für jedes Bauteil direkt auf die Geometrie-Library auf der Platte zugegriffen und dieses zeilenweise eingelesen. Schnell stellte sich jedoch heraus, dass sowohl die ständigen Plattenzugriffe, als auch der ständige Traffic über das Netzwerk zu einem sehr großen Overhead durch die Software führte. Daher wurden die Geometry-Libraries einmalig am Anfang der Software eingelesen, vorhandene Daten wurden danach in Objekten gespeichert, welche zentral abgelegt waren. So reduzierten wir die Plattenzugriffe bzw. den Netztraffic auf eine einmalige Angelegenheit und erhöhten dadurch die Performance der Software.

- **Generelle Infrastruktur innerhalb des Unternehmens:**
  Um überhaupt die Anforderungen des Projektes festzulegen, war es von großer Bedeutung, sich im Vorfeld ein Bild über das Unternehmen zu machen.
  Was anfangs relativ simpel wirkte wurde von Zeit zu Zeit sehr komplex. So musste z.B. erst festgestellt werden, wie sich die IT-Infrastruktur des Unternehmens zusammensetzt bzw. aus welchen IT-Komponenten diese Struktur besteht. Zudem musste mit erfahrenen Mitarbeitern durchgesprochen werden, wie ein idealer Prüfprozess aussieht, bzw. aussehen sollte. Durch die ständig neuen Anforderungen die dadurch Tag für Tag entstanden sind, musste die Software des Öfteren von Grund auf neu konzipiert werden, was das Thema der richtigen Planung im Vorfeld einer Softwareentwicklung für die Zukunft nochmals unterstützt.

7. Die Zukunft

Bereits heute ist diese Software schon an einzelnen Arbeitsplätzen im Einsatz und wird dort auf Herz und Nieren geprüft. Die ersten Kommentare waren durchweg positiv.
Trotzdem wurden schon während der Entwicklung der Software zukünftige Erweiterungen geplant, welche nun nachfolgend nochmals kurz beschrieben werden sollen:

- **Definition der Geometrie-Daten innerhalb der Software:**
  Bislang werden Bauteile innerhalb der Software Eagle erstellt bzw. gezeichnet und danach über das Skript export-library.ulp exportiert und von dieser Software wiederum eingelesen. In Zukunft soll dieser Step entfallen, so dass diese Bauteile direkt innerhalb der Software gezeichnet bzw. erstellt werden können.

- **Anbindung der Software an Schrittmotorensteuerung:**
  Momentan trägt die Software zwar zu einer Automatisierung der Prüfung bei, diese sollte jedoch in Zukunft vollautomatisch ablaufen. So wird in Zukunft zusätzlich zur Software eine Schrittmotorensteuerung entwickelt, welche durch diese Software angesteuert wird. Mit Hilfe dieser Schrittmotoren, an welchen eine hochauflösende Kamera angebracht ist, kann jedes Bauteil automatisch angefahren werden und über eine Bild-Analyse-Software mit einer Muster-Baugruppe verglichen werden.
  Dadurch läuft die Prüfung letztendlich voll automatisiert ab.

- **Erweiterung der Software auf THT:**
  Bislang werden innerhalb dieser Software lediglich Bauteile beachtet, welche innerhalb der SMD-Bestücklinie verwendet werden. Bauteile die in der THT-Technologie auf der Leiterplatte verwendet werden, müssen momentan noch in der traditionellen Art geprüft werden. Grund dafür ist das Bestückprogramm, welches verwendet und eingelesen wird. Dieses ist optimiert für die voll automatische SMD-Bestücklinie und beinhaltet daher keine THT-Bauteile. In Zukunft soll diese Software also dazu in der Lage sein, sowohl die SMD- als auch die THT-Bauteile zu verwenden um einen weiteren Prüfschritt einzusparen.

Am Ende dieser Dokumentation sei noch zu erwähnen, dass dieses Projekt nicht nur die Erfahrung im Bereich der Softwareentwicklung enorm erhöht hat. Vielmehr hat die enge Zusammenarbeit mit einem Unternehmen einen guten Einblick in die zukünftige Arbeitswelt gewährleistet. Durch viele Meetings mit Mitarbeitern war schnell klar, dass die zukünftige Arbeitswelt nicht mehr aus einzelnen Wissensträgern besteht, sondern vielmehr aus einem Wissenskollektiv, welches im Team an einer Lösung arbeitet.
Des Weiteren konnte ich durch dieses Projekt lernen und erfahren, was es bedeutet, ein Projekt im Vorfeld richtig durchzuplanen, um nicht während der Entwicklung an Grenzen zu stoßen.
Mit der Hoffnung, dass dieses Projekt in Zukunft die Prozesse des Unternehmens unterstützen kann, verbleibe ich in engem Kontakt und hoffe zusammen mit den Kollegen dieses Unternehmens, die Software in Zukunft kontinuierlich weiterzuentwickeln.

8. Anhang

8.1 Arc.java

```
/**
 * Klasse für die Objekte und Methoden der Klasse (Geometrie) Arc.
 *
 * Die einzelnen Objekte dieser Klasse werden innerhalb der Klasse
 * GUI_comp in Form von Geometrie-Libraries eingelesen und instantiiert.
 * Arc repräsentiert innerhalb dieser Geometrien Kreise bzw. Kreisbögen
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

public class Arc {

	/* Deklaration der Klassenvariablen */

	private double x1; // Anfangspunkt des Kreisbogens
	private double y1; // Endpunkt des Kreisbogens
	private double durchmesser1; // Durchmesser des Kreisbogens
	private double durchmesser2;
	private double angle1; // Anfangswinkel des Kreisbogens
	private double angle2; // Endwinkel des Kreisbogens
	private double width; // Linienstärke des Kreisbogens

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Koordinaten
	 *            x1 und y1, Durchmesser 1 und 2, Winkel 1 und 2, Breite
	 */

	public Arc(double x1, double y1, double durchmesser1, double
durchmesser2,
			double angle1, double angle2, double width) {
		this.x1 = x1;
		this.y1 = y1;
		this.durchmesser1 = durchmesser1;
		this.durchmesser2 = durchmesser2;
		this.angle1 = angle1;
		this.angle2 = angle2;
		this.width = width;
	}

	/**
	 * Deklaration der set- bzw. get-Methoden
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
```

```
	 * @param Übergabewerte
	 *            für die jeweilige Klassenvariablen
	 * @return jeweilige Klassenvariablen
	 */

	public void setX1(double x1) {
		this.x1 = x1;
	}

	public double getX1() {
		return x1;
	}

	public void setY1(double y1) {
		this.y1 = y1;
	}

	public double getY1() {
		return y1;
	}

	public void setDurchmesser1(double durchmesser1) {
		this.durchmesser1 = durchmesser1;
	}

	public double getDurchmesser1() {
		return durchmesser1;
	}

	public void setDurchmesser2(double durchmesser2) {
		this.durchmesser2 = durchmesser2;
	}

	public double getDurchmesser2() {
		return durchmesser2;
	}

	public void setAngle1(double angle1) {
		this.angle1 = angle1;
	}

	public double getAngle1() {
		return angle1;
	}

	public void setAngle2(double angle2) {
		this.angle2 = angle2;
	}

	public double getAngle2() {
		return angle2;
	}

	public void setWidth(double width) {
		this.width = width;
	}

	public double getWidth() {
```

```
            return width;
      }
}
```

8.2 Comp.java

```
/**
 * Klasse für die verschiedenen Bauteile, welche
 * innerhalb der Leiterplatte verwendet werden.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Shape;
import java.awt.geom.AffineTransform;
import java.awt.geom.Arc2D;
import java.awt.geom.GeneralPath;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;

import javax.swing.JPanel;

public class Comp extends JPanel {

      /* Deklaration der Klassenvariablen */

      private static final long serialVersionUID = 1L;

      int status; // Status, während des Prüfvorganges
      private String cmp_nr; // Eindeutige Sachnummer des Bauteils
      private double breite; // Breite des Bauteils
      private double laenge; // Länge des Bauteils
      private double hoehe; // Höhe des Bauteils
      private String bauform; // Bauform des Bauteils (0805 etc.)
      private String typ; // Typ des Bauteils (Widerstand etc.)
      private String library; // Pfad zur Geometrie-Library

      ArrayList<Rect> aRect = new ArrayList<Rect>(); // ArrayList, um
Objekte

      // (Geometrien) der Form

      // Rect zu speichern

      // (Rechtecke)

      ArrayList<Arc> aArc = new ArrayList<Arc>(); // ArrayList, um Objekte
                                                                        //
```

```
(Geometrien) der Form Arc zu
                                                                        //
speichern (Kreise)

      ArrayList<Line> aLine = new ArrayList<Line>(); // ArrayList, um
Objekte

      // (Geometrien) der Form

      // Line zu speichern

      // (Linien)

      /**
       * Deklaration des Konstruktors
       *
       * @author Patrick Olma
       * @version 30.06.2011
       * @param Sachnummer
       *            , breite, laenge, hoehe, bauform, typ, library,
ArrayListen
       *            für die jeweiligen Geometrien
       */

      public Comp(String cmp_nr, double breite, double laenge, double
hoehe,
                  String bauform, String typ, String library,
ArrayList<Rect> aRect,
                  ArrayList<Arc> aArc, ArrayList<Line> aLine) {
            this.cmp_nr = cmp_nr;
            this.breite = breite;
            this.laenge = laenge;
            this.hoehe = hoehe;
            this.bauform = bauform;
            this.typ = typ;
            this.library = library;
            this.aRect = aRect;
            this.aArc = aArc;
            this.aLine = aLine;
      }

      /**
       * Methode draw() um die einzelnen Bauteile anhand ihrer Koordinaten
und
       * Geometrien auf die GUI zu zeichnen
       *
       * @author Patrick Olma
       * @version 30.06.2011
       * @param Objekte
       *            der Klasse Graphics, x-Koordinate, y-Koordinate,
       *            Vergrößerungsfaktor, Sachnummer, Rotation des Bauteils,
Name
       *            und Prüfstatus.
       *
       */

      public void draw(Graphics g, double x, double y, int faktor, String
nr,
```

```
                  double rotation, String name, int status) {

            int j = 0; // Variable um abzufragen, ob GeneralPath schon
Punkt
                              // beinhaltet

            Graphics2D g2 = (Graphics2D) g; // Cast des Objektes g nach
Grpahics2D

            /* Deklaration der Affinen Transformationen. */

            AffineTransform at1 = new AffineTransform(); // Verschiebung
            AffineTransform at2 = new AffineTransform(); // Skalierung
            AffineTransform at3 = new AffineTransform(); // Rotation

            /* Deklaration der drei Shapes für die drei Geometrien (Arc,
Line, Rect. */

            Shape s1 = null; // Line
            Shape s2 = null; // Rect
            Shape s3 = null; // Arc

            at1.setToTranslation(x, y); // Verschiebung nach (x|y)
            at2.setToScale(faktor, faktor); // Vergrößerung um faktor
            at3.setToRotation(-rotation * Math.PI / 180, x, y); // Drehung
um

                  // rotation
            at1.concatenate(at2); // Verknüpfung der Transformationen 1 und
2
            at3.concatenate(at1); // Anschließende Verknüpfung 3 mit 1

            /*
             * Deklaration des Generalpaths, um die Anfangs- und Endpunkte
der
             * Line-Geometrien zu speichern.
             */

            GeneralPath gp = new GeneralPath();

            this.status = status;

            /* Anfangs- und Endpunkte der Geometrie Line in GeneralPath
speichern */

            if (aLine.size() != 0) { // Abfrage, ob ArrayList Elemente
enthält
                  for (int iLine = 0; iLine < aLine.size(); ++iLine) { //
Durchlauf

                              // der

                              // kompletten

                              // Liste
                        if (j > 0) { // Falls j>0, enthält GeneralPath
schon einen Punkt
                              gp.lineTo(aLine.get(iLine).getX1(),
```

```
aLine.get(iLine)
                                        .getY1()); // Füge Line von
Ausganspunkt zu
                                                   // übergebenem
Punkt hinzu.
                        }

                        else {
                              gp.moveTo(aLine.get(iLine).getX1(),
aLine.get(iLine)
                                        .getY1()); // Ausgangspunkt der
Linie hinzufügen
                        }
                        j++; // j erhöhen, d.h. Ausgangspunkt bereits
vorhanden!
                  }
            }

            gp.setWindingRule(GeneralPath.WIND_NON_ZERO); // Inneren
Bereich mit

            // ausfüllen
            gp.closePath(); // Pfad schließen
            s1 = at3.createTransformedShape(gp); // GeneralPath inkl.
                                                                  //
Transformationen in Shape
                                                                  //
zeichnen

            if (aArc.size() != 0) { // Abfrage, ob ArrayList Elemente
enthält
                  for (int iArc = 0; iArc < aArc.size(); ++iArc) { //
Durchlauf der

                        // kompletten

                        // Liste.
                        Arc2D.Double arc = new Arc2D.Double(
                                    // Erstellen eines neuen Kreisbogens.
                                    (aArc.get(iArc).getX1() -
(aArc.get(iArc)
                                                .getDurchmesser1() / 2)),
                                    (aArc.get(iArc).getY1() -
(aArc.get(iArc)
                                                .getDurchmesser1() / 2)),
aArc.get(iArc)
                                                .getDurchmesser1(),
aArc.get(iArc)
                                                .getDurchmesser1(),
aArc.get(iArc).getAngle1(),
                                    aArc.get(iArc).getAngle2(),
Arc2D.OPEN);
                        s3 = at3.createTransformedShape(arc); // Kreisbogen
inkl.

            // Transformationen in

            // Shape zeichnen.
```

```
                }
            }

            if (aRect.size() != 0) { // Abfrage, ob ArrayList Elemente
enthält
                for (int iRect = 0; iRect < aRect.size(); ++iRect) { //
Durchlauf

                                // der

                                // kompletten

                                // Liste
                            Rectangle2D.Double rect = new
Rectangle2D.Double(aRect.get( // Erstellen

                                                // eines

                                                // neuen

                                                // Recheckes
                                        iRect).getX1(),
(aRect.get(iRect).getY1() + (aRect.get(
                                        iRect).getY2() -
aRect.get(iRect).getY1())), (aRect
                                        .get(iRect).getX2() -
aRect.get(iRect).getX1()),
                                        -(aRect.get(iRect).getY2() -
aRect.get(iRect).getY1()));
                            s2 = at3.createTransformedShape(rect); // Rechteck
inkl.

            // Transformationen in

            // Shape zeichnen.
                            g2.fill(s2); // Shape füllen
                }
            }

            /*
             * Abfragen des Status und dementsprechendes einfärben des
Bauteils,
             * bzw. füllen des Shapes status = 0 --> Bauteil noch nicht
geprüft -->
             * keine Füllung, status = 1 --> aktuell zu prüfendes Bauteil -
-> Gelb,
             * status = 2 --> Bauteil korrekt --> Grün, status = 3 -->
Bauteil
             * inkorrekt --> Rot
             *
             * Shape s3 (Kreisbogen) wird nachdem es geprüft wurde immer
Schwarz
             * eingefärbt, da es die Polung eines Bauteils wiedergibt.
             *
             * Shape s2 wurde bereits gefüllt, da es wie der Kreisbogen die
Polung
             * eines Bauteils wiedergibt und daher grundsätzlich schwarz
eingefärbt
```

```
             * wird.
             */

            if (status == 0) {
                  if (aLine.size() != 0) {
                        g2.draw(s1);
                  }
                  if (aArc.size() != 0) {
                        g2.draw(s3);
                  }
            } else if (status == 1) {
                  if (aLine.size() != 0) {
                        g2.setPaint(Color.YELLOW);
                        g2.fill(s1);
                        g2.setColor(Color.BLACK);
                  }
                  if (aArc.size() != 0) {
                        g2.fill(s3);
                  }
            } else if (status == 2) {
                  if (aLine.size() != 0) {
                        g2.setPaint(Color.GREEN);
                        g2.fill(s1);
                        g2.setColor(Color.BLACK);
                  }
                  if (aArc.size() != 0) {
                        g2.fill(s3);
                  }
            } else if (status == 3) {
                  if (aLine.size() != 0) {
                        g2.setPaint(Color.RED);
                        g2.fill(s1);
                        g2.setColor(Color.BLACK);
                  }
                  if (aArc.size() != 0) {
                        g2.fill(s3);
                  }
            }
      }

      /**
       * Deklaration der set- bzw. get-Methoden
       *
       * @author Patrick Olma
       * @version 30.06.2011
       * @param Übergabewerte
       *            für die jeweilige Klassenvariablen
       * @return jeweilige Klassenvariablen
       */

      public void setCmp_nr(String cmp_nr) {
            this.cmp_nr = cmp_nr;
      }

      public String getCmp_nr() {
            return cmp_nr;
      }
```

```
	public void setBreite(double breite) {
		this.breite = breite;
	}

	public double getBreite() {
		return breite;
	}

	public void setLaenge(double laenge) {
		this.laenge = laenge;
	}

	public double getLaenge() {
		return laenge;
	}

	public void setLibrary(String library) {
		this.library = library;
	}

	public String getLibrary() {
		return library;
	}

	public void setHoehe(double hoehe) {
		this.hoehe = hoehe;
	}

	public double getHoehe() {
		return hoehe;
	}

	public void setBauform(String bauform) {
		this.bauform = bauform;
	}

	public String getBauform() {
		return bauform;
	}

	public void setTyp(String typ) {
		this.typ = typ;
	}

	public String getTyp() {
		return typ;
	}
}
```

8.3 Datei.java

```
/**
 * Klasse, um Textdateien über den Öffnen-Dialog zu selektieren.
 *
 * Die selektierten Textdatein werden einerseits für die Bestückdaten
 * und andererseits für die Geometrie-Datein verwendet, welche innerhalb
 * des Programmes zeilenweise eingelesen werden.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.io.File;

import javax.swing.JFileChooser;

public class Datei {

	/* Deklaration der Klassenvariablen */

	private String verz; // Ausgewählte Datei

	/**
	 * Deklaration der get-Methode
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @return ausgewählte Datei
	 */

	public String getVerz() {
		return verz;
	}

	/**
	 * Methode oeffnen() um die gewünschte Datei zu selektieren
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 */

	void oeffnen() {

		/* Neues objekt chooser der Klasse JFileChooser instantiieren
*/
		final JFileChooser chooser = new JFileChooser("Verzeichnis
wählen");
		/* Typ: Öffnen-Dialog */
		chooser.setDialogType(JFileChooser.OPEN_DIALOG);
		/* Selektions-Modus: Dateien innerhalb Verzeichnisse */

	chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
```

```
            /*
             * Neues Objekt file der Klasse File instantiieren.
Ausgangsverzeichnis
             * = Home-Verzeichnis
             */
            final File file = new File("/home");
            /* Aktuelles Verzeichnis = Home-Verzeichnis */
            chooser.setCurrentDirectory(file);
            /* Aktion-Listener hinzufügen */
            chooser.addPropertyChangeListener(new PropertyChangeListener()
{
                  public void propertyChange(PropertyChangeEvent e) {
                        if (e.getPropertyName().equals(

      JFileChooser.SELECTED_FILE_CHANGED_PROPERTY)
                                    || e.getPropertyName().equals(

      JFileChooser.DIRECTORY_CHANGED_PROPERTY)) {
                        }
                  }
            });

            /* Öffnen-Dialog sichtbar machen */
            chooser.setVisible(true);
            final int result = chooser.showOpenDialog(null);

            /* Abfrage, ob Button "OK" gedrückt wurde */
            if (result == JFileChooser.APPROVE_OPTION) {
                  /* Aktuell selektierte Datei in Variable inputVerzFile
speichern */
                  File inputVerzFile = chooser.getSelectedFile();
                  /*
                   * getPatch() gibt aktuellen Pfad als String zurück,
welcher in
                   * Variable inputVerzStr gespeichert wird
                   */
                  String inputVerzStr = inputVerzFile.getPath();
                  /* Pfad in Klassenvariable speichern */
                  verz = inputVerzStr;
            }
            /* Öffnen-Dialog unsichtbar machen */
            chooser.setVisible(false);
      }

}
```

8.4 Daten.java

```
/**
 * Klasse, um die verschiedenen Objekte in ArrayListen zu speichern.
 *
 * Dafür werden für jedes Objekt entsprechende ArrayListen der jeweiligen
Klassen
 * erstellt.
 * Beim neuen anlegen bzw. beim öffnen einer Leiterplatte, werden die
einzelnen
 * Objekte instantiiert und anschließend in diese ArrayListen gespeichert.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.util.ArrayList;

public class Daten {
	/* Objekte der Klasse PCB, welche die Leiterplatte repräsentieren */
	private static ArrayList<PCB> aPCB = new ArrayList<PCB>();
	/*
	 * Objekte der Klasse Placement, für jedes verwendete Bauteil auf der
	 * Leiterplatte
	 */
	private static ArrayList<Placement> aPlace_top = new
ArrayList<Placement>();
	private static ArrayList<Placement> aPlace_bottom = new
ArrayList<Placement>();
	/*
	 * Objekte der Klasse Comp, welche die einzelnen Geometrien der
Bauteile
	 * repräsentieren
	 */
	private static ArrayList<Comp> aComp_top = new ArrayList<Comp>();
	private static ArrayList<Comp> aComp_bottom = new ArrayList<Comp>();

	/**
	 * Deklaration der set- bzw. get-Methoden
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Übergabewerte
	 *            für die jeweilige Klassenvariablen
	 * @return jeweilige Klassenvariablen
	 */

	public static ArrayList<PCB> getAPCB() {
		return aPCB;
	}

	public static ArrayList<Placement> getAPlace_top() {
		return aPlace_top;
	}
```

```
	public void setAPCB(PCB input) {
		aPCB.add(input);
	}

	public static ArrayList<Placement> getAPlace_bottom() {
		return aPlace_bottom;
	}

	public void setAPlace_top(Placement input) {
		aPlace_top.add(input);
	}

	public void setAPlace_bottom(Placement input) {
		aPlace_bottom.add(input);
	}

	public static void setaComp_top(Comp aComp_top) {
		Daten.aComp_top.add(aComp_top);
	}

	public static ArrayList<Comp> getaComp_top() {
		return aComp_top;
	}

	public static void setaComp_bottom(Comp aComp_bottom) {
		Daten.aComp_bottom.add(aComp_bottom);
	}

	public static ArrayList<Comp> getaComp_bottom() {
		return aComp_bottom;
	}
}
```

8.5 GUI_comp_define.java

```
/**
 * Klasse für die GUI, mit welcher die einzelnen Bauteile definiert
werden.
 *
 * Definition bezieht sich hierbei auf die Abmessungen, die Sachnummern
und die
 * Angabe der Geometrie-Library.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.awt.Frame;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.WindowConstants;

public class GUI_comp_define extends JDialog {

	/* Deklaration der Klassenvariablen */

	private static final long serialVersionUID = 1L;

	public static int row;

	/* Deklaration der GUI-Komponenten */

	JLabel lBreite = new JLabel("Breite:", JLabel.LEFT);
	JLabel lLaenge = new JLabel("Laenge:", JLabel.LEFT);
	JLabel lHoehe = new JLabel("Hoehe:", JLabel.LEFT);
	JLabel lBauform = new JLabel("Bauform:", JLabel.LEFT);
	JLabel lTyp = new JLabel("Typ:", JLabel.LEFT);
	JLabel lLibrary = new JLabel("Bitte Library angeben:", JLabel.LEFT);
	JTextField tBreite = new JTextField("", JTextField.LEFT);
	JTextField tLaenge = new JTextField("", JTextField.LEFT);
	JTextField tHoehe = new JTextField("", JTextField.LEFT);
	JTextField tBauform = new JTextField("", JTextField.LEFT);
	JTextField tTyp = new JTextField("", JTextField.LEFT);
	JTextField tLibrary = new JTextField("", JTextField.LEFT);
	JButton bOpen = new JButton("...");
	JButton bOK = new JButton("OK");

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
```

```
        * @param Besitzer
        *            des Frames, Titel, Modal-Typ (True od. False)
        */

       GUI_comp_define(Frame owner, String title, boolean modal) {

             super(owner, title, modal);

             /* Definition der Positionen und der Abmessungen der GUI-
Komponenten */
             lBreite.setBounds(10, 10, 60, 20);
             lLaenge.setBounds(80, 10, 60, 20);
             lHoehe.setBounds(150, 10, 60, 20);
             tBreite.setBounds(10, 40, 60, 20);
             tLaenge.setBounds(80, 40, 60, 20);
             tHoehe.setBounds(150, 40, 60, 20);
             lBauform.setBounds(10, 70, 60, 20);
             lTyp.setBounds(80, 70, 150, 20);
             tBauform.setBounds(10, 100, 60, 20);
             tTyp.setBounds(80, 100, 150, 20);
             lLibrary.setBounds(10, 130, 200, 20);
             tLibrary.setBounds(10, 160, 200, 20);
             bOpen.setBounds(220, 160, 60, 20);
             bOK.setBounds(10, 190, 60, 20);

             /* Layout-Manager zurücksetzen */
             this.getContentPane().setLayout(null);

             /* GUI-Komponenten der ContentPane hinzufügen */
             this.getContentPane().add(lBreite);
             this.getContentPane().add(lLaenge);
             this.getContentPane().add(lHoehe);
             this.getContentPane().add(tBreite);
             this.getContentPane().add(tLaenge);
             this.getContentPane().add(tHoehe);
             this.getContentPane().add(lBauform);
             this.getContentPane().add(lTyp);
             this.getContentPane().add(tBauform);
             this.getContentPane().add(tTyp);
             this.getContentPane().add(lLibrary);
             this.getContentPane().add(tLibrary);
             this.getContentPane().add(bOpen);
             this.getContentPane().add(bOK);

             /* Abfrage ob Bauteile der Top-Seite bearbeitet werden */
             if (GUI_comp.tTop.getSelectedIndex() == 0) {
                   /*
                    * Textfelder mit aktuellen Werten der Tabelle in Klasse
GUI_comp
                    * füllen die Zeile (row) wird dabei über die Klasse
GUI_comp beim
                    * öffnen dieses Dialogs übergeben
                    */
                   tBreite.setText(GUI_comp.top[row][1]);
                   tLaenge.setText(GUI_comp.top[row][2]);
                   tHoehe.setText(GUI_comp.top[row][3]);
                   tBauform.setText(GUI_comp.top[row][4]);
                   tTyp.setText(GUI_comp.top[row][5]);
```

```
                tLibrary.setText(GUI_comp.top[row][6]);
          }
          /* Abfrage ob Bauteile der Bottom-Seite bearbeitet werden */
          if (GUI_comp.tTop.getSelectedIndex() == 1) {
                /* Gleiche Prozedur, als bei Top-Seite */
                tBreite.setText(GUI_comp.bottom[row][1]);
                tLaenge.setText(GUI_comp.bottom[row][2]);
                tHoehe.setText(GUI_comp.bottom[row][3]);
                tBauform.setText(GUI_comp.bottom[row][4]);
                tTyp.setText(GUI_comp.bottom[row][5]);
                tLibrary.setText(GUI_comp.bottom[row][6]);
          }

          /**
           * Aktion-Listener des Buttons "Öffnen"
           *
           * Hiermit werden die Geometrie-Libraries ausgewählt
           *
           * @author Patrick Olma
           * @version 30.06.2011
           *
           */
          bOpen.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                      /* Objekt der Klasse Datei instantiieren */
                      Datei lib = new Datei();
                      /* Methode oeffnen() aufrufen, um Library zu
selektieren */
                      lib.oeffnen();
                      /* Pfad dem Textfeld zuweisen */
                      tLibrary.setText(lib.getVerz());
                }
          });

          /**
           * Aktion-Listener des Buttons "OK"
           *
           * Hiermit werden die Änderungen der Bauteil-Daten zugewiesen
und in die
           * dementsprechenden Tabellen der Klasse GUI_comp geschrieben.
Diese
           * Tabellen speichern die einzelnen Daten.
           *
           * @author Patrick Olma
           * @version 30.06.2011
           *
           */
          bOK.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                      /*
                       * Prüfung, ob alle Daten angegeben wurden. Falls
nicht,
                       * erscheint ein Informations-Dialog.
                       */
                      if (tBreite.getText().equals("")) {
                            JOptionPane.showMessageDialog(null,
                                        "Bitte alle Daten angeben",
"Daten unvollständig",
```

```
                                        JOptionPane.INFORMATION_MESSAGE);

                    } else if (tLaenge.getText().equals("")) {
                        JOptionPane.showMessageDialog(null,
                                "Bitte alle Daten angeben",
"Daten unvollständig",
                                JOptionPane.INFORMATION_MESSAGE);
                    } else if (tHoehe.getText().equals("")) {
                        JOptionPane.showMessageDialog(null,
                                "Bitte alle Daten angeben",
"Daten unvollständig",
                                JOptionPane.INFORMATION_MESSAGE);
                    } else if (tBauform.getText().equals("")) {
                        JOptionPane.showMessageDialog(null,
                                "Bitte alle Daten angeben",
"Daten unvollständig",
                                JOptionPane.INFORMATION_MESSAGE);
                    } else if (tTyp.getText().equals("")) {
                        JOptionPane.showMessageDialog(null,
                                "Bitte alle Daten angeben",
"Daten unvollständig",
                                JOptionPane.INFORMATION_MESSAGE);
                    } else if (tLibrary.getText().equals("")) {
                        JOptionPane.showMessageDialog(null,
                                "Bitte alle Daten angeben",
"Daten unvollständig",
                                JOptionPane.INFORMATION_MESSAGE);
                    } else {
                        /* Abfrage, ob Bauteile der Top-Seite
bearbeitet werden */
                        if (GUI_comp.tTop.getSelectedIndex() == 0) {
                            /*
                             * Inhalte der Textfelder den
entsprechenen
                             * Tabellen-Feldern (Top) zuweisen
                             */
                            GUI_comp.top[row][1] =
tBreite.getText();
                            GUI_comp.top[row][2] =
tLaenge.getText();
                            GUI_comp.top[row][3] =
tHoehe.getText();
                            GUI_comp.top[row][4] =
tBauform.getText();
                            GUI_comp.top[row][5] = tTyp.getText();
                            GUI_comp.top[row][6] =
tLibrary.getText();
                        }

                        /* Abfrage, ob Bauteile der Bottom-Seite
bearbeitet werden */
                        if (GUI_comp.tTop.getSelectedIndex() == 1) {
                            /*
                             * Inhalte der Textfelder den
entsprechenen
                             * Tabellen-Feldern (Bottom) zuweisen
                             */
                            GUI_comp.bottom[row][1] =
```

```
tBreite.getText();
                                    GUI_comp.bottom[row][2] =
tLaenge.getText();
                                    GUI_comp.bottom[row][3] =
tHoehe.getText();
                                    GUI_comp.bottom[row][4] =
tBauform.getText();
                                    GUI_comp.bottom[row][5] =
tTyp.getText();
                                    GUI_comp.bottom[row][6] =
tLibrary.getText();
                                }
                                /* Dialog unsichtbar machen */
                                setVisible(false);
                            }
                        }
                });

                /* Packem */
                pack();
                /* Größe angeben */
                setSize(320, 320);
                /* Methode beim schließen des Dialogs */
                setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
                /* Position des Dialogs auf dem Bildschirm. Hier: Mitte des
Desktops */
                setLocation(
                        (Toolkit.getDefaultToolkit().getScreenSize().width
- getSize().width) / 2,
                        (Toolkit.getDefaultToolkit().getScreenSize().height
- getSize().height) / 2);
                /* Dialog sichtbar machen */
                setVisible(true);

        }
}
```

8.6 GUI_comp.java

```
/**
 * Klasse für die GUI, in welcher Tabellarisch die einzelnen Daten der
 * verschiedenen Bauteile angezeigt werden.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTabbedPane;
import javax.swing.JTable;
import javax.swing.WindowConstants;
import javax.swing.table.DefaultTableModel;

public class GUI_comp extends JDialog {

	/* Deklaration der Klassenvariablen */
	private static final long serialVersionUID = 1L;

	/* ArrayListen für die jeweiligen Geometrien */
	ArrayList<Rect> aRect;
	ArrayList<Arc> aArc;
	ArrayList<Line> aLine;

	/*
	 * ArrayListen für die Geometrie-Daten der Bauteile, welche bereits
mit den
	 * vorhandenen Daten gefüllt werden
	 */
	ArrayList<Comp> aComponent_top = Daten.getaComp_top();
	ArrayList<Comp> aComponent_bottom = Daten.getaComp_bottom();
```

```
	/*
	 * ArrayListen für die verschiedenen Bauteile auf der Leiterplatte,
welche
	 * bereits mit den vorhandenen Daten gefüllt werden
	 */
	ArrayList<Placement> aPlacement_top = Daten.getAPlace_top();
	ArrayList<Placement> aPlacement_bottom = Daten.getAPlace_bottom();

	/* JTabbedPane für die 2 Seiten der Leiterplatte (Top und Bottom */
	static JTabbedPane tTop = new JTabbedPane();

	/* 2 Panel, center für die Tabelle und south für die Buttons */
	JPanel center = new JPanel();
	JPanel south = new JPanel();

	/* Deklaration der GUI-Komponenten */
	JButton bDaten = new JButton("Daten einlesen");
	JButton bOK = new JButton("OK");

	/*
	 * Deklaration der Variablen row und column, um die einzelnen Felder
der
	 * Tabelle zu selektieren
	 */
	int row;
	int column;

	/*
	 * Deklaration der Arrays, um später zu prüfen, ob alle Felder der
Tabelle
	 * mit Inhalt gefüllt wurden
	 */
	int[] ready_top;
	int[] ready_bottom;

	/*
	 * Deklaration von 2 boolean Variablen, um später zu prüfen, ob alle
	 * Eingaben korrekt sind
	 */
	boolean ready;
	public static boolean comp_ready = false;

	/* Spalten der Tabelle anlegen */
	String spalten[] = { "CMP_Nr", "Breite", "Länge", "Höhe", "Bauform",
"Typ",
			"Library" };

	/* Datenmodelle der Tabelle anlegen */
	public static String top[][];
	public static String bottom[][];

	/**
	 * Deklaration der Funktion einlesen()
	 *
	 * Diese Funktion liest die angegebenen Geometrie-Libraries
zeilenweise ein,
	 * instantiiert die dementsprechenden Objekte der Geometrie-Klassen
```

```
(Arc,
	 * Line, Rect) und fügt anschließend diese Objekte den entsprechenden
	 * ArrayListen hinzu. Anschließend werden aus den angegebenen Daten
der
	 * Bauteile inkl. den eingelesenen Geometrie-Daten für jede Zeile der
	 * Tabelle Objekte der Klasse Comp instantiiert und diese
anschließend der
	 * globalen ArrayList hinzugefügt.
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Das
	 *            einzulesende 2-Dimensionale Array(col und row der
Tabelle),
	 *            die Tabelle selbst und die gewählte Seite
	 */
	public void einlesen(String[][] array, JTable table, String side) {

		/*
		 * Deklaration der Variablen, um abzufragen, welche Geometrie
innerhalb
		 * der Library einglesen werden soll
		 */
		String line;
		String geo;
		String eLine = "ElementLin";
		String eArc = "ElementArc";
		String eRect = "ElementRec";

		/*
		 * Deklaration der Variablen, welche für die Geometrien
verwendet werden
		 */
		int i, k;
		double lx1 = 0, lx2, ly1 = 0, ly2;
		double recx1 = 0, recx2 = 0, recy1 = 0, recy2 = 0;
		double ax1 = 0, ay1 = 0;
		double adurchmesser1 = 0, aangle1 = 0, adurchmesser2, aangle2 =
0, awidth = 0;

		/* Komplettes Array durchlaufen */
		for (int zaehler = 0; zaehler < array.length; zaehler++) {
			/* Neue ArrayList anlegen */
			aRect = new ArrayList<Rect>();
			aArc = new ArrayList<Arc>();
			aLine = new ArrayList<Line>();

			try {
				/* Gewünschte Textdatei öffnen */
				FileReader fread_Comp = new FileReader(
						(String) table.getValueAt(zaehler, 6));
				/* Textdatei in BufferedReader laden */
				BufferedReader in_Comp = new
BufferedReader(fread_Comp);

				/* Solange Zeile nicht leer ist */
				while ((line = in_Comp.readLine()) != null) {
```

```
                                if (line.indexOf(";") == -1) {
                                      break;
                                }
                                /* Variable i = Position des 1. ";"-Zeichens
*/
                                i = line.indexOf(";");

                                /* String geo = String von Position 0 bis zum
1. ";"-Zeichen */
                                geo = line.substring(0, i);

                                /* Abfrage ob geo == "ElementLin" ist */
                                if (geo.equals(eLine)) {
                                      /*
                                       * Verschiedene Werte, getrennt durch
";"-Zeichen
                                       * einlesen und Variablen zuweisen.
Werte müssen mit
                                       * Faktor "0.0254" multipliziert
werden, da
                                       * Geometriedaten in der Einheit mil
bestehen und in mm
                                       * umberechnet werden müssen
                                       */
                                      k = line.indexOf(";", i + 1);
                                      lx1 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;
                                      i = k;
                                      k = line.indexOf(";", i + 1);
                                      ly1 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;
                                      i = k;
                                      k = line.indexOf(";", i + 1);
                                      lx2 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;
                                      i = k;
                                      k = line.indexOf(";", i + 1);
                                      ly2 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;

                                      /* Linie in ArrayList hinzufügen */
                                      aLine.add(new Line(lx1, lx2, ly1,
ly2));

                                }
                                /* Abfrage ob geo == "ElementArc" ist */
                                else if (geo.equals(eArc)) {
                                      /*
                                       * Verschiedene Werte, getrennt durch
";"-Zeichen
                                       * einlesen und Variablen zuweisen
Werte müssen mit
                                       * Faktor "0.0254" multipliziert
werden, da
                                       * Geometriedaten in der Einheit mil
bestehen und in mm
                                       * umberechnet werden müssen
                                       */
```

```
                                    k = line.indexOf(";", i + 1);
                                    ax1 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;
                                    i = k;
                                    k = line.indexOf(";", i + 1);
                                    ay1 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;
                                    i = k;
                                    k = line.indexOf(";", i + 1);
                                    adurchmesser1 =
(Double.parseDouble(line.substring(
                                                i + 1, k))) * 0.0254;
                                    i = k;
                                    k = line.indexOf(";", i + 1);
                                    adurchmesser2 =
(Double.parseDouble(line.substring(
                                                i + 1, k))) * 0.0254;
                                    i = k;
                                    k = line.indexOf(";", i + 1);
                                    aangle1 =
Double.parseDouble(line.substring(i + 1, k));
                                    i = k;
                                    k = line.indexOf(";", i + 1);
                                    aangle2 =
Double.parseDouble(line.substring(i + 1, k));
                                    k = line.lastIndexOf(";");
                                    awidth =
Double.parseDouble(line.substring(k + 1));

                                    /* Kreisbogen in ArrayList hinzufügen
*/
                                    aArc.add(new Arc(ax1, ay1,
adurchmesser1,
                                                adurchmesser2, aangle1,
aangle2, awidth));

                                }
                                /* Abfrage ob geo == "ElementRec" ist */
                                else if (geo.equals(eRect)) {
                                    /*
                                     * Verschiedene Werte, getrennt durch
";"-Zeichen
                                     * einlesen und Variablen zuweisen.
Werte müssen mit
                                     * Faktor "0.0254" multipliziert
werden, da
                                     * Geometriedaten in der Einheit mil
bestehen und in mm
                                     * umberechnet werden müssen
                                     */
                                    k = line.indexOf(";", i + 1);
                                    recx1 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;
                                    i = k;
                                    k = line.indexOf(";", i + 1);
                                    recy1 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;
                                    i = k;
```

```
                                    k = line.indexOf(";", i + 1);
                                    recx2 =
(Double.parseDouble(line.substring(i + 1, k))) * 0.0254;
                                    i = k;
                                    k = line.lastIndexOf(";");
                                    recy2 =
(Double.parseDouble(line.substring(k + 1))) * 0.0254;

                                    /* Rechteck in ArrayList hinzufügen */
                                    aRect.add(new Rect(recx1, recx2, recy1,
recy2));
                              } else {
                                    break;
                              }
                        }
                  } catch (IOException e1) {
                        System.out.println("IO-Fehler!");
                  }
                  /*
                   * Neues Comp-Objekt mit Inhalt der Tabelle und der
eingelesenen
                   * Geometrie-Daten instantiieren
                   */
                  Comp component = new Comp((String)
table.getValueAt(zaehler, 0),
                              Double.parseDouble((String)
table.getValueAt(zaehler, 1)),
                              Double.parseDouble((String)
table.getValueAt(zaehler, 2)),
                              Double.parseDouble((String)
table.getValueAt(zaehler, 3)),
                              (String) table.getValueAt(zaehler, 4),
                              (String) table.getValueAt(zaehler, 5),
                              (String) table.getValueAt(zaehler, 6), aRect,
aArc, aLine);

                  /*
                   * Abfrage, welche Seite aktuell gewählt ist und
anschließend
                   * Comp-Objekt der globalen ArrayList hinzufügen
                   */
                  if (side.equals("top")) {
                        Daten.setaComp_top(component);
                  }
                  if (side.equals("bottom")) {
                        Daten.setaComp_bottom(component);
                  }
            }
      }

      /**
       * Methode removeDuplicates(), um Duplicate innerhalb eines Arrays zu
       * entfernen.
       *
       * Diese Funktion wird benötigt, da verschiedene Bauteile auf der
       * Leiterplatte mehrmals vorkommen können (Gleiches Bauteil,
verschiedene
       * Position). Diese sollen jedoch innerhalb dieser GUI nur einmalig
```

```
in der
	 * Tabelle erscheinen, um diese auch nur einmal zu definieren.
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param String
	 *            -Array und ArrayList der Klasse Placement für alle
verwendeten
	 *            Bauteile auf der Leiterplatte
	 * @return Von duplikaten befreites String-Array
	 */

	public String[] removeDuplicates(String[] comp,
			ArrayList<Placement> arrayList) {

		/* Komplette ArrayList durchlaufen, d.h. alle verwendeten
Bauteile */
		for (int i = 0; i < arrayList.size(); i++) {
			/* Klassenvariable cmp_nr des aktuellen Objektes in Array
speichern */
			comp[i] = arrayList.get(i).getCmp_nr();
		}

		/* Deklaration eines Sets, welches keine Duplikate enthalten
darf */
		Set<String> strings = new HashSet<String>();
		/* Durchlaufen des kompletten Arrays, d.h. alle verwendeten
Bauteile */
		for (int i = 0; i < comp.length; i++) {
			/*
			 * Wert des aktuellen Array-Feldes in Set schreiben.
Sollte nun ein
			 * Duplikat auftauchen, wird es nicht ins Set
geschrieben.
			 */
			strings.add(comp[i]);
		}

		/* Set wieder in Array umwandeln */
		comp = strings.toArray(new String[0]);
		/* Array ohne Duplikate zurückgeben */
		return comp;
	}

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Besitzer
	 *            des Frames, Titel, Modal-Typ (True od. False)
	 */
	GUI_comp(Frame owner, String title, boolean modal) {
		super(owner, title, modal);

		/*
		 * Deklaration der Arrays, in welche später für jedes Bauteil
deren
```

```
             * cmp_nr geschrieben werden
             */
            String[] comp_top = new String[aPlacement_top.size()];
            String[] comp_bottom = new String[aPlacement_bottom.size()];

            /* Duplikate entfernen */
            comp_top = removeDuplicates(comp_top, aPlacement_top);
            comp_bottom = removeDuplicates(comp_bottom, aPlacement_bottom);

            /* JDBC-Treiber laden, um verbindung mit SQL-Datenbank
aufzunehmen */
            try {
                  Class.forName("com.mysql.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                  System.out.println("Kann nicht geladen werden");
                  return;
            }
            try {
                  /* Verbindung mit Datenbank aufnehmen */
                  Connection con = DriverManager

      .getConnection("jdbc:mysql://localhost:3306/pcb_viewer",
                                          "root", "systemv");
                  Statement stmt = con.createStatement();
                  /*
                   * Deklaration der Arrays, in welche die Daten der
Bauteile
                   * geschrieben werden
                   */
                  top = new String[comp_top.length][8];
                  bottom = new String[comp_bottom.length][8];
                  /*
                   * Deklaration der Arrays, mit welchen später überprüft
wird, ob
                   * alle Daten vollständig sind
                   */
                  ready_top = new int[comp_top.length];
                  ready_bottom = new int[comp_bottom.length];

                  /* Komplettes Array mit allen verwendeten Bauteilen
durchlaufen */
                  for (int i = 0; i < comp_top.length; i++) {
                        /*
                         * SQL-Abfrage aller Bauteil-Daten des aktuell im
Array
                         * selektierten Bauteils
                         */
                        String input_top = "Select * from component where
cmp_nr=\""
                                    + comp_top[i] + "\"";
                        ResultSet result_top =
stmt.executeQuery(input_top);
                        /* Abfrage ob bereits Eintrag für Bauteil besteht
*/
                        if (result_top.next()) {
                              /* Felder der ausgelesenen SQL-Zeile in Array
schreiben */
                              top[i][0] = result_top.getString("cmp_nr");
```

```
                                top[i][1] = result_top.getString("breite");
                                top[i][2] = result_top.getString("laenge");
                                top[i][3] = result_top.getString("hoehe");
                                top[i][4] = result_top.getString("bauform");
                                top[i][5] = result_top.getString("typ");
                                top[i][6] = result_top.getString("library");
                                /* Bauteil bereits vorhanden */
                                top[i][7] = "old";
                                /* Daten eingefügt */
                                ready_top[i] = 1;
                          } else {
                                /*
                                 * Falls kein Eintrag besteht, wird lediglich
die cmp_nr ins
                                 * Array geschrieben
                                 */
                                top[i][0] = comp_top[i];
                                /* Bauteil nicht in Datenbank vorhanden */
                                top[i][7] = "new";
                          }

                    }
                    /* Die gleiche Prozedur jetzt für die Bottom-Seite */
                    for (int j = 0; j < comp_bottom.length; j++) {
                          String input_bottom = "Select * from component
where cmp_nr=\""
                                      + comp_bottom[j] + "\"";
                          ResultSet result_bottom =
stmt.executeQuery(input_bottom);
                          if (result_bottom.next()) {
                                bottom[j][0] =
result_bottom.getString("cmp_nr");
                                bottom[j][1] =
result_bottom.getString("breite");
                                bottom[j][2] =
result_bottom.getString("laenge");
                                bottom[j][3] =
result_bottom.getString("hoehe");
                                bottom[j][4] =
result_bottom.getString("bauform");
                                bottom[j][5] =
result_bottom.getString("typ");
                                bottom[j][6] =
result_bottom.getString("library");
                                bottom[j][7] = "old";
                                ready_bottom[j] = 1;
                          } else {
                                bottom[j][0] = comp_bottom[j];
                                bottom[j][7] = "new";
                          }

                    }
                    /* Verbindung zur Datenbank schliessen */
                    stmt.close();
                    con.close();
              } catch (SQLException se) {
                    System.out.println("SQL Fehler " + se.getMessage());
                    se.printStackTrace(System.out);
```

```
            }

            /*
             * Deklaration des Tabellen-Modells, bestehend aus den Daten
selbst -->
             * den gerade gefüllten 2-Dimensionalen Arrays und den
             * Spaltenbeschriftungen.
             */
            DefaultTableModel tabelle_top = new DefaultTableModel(top,
spalten);
            DefaultTableModel tabelle_bottom = new
DefaultTableModel(bottom,
                        spalten);

            /*
             * Deklaration der eigentlichen Tabellen, um die Daten
tabellarisch
             * anzuzeigen
             */
            final JTable table_top = new JTable(tabelle_top);
            final JTable table_bottom = new JTable(tabelle_bottom);

            /*
             * JTabbedPane leeren, ansonsten wird bei jedem aufruf ein
neuer Tab
             * hinzugefügt
             */
            tTop.removeAll();
            /* Tabs mit jeweiligen Tabellen hinzufügen */
            tTop.addTab("Top", new JScrollPane(table_top));
            tTop.addTab("Bottom", new JScrollPane(table_bottom));
            /* GUI-Komponenten den Panels hinzufügen */
            center.add(tTop);
            south.add(bOK);
            south.add(bDaten);

            /* Layout-Manager zurücksetzen */
            this.getContentPane().setLayout(new BorderLayout(10, 10));
            /* Panels der ContentPane hinzufügen */
            this.getContentPane().add(center, BorderLayout.CENTER);
            this.getContentPane().add(south, BorderLayout.SOUTH);

            /* Button OK deaktivieren, bis alle Daten korrekt eingegeben
wurden */
            bOK.setEnabled(false);

            /**
             * Aktion-Listener der Tabelle Top
             *
             * Durch einen Klick auf die gewünschte Zeile, wird das Fenster
zur
             * Definition der Bauteile geöffnet (GUI_comp_define). Dort
kann das
             * jeweilige Bauteil dann definiert werden.
             *
             *
             * @author Patrick Olma
             * @version 30.06.2011
```

```
             *
             */
            table_top.addMouseListener(new MouseAdapter() {
                  public void mouseClicked(MouseEvent e) {
                        /* Aktuell aktive Tabelle */
                        JTable target = (JTable) e.getSource();
                        /*
                         * row = aktuell selektierte Zeile, column =
aktuell selektierte
                         * Spalte
                         */
                        row = target.getSelectedRow();
                        column = target.getSelectedColumn();
                        /*
                         * aktuell selektierte Zeile wird der
Klassenvariable row der
                         * Klasse GUI_Comp_define zugewiesen, damit diese
weiss, welches
                         * Bauteil definiert werden soll.
                         */
                        GUI_comp_define.row = row;
                        /* Dialog zu definition des Bauteils sichtbar
machen */
                        GUI_comp_define comp_define = new
GUI_comp_define(null,
                                    "Bauteildefinition", true);
                        /*
                         * Nach schließen des Dialogs werden Daten aus
Dialog übernommen
                         * und den entsprechenden Feldern der Tabelle
zugewiesen
                         */
                        target.setValueAt(comp_define.tBreite.getText(),
row, 1);
                        target.setValueAt(comp_define.tLaenge.getText(),
row, 2);
                        target.setValueAt(comp_define.tHoehe.getText(),
row, 3);
                        target.setValueAt(comp_define.tBauform.getText(),
row, 4);
                        target.setValueAt(comp_define.tTyp.getText(), row,
5);
                        target.setValueAt(comp_define.tLibrary.getText(),
row, 6);
                        /* Daten definiert */
                        ready_top[row] = 1;
                  }
            });

            /**
             * Aktion-Listener der Tabelle Bottom
             *
             * Durch einen Klick auf die gewünschte Zeile, wird das Fenster
zur
             * Definition der Bauteile geöffnet (GUI_comp_define). Dort
kann das
             * jeweilige Bauteil dann definiert werden.
             *
```

```
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            table_bottom.addMouseListener(new MouseAdapter() {
                  public void mouseClicked(MouseEvent e) {
                        /* Aktuell aktive Tabelle */
                        JTable target = (JTable) e.getSource();
                        /*
                         * row = aktuell selektierte Zeile, column =
aktuell selektierte
                         * Spalte
                         */
                        row = target.getSelectedRow();
                        column = target.getSelectedColumn();
                        /*
                         * aktuell selektierte Zeile wird der
Klassenvariable row der
                         * Klasse GUI_Comp_define zugewiesen, damit diese
weiss, welches
                         * Bauteil definiert werden soll.
                         */
                        GUI_comp_define.row = row;
                        /* Dialog zu definition des Bauteils sichtbar
machen */
                        GUI_comp_define comp_define = new
GUI_comp_define(null,
                                    "Bauteildefinition", true);
                        /*
                         * Nach schließen des Dialogs werden Daten aus
Dialog übernommen
                         * und den entsprechenden Feldern der Tabelle
zugewiesen
                         */
                        target.setValueAt(comp_define.tBreite.getText(),
row, 1);
                        target.setValueAt(comp_define.tLaenge.getText(),
row, 2);
                        target.setValueAt(comp_define.tHoehe.getText(),
row, 3);
                        target.setValueAt(comp_define.tBauform.getText(),
row, 4);
                        target.setValueAt(comp_define.tTyp.getText(), row,
5);
                        target.setValueAt(comp_define.tLibrary.getText(),
row, 6);
                        /* Daten definiert */
                        ready_bottom[row] = 1;
                  }
            });

            /**
             * Aktion-Listener des Button "Daten einlesen"
             *
             * Durch einen Klick auf diesen Button wird die Methode
einlesen()
             * aufgerufen
```

```
             *
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            bDaten.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* ArrayListen leeren */
                        aComponent_top.clear();
                        aComponent_bottom.clear();
                        /* Abfrage ob Bestückdaten der Top-Seite vorhanden
sind */
                        if (GUI.top) {
                              /*
                               * Kompletter Durchlauf des Arrays, für jede
Zeile der
                               * Tabelle wurde bei vollständiger eingabe
der Wert 1 ins
                               * Array geschriben
                               */
                              for (int i = 0; i < ready_top.length; i++) {
                                    /* Abfrage ob Daten vollständig
eingegeben wurden */
                                    if (ready_top[i] == 0) {
                                          ready = false;
                                    } else {
                                          ready = true;
                                    }
                              }
                        }
                        /* Abfrage ob Bestückdaten der Bottom-Seite
vorhanden sind */
                        if (GUI.bottom) {
                              /* Gleiche Prozedur, als bei Top-Seite */
                              for (int j = 0; j < ready_bottom.length; j++)
{
                                    if (ready_bottom[j] == 0) {
                                          ready = false;
                                    } else {
                                          ready = true;
                                    }
                              }
                        }

                        /*
                         * Abfrage ob Daten der beiden Seiten vollständig
abgegeben
                         * wurden
                         */
                        if (ready) {
                              /* Abfrage, ob Bestückdaten der Top-Seite
vorhanden sind */
                              if (GUI.top) {
                                    /*
                                     * Daten der Tabelle einlesen, durch
Methodenaufruf
                                     * enlesen()
```

```
                                        */
                                       einlesen(top, table_top, "top");
                                       /* Alle Bauteil-Daten eingelesen */
                                       comp_ready = true;
                                 }
                                 /* Abfrage, ob Bestückdaten der Bottom-Seite
vorhanden sind */
                                 if (GUI.bottom) {
                                       /* Gleiche Prozedur, als bei Top-Seite
*/
                                       einlesen(bottom, table_bottom,
"bottom");
                                       comp_ready = true;
                                 }
                                 /* OK-Button aktivieren */
                                 bOK.setEnabled(true);
                           } else {
                                 JOptionPane.showMessageDialog(null,
                                             "Daten nicht vollständig!",
"Daten unvollständig",
                                             JOptionPane.INFORMATION_MESSAGE);
                           }
                           /* Layout auf GUI neu zeichnen */
                           GUI.hd.aktualisieren();
                     }
               });

               /**
                * Aktion-Listener des Button "OK"
                *
                * Durch einen Klick auf diesen Button werden alle
eingetragenen Daten
                * der Bauteile in die Datenbank geschrieben, um diese zu
späteren
                * Zeitpunkten wieder verwenden zu können. Zusätzlich wird der
Dialog
                * geschlossen.
                *
                *
                * @author Patrick Olma
                * @version 30.06.2011
                *
                */

               bOK.addActionListener(new ActionListener() {
                     public void actionPerformed(ActionEvent e) {
                           /*
                            * JDBC-Treiber laden um Verbindung mit der
Datenbank
                            * aufzunehmen
                            */
                           try {
                                 Class.forName("com.mysql.jdbc.Driver");
                           } catch (ClassNotFoundException ee) {
                                 System.out.println("Kann nicht geladen
werden");
                                 return;
                           }
```

```
                        try {
                              /* Verbindung mit der Datenbank aufnehmen */
                              Connection con = DriverManager.getConnection(

      "jdbc:mysql://localhost:3306/pcb_viewer", "root",
                                          "systemv");
                              Statement stmt = con.createStatement();
                              String input;
                              /* Komplettes Array durchlaufen, also alle
Bauteildaten */
                              for (int i = 0; i < top.length; i++) {
                                    /*
                                     * Pfad der Geometrie-Library
umwandeln, damit dieser
                                     * mit korrekten Slashes in die
Datenbank eingefügt wird
                                     */
                                    String pfad = top[i][6];
                                    pfad = pfad.replace("\\", "\\\\");
                                    /*
                                     * Abfrage ob Bauteil-Daten bereits
vorhanden waren. In
                                     * diesem Fall müssen die Daten über
das
                                     * update-Statement geändert werden.
                                     */
                                    if (top[i][7].equals("old")) {
                                          /* Felder des Arrays in
Datenbanken aktualieren */
                                          input = "update component set
cmp_nr=\""
                                                      + top[i][0] + "\",
breite=\"" + top[i][1]
                                                      + "\", laenge=\"" +
top[i][2]
                                                      + "\", hoehe=\"" +
top[i][3]
                                                      + "\", bauform=\"" +
top[i][4]
                                                      + "\", typ=\"" +
top[i][5]
                                                      + "\", library=\"" +
pfad
                                                      + "\" where
cmp_nr=\"" + top[i][0] + "\"";
                                          stmt.executeUpdate(input);
                                    } else {
                                          /*
                                           * Falls die Bauteil-Daten noch
nicht vorhanden
                                           * waren, müssen sie über das
insert-Statement neu
                                           * hinzugefügt werden
                                           */
                                          input = "insert into component
(cmp_nr, breite, laenge, hoehe, bauform, typ, library) values (\""
                                                      + top[i][0]
                                                      + "\", \""
```

```
                                                + top[i][1]
                                                + "\", \""
                                                + top[i][2]
                                                + "\", \""
                                                + top[i][3]
                                                + "\", \""
                                                + top[i][4]
                                                + "\", \""
                                                + top[i][5]
                                                + "\", \""
                                                + pfad
                                                + "\")";
                                          stmt.executeUpdate(input);
                                    }
                              }

                              /* Gleiche Prozedur, als bei Top-Seite */
                              for (int j = 0; j < bottom.length; j++) {
                                    String pfad = bottom[j][6];
                                    pfad = pfad.replace("\\", "\\\\");
                                    if (bottom[j][7].equals("old")) {
                                          input = "update component set
cmp_nr=\""
                                                + bottom[j][0] + "\",
breite=\""
                                                + bottom[j][1] + "\",
laenge=\""
                                                + bottom[j][2] + "\",
hoehe=\""
                                                + bottom[j][3] + "\",
bauform=\""
                                                + bottom[j][4] + "\",
typ=\""
                                                + bottom[j][5] + "\",
library=\"" + pfad
                                                + "\" where
cmp_nr=\"" + bottom[j][0]
                                                + "\"";
                                          stmt.executeUpdate(input);
                                    } else {
                                          input = "insert into component
(cmp_nr, breite, laenge, hoehe, bauform, typ, library) values (\""
                                                + bottom[j][0]
                                                + "\", \""
                                                + bottom[j][1]
                                                + "\", \""
                                                + bottom[j][2]
                                                + "\", \""
                                                + bottom[j][3]
                                                + "\", \""
                                                + bottom[j][4]
                                                + "\", \""
                                                + bottom[j][5]
                                                + "\", \""
                                                + bottom[j][6] +
"\")";
                                          stmt.executeUpdate(input);
                                    }
```

```
                                }
                                /* Verbindung schließen */
                                stmt.close();
                                con.close();
                          } catch (SQLException se) {
                                System.out.println("SQL Fehler " +
se.getMessage());
                                se.printStackTrace(System.out);
                          }

                          /* Dialog schließen */
                          setVisible(false);
                    }
              });

              /* Packen */
              pack();
              /* Größe angeben */
              setSize(500, 550);
              /* Methode beim schließen des Dialogs */
              setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
              /* Position des Dialogs auf dem Bildschirm. Hier: Mitte des
Desktops */
              setLocation(
                          (Toolkit.getDefaultToolkit().getScreenSize().width
- getSize().width) / 2,
                          (Toolkit.getDefaultToolkit().getScreenSize().height
- getSize().height) / 2);
              /* Dialog sichtbar machen */
              setVisible(true);
        }
}
```

8.7 GUI_new.java

```
/**
 * Klasse für die GUI, in welcher über einen Öffnen-Dialog
 * die einzelnen Bestückfiles der jeweiligen Seiten der Leiterplatte
 * eingelesen werden.
 * Die Textdateien werden zeilenweise eingelesen und für jede Zeile wird
 * ein entsprechendes Objekt erstellt.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.ArrayList;

import javax.swing.*;

public class GUI_new extends JDialog {

	/* Deklaration der Klassenvariablen */
	private static final long serialVersionUID = 1L;

	/*
	 * ArrayListen für die Daten der der Leiterplatte, welche bereits mit
den
	 * vorhandenen Daten gefüllt werden
	 */
	ArrayList<PCB> aLeiterplatte = Daten.getAPCB();

	/*
	 * ArrayListen für die Daten der Bauteile, welche bereits mit den
	 * vorhandenen Daten gefüllt werden
	 */
	ArrayList<Placement> aPlacement_top = Daten.getAPlace_top();
	ArrayList<Placement> aPlacement_bottom = Daten.getAPlace_bottom();

	/*
	 * ArrayListen für die Geometrie-Daten der Bauteile, welche bereits
mit den
	 * vorhandenen Daten gefüllt werden
	 */
	ArrayList<Comp> aComponent_top = Daten.getaComp_top();
	ArrayList<Comp> aComponent_bottom = Daten.getaComp_bottom();

	/* Deklaration der GUI-Komponenten */
	JLabel lPlace_top = new JLabel("Bitte Placement-Datei [TOP]
angeben:",
			JLabel.LEFT);
	JLabel lPlace_bottom = new JLabel(
			"Bitte Placement-Datei [BOTTOM] angeben:", JLabel.LEFT);
	JTextField tPlace_top = new JTextField("", JTextField.LEFT);
	JTextField tPlace_bottom = new JTextField("", JTextField.LEFT);
```

```
	JButton bOpenPlace_top = new JButton("...");
	JButton bOpenPlace_bottom = new JButton("...");
	JButton bOK = new JButton("OK");

	JCheckBox rTop = new JCheckBox("Top");
	JCheckBox rBottom = new JCheckBox("Bottom");

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Besitzer
	 *            des Frames, Titel, Modal-Typ (True od. False)
	 */
	GUI_new(Frame owner, String title, boolean modal) {

		super(owner, title, modal);

		/* Position und Größe der GUI-Komponenten angeben */
		lPlace_top.setBounds(10, 10, 200, 20);
		lPlace_bottom.setBounds(10, 80, 200, 20);
		tPlace_top.setBounds(10, 40, 200, 22);
		tPlace_bottom.setBounds(10, 110, 200, 22);
		bOpenPlace_top.setBounds(220, 40, 60, 20);
		bOpenPlace_bottom.setBounds(220, 110, 60, 20);
		bOK.setBounds(10, 140, 60, 20);
		rTop.setBounds(100, 140, 60, 20);
		rBottom.setBounds(170, 140, 60, 20);

		/* Textfelder und Buttons deaktivieren, bis Checkbox gesetzt
wird */
		tPlace_top.setEnabled(false);
		tPlace_bottom.setEnabled(false);
		bOpenPlace_top.setEnabled(false);
		bOpenPlace_bottom.setEnabled(false);

		/* Layout-Manager zurücksetzen */
		this.getContentPane().setLayout(null);

		/* GUI-Komponenten der Content-Pane hinzufügen */
		this.getContentPane().add(lPlace_top);
		this.getContentPane().add(lPlace_bottom);
		this.getContentPane().add(tPlace_top);
		this.getContentPane().add(tPlace_bottom);
		this.getContentPane().add(bOpenPlace_top);
		this.getContentPane().add(bOpenPlace_bottom);
		this.getContentPane().add(bOK);
		this.getContentPane().add(rTop);
		this.getContentPane().add(rBottom);

		/**
		 * Aktion-Listener der Checkbox "Top"
		 *
		 * Durch aktivieren dieser Checkbox, wird das Textfeld und der
Button
		 * für die Top-Seite aktiviert
		 *
```

```
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            rTop.addItemListener(new ItemListener() {
                  public void itemStateChanged(ItemEvent e) {
                        /* Abfrage, ob Checkbox selektiert wurde */
                        if (e.getStateChange() == ItemEvent.SELECTED) {
                              /* Textfeld und Button aktivieren */
                              tPlace_top.setEnabled(true);
                              bOpenPlace_top.setEnabled(true);
                              /* Abfrage, ob Checkbox deselektiert wurde */
                        } else if (e.getStateChange() ==
ItemEvent.DESELECTED) {
                              /* Textfeld und Button deaktivieren */
                              tPlace_top.setEnabled(false);
                              bOpenPlace_top.setEnabled(false);
                        }
                  }
            });

            /**
             * Aktion-Listener der Checkbox "Bottom"
             *
             * Durch aktivieren dieser Checkbox, wird das Textfeld und der
Button
             * für die Bottom-Seite aktiviert
             *
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            rBottom.addItemListener(new ItemListener() {
                  public void itemStateChanged(ItemEvent e) {
                        /* Abfrage, ob Checkbox selektiert wurde */
                        if (e.getStateChange() == ItemEvent.SELECTED) {
                              /* Textfeld und Button aktivieren */
                              tPlace_bottom.setEnabled(true);
                              bOpenPlace_bottom.setEnabled(true);
                              /* Abfrage, ob Checkbox deselektiert wurde */
                        } else if (e.getStateChange() ==
ItemEvent.DESELECTED) {
                              /* Textfeld und Button deaktivieren */
                              tPlace_bottom.setEnabled(false);
                              bOpenPlace_bottom.setEnabled(false);
                        }
                  }
            });

            /**
             * Aktion-Listener des Buttons "Öffnen" der Top-Seite
             *
             * Durch Klick auf diesen Button, wird der Öffnen-Dialog
geöffnet, um
             * das jeweilige Bestückfile anzugeben
             *
```

```
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            bOpenPlace_top.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Öffnen-Dialog anzeigen */
                        Datei place_top = new Datei();
                        /* Pfad zur Bestückdatei angeben */
                        place_top.oeffnen();
                        /* Pfad in Textfeld schreiben */
                        tPlace_top.setText(place_top.getVerz());
                  }
            });

            /**
             * Aktion-Listener des Buttons "Öffnen" der Bottom-Seite
             *
             * Durch Klick auf diesen Button, wird der Öffnen-Dialog
geöffnet, um
             * das jeweilige Bestückfile anzugeben
             *
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            bOpenPlace_bottom.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Öffnen-Dialog anzeigen */
                        Datei place_bottom = new Datei();
                        /* Pfad zur Bestückdatei angeben */
                        place_bottom.oeffnen();
                        /* Pfad in Textfeld schreiben */
                        tPlace_bottom.setText(place_bottom.getVerz());
                  }
            });

            /**
             * Aktion-Listener des Buttons "OK"
             *
             * Durch Klick auf diesen Button, werden die angegebenen
Bestückdateien
             * zeilenweise eingelesen und für jede Zeile wird ein Objekt
der Klasse
             * Component instantiiert, welches das Bauteil auf der
Leiterplatte
             * inkl. deren Daten wie Position, Rotation, Name etc.
repräsentiert.
             *
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            bOK.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
```

```
                        /*
                         * Deklaration der Objekte der Klasse Daten für
Top- und
                         * Bottom-Seite
                         */
                        Daten aPlace_top = new Daten();
                        Daten aPlace_bottom = new Daten();

                        /*
                         * Vorhandene ArrayListen leeren, um neue Daten
hinzuzufügen
                         */
                        aComponent_top.clear();
                        aComponent_bottom.clear();
                        aLeiterplatte.clear();

                        /*
                         * Deklaration der String-Variablen, welche beim
einlesen für
                         * die jeweilige Zeile stehen
                         */
                        String line_Place_top;
                        String line_Place_bottom;

                        /* Abfrage, ob Checkbox der Top-Seite aktiviert ist
*/
                        if (rTop.isSelected()) {
                              /*
                               * Vorhandene ArrayList leeren, um neue Daten
hinzuzufügen
                               */
                              aPlacement_top.clear();
                              try {
                                    /* Öffnet die gewünschte Bestückdatei
*/
                                    FileReader fread_Place_top = new
FileReader(tPlace_top
                                                .getText());
                                    /* Bestückdatei wird in den
BufferedReader geladen */
                                    BufferedReader in_Place_top = new
BufferedReader(
                                                fread_Place_top);

                                    /* Solange Zeile nicht leer ist */
                                    while ((line_Place_top =
in_Place_top.readLine()) != null) {
                                          /* neues Objekt der Klasse
Placement instantiieren */
                                          Placement place_top_input = new
Placement();
                                          /*
                                           * Zeile trennen und Werte für
Klassenvariablen
                                           * setzen
                                           */

      place_top_input.setValue_top(line_Place_top);
```

```
                                        /* Objekt in ArrayList hinzufügen
*/

      aPlace_top.setAPlace_top(place_top_input);
                                  }

                              } catch (IOException e1) {
                                    System.out.println("IO-Fehler!");
                              }

                              /*
                               * Boolean-Wert der Klasse GUI auf true
setzen, d.h. die
                               * Seite Top wurde eingelesen
                               */
                              GUI.top = true;
                        }
                        /* Selbe Prozedur, als bei Top-Seite */
                        if (rBottom.isSelected()) {
                              aPlacement_bottom.clear();
                              try {
                                    FileReader fread_Place_bottom = new
FileReader(
                                                tPlace_bottom.getText());
                                    BufferedReader in_Place_bottom = new
BufferedReader(
                                                fread_Place_bottom);
                                    while ((line_Place_bottom =
in_Place_bottom.readLine()) != null) {
                                          Placement place_bottom_input =
new Placement();
                                          place_bottom_input

      .setValue_bottom(line_Place_bottom);

      aPlace_bottom.setAPlace_bottom(place_bottom_input);
                                    }
                              } catch (IOException e1) {
                                    System.out.println("IO-Fehler!");
                              }

                              GUI.bottom = true;
                        }
                        /* Es wurde eine neue Leiterplatte eingelesen */
                        GUI.neu = true;
                        /* Es wurden noch nicht alle Leiterplatten-Daten
angegeben */
                        GUI_pcb.pcb_ready = false;
                        /* Eingelesene Leiterplatte der GUI hinzufügen */
                        GUI.pCenter.add(GUI.hd);
                        /* Leiterplatte neu zeichnen */
                        GUI.hd.aktualisieren();
                        /* Dialog unsichtbar machen */
                        setVisible(false);
                  }
            }

            );
```

```
            /* Packen */
            pack();
            /* Größe angeben */
            setSize(300, 200);
            /* Methode beim schließen des Dialogs */
            setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
            /* Position des Dialogs auf dem Bildschirm. Hier: Mitte des
Desktops */
            setLocation(
                        (Toolkit.getDefaultToolkit().getScreenSize().width
- getSize().width) / 2,
                        (Toolkit.getDefaultToolkit().getScreenSize().height
- getSize().height) / 2);
            /* Dialog sichtbar machen */
            setVisible(true);
      }

}
```

8.8 GUI_open.java

```
/**
 * Klasse für die GUI, in welcher über einen Dialog
 * die eindeutige Seriennummer einer bereits gespeicherten
 * Leiterplatte angegeben werden kann, um diese zu späteren
 * Zeitpunkten wieder öffnen zu können.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.awt.Frame;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.WindowConstants;

public class GUI_open extends JDialog {

	/* Deklaration der Klassenvariablen */
	private static final long serialVersionUID = 1L;

	/* Deklaration der GUI-Komponenten */
	JLabel lOpen = new JLabel("Bitte Seriennummer angeben:",
JLabel.LEFT);
	JTextField tOpen = new JTextField("");
	JButton bOK = new JButton("Öffnen");

	/**
	 * Deklaration der get-Methode
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @return Wert des Textfeldes, welche die zu öffnende Seriennummer
	 *         beinhaltet.
	 */
	public String getSNR() {
		return tOpen.getText();
	}

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Besitzer
	 *            des Frames, Titel, Modal-Typ (True od. False)
	 */
	GUI_open(Frame owner, String title, boolean modal) {
```

```
            super(owner, title, modal);

            /* Position und Größe der GUI-Komponenten angeben */
            lOpen.setBounds(10, 10, 200, 20);
            tOpen.setBounds(10, 30, 200, 20);
            bOK.setBounds(10, 70, 100, 20);

            /* Layout-Manager zurücksetzen */
            this.getContentPane().setLayout(null);

            /* GUI-Komponenten der Content-Pane hinzufügen */
            this.getContentPane().add(lOpen);
            this.getContentPane().add(tOpen);
            this.getContentPane().add(bOK);

            /**
             * Aktion-Listener des Buttons "OK"
             *
             * Durch Klick auf diesen Button, wird der Dialog geschlossen
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            bOK.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Dialog unsichtbar machen */
                        setVisible(false);
                  }
            });

            /* Packen */
            pack();
            /* Größe angeben */
            setSize(250, 130);
            /* Methode beim schließen des Dialogs */
            setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
            /* Position des Dialogs auf dem Bildschirm. Hier: Mitte des
Desktops */
            setLocation(
                        (Toolkit.getDefaultToolkit().getScreenSize().width
- getSize().width) / 2,
                        (Toolkit.getDefaultToolkit().getScreenSize().height
- getSize().height) / 2);
            /* Dialog sichtbar machen */
            setVisible(true);
      }
}
```

8.9 GUI_pcb.java

```
/**
 * Klasse für die GUI, mit welcher die Leiterplatte definiert wird.
 *
 * Definition bezieht sich hierbei auf die Abmessungen, die Sachnummern
und
 * andere generelle Daten zur Leiterplatte.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.awt.Frame;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.WindowConstants;

public class GUI_pcb extends JDialog {

	/* Deklaration der Klassenvariablen */
	private static final long serialVersionUID = 1L;
	public static int row;
	public static boolean pcb_ready = false;

	/*
	 * ArrayListen für die Daten der der Leiterplatte, welche bereits mit
den
	 * vorhandenen Daten gefüllt werden
	 */
	ArrayList<PCB> aPCB = Daten.getAPCB();

	/* Deklaration der GUI-Komponenten */
	JLabel lBreite = new JLabel("Breite:", JLabel.LEFT);
	JLabel lHoehe = new JLabel("Hoehe:", JLabel.LEFT);
	JLabel lDicke = new JLabel("Dicke:", JLabel.LEFT);
	JLabel lSNR = new JLabel("Sachnummer:", JLabel.LEFT);
	JLabel lBezeichnung = new JLabel("Bezeichnung:", JLabel.LEFT);
	JLabel lSeriennummer = new JLabel("Seriennummer:", JLabel.LEFT);
	JTextField tBreite = new JTextField("", JTextField.LEFT);
	JTextField tHoehe = new JTextField("", JTextField.LEFT);
	JTextField tDicke = new JTextField("", JTextField.LEFT);
	JTextField tSNR = new JTextField("", JTextField.LEFT);
	JTextField tBezeichnung = new JTextField("", JTextField.LEFT);
	JTextField tSeriennummer = new JTextField("", JTextField.LEFT);
	JButton bOK = new JButton("OK");
```

```
	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Besitzer
	 *            des Frames, Titel, Modal-Typ (True od. False)
	 */
	GUI_pcb(Frame owner, String title, boolean modal) {

		super(owner, title, modal);

		/* Position und Größe der GUI-Komponenten angeben */
		lBreite.setBounds(10, 10, 60, 20);
		lHoehe.setBounds(80, 10, 60, 20);
		lDicke.setBounds(150, 10, 60, 20);
		tBreite.setBounds(10, 30, 60, 20);
		tHoehe.setBounds(80, 30, 60, 20);
		tDicke.setBounds(150, 30, 60, 20);
		lSNR.setBounds(10, 70, 100, 20);
		tSNR.setBounds(10, 90, 200, 20);
		lSeriennummer.setBounds(10, 130, 100, 20);
		tSeriennummer.setBounds(10, 150, 200, 20);
		lBezeichnung.setBounds(10, 190, 150, 20);
		tBezeichnung.setBounds(10, 210, 200, 20);
		bOK.setBounds(10, 240, 60, 20);

		/*
		 * Textfeld der Seriennummer deaktivieren. Dieses Feld wird
beim neuen
		 * anlegen einer Leiterplatte angegeben und darf danach nicht
mehr
		 * geändert werden.
		 */
		tSeriennummer.setEnabled(false);

		/* Layout-Manager zurücksetzen */
		this.getContentPane().setLayout(null);

		/* GUI-Komponenten der Content-Pane hinzufügen */
		this.getContentPane().add(lBreite);
		this.getContentPane().add(lHoehe);
		this.getContentPane().add(lDicke);
		this.getContentPane().add(tBreite);
		this.getContentPane().add(tHoehe);
		this.getContentPane().add(tDicke);
		this.getContentPane().add(lSNR);
		this.getContentPane().add(lBezeichnung);
		this.getContentPane().add(tSNR);
		this.getContentPane().add(tBezeichnung);
		this.getContentPane().add(lSeriennummer);
		this.getContentPane().add(tSeriennummer);
		this.getContentPane().add(bOK);

		/* Abfrage, ob ArrayList bereits Daten enthält */
		if (aPCB.size() != 0) {
			/*
			 * Daten des letzten (aktuellster) Eintrag der ArrayList
```

```
den
                     * entsprechenden Textfeldern zuweisen
                     */
                    tBreite.setText(String.valueOf(aPCB.get(aPCB.size() - 1)
                                .getBreite()));
                    tHoehe.setText(String.valueOf(aPCB.get(aPCB.size() -
1).getHoehe()));
                    tDicke.setText(String.valueOf(aPCB.get(aPCB.size() -
1).getDicke()));
                    tSNR.setText(aPCB.get(aPCB.size() - 1).getSachnummer());
                    tBezeichnung.setText(aPCB.get(aPCB.size() -
1).getBezeichnung());
                    /*
                     * Wert der Seriennummer aus der GUI dem entsprechenden
Textfeld
                     * zuweisen
                     */
                    tSeriennummer.setText(GUI.lSNR_input.getText());
              } else {
                    /* Textfelder bleiben leer, sofern keine Daten vorhanden
sind */
                    tBreite.setText("");
                    tHoehe.setText("");
                    tDicke.setText("");
                    tSNR.setText("");
                    tBezeichnung.setText("");
                    /*
                     * Wert der Seriennummer aus der GUI dem entsprechenden
Textfeld
                     * zuweisen
                     */
                    tSeriennummer.setText(GUI.lSNR_input.getText());
              }

              /**
               * Aktion-Listener des Buttons "OK"
               *
               * Durch Klick auf diesen Button, wird geprüft, ob alle Daten
korrekt
               * angegeben wurden. Anschließend wird ein Objekt der Klasse
PCB mit den
               * angegebenen Daten instantiiert und der ArrayList
hinzugefügt.
               *
               * @author Patrick Olma
               * @version 30.06.2011
               *
               */
              bOK.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                          /*
                           * Prüfen, ob alle Textfelder einen Inhalt haben.
Sollte dies
                           * nicht der Fall sein, erscheint eine Meldung
                           */
                          if (tBreite.getText().equals("")) {
                                JOptionPane.showMessageDialog(null,
                                            "Bitte alle Daten angeben",
```

```
"Daten unvollständig",
                                          JOptionPane.INFORMATION_MESSAGE);

                        } else if (tHoehe.getText().equals("")) {
                              JOptionPane.showMessageDialog(null,
                                          "Bitte alle Daten angeben",
"Daten unvollständig",
                                          JOptionPane.INFORMATION_MESSAGE);
                        } else if (tDicke.getText().equals("")) {
                              JOptionPane.showMessageDialog(null,
                                          "Bitte alle Daten angeben",
"Daten unvollständig",
                                          JOptionPane.INFORMATION_MESSAGE);
                        } else if (tSNR.getText().equals("")) {
                              JOptionPane.showMessageDialog(null,
                                          "Bitte alle Daten angeben",
"Daten unvollständig",
                                          JOptionPane.INFORMATION_MESSAGE);
                        } else if (tBezeichnung.getText().equals("")) {
                              JOptionPane.showMessageDialog(null,
                                          "Bitte alle Daten angeben",
"Daten unvollständig",
                                          JOptionPane.INFORMATION_MESSAGE);
                        }
                        /* Alle Textfelder beinhalten Daten */
                        else {
                              /*
                               * neues Objekt der Klasse PCB mit angegeben
Daten
                               * instantiieren
                               */
                              PCB pcb = new PCB(tSNR.getText(), Double
                                          .parseDouble(tBreite.getText()),
Double
                                          .parseDouble(tHoehe.getText()),
Double
                                          .parseDouble(tDicke.getText()),
tBezeichnung
                                          .getText());
                              /* Objekt der ArrayList hinzufügen */
                              aPCB.add(pcb);
                              /* Es wurden alle Leiterplatten-Daten
angegeben */
                              pcb_ready = true;
                              /*
                               * Wert der Seriennummer in entsprechendes
Textfeld der GUI
                               * schreiben
                               */

      GUI.lSNR_input.setText(tSeriennummer.getText());
                              /* Dialog schließen */
                              setVisible(false);
                        }
                  }
            });

            /* Packen */
```

```
            pack();
            /* Größe angeben */
            setSize(320, 320);
            /* Methode beim schließen des Dialogs */
            setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
            /* Position des Dialogs auf dem Bildschirm. Hier: Mitte des
Desktops */
            setLocation(
                        (Toolkit.getDefaultToolkit().getScreenSize().width
- getSize().width) / 2,
                        (Toolkit.getDefaultToolkit().getScreenSize().height
- getSize().height) / 2);
            /* Dialog sichtbar machen */
            setVisible(true);

      }
}
```

8.10 GUI.java

```
/**
 * Klasse für die Haupt-GUI, in welcher das Layout der Leiterplatte
 * gezeichnet wird und die Prüfung gesteuert werden kann.
 *
 * Bestandteile dieser GUI sind zum einen der Hauptbereich im Zentrum,
 * in welchem das Layout gezeichnet wird. Der Informationsbereich auf der
 * rechten Seite, in welchem generelle Informationen zur Leiterplatte und
den zu
 * prüfenden Bauteilen dargestellt werden. Der Steuerbereich im unteren
Teil der
 * GUI, um die Prüfung zu steuern. Und ein traditioneller
Navigationsbereich im oberen
 * Teil der GUI.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.awt.*;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import javax.swing.*;

public class GUI extends JFrame {

	/* Deklaration der Klassenvariablen */
	private static final long serialVersionUID = 1L;
	private static String side = "top";
	private int schritte_max;
	public static boolean top = false;
	public static boolean bottom = false;
	public static boolean neu = true;
	boolean vorhanden;
	boolean pruefung = false;
	boolean find = false;

	/*
	 * Instantiierung eines Objektes der Klasse Prüfung. Dieses Objekt
steht für
	 * die Prüfung an sich.
	 */
	Pruefung pruef = new Pruefung();

	/**
	 * Methode addComponent() um die gewünschten Panels einem GridBad-
Layout
	 * hinzuzufügen
```

```
        *
        * @author Patrick Olma
        * @version 30.06.2011
        * @param Der
        *             eigentliche Container, das vorhandene GridBagLayout,
die
        *             hinzuzufügende Komponente, die Position (x,y) die Größe
        *             (width, height) der Komponente und die Größe des
Layouts
        *             (weightx, weighty)
        */
       static void addComponent(Container cont, GridBagLayout gbl, Component
c,
                     int x, int y, int width, int height, double weightx,
double weighty) {
             /* Neuen GridBadConstraint instantiieren */
             GridBagConstraints gbc = new GridBagConstraints();
             /*
              * Die Höhe und die Breite der Komponente werden so verändert,
dass sie
              * den Anzeigebereich voll ausfüllt.
              */
             gbc.fill = GridBagConstraints.BOTH;
             /* Eigenschaften des GridBags definieren */
             gbc.gridx = x;
             gbc.gridy = y;
             gbc.gridwidth = width;
             gbc.gridheight = height;
             gbc.weightx = weightx;
             gbc.weighty = weighty;
             /* Constraints (Eigenschaften) dem aktuellen GridBagLayout
zuweisen */
             gbc.setConstraints(c, gbc);
             /* Komponente hinzufügen */
             cont.add(c);
       }

       /* Deklaration der GUI-Komponenten */
       JMenuBar mBar = new JMenuBar();

       JMenu mFile = new JMenu("Datei");
       JMenu mEdit = new JMenu("Bearbeiten");
       JMenu mZoom = new JMenu("Zoom");
       JMenu mRotate = new JMenu("Drehung");
       JMenu mData = new JMenu("Daten");
       JMenu mAnsicht = new JMenu("Ansicht");

       JMenuItem iNew = new JMenuItem("Neu");
       JMenuItem iOpen = new JMenuItem("Öffnen...");
       JMenuItem iClose = new JMenuItem("Schließen");
       JMenuItem iSave = new JMenuItem("Speichern");
       JMenuItem iExit = new JMenuItem("Beenden");

       JMenuItem iFind = new JMenuItem("Suche");
       JMenuItem iGroß = new JMenuItem("Vergrößern");
       JMenuItem iKlein = new JMenuItem("Verkleinern");
       JMenuItem iSeite = new JMenuItem("Auf Seite");
       JMenuItem i90UZS = new JMenuItem("90° UZS");
```

```
        JMenuItem i90GUZS = new JMenuItem("90° GUZS");
        JMenuItem i180 = new JMenuItem("180°");

        JMenuItem iPCB = new JMenuItem("PCB");
        JMenuItem iComp = new JMenuItem("Bauteil");

        JMenuItem iTop = new JMenuItem("Top");
        JMenuItem iBottom = new JMenuItem("Bottom");

        JToolBar toolbar = new JToolBar();

        JLabel lDaten = new JLabel("PCB-Daten", JLabel.LEFT);
        JLabel lPCB = new JLabel("PCB:", JLabel.LEFT);
        JLabel lSNR = new JLabel("SNR:", JLabel.LEFT);
        JLabel lUser = new JLabel("Prüfer:", JLabel.LEFT);
        JLabel lPruef = new JLabel("Prüfung", JLabel.LEFT);
        JLabel lComp = new JLabel("Bauteil:", JLabel.LEFT);
        JLabel lCompnr = new JLabel("SNR:", JLabel.LEFT);
        JLabel lRotation = new JLabel("Rotation:", JLabel.LEFT);
        JLabel lTyp = new JLabel("Typ:", JLabel.LEFT);
        JLabel lBauform = new JLabel("Bauform:", JLabel.LEFT);
        JLabel lSchritt = new JLabel("Prüfschritt:", JLabel.LEFT);
        public static JLabel lPCB_input = new JLabel("", JLabel.LEFT);
        public static JLabel lSNR_input = new JLabel("", JLabel.LEFT);
        public static JLabel lUser_input = new JLabel("", JLabel.LEFT);
        public static JLabel lComp_input = new JLabel("", JLabel.LEFT);
        public static JLabel lCompnr_input = new JLabel("", JLabel.LEFT);
        public static JLabel lRotation_input = new JLabel("", JLabel.LEFT);
        public static JLabel lTyp_input = new JLabel("", JLabel.LEFT);
        public static JLabel lBauform_input = new JLabel("", JLabel.LEFT);
        public static JLabel lSchritt_input = new JLabel("", JLabel.LEFT);
        JLabel lFortschritt = new JLabel("Fortschritt", JLabel.LEFT);
        JLabel lSchritt_manu = new JLabel("Schritt:", JLabel.LEFT);
        JLabel lTest = new JLabel("Test", JLabel.LEFT);

        JTextField tSchritt_manu = new JTextField("Bitte Bauteil eingeben",
                    JTextField.LEFT);

        JProgressBar pProgress = new JProgressBar(0, 100);

        JRadioButton rAuto = new JRadioButton("Automatischer Test");
        JRadioButton rManu = new JRadioButton("Manueller Test");
        ButtonGroup gTest = new ButtonGroup();

        JLabel lCenter = new JLabel("Center", JLabel.LEFT);

        JButton bPrev = new JButton("Zurück");
        JButton bNext = new JButton("Vor");
        JButton bStop = new JButton("Stop");
        JButton bStart = new JButton("Start");
        JButton bOK = new JButton("OK");
        JButton bNOT = new JButton("Nicht OK");

        /* Deklaration der Panels */

        JPanel pNorth = new JPanel();
        JPanel pEast = new JPanel();
        JPanel pSouth = new JPanel();
```

```
	public static JPanel pCenter = new JPanel();

	/*
	 * Neues Objekt der Klasse Layout instantiieren. Dieses Objekt
repräsentiert
	 * das Layout der Leiterplatte
	 */
	public static Layout hd = new Layout();

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 *
	 */
	public GUI() {

		/* Titel des Tools */
		super("PCB-Viewer v1.0");

		/* Methode beim schließen des Tools */
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		/*
		 * Neuen Container anlgen mit Inhalt der aktuellen ContenPane
		 */
		Container c = this.getContentPane();

		/* Neues GridBagLayout anlegen */
		GridBagLayout gbl = new GridBagLayout();

		/* GridBagLayout dem Container zuweisen */
		c.setLayout(gbl);

		/* Navigationskomponenten hinzufügen */
		mFile.add(iNew);
		mFile.add(iOpen);
		mFile.addSeparator();
		mFile.add(iClose);
		mFile.addSeparator();
		mFile.add(iSave);
		mFile.addSeparator();
		mFile.add(iExit);

		mEdit.add(iFind);
		mEdit.addSeparator();
		mEdit.add(mZoom);
		mEdit.addSeparator();
		mEdit.add(mRotate);
		mZoom.add(iGroß);
		mZoom.add(iKlein);
		mZoom.add(iSeite);
		mRotate.add(i90UZS);
		mRotate.add(i90GUZS);
		mRotate.add(i180);

		mData.add(iPCB);
```

```
            mData.add(iComp);

            mAnsicht.add(iTop);
            mAnsicht.add(iBottom);

            mBar.add(mFile);
            mBar.add(mEdit);
            mBar.add(mData);
            mBar.add(mAnsicht);

            /* Menü zuweisen */
            setJMenuBar(mBar);

            /* Art der Ränder definieren */
            pNorth.setBorder(BorderFactory.createEtchedBorder());
            pEast.setBorder(BorderFactory.createEtchedBorder());
            pSouth.setBorder(BorderFactory.createEtchedBorder());
            pCenter.setBorder(BorderFactory.createEtchedBorder());

            /* Layout-Manager zurücksetzen */
            pEast.setLayout(null);
            pCenter.setLayout(null);

            /* Hintergrund des Panels auf Weiss setzen */
            pCenter.setBackground(Color.WHITE);

            /*
             * Position, Größe, Sichtbarkeit und Art des Randes der GUI-
Komponenten
             * angeben
             */
            lDaten.setBounds(10, 10, 175, 22);
            lDaten.setBorder(BorderFactory.createBevelBorder(0));
            lPCB.setBounds(10, 40, 60, 22);
            lSNR.setBounds(10, 70, 60, 22);
            lUser.setBounds(10, 100, 60, 22);
            lPruef.setBounds(10, 140, 175, 22);
            lPruef.setBorder(BorderFactory.createBevelBorder(0));
            lComp.setBounds(10, 170, 60, 22);
            lCompnr.setBounds(10, 200, 60, 22);
            lRotation.setBounds(10, 230, 60, 22);
            lTyp.setBounds(10, 260, 60, 22);
            lBauform.setBounds(10, 290, 60, 22);
            lFortschritt.setBounds(10, 330, 175, 22);
            lFortschritt.setBorder(BorderFactory.createBevelBorder(0));
            lSchritt.setBounds(10, 360, 60, 22);

            lPCB_input.setBounds(80, 40, 100, 22);
            lPCB_input.setOpaque(true);
            lPCB_input.setBackground(new Color(0xffffff));
            lSNR_input.setBounds(80, 70, 100, 22);
            lSNR_input.setOpaque(true);
            lSNR_input.setBackground(new Color(0xffffff));
            lUser_input.setBounds(80, 100, 100, 22);
            lUser_input.setOpaque(true);
            lUser_input.setBackground(new Color(0xffffff));
            lComp_input.setBounds(80, 170, 100, 22);
            lComp_input.setOpaque(true);
```

```
        lComp_input.setBackground(new Color(0xffffff));
        lCompnr_input.setBounds(80, 200, 100, 22);
        lCompnr_input.setOpaque(true);
        lCompnr_input.setBackground(new Color(0xffffff));
        lRotation_input.setBounds(80, 230, 100, 22);
        lRotation_input.setOpaque(true);
        lRotation_input.setBackground(new Color(0xffffff));
        lTyp_input.setBounds(80, 260, 100, 22);
        lTyp_input.setOpaque(true);
        lTyp_input.setBackground(new Color(0xffffff));
        lBauform_input.setBounds(80, 290, 100, 22);
        lBauform_input.setOpaque(true);
        lBauform_input.setBackground(new Color(0xffffff));

        pProgress.setBounds(10, 390, 175, 22);
        pProgress.setStringPainted(false);

        lSchritt_input.setBounds(80, 360, 100, 22);
        lSchritt_input.setOpaque(true);
        lSchritt_input.setBackground(new Color(0xffffff));

        lTest.setBounds(10, 430, 175, 22);
        lTest.setBorder(BorderFactory.createBevelBorder(0));

        gTest.add(rAuto);
        gTest.add(rManu);
        rAuto.setBounds(10, 460, 175, 22);
        rManu.setBounds(10, 490, 175, 22);

        /* Checkbox "Automatischer Test" selektieren */
        rAuto.setSelected(true);

        lSchritt_manu.setBounds(10, 520, 60, 22);
        tSchritt_manu.setBounds(80, 520, 100, 22);

        /* Textfeld deaktivieren */
        tSchritt_manu.setEnabled(false);

        bStart.setBounds(10, 560, 70, 22);
        bStop.setBounds(100, 560, 70, 22);

        bStop.setEnabled(false);
        bPrev.setEnabled(false);
        bNext.setEnabled(false);
        bOK.setEnabled(false);
        bNOT.setEnabled(false);

        /* Menüpunkt Save deaktiveren */
        iSave.setEnabled(false);

        /* GUI-Komponenten den Panels hinzufügen */
        pEast.add(lDaten);
        pEast.add(lPCB);
        pEast.add(lPCB_input);
        pEast.add(lSNR);
        pEast.add(lSNR_input);
        pEast.add(lUser);
        pEast.add(lUser_input);
```

```
            pEast.add(lPruef);
            pEast.add(lComp);
            pEast.add(lComp_input);
            pEast.add(lCompnr);
            pEast.add(lCompnr_input);
            pEast.add(lRotation);
            pEast.add(lRotation_input);
            pEast.add(lRotation);
            pEast.add(lRotation_input);
            pEast.add(lTyp);
            pEast.add(lTyp_input);
            pEast.add(lBauform);
            pEast.add(lBauform_input);
            pEast.add(lSchritt);
            pEast.add(lSchritt_input);
            pEast.add(lFortschritt);
            pEast.add(pProgress);
            pEast.add(lTest);
            pEast.add(rAuto);
            pEast.add(rManu);
            pEast.add(lSchritt_manu);
            pEast.add(tSchritt_manu);
            pEast.add(bStart);
            pEast.add(bStop);

            pSouth.add(bPrev);
            pSouth.add(bNext);
            pSouth.add(bOK);
            pSouth.add(bNOT);

            pNorth.add(toolbar);

            /**
             * Aktion-Listener des Menüpunktes "Datei --> Neu"
             *
             * Durch Klick auf diesen Menüpunkt wird ein Dialog geöffnet,
mit
             * welchem man in die Lage ist, die jeweiligen Bestückfiles für
die
             * Leiterplatte anzugeben. Diese Bestückfiles werden dann
zeilenweise
             * eingelesen und das entstehende Layout wird auf die GUI
gezeichnet.
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            iNew.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Dialog GUI_new sichtbar machen */
                        GUI_new neu = new GUI_new(null, "Neue PCB", true);
                        /*
                         * Nachdem Bestückfiles eingelesen wurden, Dialog
GUI_comp
                         * sichtbar machen, um Geometrie-Daten der Bauteile
zu
                         * definieren
```

```
                         */
                        GUI_comp comp = new GUI_comp(null, "Bauteil",
true);
                        /*
                         * Nachdem Geometrie-Daten der Bauteile eingelesen
wurden,
                         * Dialog GUI_pcb sichtbar machen, um
Leiterplatten-Daten zu
                         * definieren
                         */
                        GUI_pcb pcb = new GUI_pcb(null, "PCB", true);
                        /* Button des Steuerungsbereiches deaktivieren */
                        bPrev.setEnabled(false);
                        bNext.setEnabled(false);
                        bOK.setEnabled(false);
                        bNOT.setEnabled(false);
                        /* Button Start aktivieren --> Prüfung starten */
                        bStart.setEnabled(true);
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Daten --> Bauteil"
             *
             * Durch Klick auf diesen Menüpunkt wird ein Dialog geöffnet,
mit
             * welchem man in die Lage ist, die jeweiligen Geometrie-Daten
der
             * einzelnen Bauteile zu definieren
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            iComp.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Dialog GUI_comp sichtbar machen */
                        GUI_comp comp_daten = new GUI_comp(null, "Bauteil-
Daten", true);
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Daten --> Leiterplatte"
             *
             * Durch Klick auf diesen Menüpunkt wird ein Dialog geöffnet,
mit
             * welchem man in die Lage ist, die jeweiligen Daten der
Leiterplatte zu
             * definieren
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            iPCB.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Dialog GUI_pcb sichtbar machen */
```

```
                        GUI_pcb pcb_daten = new GUI_pcb(null, "PCB-Daten",
true);
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Ansicht --> Top"
             *
             * Durch Klick auf diesen Menüpunkt wird die Top-Seite der
Leiterplatte
             * auf die GUI gezeichnet
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            iTop.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Globale Variable auf "top" setzen */
                        setSide("top");
                        /* Layout neu zeichnen */
                        hd.aktualisieren();
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Ansicht --> Bottom"
             *
             * Durch Klick auf diesen Menüpunkt wird die Bottom-Seite der
             * Leiterplatte auf die GUI gezeichnet
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            iBottom.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Globale Variable auf "bottom" setzen */
                        setSide("bottom");
                        /* Layout neu zeichnen */
                        hd.aktualisieren();
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Bearbeiten --> Drehen -->
90° UZS"
             *
             * Durch Klick auf diesen Menüpunkt wird das Layout der
Leiterplatte um
             * 90° im Uhrzeigersinn gedreht
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            i90UZS.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
```

```
                        /* Methode rotate() der Klasse Layout aufrufen und
um 90° drehen */
                        hd.rotate(90);
                        /* Layout neu zeichnen */
                        hd.aktualisieren();
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Bearbeiten --> Drehen -->
90° GUZS"
             *
             * Durch Klick auf diesen Menüpunkt wird das Layout der
Leiterplatte um
             * 90° gegen den Uhrzeigersinn gedreht
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            i90GUZS.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /*
                         * Methode rotate() der Klasse Layout aufrufen und
um -90°
                         * drehen
                         */
                        hd.rotate(-90);
                        /* Layout neu zeichnen */
                        hd.aktualisieren();
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Bearbeiten --> Drehen -->
180°"
             *
             * Durch Klick auf diesen Menüpunkt wird das Layout der
Leiterplatte um
             * 180° gedreht
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            i180.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /*
                         * Methode rotate() der Klasse Layout aufrufen und
um 180°
                         * drehen
                         */
                        hd.rotate(180);
                        /* Layout neu zeichnen */
                        hd.aktualisieren();
                  }
            });
```

```
		/**
		 * Aktion-Listener des Menüpunktes "Bearbeiten --> Zoom -->
Vergrößern"
		 *
		 * Durch Klick auf diesen Menüpunkt wird das aktuell zu
prüfende Bauteil
		 * ins Zentrum der GUI verschoben und um einen bestimmten
Faktor
		 * vergrößert.
		 *
		 * @author Patrick Olma
		 * @version 30.06.2011
		 *
		 */
		iGroß.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				/*
				 * Methode scale_groß der Klasse Layout aufrufen.
Aktuelles
				 * Bauteil wird ins Zentrum verschoben und
vergrößert
				 */
				hd.scale_groß(pruef.getZaehler());
			}
		});

		/**
		 * Aktion-Listener des Menüpunktes "Bearbeiten --> Zoom -->
Verkleinern"
		 *
		 * Durch Klick auf diesen Menüpunkt wird das aktuell zu
prüfende Bauteil
		 * ins Zentrum der GUI verschoben und um einen bestimmten
Faktor
		 * verkleinert.
		 *
		 * @author Patrick Olma
		 * @version 30.06.2011
		 *
		 */
		iKlein.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				/*
				 * Methode scale_klein der Klasse Layout aufrufen.
Aktuelles
				 * Bauteil wird ins Zentrum verschoben und
verkleinert
				 */
				hd.scale_klein(pruef.getZaehler());
			}
		});

		/**
		 * Aktion-Listener des Menüpunktes "Bearbeiten --> Zoom -->
Seite"
		 *
		 * Durch Klick auf diesen Menüpunkt wird der Vergrößerungs/
		 * Verkleinerungsfaktor zurückgesetzt und das komplette Layout
```

```
wird
		 * wieder auf die GUI gezeichnet.
		 *
		 * @author Patrick Olma
		 * @version 30.06.2011
		 *
		 */
		iSeite.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				/*
				 * Methode scale_reset() der Klasse Layout
aufrufen. Faktor wird
				 * zurückgesetzt und komplettes Layout wieder auf
GUI gezeichnet
				 */
				hd.scale_reset();
			}
		});

		/**
		 * Aktion-Listener des Menüpunktes "Datei --> Speichern"
		 *
		 * Durch Klick auf diesen Menüpunkt wird der aktuelle Stand der
		 * Leiterplatte inkl. aller Daten und Ergebnisse ein eine
vorhandene
		 * Datenbank geschrieben
		 *
		 * @author Patrick Olma
		 * @version 30.06.2011
		 *
		 */
		iSave.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				/*
				 * Aufruf der Funktion speichern() um Daten in
Datenbank zu
				 * schreiben
				 */
				speichern();
			}
		});

		/**
		 * Aktion-Listener des Menüpunktes "Datei --> Öffnen"
		 *
		 * Durch Klick auf diesen Menüpunkt werden gespeicherte Daten
aus der
		 * Datenbank eingelesen
		 *
		 * @author Patrick Olma
		 * @version 30.06.2011
		 *
		 */
		iOpen.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				/* Dialog GUI_open sichtbar machen um Seriennummer
anzugeben */
				GUI_open open = new GUI_open(null, "Öffnen", true);
```

```
                        /*
                         * Funktion öffnen aufrufen um Daten mit
eingegebener
                         * Seriennumer einzulesen
                         */
                        oeffnen(open.getSNR());
                        /* Abfrage, ob Datensatz in Datenbank vorhanden ist
*/
                        if (vorhanden) {
                              /*
                               * Dialog GUI_comp sichtbar machen, um
Geometrie-Daten der
                               * Bauteile einzulesen
                               */
                              GUI_comp comp_daten = new GUI_comp(null,
"Bauteil-Daten",
                                          true);
                              /*
                               * Methode scale_reset() der Klasse Layout
aufrufen. Faktor
                               * wird zurückgesetzt und komplettes Layout
auf GUI
                               * gezeichnet
                               */
                              hd.scale_reset();
                              /* Abfrage, ob Ansicht --> Top gewählt wurde
*/
                              if (getSide().equals("top")) {
                                    /*
                                     * Maximale Anzahl an Schritten =
Anzahl an Objekten in
                                     * ArrayList
                                     */
                                    schritte_max =
Daten.getAPlace_top().size();
                              }
                              /* Abfrage, ob Ansicht --> Bottom gewählt
wurde */
                              if (getSide().equals("bottom")) {
                                    /*
                                     * Maximale Anzahl an Schritten =
Anzahl an Objekten in
                                     * ArrayList
                                     */
                                    schritte_max =
Daten.getAPlace_bottom().size();
                              }
                        }
                        /* Wert der Fortschrittsanzeige setzen */
                        pProgress.setValue(((pruef.getZaehler() + 1) * (100
/ schritte_max)));
                        /*
                         * Status auf 1 setzen, d.h. aktuelles Bauteil wird
gelb
                         * eingefärbt
                         */
                        pruef.setStatus(1);
                        /* Prüfung beginnen */
```

```
                        pruef.pruefe();
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Datei --> Schließen"
             *
             * Durch Klick auf diesen Menüpunkt werden alle Daten
zurückgesetzt und
             * das aktuelle Layout wird von der GUI entfernt
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            iClose.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /*
                         * Dialog sichtbar machen, um abzufragen, ob
gespeichert werden
                         * soll
                         */
                        int result = JOptionPane.showConfirmDialog(null,
                                    "Prüfung speichern?", "Speichern",
                                    JOptionPane.YES_NO_CANCEL_OPTION);
                        /* Speichern... */
                        if (result == JOptionPane.YES_OPTION) {
                              /*
                               * Methode speichern() aufrufen, um Daten in
Datenbank zu
                               * schreiben
                               */
                              speichern();
                              /*
                               * ArrayListen für die Daten der Leiterplatte
bzw. Bauteile,
                               * welche bereits mit den vorhandenen Daten
gefüllt werden
                               */
                              ArrayList<PCB> aLeiterplatte =
Daten.getAPCB();
                              ArrayList<Placement> aPlacement_top =
Daten.getAPlace_top();
                              ArrayList<Placement> aPlacement_bottom =
Daten
                                          .getAPlace_bottom();

                              /* ArrayListen leeren */
                              aLeiterplatte.clear();
                              aPlacement_top.clear();
                              aPlacement_bottom.clear();
                              /* Layout neu zeichnen */
                              hd.aktualisieren();
                              /* Buttons des Steuerungsbereiches
deaktivieren */
                              bPrev.setEnabled(false);
                              bNext.setEnabled(false);
                              bOK.setEnabled(false);
```

```
                                bNOT.setEnabled(false);
                                /* Button Start aktivieren --> Prüfung
starten */
                                bStart.setEnabled(true);
                          }
                          /*
                           * Nicht speichern... --> selbe Prozedur ohne
Methodenaufruf
                           * speichern()
                           */
                          else if (result == JOptionPane.NO_OPTION) {
                                ArrayList<PCB> aLeiterplatte =
Daten.getAPCB();
                                ArrayList<Placement> aPlacement_top =
Daten.getAPlace_top();
                                ArrayList<Placement> aPlacement_bottom =
Daten
                                            .getAPlace_bottom();
                                aLeiterplatte.clear();
                                aPlacement_top.clear();
                                aPlacement_bottom.clear();
                                hd.aktualisieren();
                                bPrev.setEnabled(false);
                                bNext.setEnabled(false);
                                bOK.setEnabled(false);
                                bNOT.setEnabled(false);
                                bStart.setEnabled(true);
                          }
                          /* Fortschrittsanzeige zurücksetzen */
                          pProgress.setValue(0);
                          /* Textfelder leeren */
                          lPCB_input.setText("");
                          lSNR_input.setText("");
                          lUser_input.setText("");
                          lComp_input.setText("");
                          lCompnr_input.setText("");
                          lRotation_input.setText("");
                          lTyp_input.setText("");
                          lBauform_input.setText("");
                          lSchritt_input.setText("");
                    }
              });

              /**
               * Aktion-Listener des Menüpunktes "Bearbeiten --> Suche"
               *
               * Durch Klick auf diesen Menüpunkt können einzelne Bauteile
direkt
               * gesucht werden.
               *
               * @author Patrick Olma
               * @version 30.06.2011
               *
               */
              iFind.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                          /* Deklaration der ArrayList */
                          ArrayList<Placement> aPlacement = null;
```

```
                        /* Zähler zurücksetzen */
                        int zaehler = 0;
                        /* Variable find um anzugeben, ob Bauteil gefunden
wurde */
                        find = false;
                        /* Dialog sichtbar machen, um zu suchendes Bauteil
anzugeben */
                        String cmp = JOptionPane.showInputDialog(null,
                                    "Bitte zu suchendes Bauteil angeben",
"Bauteil",
                                    JOptionPane.PLAIN_MESSAGE);
                        /* Abfrage, ob Ansicht --> Top gewählt wurde */
                        if (getSide().equals("top")) {
                              /* ArrayList füllen */
                              aPlacement = Daten.getAPlace_top();
                        }
                        /* Abfrage, ob Ansicht --> Buttom gewählt wurde */
                        else if (getSide().equals("bottom")) {
                              /* ArrayList füllen */
                              aPlacement = Daten.getAPlace_bottom();
                        }
                        /* Komplette ArrayList durchlaufen */
                        for (int i = 0; i < aPlacement.size(); i++) {
                              /* Abfrage, ob Bauteile gefunden wurde */
                              if
(aPlacement.get(i).getCmp_name().equals(cmp)) {
                                    /* Zähler auf aktuelles Bauteil setzen
*/
                                    zaehler = i;
                                    /* Bauteil wurde gefunden... */
                                    find = true;
                              }
                        }
                        /* Abfrage, ob Bauteil gefunden wurde */
                        if (find) {
                              /* Zähler setzen */
                              pruef.setZaehler(zaehler);
                              /* Status des Bauteils auf 1 setzen, d.h.
gelb einfärben */
                              pruef.setStatus(1);
                              /* Prüfung des Bauteils durchführen */
                              pruef.pruefe();
                              /* Layout neu zeichnen */
                              hd.aktualisieren();
                              /*
                               * Buttons aktivieren, um festzustellen ob
Bauteil
                               * korrekt/fehlerhaft ist
                               */
                              bOK.setEnabled(true);
                              bNOT.setEnabled(true);
                        }
                        /* Bauteil nicht gefunden... */
                        else {
                              JOptionPane.showMessageDialog(null,
                                          "Bauteil nicht gefunden!",
"Fehler",
                                          JOptionPane.ERROR_MESSAGE);
```

```
                              find = false;
                        }
                        bOK.setEnabled(true);
                        bNOT.setEnabled(true);
                  }
            });

            /**
             * Aktion-Listener des Menüpunktes "Datei --> Beenden"
             *
             * Durch Klick auf diesen Menüpunkt wird das Programm beendet.
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            iExit.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /*
                         * Dialog sichtbar machen, um abzufragen, ob
aktueller Stand
                         * gespeichert werden soll
                         */
                        int result = JOptionPane.showConfirmDialog(null,
                                    "Prüfung speichern?", "Speichern",
                                    JOptionPane.YES_NO_CANCEL_OPTION);
                        /* Speichern... */
                        if (result == JOptionPane.YES_OPTION) {
                              /*
                               * Methodenaufruf speichern() um Daten in
Datenbank zu
                               * schreiben.
                               */
                              speichern();
                              /* Programm beenden */
                              System.exit(0);
                        }
                        /*
                         * Nicht speichern... --> gleiche Prozedur ohne
Methodenaufruf
                         * speichern()
                         */
                        else if (result == JOptionPane.NO_OPTION) {
                              System.exit(0);
                        }
                  }
            });

            /**
             * Aktion-Listener des Buttons "Start"
             *
             * Durch Klick auf diesen Button beginnt die Prüfung direkt
gesucht
             * werden.
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
```

```
		 */
		bStart.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {

				/* Abfrage, ob Bestückfiles eingelesen wurden */
				if (!top && !bottom) {
					JOptionPane.showMessageDialog(null,
							"Keine Bestückdaten vorhanden!",
"Keine Daten",
							JOptionPane.INFORMATION_MESSAGE);
				} else {
					/* Dialog sichtbar machen, um Seriennummer
anzugeben */
					String pcb_snr =
JOptionPane.showInputDialog(null,
							"Bitte Seriennummer der PCB
angeben",
							"Seriennummer",
JOptionPane.PLAIN_MESSAGE);
					/* Seriennummer zuweisen */
					lSNR_input.setText(pcb_snr);
					/* Dialog sichtbar machen, um Prüfer
anzugeben */
					String pcb_pruefer =
JOptionPane.showInputDialog(null,
							"Bitte Name des Prüfers angeben",
"Prüfer",
							JOptionPane.PLAIN_MESSAGE);
					/* Prüfer zuweisen */
					lUser_input.setText(pcb_pruefer);
				}

				/* Abfrage, ob Bestückfile der Top-Seite eingelesen
wurde */
				if (top) {
					/* Ansicht --> Top auswählen */
					setSide("top");
				}
				/* Abfrage, ob Bestückfile der Bottom-Seite
eingelesen wurde */
				else if (bottom) {
					/* Ansicht --> Bottom auswählen */
					setSide("bottom");
				}

				/* Abfrage, ob alle Leiterplatten-Daten korrekt
angegeben wurden */
				if (!GUI_pcb.pcb_ready) {
					JOptionPane.showMessageDialog(null,
							"Bitte alle PCB-Daten angeben",
							"PCB-Daten unvollständig",
							JOptionPane.INFORMATION_MESSAGE);
					GUI_pcb pcb_daten = new GUI_pcb(null, "PCB-
Daten", true);
				}
				/*
				 * Abfrage, ob alle Geometrie-Daten der Bauteile
angegeben
```

```
                         * wurden
                         */
                        if (!GUI_comp.comp_ready) {
                              JOptionPane.showMessageDialog(null, "Bitte
Daten einlesen",
                                          "Daten wurden nicht eingelesen",
                                          JOptionPane.INFORMATION_MESSAGE);
                              GUI_comp comp_daten = new GUI_comp(null,
"Bauteil-Daten",
                                          true);
                        } else {
                              /*
                               * Abfrage, ob bereits Prüfung stattgefunden
hat. Bei erstem
                               * Start sind keine Daten im Placement_Array!
Diese werden
                               * erst nach dem ersten Aufruf der Funktion
pruefe() in
                               * Klasse Pruefung gespeichert
                               */
                              if (pruefung) {
                                    /* Status der Bauteile zurücksetzen */
                                    pruef.reset_status();
                              }
                              /*
                               * Abfrage, ob Daten geöffnet wurden. Nach
dem Öffnen sind
                               * keine Daten im Placement_Array! Diese
werden erst nach
                               * dem ersten Aufruf der Funktion pruefe() in
Klasse
                               * Pruefung gespeichert
                               */
                              if (!neu) {
                                    /* Status der Bauteile zurücksetzen */
                                    pruef.reset_status();
                              }
                              /* Abfrage, ob nach Bauteil gesucht wurde */
                              if (find) {
                                    /* Status der Bauteile zurücksetzen */
                                    pruef.reset_status();
                              }
                              /* Prüfung hat begonnen */
                              pruefung = true;
                              /* Neu eingelesene Daten */
                              neu = true;
                              /* Es wurde kein Bauteil gesucht */
                              find = false;
                              /* Speichern-Button deaktivieren */
                              iSave.setEnabled(false);
                              /* Vergrößerungs-/Verkleinerungsfaktor
zurücksetzen */
                              hd.scale_reset();
                              /* Zähler zurücksetzen */
                              pruef.reset_zaehler();
                              /* 1. zu prüfendes Bauteil gelb einfärben */
                              pruef.setStatus(1);
                              /* Layout neu zeichnen */
```

```
                                hd.aktualisieren();
                                /* Prüfung des Bauteils beginnen */
                                pruef.pruefe();
                                /* Maximale Anzahl an Prüfschritten zuweisen
*/
                                schritte_max = pruef.getZaehler_max();
                                /* Fortschrittsbalken zurücksetzen */
                                pProgress.setValue(0);
                                /* Steuerungsbuttons deaktivieren */
                                bPrev.setEnabled(false);
                                bNext.setEnabled(false);
                                bStart.setEnabled(false);
                                /*
                                 * Buttons aktivieren, um festzustellen, ob
Bauteil
                                 * korrekt/fehlerhaft ist
                                 */
                                bOK.setEnabled(true);
                                bNOT.setEnabled(true);
                        }
                }
        });

        /**
         * Aktion-Listener des Buttons "Stop"
         *
         * Durch Klick auf diesen Button wird die Prüfung gestoppt.
         *
         * @author Patrick Olma
         * @version 30.06.2011
         *
         */
        bStop.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                    /* Alle Steuerungs-Buttons deaktivieren */
                    bPrev.setEnabled(false);
                    bNext.setEnabled(false);
                    bOK.setEnabled(false);
                    bNOT.setEnabled(false);
                    /* Prüfung kann neu gestartet werden */
                    bStart.setEnabled(true);
                    /* Speichern jetzt möglich */
                    iSave.setEnabled(true);
              }
        });

        /**
         * Aktion-Listener des Buttons "Next"
         *
         * Durch Klick auf diesen Button gelangt man zum nächsten
Bauteil,
         * welches geprüft werden soll.
         *
         * @author Patrick Olma
         * @version 30.06.2011
         *
         */
        bNext.addActionListener(new ActionListener() {
```

```
                  public void actionPerformed(ActionEvent e) {
                        /* Vergrößerungs-/Verkleinerungsfaktor zurücksetzen
*/
                        hd.scale_reset();
                        /* Aktuelles Bauteil nicht mehr einfärben */
                        pruef.setStatus(0);
                        /* Prüfung des aktuellen Bauteils beginnen */
                        pruef.pruefe();
                        /* Schritt erhöhen, d.h. auf nächstes Bauteil
springen */
                        pruef.erhoehe_zaehler();
                        /* Aktuelles Bauteil gelb einfärben */
                        pruef.setStatus(1);
                        /* Prüfung des Bauteils beginnen */
                        pruef.pruefe();
                        /* Fortschrittsbalken setzen */
                        pProgress.setValue(((pruef.getZaehler() + 1) * (100
/ schritte_max)));
                        /* Layout neu zeichnen */
                        hd.aktualisieren();
                        /*
                         * Buttons aktivieren, um festzustellen, ob Bauteil
                         * korrekt/fehlerhaft ist
                         */
                        bOK.setEnabled(true);
                        bNOT.setEnabled(true);
                        /*
                         * Stop der Prüfung erst möglich, wenn Bauteil
identifiziert
                         * wurde
                         */
                        bStop.setEnabled(false);
                  }
            });

            /**
             * Aktion-Listener des Buttons "Prev"
             *
             * Durch Klick auf diesen Button gelangt man zum vorherigen
Bauteil,
             * welches geprüft werden soll.
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            bPrev.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Vergrößerungs-/Verkleinerungsfaktor zurücksetzen
*/
                        hd.scale_reset();
                        /* Aktuelles Bauteil nicht mehr einfärben */
                        pruef.setStatus(0);
                        /* Prüfung des aktuellen Bauteils beginnen */
                        pruef.pruefe();
                        /* Schritt erniedrigen, d.h. auf vorheriges Bauteil
springen */
                        pruef.erniedrige_zaehler();
```

```
                        /* Aktuelles Bauteil gelb einfärben */
                        pruef.setStatus(1);
                        /* Prüfung des Bauteils beginnen */
                        pruef.pruefe();
                        /* Fortschrittsbalken setzen */
                        pProgress.setValue(((pruef.getZaehler() + 1) * (100
/ schritte_max)));
                        /* Layout neu zeichnen */
                        hd.aktualisieren();
                        /*
                         * Buttons aktivieren, um festzustellen, ob Bauteil
                         * korrekt/fehlerhaft ist
                         */
                        bOK.setEnabled(true);
                        bNOT.setEnabled(true);
                        /*
                         * Stop der Prüfung erst möglich, wenn Bauteil
identifiziert
                         * wurde
                         */
                        bStop.setEnabled(false);
                  }
            });

            /**
             * Aktion-Listener des Buttons "OK"
             *
             * Durch Klick auf diesen Button wird das Bauteil als korrekt
             * identifiziert
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            bOK.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /* Aktuelles Bauteil grün einfärben */
                        pruef.setStatus(2);
                        /* Prüfung des aktuellen Bauteils beginnen */
                        pruef.pruefe();
                        /* Vergrößerungs-/Verkleinerungsfaktor zurücksetzen
*/
                        hd.scale_reset();
                        /* Fortschrittsbalken setzen */
                        pProgress.setValue(((pruef.getZaehler() + 1) * (100
/ schritte_max)));
                        /* Layout neu zeichnen */
                        hd.aktualisieren();
                        /* Steuerungsbuttons aktivieren */
                        bNext.setEnabled(true);
                        bPrev.setEnabled(true);
                        bStop.setEnabled(true);

                        /* Abfrage, ob aktuelles Bauteil auch letztes
Bauteil ist */
                        if ((pruef.getZaehler() + 1) ==
pruef.getZaehler_max()) {
                              /*
```

```
                                * Dialog sichtbar machen, um abzufragen, ob
gespeichert
                                * werden soll
                                */
                               int result =
JOptionPane.showConfirmDialog(null,
                                                "Prüfung speichern?",
"Speichern",

      JOptionPane.YES_NO_CANCEL_OPTION);
                               /* Speichern... */
                               if (result == JOptionPane.YES_OPTION) {
                                     /*
                                      * Methodenaufruf speichern(), um Daten
in Datenbank zu
                                      * schreiben
                                      */
                                     speichern();
                                     /* Steuerungsbuttons deaktivieren */
                                     bPrev.setEnabled(false);
                                     bNext.setEnabled(false);
                                     bOK.setEnabled(false);
                                     bNOT.setEnabled(false);
                                     /* Prüfung kann neu gestartet werden */
                                     bStart.setEnabled(true);
                                     /* Prüfung kann gespeichert werden */
                                     iSave.setEnabled(true);
                                     /* Status der Bauteile zurücksetzen */
                                     pruef.reset_status();
                                     /* Zähler zurücksetzen */
                                     pruef.reset_zaehler();
                                     /* Neue Prüfung = neue Leiterplatte */
                                     neu = true;
                               }
                               /*
                                * Nicht speichern... --> gleiche Prozedur
ohne
                                * Methodenaufruf speichern()
                                */
                               else if (result == JOptionPane.NO_OPTION) {
                                     bPrev.setEnabled(false);
                                     bNext.setEnabled(false);
                                     bOK.setEnabled(false);
                                     bNOT.setEnabled(false);
                                     bStart.setEnabled(true);
                                     iSave.setEnabled(true);
                                     pruef.reset_status();
                                     pruef.reset_zaehler();
                                     neu = true;
                               }
                          } else {
                               /* Abfrage, ob nicht nach Bauteil gesucht
wurde */
                               if (!find) {
                                     /* Zähler erhöhen --> nächstes Bauteil
*/
                                     pruef.erhoehe_zaehler();
                                     /* Bauteil gelb einfärben */
```

```
                                    pruef.setStatus(1);
                                    /* Prüfung beginnen */
                                    pruef.pruefe();
                              }
                              /* Abfrage, ob nach Bauteil gesucht wurde */
                              else if (find) {
                                    /* Zähler zurücksetzen */
                                    pruef.reset_zaehler();
                                    /* Steuerungsbuttons deaktivieren */
                                    bOK.setEnabled(false);
                                    bNOT.setEnabled(false);
                                    bNext.setEnabled(false);
                                    bPrev.setEnabled(false);
                              }
                        }
                  }
            });

            /**
             * Aktion-Listener des Buttons "NOT"
             *
             * Durch Klick auf diesen Button wird das Bauteil als
fehlerhaft
             * identifiziert
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            bNOT.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        /*
                         * Gleiche Prozedur, wie AktionLstener des Buttons
OK. Hier wird
                         * das Bauteil jedoch nicht grün eingefärbt,
sondern rot, indem
                         * der Status nun auf 3 gesetzt wird, statt auf 2
                         */
                        String fehler = JOptionPane.showInputDialog(null,
                                    "Bitte Art des Fehlers angeben",
"Fehlerbeschreibung",
                                    JOptionPane.PLAIN_MESSAGE);
                        Pruefung.fehler = true;
                        pruef.setBeschreibung(fehler);
                        pruef.setStatus(3);
                        pruef.pruefe();
                        hd.scale_reset();
                        pProgress.setValue(((pruef.getZaehler() + 1) * (100
/ schritte_max)));
                        hd.aktualisieren();
                        bNext.setEnabled(true);
                        bPrev.setEnabled(true);
                        bStop.setEnabled(true);
                        if ((pruef.getZaehler() + 1) ==
pruef.getZaehler_max()) {
                              int result =
JOptionPane.showConfirmDialog(null,
                                          "Prüfung speichern?",
```

```
"Speichern",

      JOptionPane.YES_NO_CANCEL_OPTION);
                              if (result == JOptionPane.YES_OPTION) {
                                    speichern();
                                    bPrev.setEnabled(false);
                                    bNext.setEnabled(false);
                                    bOK.setEnabled(false);
                                    bNOT.setEnabled(false);
                                    bStart.setEnabled(true);
                                    iSave.setEnabled(true);
                                    pruef.reset_status();
                                    pruef.reset_zaehler();
                                    neu = true;
                              } else if (result == JOptionPane.NO_OPTION) {
                                    bPrev.setEnabled(false);
                                    bNext.setEnabled(false);
                                    bOK.setEnabled(false);
                                    bNOT.setEnabled(false);
                                    bStart.setEnabled(true);
                                    iSave.setEnabled(true);
                                    pruef.reset_status();
                                    pruef.reset_zaehler();
                                    neu = true;
                              }
                        } else {
                              if (!find) {
                                    pruef.erhoehe_zaehler();
                                    pruef.setStatus(1);
                                    pruef.pruefe();
                              } else if (find) {
                                    pruef.reset_zaehler();
                                    bOK.setEnabled(false);
                                    bNOT.setEnabled(false);
                                    bNext.setEnabled(false);
                                    bPrev.setEnabled(false);
                              }
                        }
                  }
            });

            /**
             * Aktion-Listener der Checkbox "Automatischer Test"
             *
             * Durch Aktivieren dieser Checkbox, können in Zukunft alle
Bauteile
             * voll-automatisch geprüft werden.
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            rAuto.addItemListener(new ItemListener() {
                  public void itemStateChanged(ItemEvent e) {
                        /* Abfrage, ob Checkbox selektiert wurde */
                        if (e.getStateChange() == ItemEvent.SELECTED) {
                              /* Textfeld deaktivieren */
                              tSchritt_manu.setEnabled(false);
```

```
                        }
                  }
            });

            /**
             * Aktion-Listener der Checkbox "Manueller Test"
             *
             * Durch Aktivieren dieser Checkbox, können in Zukunft alle
Bauteile
             * manuell und einzeln geprüft werden.
             *
             * @author Patrick Olma
             * @version 30.06.2011
             *
             */
            rManu.addItemListener(new ItemListener() {
                  public void itemStateChanged(ItemEvent e) {
                        /* Abfrage, ob Checkbox selektiert wurde */
                        if (e.getStateChange() == ItemEvent.SELECTED) {
                              /* Textfeld aktivieren */
                              tSchritt_manu.setEnabled(true);
                        }
                  }
            });

            /* Panels dem GridBagLayout hinzufügen */
            addComponent(c, gbl, pNorth, 0, 0, 4, 1, 1.0, 0);
            addComponent(c, gbl, pCenter, 0, 1, 2, 1, 4.0, 1.0);
            addComponent(c, gbl, pEast, 2, 1, 1, 1, 1.0, 1.0);
            addComponent(c, gbl, pSouth, 0, 3, 4, 1, 1.0, 0);

            /* Packen */
            pack();
            /* Größe angeben */
            setSize(1024, 768);
            /* GUI sichtbar machen */
            setVisible(true);
            /* Position des Dialogs auf dem Bildschirm. Hier: Mitte des
Desktops */
            setLocation(
                        (Toolkit.getDefaultToolkit().getScreenSize().width
- getSize().width) / 2,
                        (Toolkit.getDefaultToolkit().getScreenSize().height
- getSize().height) / 2);

      }

      /**
       * Methode speichern() um alle entstandenen Daten in die Datenbank zu
       * schreiben
       *
       * @author Patrick Olma
       * @version 30.06.2011
       */
      public void speichern() {
            /*
             * ArrayListen für die Daten der Leiterplatten bzw. Bauteile,
welche
```

```
		 * bereits mit den vorhandenen Daten gefüllt werden
		 */
		ArrayList<PCB> aPCB = Daten.getAPCB();
		ArrayList<Placement> aPlace_top = Daten.getAPlace_top();
		ArrayList<Placement> aPlace_bottom = Daten.getAPlace_bottom();

		try {
			/* JDBC-Treiber laden */
			Class.forName("com.mysql.jdbc.Driver");
		} catch (ClassNotFoundException e) {
			System.out.println("Kann nicht geladen werden");
			return;
		}
		try {
			/* Verbindung mit Datenbank aufnehmen */
			Connection con = DriverManager

	.getConnection("jdbc:mysql://localhost:3306/pcb_viewer",
							"root", "systemv");
			Statement stmt = con.createStatement();
			String input;
			/* Abfrage, ob Leiterplatte neu ist */
			if (neu) {
				/*
				 * Komplette ArrayList der PCB durchlaufen und
Daten in
				 * Datenbank schreiben
				 */
				for (int i = 0; i < aPCB.size(); i++) {
					input = "insert into pcb (pcb_seriennummer,
pcb_sachnummer, breite, hoehe, dicke, bezeichnung, zaehler, pruefer, date)
values (\""
							+ GUI.lSNR_input.getText()
							+ "\", \""
							+ aPCB.get(i).getSachnummer()
							+ "\","
							+ aPCB.get(i).getBreite()
							+ ", \""
							+ aPCB.get(i).getHoehe()
							+ "\", \""
							+ aPCB.get(i).getDicke()
							+ "\", \""
							+ aPCB.get(i).getBezeichnung()
							+ "\", \""
							+ ((pruef.getZaehler()) - 1)
							+ "\", \""
							+ lUser_input.getText()
							+ "\", \""
							+ new java.sql.Date(new
java.util.Date().getTime())
							+ "\")";
					stmt.executeUpdate(input);
				}
				/* Abfrage, ob Bestückfile der Top-Seite eingelesen
wurde */
				if (GUI.top) {
					/*
					 * Komplette ArrayList der Bauteile
```

```
durchlaufen und Daten in
                                * Datenbank schreiben
                                */
                               for (int i = 0; i < aPlace_top.size(); i++) {
                                     input = "insert into placement
(pcb_seriennummer, cmp_nr, x, y, rotation, cmp_name, status, side,
beschreibung) values (\""
                                                 + GUI.lSNR_input.getText()
                                                 + "\", \""
                                                 +
aPlace_top.get(i).getCmp_nr()
                                                 + "\","
                                                 + aPlace_top.get(i).getX()
                                                 + ", \""
                                                 + aPlace_top.get(i).getY()
                                                 + "\", \""
                                                 +
aPlace_top.get(i).getRotation()
                                                 + "\", \""
                                                 +
aPlace_top.get(i).getCmp_name()
                                                 + "\", \""
                                                 +
aPlace_top.get(i).getStatus()
                                                 + "\", \"to\", \""
                                                 +
aPlace_top.get(i).getBeschreibung() + "\")";
                                     stmt.executeUpdate(input);
                               }
                          }
                          /* Abfrage, ob Bestückfile der Bottom-Seite
eingelesen wurde */
                          if (GUI.bottom) {
                               /* Gleiche Prozedur, als bei Top-Seite */
                               for (int i = 0; i < aPlace_bottom.size();
i++) {
                                     input = "insert into placement
(pcb_seriennummer, cmp_nr, x, y, rotation, cmp_name, status, side,
beschreibung) values (\""
                                                 + GUI.lSNR_input.getText()
                                                 + "\", \""
                                                 +
aPlace_bottom.get(i).getCmp_nr()
                                                 + "\","
                                                 +
aPlace_bottom.get(i).getX()
                                                 + ", \""
                                                 +
aPlace_bottom.get(i).getY()
                                                 + "\", \""
                                                 +
aPlace_bottom.get(i).getRotation()
                                                 + "\", \""
                                                 +
aPlace_bottom.get(i).getCmp_name()
                                                 + "\", \""
                                                 +
aPlace_bottom.get(i).getStatus()
```

```
                                                + "\", \"bo\", \""
                                                +
aPlace_bottom.get(i).getBeschreibung()
                                                + "\")";
                                          stmt.executeUpdate(input);
                                    }
                              }
                        }
                        /* Leiterplatte war nicht neu --> wurde geöffnet */
                        else {
                              /*
                               * Komplette ArrayList der PCB durchlaufen und
Daten in
                               * Datenbank aktualisieren
                               */
                              for (int i = 0; i < aPCB.size(); i++) {
                                    input = "update pcb set pcb_sachnummer=\""
                                                + aPCB.get(i).getSachnummer() +
"\", breite=\""
                                                + aPCB.get(i).getBreite() + "\",
hoehe=\""
                                                + aPCB.get(i).getHoehe() + "\",
dicke=\""
                                                + aPCB.get(i).getDicke() + "\",
bezeichnung=\""
                                                + aPCB.get(i).getBezeichnung() +
"\", zaehler=\""
                                                + ((pruef.getZaehler()) - 1) +
"\", pruefer=\""
                                                + lUser_input.getText() + "\",
date=\""
                                                + new java.sql.Date(new
java.util.Date().getTime())
                                                + "\" where pcb_seriennummer='"
                                                + lSNR_input.getText() + "'";
                                    stmt.executeUpdate(input);
                              }
                              /* Abfrage, ob Top-Seite der Leiterplatte geöffnet
wurde */
                              if (GUI.top) {
                                    /*
                                     * Komplette ArrayList der Bauteile
durchlaufen und Daten in
                                     * Datenbank aktualisieren
                                     */
                                    for (int i = 0; i < aPlace_top.size(); i++) {
                                          input = "update placement set
status=\""
                                                +
aPlace_top.get(i).getStatus()
                                                + "\", beschreibung=\""
                                                +
aPlace_top.get(i).getBeschreibung()
                                                + "\" where
pcb_seriennummer=\""
                                                + lSNR_input.getText() +
"\" and cmp_nr=\""
                                                +
```

```
aPlace_top.get(i).getCmp_nr()
                                                + "\" and cmp_name=\""
                                                +
aPlace_top.get(i).getCmp_name() + "\"";
                                    stmt.executeUpdate(input);
                              }
                        }
                        /* Abfrage, ob Bottom-Seite der Leiterplatte
geöffnet wurde */
                        if (GUI.bottom) {
                              /* Gleiche Prozedur, als bei Top-Seite */
                              for (int i = 0; i < aPlace_bottom.size();
i++) {
                                    input = "update placement set
status=\""
                                                +
aPlace_bottom.get(i).getStatus()
                                                + "\", beschreibung=\""
                                                +
aPlace_bottom.get(i).getBeschreibung()
                                                + "\" where
pcb_seriennummer=\""
                                                + lSNR_input.getText() +
"\" AND cmp_nr=\""
                                                +
aPlace_bottom.get(i).getCmp_nr()
                                                + "\" and cmp_name=\""
                                                +
aPlace_top.get(i).getCmp_name() + "\"";
                                    stmt.executeUpdate(input);
                              }
                        }
                  }
                  stmt.close();
                  con.close();
            } catch (SQLException se) {
                  JOptionPane.showMessageDialog(null, se.getMessage(),
"Fehler",
                              JOptionPane.ERROR_MESSAGE);
                  se.printStackTrace(System.out);
            }
      }

      /**
       * Methode oeffnen() um bereits gespeicherte Daten wieder aus der
Datenbank
       * auszulesen.
       *
       * @author Patrick Olma
       * @version 30.06.2011
       * @param String
       *            der Seriennummer
       */
      public void oeffnen(String SNR) {
            /*
             * ArrayListen für die Daten der der Leiterplatte, welche
bereits mit
             * den vorhandenen Daten gefüllt werden
```

```
             */
            ArrayList<PCB> aLeiterplatte = Daten.getAPCB();
            ArrayList<Placement> aPlacement_top = Daten.getAPlace_top();
            ArrayList<Placement> aPlacement_bottom =
Daten.getAPlace_bottom();

            /* ArrayListen leeren */
            aLeiterplatte.clear();
            aPlacement_top.clear();
            aPlacement_bottom.clear();

            /* Variable, um anzugeben, dass Daten vorhanden sind */
            vorhanden = false;
            try {
                  /* JDBC-Treiber laden */
                  Class.forName("com.mysql.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                  System.out.println("Kann nicht geladen werden");
                  return;
            }
            try {
                  /* Verbindung mit Datenbank aufnehmen */
                  Connection con = DriverManager

      .getConnection("jdbc:mysql://localhost:3306/pcb_viewer",
                                          "root", "systemv");
                  Statement stmt = con.createStatement();
                  /* Selektion der Daten, anhand der eindeutigen
Seriennummer */
                  String pcb = "Select * from pcb where
pcb_seriennummer=\"" + SNR
                              + "\"";
                  ResultSet result_pcb = stmt.executeQuery(pcb);
                  /* Abfrage, ob Daten vorhanden sind */
                  if (result_pcb.next()) {
                        /* Neues Objekt der Klasse PCB anlegen */
                        PCB leiterplatte = new PCB(
                                    result_pcb.getString("pcb_sachnummer"),

      Double.parseDouble(result_pcb.getString("breite")),

      Double.parseDouble(result_pcb.getString("hoehe")),

      Double.parseDouble(result_pcb.getString("dicke")),
                                    result_pcb.getString("bezeichnung"));
                        /* Objekt in ArrayList hinzufügen */
                        aLeiterplatte.add(leiterplatte);
                        /* Prüfer in entsprechendes Textfeld der GUI
schreiben */

      lUser_input.setText(result_pcb.getString("pruefer"));
                        pruef.setZaehler((Integer.parseInt(result_pcb
                                    .getString("zaehler"))) + 1);
                        /* Sachnummer in entsprechendes Textfeld der GUI
schreiben */

      lPCB_input.setText(result_pcb.getString("pcb_sachnummer"));
                        /* Seriennummer in entsprechendes Textfeld der GUI
```

```
schreiben */
                        lSNR_input.setText(SNR);
                        /* Daten sind vorhanden */
                        vorhanden = true;
                  } else {
                        JOptionPane.showMessageDialog(null,
                                    "Datensatz nicht vorhanden", "Fehler",
                                    JOptionPane.ERROR_MESSAGE);
                  }
                  /* Daten vorhanden... */
                  if (vorhanden) {
                        /* Selektion der Bauteil-Daten */
                        String place_top = "Select * from placement where
pcb_seriennummer=\""
                                    + SNR + "\" and side='to'";
                        ResultSet result_place_top =
stmt.executeQuery(place_top);
                        /* Für jede Zeile... */
                        while (result_place_top.next()) {
                              /* Neues Objekt der Klasse Placement anlegen
(Top-Seite) */
                              Placement place_to = new Placement();
                              /* Klassenvariablen setzen */

      place_to.setCmp_nr(result_place_top.getString("cmp_nr"));

      place_to.setX(Double.parseDouble(result_place_top
                                    .getString("x")));

      place_to.setY(Double.parseDouble(result_place_top
                                    .getString("y")));

      place_to.setRotation(Double.parseDouble(result_place_top
                                    .getString("rotation")));

      place_to.setCmp_name(result_place_top.getString("cmp_name"));

      place_to.setStatus(Integer.parseInt(result_place_top
                                    .getString("status")));
                              place_to.setBeschreibung(result_place_top
                                    .getString("beschreibung"));
                              /* Objekte der ArrayList hinzufügen */
                              aPlacement_top.add(place_to);
                              /* Top-Seite wurde eingelesen */
                              top = true;
                              /*
                               * Maximale Anzahl an Prüfschritten = Anzahl
der Objekte in
                               * ArrayList
                               */
                              pruef.setZaehler_max(aPlacement_top.size());
                        }

                        /* Gleiche Prozedur wie Top-Seite für Bottom-Seite
*/
                        String place_bottom = "Select * from placement
where pcb_seriennummer=\""
                                    + SNR + "\" and side='bo'";
```

```
                        ResultSet result_place_bottom =
stmt.executeQuery(place_bottom);
                        while (result_place_bottom.next()) {
                              Placement place_bo = new Placement();

      place_bo.setCmp_nr(result_place_top.getString("cmp_nr"));

      place_bo.setX(Double.parseDouble(result_place_top
                                    .getString("x")));

      place_bo.setY(Double.parseDouble(result_place_top
                                    .getString("y")));

      place_bo.setRotation(Double.parseDouble(result_place_top
                                    .getString("rotation")));

      place_bo.setCmp_name(result_place_top.getString("cmp_name"));

      place_bo.setStatus(Integer.parseInt(result_place_top
                                    .getString("status")));
                              place_bo.setBeschreibung(result_place_top
                                    .getString("beschreibung"));

                              aPlacement_bottom.add(place_bo);
                              bottom = true;

      pruef.setZaehler_max(aPlacement_bottom.size());
                        }
                  }
                  stmt.close();
                  con.close();
            } catch (SQLException se) {
                  JOptionPane.showMessageDialog(null, se.getMessage(),
"Fehler",
                              JOptionPane.ERROR_MESSAGE);
                  se.printStackTrace(System.out);
            }
            /* Steuerungsbuttons aktivieren */
            bOK.setEnabled(true);
            bNOT.setEnabled(true);
            bNext.setEnabled(true);
            bPrev.setEnabled(true);
            /* Ansicht standard auf top setzen */
            setSide("top");
            /* Leiterplatte wurde geöffnet */
            neu = false;
            /* Leiterplatten-Daten korrekt angegeben */
            GUI_pcb.pcb_ready = true;
            /* Layout in pCenter zeichnen */
            GUI.pCenter.add(GUI.hd);
      }

      /**
       * Deklaration der set- bzw. get-Methoden
       *
       * @author Patrick Olma
       * @version 30.06.2011
       * @param Übergabewerte
```

```
         *            für die jeweilige Klassenvariablen
         * @return jeweilige Klassenvariablen
         */
        public void setSide(String side) {
              GUI.side = side;
        }

        public static String getSide() {
              return side;
        }
}
```

8.11 Layout.java

```
/**
 * Klasse für das Layout.
 *
 * Diese Klassse beinhaltet Funktion, um das Layout der Leiterplatte
 * auf die GUI zu zeichnen.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.util.ArrayList;
import java.util.Iterator;
import javax.swing.JPanel;

public class Layout extends JPanel {

	/* Deklaration der Klassenvariablen */
	private static final long serialVersionUID = 1L;

	/*
	 * ArrayList für die Daten der Leiterplatte, welche bereits mit den
	 * vorhandenen Daten gefüllt werden
	 */
	ArrayList<PCB> aLeiterplatte = Daten.getAPCB();
	/* ArrayList für die Daten der Bauteile */
	ArrayList<Placement> aPlacement;
	/* ArrayList für die Geometrie-Daten der Bauteile */
	ArrayList<Comp> aComp;

	double PCB_x;
	double PCB_y;
	double PCB_pitch_x;
	double PCB_pitch_y;

	double Place_x;
	double Place_y;
	double rotation;
	double rotation_pcb = 0;

	double xmin = 0;
	double xmax = 0;
	double ymin = 0;
	double ymax = 0;

	int deltax = 0;
	int deltay = 0;

	boolean trans = false;

	int width = 650;
```

```
	int width_max = 790;
	int height = 550;
	int height_max = 640;
	int faktor = 1;
	int faktor_scale = 1;
	int status;

	String nr;

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 *
	 */
	Layout() {
		/* Größe angeben */
		setSize(width_max, height_max);
		/* Layout sichtbar machen */
		setVisible(true);
		/* Position im Panel angeben */
		setLocation(10, 10);
		/* Hintergrundfarbe auf Weiss setzen */
		setBackground(Color.WHITE);
	}

	/**
	 * Methode zeichneLayout() um das Layout auf die GUI zu zeichnen
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Graphics
	 *            Objekt g
	 */
	protected void zeichneLayout(Graphics g) {

		/* Abfrage, ob Ansicht --> Top gewählt wurde */
		if (GUI.getSide() == "top") {
			/* ArrayListen füllen */
			aPlacement = Daten.getAPlace_top();
			aComp = Daten.getaComp_top();
		}
		/* Abfrage, ob Ansicht --> Bottom gewählt wurde */
		else if (GUI.getSide() == "bottom") {
			/* ArrayListen füllen */
			aPlacement = Daten.getAPlace_bottom();
			aComp = Daten.getaComp_bottom();
		}

		/* Farbe auf Schwarz setzen */
		g.setColor(Color.BLACK);

		/*
		 * Temporäre Arrays yTemp, xTemp erstellen, um nach x-/y-
Koordinaten zu
		 * sortieren
		 */
```

```
		double[] yTemp = new double[aPlacement.size()];
		double[] xTemp = new double[aPlacement.size()];
		/* Komplette ArrayList durchlaufen */
		for (int j = 0; j < aPlacement.size(); j++) {
			/* y-Koordinaten in Array schreiben */
			yTemp[j] = aPlacement.get(j).getY();
		}
		/* Komplette ArrayList durchlaufen */
		for (int k = 0; k < aPlacement.size(); k++) {
			/* x-Koordninaten in Array schreiben */
			xTemp[k] = aPlacement.get(k).getX();
		}

		/* Arrays sortieren */
		java.util.Arrays.sort(yTemp);
		java.util.Arrays.sort(xTemp);

		/* Iterator anlegen, um durch ArrayList zu laufen */
		Iterator<Placement> iterPlace = aPlacement.iterator();
		/* Solange Objekt vorhanden ist... */
		while (iterPlace.hasNext()) {
			Object o = iterPlace.next();

			/* Minimale Koordinaten x und y */
			xmin = xTemp[0];
			ymin = yTemp[0];
			/* Maximale Koordinaten x und y */
			xmax = xTemp[aPlacement.size() - 1];
			ymax = yTemp[aPlacement.size() - 1];

			/* Breite der Leiterplatte */
			PCB_x = ((xmax - xmin));
			/* Höhe der Leiterplatte */
			PCB_y = ((ymax - ymin));

			/* Abfrage, ob keine Transformierung vorhanden ist */
			if (!trans) {
				/* Faktor setzen */
				faktor = (width / ((int) PCB_x));
				/*
				 * Abfrage ob Breite * Faktor größer ist als
maximale breite des
				 * Panels, damit Layout nicht über Rand hinaus
gezeichnet wird
				 */
				if (((int) PCB_x * faktor) > width) {
					/* Solange größer... */
					while (((int) PCB_x * faktor) > width) {
						/* Faktor um 1 verringern */
						faktor--;
					}
				}
				/* Gleiche Prozedur wie Breite */
				else if (((int) PCB_y * faktor) > height) {
					while (((int) PCB_y * faktor) > height) {
						faktor--;
					}
				}
```

```
                    }

                    /*
                     * Rechteck in die Mitte des Panels zeichnen mit
Breite*Faktor und
                     * Höhe*Faktor. Dieses Rechteck steht für die äußere Form
der
                     * Leiterplatte
                     */
                    g.drawRect(
                                    ((width_max / 2) - ((((int) PCB_x) * faktor +
(faktor + 50)) / 2)),
                                    ((height_max / 2) - ((((int) PCB_y) * faktor
+ (faktor + 50)) / 2)),
                                    ((((int) PCB_x) * faktor) + (faktor + 50)),
                                    ((((int) PCB_y) * faktor) + (faktor + 50)));

                    /*
                     * Pitch in x- und y-Richtung berechnen. Pitch bezeichnet
den
                     * Abstand vom Mittelpunkt der Leiterplatte bis zu den
äußeren
                     * Ecken.
                     */
                    PCB_pitch_x = (((width_max / 2) - ((int) PCB_x * faktor)
/ 2) - (xmin * faktor));
                    PCB_pitch_y = (((height_max / 2) - ((int) PCB_y * faktor)
/ 2) - (ymin * faktor));

                    /* Koordinate x des Bauteils einlesen */
                    Place_x = ((Placement) o).getX();
                    /* Koordinate y des Bauteils einlesen */
                    Place_y = ((Placement) o).getY();
                    /* Rotation des Bauteils einlesen */
                    rotation = ((Placement) o).getRotation();
                    /* Status des Bauteils einlesen */
                    status = ((Placement) o).getStatus();
                    /* Nummer des Bauteils einlesen */
                    nr = ((Placement) o).getCmp_nr();

                    /* Komplette ArrayList der Geometrie-Daten durchlaufen */
                    for (int y = 0; y < aComp.size(); y++) {
                          /*
                           * Abfrage, ob Objekt in Geometrie-Daten-ArrayList
gleich dem
                           * Objekt der Placement-ArrayList ist
                           */
                          if (aComp.get(y).getCmp_nr()
                                        .equals(((Placement) o).getCmp_nr())) {
                                /* Aktuelle Geometrie für dieses Bauteil
zeichnen */
                                aComp.get(y).draw(g, PCB_pitch_x + (Place_x *
faktor),
                                            (PCB_pitch_y + (Place_y *
faktor)), faktor, nr,
                                            rotation, ((Placement)
o).getCmp_name(), status);
                          }
```

```
                }
            }
        }

        /**
         * Methode paintComponent()
         *
         * Std. Methode der Klasse Graphics, welche hier überschrieben wird
         *
         * @author Patrick Olma
         * @version 30.06.2011
         * @param Graphics
         *            Objekt g
         */
        public void paintComponent(Graphics g) {
            /* Cast auf Graphics2d */
            Graphics2D g2 = (Graphics2D) g;
            /* Aufruf der Methode der Super-Klasse (Graphics) */
            super.paintComponent(g);
            /* Abfrage, ob Ansicht --> Top gewählt wurde */
            if (GUI.getSide() == "top") {
                /* String auf GUI zeichnen */
                g.drawString("Top-Side View", 10, 10);
            }
            /* Abfrage, ob Ansicht --> Bottom gewählt wurde */
            else if (GUI.getSide() == "bottom") {
                /* String auf GUI zeichnen */
                g.drawString("Bottom-Side View", 10, 10);
            }
            /* Farbe schwarz zuweisen */
            g2.setColor(Color.BLACK);
            /* Komplettes Layout rotieren, abhängig von rotation_pcb um
Mittelpunkt */
            g2.rotate(rotation_pcb * Math.PI / 180, width_max / 2,
height_max / 2);
            /* Verschiebung des kompletten Layouts in x und y Richtung */
            g2.translate(deltax, deltay);
            /* Aufruf der Methode zeichneLayout() um Layout zu zeichnen */
            zeichneLayout(g2);
        }

        /**
         * Methode aktualisieren() um das Layout neu zu zeichnen
         *
         * @author Patrick Olma
         * @version 30.06.2011
         */
        public void aktualisieren() {
            /* Aufruf der repaint()-Funktion, geerbt von Graphics */
            repaint();
        }

        /**
         * Methode rotate() um das Layout zu rotieren
         *
         * @author Patrick Olma
         * @version 30.06.2011
         * @param Wert
```

```
	 *            der rotation
	 */
	public void rotate(double rotation) {
		/* Klassenvariable setzen */
		rotation_pcb = rotation_pcb + rotation;
	}

	/**
	 * Methode scale_groß() um das Layout zu vergrößern
	 *
	 * Dabe wird das aktuelle Bauteil in die Mitte der GUI verschoben.
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Zaehler
	 *            des aktuellen Bauteils
	 */
	public void scale_groß(int zaehler) {
		/* Abstand des aktuellen Bauteils zum Mittelpunkt der
Leiterplatte in x */
		deltax = (int) ((width_max / 2) - (((PCB_pitch_x + ((aPlacement
				.get(zaehler).getX()) * faktor)))));
		/* Abstand des aktuellen Bauteils zum Mittelpunkt der
Leiterplatte in y */
		deltay = (int) ((height_max / 2) - (((PCB_pitch_y +
((aPlacement
				.get(zaehler).getY()) * faktor)))));
		/* Layout wurde transformiert... */
		trans = true;
		/* Faktor erhöhen */
		faktor = faktor + 2;
		/* Layout neu zeichnen */
		aktualisieren();
	}

	/**
	 * Methode scale_klein() um das Layout zu verkleinern.
	 *
	 * Dabe wird das aktuelle Bauteil in die Mitte der GUI verschoben.
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Zaehler
	 *            des aktuellen Bauteils
	 */
	public void scale_klein(int zaehler) {
		/* Abstand des aktuellen Bauteils zum Mittelpunkt der
Leiterplatte in x */
		deltax = (int) ((width_max / 2) - (((PCB_pitch_x + ((aPlacement
				.get(zaehler).getX()) * faktor)))));
		/* Abstand des aktuellen Bauteils zum Mittelpunkt der
Leiterplatte in y */
		deltay = (int) ((height_max / 2) - (((PCB_pitch_y +
((aPlacement
				.get(zaehler).getY()) * faktor)))));
		/* Layout wurde transformiert... */
		trans = true;
		/* Faktor verringern */
```

```
            faktor = faktor - 2;
            /* Layout neu zeichnen */
            aktualisieren();
      }

      /**
       * Methode scale_reset() um die Transformationen zurückzusetzen.
       *
       * @author Patrick Olma
       * @version 30.06.2011
       */
      public void scale_reset() {
            /* Keine Verschiebung in x und y */
            deltax = 0;
            deltay = 0;
            /* Keine Transformation */
            trans = false;
            /* Layout neu zeichnen */
            aktualisieren();
      }
}
```

8.12 Line.java

```
/**
 * Klasse für die Objekte und Methoden der Klasse (Geometrie) Line.
 *
 * Die einzelnen Objekte dieser Klasse werden innerhalb der Klasse
 * GUI_comp in Form von Geometrie-Libraries eingelesen und instantiiert.
 * Line repräsentiert innerhalb dieser Geometrien Geraden von Anfangs zu
Endpunkten
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

public class Line {

	/* Deklaration der Klassenvariablen */
	private double x1;
	private double x2;
	private double y1;
	private double y2;

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Anfangspunkt
	 *            (x1,y1) und Endpunkt (x2,y2)
	 */
	public Line(double x1, double x2, double y1, double y2) {
		this.x1 = x1;
		this.x2 = x2;
		this.y1 = y1;
		this.y2 = y2;
	}

	/**
	 * Deklaration der set- bzw. get-Methoden
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Übergabewerte
	 *            für die jeweilige Klassenvariablen
	 * @return jeweilige Klassenvariablen
	 */
	public void setX1(double x1) {
		this.x1 = x1;
	}

	public double getX1() {
		return x1;
	}

	public void setX2(double x2) {
```

```
            this.x2 = x2;
      }

      public double getX2() {
            return x2;
      }

      public void setY1(double y1) {
            this.y1 = y1;
      }

      public double getY1() {
            return y1;
      }

      public void setY2(double y2) {
            this.y2 = y2;
      }

      public double getY2() {
            return y2;
      }
}
```

8.13 PCB.java

```
/**
 * Klasse für die Objekte, welche die Leiterplatte repräsentieren.
 *
 * Eine Leiterplatte wird hierbei repräsentiert durch ihre Sachnummer, die
Abmessungen
 * und deren Bezeichnung
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

public class PCB {

	/* Deklaration der Klassenvariablen */
	private String Sachnummer;
	private double breite;
	private double hoehe;
	private double dicke;
	private String bezeichnung;

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Sachnummer, Abmessungen (breite, hoehe, dicke) und
Bezeichnung
	 */
	public PCB(String Sachnummer, double breite, double hoehe, double
dicke,
			String bezeichnung) {
		this.Sachnummer = Sachnummer;
		this.breite = breite;
		this.hoehe = hoehe;
		this.dicke = dicke;
		this.bezeichnung = bezeichnung;
	}

	/**
	 * Deklaration der set- bzw. get-Methoden
	 * @author  Patrick Olma
	 * @version  30.06.2011
	 * @param Übergabewerte  für die jeweilige Klassenvariablen
	 * @return  jeweilige Klassenvariablen
	 */
	public void setSachnummer(String Sachnummer) {
		this.Sachnummer = Sachnummer;
	}

	public String getSachnummer() {
		return Sachnummer;
	}
```

```
	public void setBreite(double breite) {
		this.breite = breite;
	}

	public double getBreite() {
		return breite;
	}

	public void setHoehe(double hoehe) {
		this.hoehe = hoehe;
	}

	public double getHoehe() {
		return hoehe;
	}

	public void setBezeichnung(String bezeichnung) {
		this.bezeichnung = bezeichnung;
	}

	public String getBezeichnung() {
		return bezeichnung;
	}

	public void setDicke(double dicke) {
		this.dicke = dicke;
	}

	public double getDicke() {
		return dicke;
	}
}
```

8.14 Placement.java

```
/**
 * Klasse für die Objekte, welche die einzelnen Bauteile
 * auf der Leiterplatte repräsentieren.
 *
 * Ein Bauteil wird hierbei repräsentiert durch dessen eindeutige
Seriennummer,
 * die Abmessungen, die Rotation auf der Leiterplatte, dessen Name, der
Status der
 * Prüfung, dessen Name und der Fehler-Beschreibung.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

public class Placement {

	/* Deklaration der Klassenvariablen */
	private String cmp_nr;
	private double x;
	private double y;
	private double rotation;
	private String cmp_name;
	private int status;
	private String beschreibung;

	/**
	 * Methode setValue(), um jede Zeile des Bestückfiles zu trennen und
die
	 * Klassenvariablen zu setzen.
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Zeile
	 *            des Bestückfiles
	 *
	 */

	/*
	 * Methode für die Top-Seite einer Leiterplatte Hier wird im
gegensatz zur
	 * Bottom-Seite an der y-Achse gespiegelt. Grund hierfür ist, dass
beim
	 * Layouten durch die Bottom-Seite "hindurch" geschaut wird, d.h. die
	 * Top-Seite spiegelverkehrt ist.
	 */
	public void setValue_top(String input) {
		/* Zeilen durch Trennzeichen ";" trennen und Klassenvariablen
setzen */
		int i, k;
		i = input.indexOf(";");
		cmp_nr = input.substring(0, i);
		k = input.indexOf(";", i + 1);
		x = Double.parseDouble(input.substring(i + 1, k));
```

```
            i = k;
            k = input.indexOf(";", i + 1);
            y = -(Double.parseDouble(input.substring(i + 1, k)));
            i = k;
            k = input.indexOf(";", i + 1);
            rotation = Double.parseDouble(input.substring(i + 1, k));
            k = input.lastIndexOf(";");
            cmp_name = input.substring(k + 1);
      }

      /* Methode für die Bottom-Seite einer Leiterplatte */
      public void setValue_bottom(String input) {
            /* Zeilen durch Trennzeichen ";" trennen und Klassenvariablen
setzen */
            int i, k;
            i = input.indexOf(";");
            cmp_nr = input.substring(0, i);
            k = input.indexOf(";", i + 1);
            x = Double.parseDouble(input.substring(i + 1, k));
            i = k;
            k = input.indexOf(";", i + 1);
            y = (Double.parseDouble(input.substring(i + 1, k)));
            i = k;
            k = input.indexOf(";", i + 1);
            rotation = Double.parseDouble(input.substring(i + 1, k));
            k = input.lastIndexOf(";");
            cmp_name = input.substring(k + 1);
      }

      /**
       * Deklaration der set- bzw. get-Methoden
       *
       * @author Patrick Olma
       * @version 30.06.2011
       * @param Übergabewerte
       *            für die jeweilige Klassenvariablen
       * @return jeweilige Klassenvariablen
       */
      public void setCmp_nr(String cmp_nr) {
            this.cmp_nr = cmp_nr;
      }

      public String getCmp_nr() {
            return cmp_nr;
      }

      public void setX(double x) {
            this.x = x;
      }

      public double getX() {
            return x;
      }

      public void setY(double y) {
            this.y = y;
      }
```

```
	public double getY() {
		return y;
	}

	public void setRotation(double rotation) {
		this.rotation = rotation;
	}

	public double getRotation() {
		return rotation;
	}

	public void setCmp_name(String cmp_name) {
		this.cmp_name = cmp_name;
	}

	public String getCmp_name() {
		return cmp_name;
	}

	public void setStatus(int status) {
		this.status = status;
	}

	public int getStatus() {
		return status;
	}

	public void setBeschreibung(String beschreibung) {
		this.beschreibung = beschreibung;
	}

	public String getBeschreibung() {
		return beschreibung;
	}
}
```

8.15 Pruefung.java

```
/**
 * Klasse für die Methoden der Prüfung der einzelnen Bauteile auf der
Leiterplatte
 *
 * Das aktuell zu prüfende Bauteil wird während der Prüfung gelb
eingefärbt.
 * Nachdem der Prüfer das Bauteil geprüft hat, kann er angeben, ob dieses
 * korrekt ist, dann wird es grün eingefärbt. Bzw. fehlerhat, dann wird es
rot eingefärbt.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import java.util.ArrayList;

public class Pruefung {

	/* Deklaration der Klassenvariablen */
	private int zaehler = 0;
	private int zaehler_max;
	private int status;
	private String beschreibung;
	public static boolean fehler = false;

	/*
	 * ArrayList für die Daten der Leiterplatte, welche bereits mit den
	 * vorhandenen Daten gefüllt werden
	 */
	ArrayList<PCB> aLeiterplatte = Daten.getAPCB();
	/* ArrayList für die Daten der Bauteile */
	ArrayList<Placement> aPlacement;
	/* ArrayList für die Geometrie-Daten der Bauteile */
	ArrayList<Comp> aComp;

	/**
	 * Methode zeichneLayout() um das Bauteil zu prüfen
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 */
	public void pruefe() {

		/* Abfrage, ob Ansicht --> Top gewählt wurde */
		if (GUI.getSide() == "top") {
			/* ArrayListen füllen */
			aPlacement = Daten.getAPlace_top();
			aComp = Daten.getaComp_top();
		}
		/* Abfrage, ob Ansicht --> Bottom gewählt wurde */
		else if (GUI.getSide() == "bottom") {
			/* ArrayListen füllen */
			aPlacement = Daten.getAPlace_bottom();
```

```
                  aComp = Daten.getaComp_bottom();
            }

            /* Textfelder der GUI füllen mit Daten des aktuellen Bauteils
*/
            GUI.lComp_input.setText(aPlacement.get(zaehler).getCmp_name());
            GUI.lCompnr_input.setText(aPlacement.get(zaehler).getCmp_nr());

      GUI.lRotation_input.setText(Double.toString(aPlacement.get(zaehler)
                        .getRotation()));
            GUI.lSchritt_input.setText("" + (zaehler + 1));
            GUI.lPCB_input.setText(aLeiterplatte.get(aLeiterplatte.size() -
1)
                        .getSachnummer());
            /* Komplette ArrayList durchlaufen */
            for (int y = 0; y < aComp.size(); y++) {
                  /*
                   * Abfrage ob aktuelles Bauteil der Geometrie-Daten
gleich dem
                   * aktuellen Bauteil der Placement-Daten ist.
                   */
                  if (aComp.get(y).getCmp_nr()
                              .equals(aPlacement.get(zaehler).getCmp_nr()))
{
                        /* Textfeld für Bauform des Bauteils füllen */

      GUI.lBauform_input.setText(aComp.get(y).getBauform());
                        /* Textfeld für Typ des Bauteils füllen */
                        GUI.lTyp_input.setText(aComp.get(y).getTyp());
                  }
            }
            /* Status des aktuellen Bauteils setzen */
            aPlacement.get(zaehler).setStatus(status);
            /* Abfrage, ob Bauteil fehlerhaft war */
            if (fehler) {
                  /* Beschreibung des Fehlers setzen */
                  aPlacement.get(zaehler).setBeschreibung(beschreibung);
            }
            /* Kein Fehler vorhanden... */
            fehler = false;
      }

      /**
       * Methode erhoehe_zaehler() um den Zähler zu erhöhen, d.h. zum
nächsten
       * Bauteil zu springen
       *
       * @author Patrick Olma
       * @version 30.06.2011
       */
      public void erhoehe_zaehler() {
            /* Abfrage, ob Zaehler noch kleiner als die maximalen Schritte
sind */
            if (zaehler < (aPlacement.size() - 1)) {
                  /* Zähler um 1 erhöhen */
                  zaehler++;
            }
      }
```

```
	/**
	 * Methode erniedrige_zaehler() um den Zähler zu erniedrigen, d.h.
zum
	 * vorherigen Bauteil zu springen
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 */
	public void erniedrige_zaehler() {
		/* Abfrage, ob Zähler größer als der 1. Schritt ist */
		if (zaehler > 0) {
			/* Zähler um 1 verringern */
			zaehler--;
		}
	}

	/**
	 * Methode reset_zaehler() um den Zähler auf 0 zurückzusetzen
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 */
	public void reset_zaehler() {
		zaehler = 0;
	}

	/**
	 * Methode reset_status() um den Status aller Bauteile zurückzusetzen
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 */
	public void reset_status() {
		/* Komplette ArrayList durchlaufen */
		for (int i = 0; i < aPlacement.size(); i++) {
			/* Status aller Bauteile auf 0 setzen */
			aPlacement.get(i).setStatus(0);
		}
	}

	/**
	 * Deklaration der set- bzw. get-Methoden
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Übergabewerte
	 *            für die jeweilige Klassenvariablen
	 * @return jeweilige Klassenvariablen
	 */
	public int getZaehler() {
		return zaehler;
	}

	public void setZaehler(int zaehler) {
		this.zaehler = zaehler;
	}
```

```
	public int getZaehler_max() {
		zaehler_max = aPlacement.size();
		return zaehler_max;
	}

	public void setZaehler_max(int zaehler_max) {
		this.zaehler_max = zaehler_max;
	}

	public void setStatus(int status) {
		this.status = status;
	}

	public int getStatus() {
		return status;
	}

	public void setBeschreibung(String beschreibung) {
		this.beschreibung = beschreibung;
	}

	public String getBeschreibung() {
		return beschreibung;
	}
}
```

8.16 Rect.java

```
/**
 * Klasse für die Objekte und Methoden der Klasse (Geometrie) Rect.
 *
 * Die einzelnen Objekte dieser Klasse werden innerhalb der Klasse
 * GUI_comp in Form von Geometrie-Libraries eingelesen und instantiiert.
 * Rect repräsentiert innerhalb dieser Geometrien Rechtecke vom Punkt A
nach B
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

public class Rect {

	/* Deklaration der Klassenvariablen */
	private double x1;
	private double x2;
	private double y1;
	private double y2;

	/**
	 * Deklaration des Konstruktors
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Anfangspunkt
	 *            (x1,y1) und Endpunkt (x2,y2)
	 */
	public Rect(double x1, double x2, double y1, double y2) {
		this.x1 = x1;
		this.x2 = x2;
		this.y1 = y1;
		this.y2 = y2;
	}

	/**
	 * Deklaration der set- bzw. get-Methoden
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param Übergabewerte
	 *            für die jeweilige Klassenvariablen
	 * @return jeweilige Klassenvariablen
	 */
	public void setX1(int x1) {
		this.x1 = x1;
	}

	public double getX1() {
		return x1;
	}

	public void setX2(double x2) {
```

```
		this.x2 = x2;
	}

	public double getX2() {
		return x2;
	}

	public void setY1(double y1) {
		this.y1 = y1;
	}

	public double getY1() {
		return y1;
	}

	public void setY2(double y2) {
		this.y2 = y2;
	}

	public double getY2() {
		return y2;
	}

}
```

8.17 Viewer.java

```
/**
 * Hautpklasse dieser Software.
 *
 * Diese Klasse ist der Einstiegspunkt in die Software und beinhaltet die
main()-Methode
 * In dieser Klasse wird das Look-And-Feel definiert und die GUI gestartet.
 *
 * @author Patrick Olma
 * @version 30.06.2011
 */

package pcb_viewer;

import javax.swing.UIManager;

public class Viewer {

	/**
	 * Methode main()
	 *
	 * @author Patrick Olma
	 * @version 30.06.2011
	 * @param args
	 */
	public static void main(String[] args) {
		try {
			/* Look-And-Feel setzen */
			UIManager

	.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel")
;
		} catch (Exception e) {
		}
		/* GUI starten */
		new GUI();
	}

}
```

8.18 export-library.ulp

```
int u2geda (int value)
{
	return u2mil (value);
}

string footprint_line (string val)
{
	return val + "\n";
}

string get_file_name (string pkg_name)
{
	string masked = ":?*/\\><| ";
	string result;
	for (int i = 0; pkg_name [i]; i ++)
		result [i] = strchr (masked, pkg_name [i]) >= 0 ? '_' :
pkg_name [i];
	return result;
}

string pcb_circle (UL_CIRCLE circle)
{
	if (!circle) return "";
	string result;
	sprintf (
		result,
		"ElementArc;%d;%d;%d;%d;%d;%d;%d",
		u2geda (circle.x),
		u2geda (-circle.y),
		u2geda (circle.radius),
		u2geda (circle.radius),
		0,
		360,
		u2geda (circle.width)
	);
	return result;
}

string pcb_wire (UL_WIRE wire)
{
	if (!wire) return "";
	string result;
	if (!wire.arc)
		sprintf (
			result,
			"ElementLin;%d;%d;%d;%d;%d",
			u2geda (wire.x1),
			u2geda (-wire.y1),
			u2geda (wire.x2),
			u2geda (-wire.y2),
			u2geda (wire.width)
		);
	else
		sprintf (
			result,
```

```
                  "ElementArc;%d;%d;%d;%d;%d;%d;%d",
                  (u2geda (wire.arc.xc) + u2geda (wire.arc.radius/2)),
                  (u2geda (-wire.arc.yc) + u2geda (wire.arc.radius/2)),
                  u2geda (wire.arc.radius),
                  u2geda (wire.arc.radius),
                  int (wire.arc.angle1),
                  int (wire.arc.angle2 - wire.arc.angle1),
                  u2geda (wire.width)
            );
      return result;
}

string pcb_rectangle (UL_RECTANGLE rect)
{
      if (!rect) return "";
      string result;
      int x1 = u2geda (rect.x1);
      int y1 = u2geda (-rect.y1);
      int x2 = u2geda (rect.x2);
      int y2 = u2geda (-rect.y2);
      sprintf (
            result,

            "ElementRec;%d;%d;%d;%d",
            x1, y1,
            x2, y2
      );
      return result;
}


string pcb_package (UL_PACKAGE pkg)
{
      if (!pkg) return "";
      string result;
      //sprintf (result, pkg.headline);
      pkg.circles (circle)    result += footprint_line (pcb_circle
(circle));
      pkg.wires (wire)        result += footprint_line (pcb_wire (wire));
      pkg.rectangles (rect)   result += footprint_line (pcb_rectangle
(rect));
      return result;
}

////////////////////////////////////////////////////////
string path = dlgDirectory ("Select output path", "");

library (lib)
{
      lib.packages (pack)
      {
            output (path + "/" + get_file_name (pack.name) + ".fp", "w")
            {
                  printf ("%s\n", pcb_package (pack));
            }
      }
}
```