```java
// ------------------------------------------------------------------------------------
// ---------- Diplomarbeit: Fehlerdatenauswertung in Java ------------------------------
// ---------- Hochschule Mittweida, 2017 -----------------------------------------------
// ---------- Autor: Strohmaier Markus -------------------------------------------------
// ------------------------------------------------------------------------------------

// ------- Klasse _Main_Window.java -----------------------------------------------------

// Name des Packages
package aoi_java;

// Imports aus Standard-Bibliotheken
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFormattedTextField;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JTabbedPane;
import javax.swing.SwingConstants;
import javax.swing.UIManager;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import java.sql.SQLException;

// Import der externen JDatePicker-Bibliothek
```

```java
import org.jdatepicker.impl.JDatePickerImpl;

// Klassenkopf (Start der Klasse)
public class _Main_Window extends JFrame
{
    // Globale Variable Tab-Fenster
    private static JTabbedPane tabbedPane;

    // Konstruktor der Klasse
    public _Main_Window()
    {
        // Fenstertitel (wird in der Titelleiste angezeigt)
        setTitle("AOI Fehlerdatenauswertung");

        // Standard-Aktion für Fenster schließen ändern
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

        // Bei Klick aufs Schließen-Symbol Tabs statt Fenster schließen
        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent arg0)
            {

                // Wenn mehrere Tabs offen
                if (tabbedPane.getTabCount()>1)
                    switch (tabbedPane.getSelectedIndex())
                    {
                        // Wenn Home-Tab = aktiv, dann letzten Tab schließen
                        case 0:tabbedPane.removeTabAt(tabbedPane.getTabCount()-1); break;

                        // sonst aktiven Tab schließen
                        default: tabbedPane.removeTabAt(tabbedPane.getSelectedIndex()); break;
                    }

                // Nur noch Home-Tab offen
                else
                {
                    try
                    {
                        // Verbindung trennen
```

```
81                      SQL_Handling.disconnect();
82                  }
83                  // SQL-Fehler in der Konsole ausgeben
84                  catch (SQLException e1) {
85                      e1.printStackTrace();
86                  }
87                  // Komplettes Fenster schließen
88                  System.exit(0);
89              }
90          }
91      });
92
93      // Grundfenster mit 5 Pixel Abstand zum Rand erstellen
94      JPanel contentPane = new JPanel();
95      contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
96      contentPane.setLayout(new BorderLayout(0, 0));
97      setContentPane(contentPane);
98
99      // neues Tab-Fenster einfügen
100     tabbedPane = new JTabbedPane();
101     contentPane.add(tabbedPane, BorderLayout.CENTER);
102
103     // neuen Tab mit dem Titel "Home" zum Tab-Fenster hinzufügen
104     // Leeres Panel in den erstellten Tab einfügen
105     JPanel panel = new JPanel();
106     tabbedPane.addTab("Home", null, panel, null);
107     panel.setLayout(new BorderLayout(0, 0));
108
109     // Panel für den Selektionsmodus einfügen
110     JPanel panel_sel = new JPanel();
111     panel_sel.setBorder(new TitledBorder(null, "Modus auswählen",
112             TitledBorder.LEADING, TitledBorder.TOP, null, null));
113     panel.add(panel_sel, BorderLayout.NORTH);
114     panel_sel.setLayout(new GridLayout(0, 2, 0, 0));
115
116     // Auswahl-Schaltfläche für Mehrfachselektion erstellen und hinzufügen
117     // Größe und Schriftart anpassen
118     // Mehrfachselektion ist beim Start aktiv
119     JRadioButton rdbtnDate = new JRadioButton("Auswertung über Zeitraum");
120     rdbtnDate.setFont(new Font("Tahoma", Font.PLAIN, 24));
```

```
121         rdbtnDate.setSelected(true);
122         panel_sel.add(rdbtnDate);
123 
124         // Auswahl-Schaltfläche für Einzelselektion erstellen und hinzufügen
125         // Größe und Schriftart anpassen
126         JRadioButton rdbtnOrder = new JRadioButton("Einzelnen Auftrag auswerten");
127         rdbtnOrder.setFont(new Font("Tahoma", Font.PLAIN, 24));
128         panel_sel.add(rdbtnOrder);
129 
130         // Schaltfläche als Gruppe festlegen, somit kann nur 1 Schaltfläche aktiv sein
131         // Gegenseitiger Ausschluss
132         ButtonGroup group = new ButtonGroup();
133         group.add(rdbtnDate);
134         group.add(rdbtnOrder);
135 
136         // Hilfspanel für die Auswertungsdetails einfügen
137         JPanel panel_1 = new JPanel();
138         panel_1.setBorder(new TitledBorder(UIManager.getBorder("TitledBorder.border"),
139                 "Details eingeben", TitledBorder.LEADING, TitledBorder.TOP, null, new Color(0, 0, 0)));
140         panel.add(panel_1, BorderLayout.CENTER);
141         panel_1.setLayout(new GridLayout(0, 2, 0, 0));
142 
143         // Detailpanel für den Mehrfachselektionsmodus einfügen
144         JPanel panel_multi = new JPanel();
145         panel_multi.setBorder(new EmptyBorder(5, 10, 5, 10));
146         panel_multi.setLayout(new GridLayout(0, 1, 10, 10));
147         panel_1.add(panel_multi);
148 
149         // 2 JDatePicker-Instanzen erstellen mit Textfeldern erstellen
150         //      a) Zur Eingabe des Start-Datums
151         //      b) Zur Eingabe des Ziel-Datums
152         JDatePickerImpl dpFrom = Date_Handling.createDatePicker();
153         JDatePickerImpl dpTo = Date_Handling.createDatePicker();
154         JFormattedTextField tfFrom = dpFrom.getJFormattedTextField();
155         JFormattedTextField tfTo = dpTo.getJFormattedTextField();
156 
157         // Textfelder nicht auswählbar und editierbar machen
158         // Diese werden über das DatePicker-Auswahlfenster befüllt
159         tfFrom.setFocusable(false);
160         tfTo.setFocusable(false);
```

```
161
162         // Hilfspanel #1 für den Mehrfachselektionsmodus
163         // für Datumsauswahl
164         JPanel panel_m1 = new JPanel();
165         panel_multi.add(panel_m1);
166         panel_m1.setLayout(new GridLayout(0, 1, 0, 10));
167
168         // Hilfspanel #2 für den Mehrfachselektionsmodus
169         // für Filterauswahl
170         JPanel panel_m2 = new JPanel();
171         panel_m2.setBorder(new EmptyBorder(20, 0, 20, 0));
172         panel_multi.add(panel_m2);
173         panel_m2.setLayout(new GridLayout(0, 1, 0, 5));
174
175         // Bezeichner für Produktfilter einfügen
176         JLabel lblProduktfilter = new JLabel("Produktfilter:");
177         lblProduktfilter.setFont(new Font("Tahoma", Font.PLAIN, 16));
178         lblProduktfilter.setVerticalAlignment(SwingConstants.BOTTOM);
179         panel_m2.add(lblProduktfilter);
180
181         // Drop-Down-Box für Produktfilter einfügen
182         // Standardwert "alle Produkte" zum Menü hinzufügen
183         JComboBox<String> cbProducts = new JComboBox<String>();
184         cbProducts.setFont(new Font("Tahoma", Font.PLAIN, 14));
185         panel_m2.add(cbProducts);
186         cbProducts.addItem("--- ALLE PRODUKTE ---");
187
188         // Bezeichner für Maschinenfilter einfügen
189         JLabel lblMaschinenfilter = new JLabel("Maschinenfilter:");
190         lblMaschinenfilter.setFont(new Font("Tahoma", Font.PLAIN, 16));
191         lblMaschinenfilter.setVerticalAlignment(SwingConstants.BOTTOM);
192         panel_m2.add(lblMaschinenfilter);
193
194         // Drop-Down-Box für Maschinenfilter einfügen
195         // Standardwert "alle AOIs" zum Menü hinzufügen
196         JComboBox<String> cbMachines = new JComboBox<String>();
197         cbMachines.setFont(new Font("Tahoma", Font.PLAIN, 14));
198         panel_m2.add(cbMachines);
199         cbMachines.addItem("--- ALLE AOIs ---");
200
```

```
201         // Drop-Down-Boxen "Produkt" und "Maschinen" mit den Daten aus der Datenbank füllen
202         try
203         {
204             // Verbindungsaufbau, wenn keine Verbindung besteht
205             if (SQL_Handling.getConnection() == null)
206                 SQL_Handling.connect();
207 
208             // alle verfügbaren Produkte abrufen
209             SQL_Handling.query("SELECT DISTINCT File_Name FROM RECIPE");
210 
211             // Abgefragte Datensätze zum Produkt-Drop-Down-Menü hinzufügen
212             while(SQL_Handling.getResultSet().next())
213                 cbProducts.addItem(SQL_Handling.getResultSet().getObject(1).toString());
214 
215             // alle verfügbaren Maschinen abrufen
216             SQL_Handling.query("SELECT Machine_Name FROM MACHINE WHERE Machine_Type_Name Like 'Vision'");
217 
218             // Abgefragte Datensätze zum Produkt-Drop-Down-Menü hinzufügen
219             while(SQL_Handling.getResultSet().next())
220                 cbMachines.addItem(SQL_Handling.getResultSet().getObject(1).toString());
221         }
222         // Bei Fehler(n), Fehlermeldungen in der Kommandozeile ausgeben und Dialog anzeigen
223         catch (SQLException ex)
224         {
225             JOptionPane.showMessageDialog(rootPane, "Verbindungsfehler");
226             ex.printStackTrace();
227         }
228 
229         // Unterpanel #3 für den Mehrfachselektionsmodus
230         // Für Datumsschnellauswahl
231         JPanel panel_m3 = new JPanel();
232         panel_m1.add(panel_m3);
233         panel_m3.setLayout(new GridLayout(0, 1, 0, 0));
234 
235         // Bezeichnner "Schnellauswahl" einfügen
236         JLabel lblDateFast = new JLabel("Datum (Schnellauswahl):");
237         lblDateFast.setFont(new Font("Tahoma", Font.PLAIN, 16));
238         panel_m3.add(lblDateFast);
239 
240         // Drop-Down-Box für die "Schnellauswahl" einfügen
```

```
241         JComboBox<String> cbDate = new JComboBox<String>();
242         panel_m3.add(cbDate);
243         cbDate.setFont(new Font("Tahoma", Font.PLAIN, 14));
244         cbDate.addItem("Manuell");
245         cbDate.addItem("Letzte Woche");
246         cbDate.addItem("Letzter Monat");
247         cbDate.addItem("Voretzter Monat");
248 
249         // Listener für die Drop-Down-Box => Bei Änderung der Auswahl
250         cbDate.addItemListener(new ItemListener()
251         {
252             public void itemStateChanged(ItemEvent ev)
253             {
254                 if(ev.getStateChange() == ItemEvent.SELECTED)
255                 {
256                     // Startwerte => Textfelder Start und Ende leer
257                     String [] dates = {"", ""};
258 
259                     // Abfrage des gewählten Index
260                     switch (cbDate.getSelectedIndex())
261                     {
262                         // Auswahl: "letzte Woche"
263                         // Abfrage des Start- und Enddatums über die Date_Handling-Klasse
264                         case 1:
265                             dates = Date_Handling.getLastYearWeek();
266                             tfFrom.setText(dates[0]);
267                             tfTo.setText(dates[1]);
268                             break;
269 
270                         // Auswahl: "letzter Monat"
271                         // Abfrage des Start- und Enddatums über die Date_Handling-Klasse
272                         case 2:
273                             dates = Date_Handling.getLastMonth();
274                             tfFrom.setText(dates[0]);
275                             tfTo.setText(dates[1]);
276                             break;
277 
278                         // Auswahl: "vorletzter Monat"
279                         // Abfrage des Start- und Enddatums über die Date_Handling-Klasse
280                         case 3:
```

```
281                         dates = Date_Handling.getMonthBeforeLast();
282                         tfFrom.setText(dates[0]);
283                         tfTo.setText(dates[1]);
284                         break;
285 
286                     // Auswahl: "Manuell"
287                     // Heutigen Tag über Date_Handling-Klasse abfragen und temporär eintragen
288                     default:
289                         tfFrom.setText(Date_Handling.getToday());
290                         tfTo.setText(Date_Handling.getToday());
291                         break;
292                 }
293             }
294         }
295     });
296 
297     // Unterpanel #4 für den Mehrfachselektionsmodus
298     JPanel panel_m4 = new JPanel();
299     panel_m1.add(panel_m4);
300     panel_m4.setLayout(new GridLayout(0, 1, 0, 0));
301 
302     // Bezeichner für das Startdatum einfügen
303     // Date-Picker für das Startdatum zum Panel hinzufügen
304     JLabel lblStart = new JLabel("Start: ");
305     lblStart.setFont(new Font("Tahoma", Font.PLAIN, 16));
306     lblStart.setVerticalAlignment(SwingConstants.BOTTOM);
307     panel_m4.add(lblStart);
308     panel_m4.add(dpFrom);
309 
310     // Unterpanel #5 für den Mehrfachselektionsmodus
311     JPanel panel_m5 = new JPanel();
312     panel_m1.add(panel_m5);
313     panel_m5.setLayout(new GridLayout(0, 1, 0, 0));
314 
315     // Bezeichner für das Enddatum einfügen
316     // Date-Picker für das Enddatum zum Panel hinzufügen
317     JLabel lblEnde = new JLabel("Ende: ");
318     lblEnde.setFont(new Font("Tahoma", Font.PLAIN, 16));
319     lblEnde.setVerticalAlignment(SwingConstants.BOTTOM);
320     panel_m5.add(lblEnde);
```

```
321         panel_m5.add(dpTo);
322
323         // Heutigen Tag über Date_Handling-Klasse abfragen und als Startwerte eintragen
324         tfFrom.setText(Date_Handling.getToday());
325         tfTo.setText(Date_Handling.getToday());
326
327         // Detailpanel für den Einzelselektionsmodus einfügen
328         JPanel panel_single = new JPanel();
329         panel_single.setBorder(new EmptyBorder(5, 10, 5, 10));
330         panel_1.add(panel_single);
331         panel_single.setLayout(new GridLayout(0, 1, 0, 0));
332
333         // Hilfspanel #1 für den Einzelselektionsmodus einfügen
334         JPanel panel_s1 = new JPanel();
335         panel_single.add(panel_s1);
336         panel_s1.setLayout(new GridLayout(0, 1, 0, 0));
337
338         // Bezeichner für den Einzelauftrag einfügen
339         JLabel lblFertigungsauftrag = new JLabel("Fertigungsauftrag eingeben:");
340         lblFertigungsauftrag.setFont(new Font("Tahoma", Font.PLAIN, 16));
341         panel_s1.add(lblFertigungsauftrag);
342
343         // Textfeld zur Benutzereingabe für den Auftrag einfügen
344         JFormattedTextField txtOrder = new JFormattedTextField();
345         txtOrder.setFont(new Font("Tahoma", Font.PLAIN, 16));
346         panel_s1.add(txtOrder);
347
348         // Parameter für das Textfeld
349         // Quickinfo (Tooltip) hinzufügen
350         // Zentrierte Schrift aktivieren
351         // Beim Start inaktiv => Mehrfachselektion ist am Anfang aktiviert
352         txtOrder.setToolTipText("Auftrag eingeben");
353         txtOrder.setHorizontalAlignment(SwingConstants.CENTER);
354         txtOrder.setEnabled(false);
355
356         // Startwert zum Testen der Einzelabfrage
357         txtOrder.setText("2013890131");
358
359         // leeres Hilfslabel zur besseren Darstellung hinzufügen
360         JLabel label = new JLabel("");
```

```
361         panel_s1.add(label);
362
363         // Schaltfläche zum Aktivieren der Einzelabfrage hinzufügen
364         // Quickinfo (Tooltip) hinzufügen
365         // Beim Start inaktiv => Mehrfachselektion ist am Anfang aktiviert
366         JButton btnTable = new JButton("Tabelle");
367         panel_s1.add(btnTable);
368         btnTable.setFont(new Font("Tahoma", Font.PLAIN, 20));
369         btnTable.setToolTipText("Tabelle mit allen Details abrufen");
370         btnTable.setEnabled(false);
371
372         // leeres Hilfspanel zur besseren Darstellung hinzufügen
373         JPanel panel_s2 = new JPanel();
374         panel_single.add(panel_s2);
375         panel_s2.setLayout(new GridLayout(0, 1, 0, 0));
376
377         // Gegenseitiger Ausschluss der Selektionskomponenten
378         // ActionListener => Wenn Mehrfachselektion gewählt wird
379         rdbtnDate.addActionListener(new ActionListener()
380         {
381             public void actionPerformed(ActionEvent e)
382             {
383                 // Alle Einzelselektions-Komponenten deaktivieren
384                 for(int i = 0; i < panel_s1.getComponents().length; i++)
385                     panel_s1.getComponent(i).setEnabled(false);
386
387                 // Alle Mehrfachselektionskomponenten aktivieren
388                 for(int i = 0; i < panel_m1.getComponents().length; i++)
389                     panel_m1.getComponent(i).setEnabled(true);
390                 for(int i = 0; i < panel_m2.getComponents().length; i++)
391                     panel_m2.getComponent(i).setEnabled(true);
392                 for(int i = 0; i < panel_m3.getComponents().length; i++)
393                     panel_m3.getComponent(i).setEnabled(true);
394                 for(int i = 0; i < panel_m4.getComponents().length; i++)
395                     panel_m4.getComponent(i).setEnabled(true);
396                 for(int i = 0; i < panel_m5.getComponents().length; i++)
397                     panel_m5.getComponent(i).setEnabled(true);
398
399                 // Textfelder für Datumsauswahl manuell aktivieren
400                 tfFrom.setEnabled(true);
```

```
401             tfTo.setEnabled(true);
402         }
403     });
404
405     // ActionListener => Wenn Einzelselektion gewählt wird
406     rdbtnOrder.addActionListener(new ActionListener()
407     {
408         public void actionPerformed(ActionEvent e)
409         {
410             // Alle Einzelselektions-Komponenten aktivieren
411             for(int i = 0; i < panel_s1.getComponents().length; i++)
412                 panel_s1.getComponent(i).setEnabled(true);
413
414             // Alle Mehrfachselektionskomponenten deaktivieren
415             for(int i = 0; i < panel_m1.getComponents().length; i++)
416                 panel_m1.getComponent(i).setEnabled(false);
417             for(int i = 0; i < panel_m2.getComponents().length; i++)
418                 panel_m2.getComponent(i).setEnabled(false);
419             for(int i = 0; i < panel_m3.getComponents().length; i++)
420                 panel_m3.getComponent(i).setEnabled(false);
421             for(int i = 0; i < panel_m4.getComponents().length; i++)
422                 panel_m4.getComponent(i).setEnabled(false);
423             for(int i = 0; i < panel_m5.getComponents().length; i++)
424                 panel_m5.getComponent(i).setEnabled(false);
425
426             // Textfelder für Datumsauswahl manuell deaktivieren
427             tfFrom.setEnabled(false);
428             tfTo.setEnabled(false);
429         }
430     });
431
432     // Zusatzpanel für FPY und Diagramm
433     // Diese beiden Schaltflächen sind in beiden Selektionsmodus verfügbar
434     JPanel panel_opt = new JPanel();
435     panel_opt.setBorder(new TitledBorder(null, "Grundfunktionen",
436             TitledBorder.LEADING, TitledBorder.TOP, null, null));
437     panel.add(panel_opt, BorderLayout.SOUTH);
438     panel_opt.setLayout(new GridLayout(0, 1, 0, 10));
439
440     // Schaltfläche für First Pass Yield
```

```
441         JButton btnFPY = new JButton("First Pass Yields");
442         btnFPY.setFont(new Font("Tahoma", Font.PLAIN, 20));
443         panel_opt.add(btnFPY);
444         btnFPY.setToolTipText("FPY(s) anzeigen");
445 
446         // Schaltfläche für Diagramm
447         JButton btnChart = new JButton("Diagramm");
448         btnChart.setFont(new Font("Tahoma", Font.PLAIN, 20));
449         panel_opt.add(btnChart);
450         btnChart.setToolTipText("gefiltertes Diagramm anzeigen");
451 
452         // Tabellen-Schaltfläche Action-Listener
453         btnTable.addActionListener(new ActionListener()
454         {
455             public void actionPerformed(ActionEvent e)
456             {
457                 try
458                 {
459                     // Eingegebenen Auftrag (Text) in eine Zahl umwandeln
460                     long order = (long) Double.parseDouble(txtOrder.getText());
461 
462                     // Wenn Zahl im richtigen Wertebereich
463                     if (order > 2013799999 && order < 2017900000)
464                     {
465                         // Verbindungsversuch, wenn keine Verbindung besteht
466                         try
467                         {
468                             if (SQL_Handling.getConnection() == null)
469                                 SQL_Handling.connect();
470 
471                             // SQL-Abfrage mit allen Details zum eingegebenen Fertigungsauftrag
472                             SQL_Handling.query("SELECT Panel_Bar_Code AS Panel, Card_Bar_Code AS Card, Card_Number, "
473                                 + "CONVERT(VARCHAR(17), DATEADD(SECOND, Panel_Numeric_Date, '1970-01-01 01:00'), 13) "
474                                 + "AS Inspection_Time, "
475                                 + "Machine_Name AS Machine, File_Name AS Product, Face, Test_Time, "
476                                 + "CARDS.Anomaly_BR AS Before_Repair, "
477                                 + "CARDS.Anomaly_AR AS After_Repair, Topology, Part_Number, JEDEC_Name AS Jedec, "
478                                 + "Repair_Error_Comment AS Failure_Code, Operator_Name AS Operator, "
479                                 + "CONVERT(VARCHAR(17), "
480                                 + "DATEADD(SECOND, Repair_Numeric_Date_Hour, '1970-01-01 01:00'), 13) AS Repair_Time "
```

```
481                             + "FROM TESTED_OBJECT "
482                             + "RIGHT JOIN CARDS ON CARDS.Card_Id = TESTED_OBJECT.Card_Id "
483                             + "INNER JOIN PANELS ON PANELS.Panel_Id = CARDS.Panel_Id "
484                             + "INNER JOIN MACHINE ON MACHINE.Machine_Id = PANELS.Machine_Id "
485                             + "INNER JOIN RECIPE ON RECIPE.RECIPE_ID = PANELS.RECIPE_ID "
486                             + "LEFT JOIN OPERATOR ON TESTED_OBJECT.Operator_Id = OPERATOR.Operator_Id  "
487                             + "LEFT JOIN PART_NUMBER ON TESTED_OBJECT.Part_Number_Id = PART_NUMBER.Part_Number_Id "
488                             + "LEFT JOIN JEDEC ON PART_NUMBER.JEDEC_Id = JEDEC.JEDEC_Id "
489                             + "WHERE Card_Bar_Code LIKE '" + order + "%' "
490                             + "ORDER BY Panel_Bar_Code ASC");
491 
492                         // Tabelle mit Hilfe der Klasse Table_Handling erstellen
493                         Table_Handling.createTable(order);
494                     }
495                     // SQL-Fehler
496                     // a) Dialogfeld anzeigen
497                     // b) Fehlerdetails in der Konsole ausgeben
498                     catch (SQLException e_tbl1)
499                     {
500                         JOptionPane.showMessageDialog(rootPane, "Es konnte keine Verbindung aufgebaut werden");
501                         e_tbl1.printStackTrace();
502                     }
503                 }
504             }
505             // Allgemeiner Fehler
506             // a) Dialogfeld anzeigen
507             // b) Fehlerdetails in der Konsole ausgeben
508             catch (Exception e_tbl2)
509             {
510                 JOptionPane.showMessageDialog(rootPane, "Verbindungsfehler");
511                 e_tbl2.printStackTrace();
512             }
513         }
514     });
515 
516     // Diagramm-Schaltfläche Action-Listener
517     btnChart.addActionListener(new ActionListener()
518     {
519         public void actionPerformed(ActionEvent e)
520         {
```

```
521                 // Neue Instanz der Klasse Chart_Options erstellen
522                 // Dialogfeld für die Diagrammoptionen
523                 new Chart_Options();
524
525                 // Wenn das Dialogfeld mit ok bestätigt wird, dann fortfahren
526                 if(!Chart_Options.getAbort())
527                 {
528                     try
529                     {
530                         // Lokale Variablen deklarieren und mit Startwerten versehen
531                         String var = "";
532                         long order = 0;
533                         String machine = cbMachines.getSelectedItem().toString();
534                         String product = cbProducts.getSelectedItem().toString();
535
536                         // Bei Einzelselektion
537                         if (rdbtnOrder.isSelected())
538                         {
539                             // Eingegebenen Auftrag (Text) in eine Zahl umwandeln
540                             order = (long) Double.parseDouble(txtOrder.getText());
541
542                             // Wenn Zahl im richtigen Wertebereich
543                             // Variablen String für SQL-Abfrage generieren (Einzelauftrag)
544                             if (order > 2013799999 && order < 2017900000)
545                                 var = "Card_Bar_Code LIKE '" + order + "%' ";
546                         }
547                         // Bei Mehrfachselektion
548                         else
549                             // Variablen String für SQL-Abfrage generieren (Datumsintervall)
550                             var = "DATEADD(SECOND, Panel_Numeric_Date, '1970-01-01 01:00') BETWEEN '"
551                                     + tfFrom.getText() + " 00:00:00.000' AND '" + tfTo.getText() + " 23:59:59.000' ";
552
553                         // Weitere lokale Variablen + Startparameter
554                         String face = "ALLE";
555                         String error = "LIKE '%'";
556                         String filter = Chart_Options.getFilter();
557                         String filter_code = "Topology";
558
559                         // Wenn eine Seite ausgewählt wurde
560                         // default = beide Seiten ("ALLE")
```

```
561                 switch(Chart_Options.getFace())
562                 {
563                     // Variable "face" ändern
564                     case "nur Top": face = "TOP"; break;
565                     case "nur Bottom": face = "BOTTOM"; break;
566                 }
567 
568                 // Wenn eine Fehlertype ausgewählt wurde
569                 // default = alle Fehler ("%")
570                 switch(Chart_Options.getError())
571                 {
572                     case "nur Pseudo": error = "LIKE '810'"; break;
573                     case "nur echte": error = "NOT LIKE '810'"; break;
574                 }
575 
576                 // Abfrage nach welchem Merkmal gefiltert werden soll
577                 // default = Positionsbezeichnung ("Topology")
578                 switch (filter)
579                 {
580                     case "Bauteil": filter_code = "Part_Number"; break;
581                     case "Gehäuseform": filter_code = "JEDEC_Name"; break;
582                     case "Art": filter_code = "Repair_Error_Comment"; break;
583                 }
584 
585                 try
586                 {
587                     // Verbindungsversuch, wenn keine Verbindung besteht
588                     if (SQL_Handling.getConnection() == null)
589                         SQL_Handling.connect();
590 
591                     // SQL-Abfrage mit variablen Filtereinstellungen durchführen
592                     SQL_Handling.query("SELECT Count(" + filter_code + ") AS 'Counter', " + filter_code +
593                             ((face.equals("TOP") || face.equals("BOTTOM")) ? ", FACE " : " ")
594                             + "FROM TESTED_OBJECT  "
595                             + "RIGHT JOIN CARDS ON CARDS.Card_Id = TESTED_OBJECT.Card_Id "
596                             + "INNER JOIN PANELS ON PANELS.Panel_Id = CARDS.Panel_Id "
597                             + "INNER JOIN MACHINE ON MACHINE.MACHINE_ID = PANELS.MACHINE_ID "
598                             + "INNER JOIN RECIPE ON RECIPE.RECIPE_ID = PANELS.RECIPE_ID "
599                             + "LEFT JOIN PART_NUMBER ON TESTED_OBJECT.Part_Number_Id = PART_NUMBER.Part_Number_Id "
600                             + "LEFT JOIN JEDEC ON PART_NUMBER.JEDEC_Id = JEDEC.JEDEC_Id "
```

```
601                                 + "WHERE " + var + (rdbtnOrder.isSelected() ? "" : (machine.equals("--- ALLE AOIs ---") ? "" : " "
602                                 + "AND MACHINE_NAME LIKE '" + machine + "'") + " "
603                                 + (product.equals("--- ALLE PRODUKTE ---") ? "" : " AND File_Name LIKE '" + product + "'")) + " "
604                                 + "AND Repair_Error_Comment " + error + "AND Repair_Error_Comment NOT LIKE '' "
605                                 + ((face.equals("TOP") || face.equals("BOTTOM")) ? " AND FACE LIKE '" + face + "%' " : " ")
606                                 + "GROUP BY " + filter_code + ((face.equals("TOP") || face.equals("BOTTOM")) ? ", FACE " : " ")
607                                 + "ORDER BY COUNT(" + filter_code + ") DESC");
608 
609                             // Diagramm erstellen über die Klasse Chart_Handling
610                             Chart_Handling.createChart(filter);
611                         }
612                         // SQL-Fehler
613                         // a) Dialogfeld anzeigen
614                         // b) Fehlerdetails in der Konsole ausgeben
615                         catch (SQLException e_chart1) {
616                             JOptionPane.showMessageDialog(rootPane, "Verbindungsfehler");
617                             e_chart1.printStackTrace();
618                         }
619                     }
620                     // allgemeiner Fehler
621                     // a) Dialogfeld anzeigen
622                     // b) Fehlerdetails in der Konsole ausgeben
623                     catch (Exception e_chart2) {
624                         JOptionPane.showMessageDialog(rootPane, "ungültige Selektion");
625                         e_chart2.printStackTrace();
626                     }
627                 }
628             }
629         });
630 
631         // First Pass Yield Action Listener
632         btnFPY.addActionListener(new ActionListener()
633         {
634             public void actionPerformed(ActionEvent e)
635             {
636                 // Lokale Variablen deklarieren und mit Startwerten versehen
637                 String var = "";
638                 String machine = cbMachines.getSelectedItem().toString();
639                 String product = cbProducts.getSelectedItem().toString();
640 
```

```
641                 try
642                 {
643                     // Bei Einzelselektion
644                     if (rdbtnOrder.isSelected())
645                     {
646                         // Eingegebenen Auftrag (Text) in Zahl umwandeln
647                         long order = (long) Double.parseDouble(txtOrder.getText());
648 
649                         // Wenn Zahl im richtigen Wertebereich
650                         // Variablen String für SQL-Abfrage generieren (Einzelauftrag)
651                         if (order > 2013799999 && order < 2017900000)
652                             var = "Card_Bar_Code LIKE '" + order + "%' ";
653                     }
654                     // Bei Mehrfachselektion
655                     else
656                         // Variablen String für SQL-Abfrage generieren (Datumsintervall)
657                         var = "DATEADD(SECOND, Panel_Numeric_Date, '1970-01-01 01:00') BETWEEN '"
658                                 + tfFrom.getText() + " 00:00:00.000' AND '" + tfTo.getText() + " 23:59:59.000' ";
659 
660                     try
661                     {
662                         // Verbindungsversuch, wenn keine Verbindung besteht
663                         if (SQL_Handling.getConnection() == null)
664                             SQL_Handling.connect();
665 
666                         // SQL-Abfrage mit variablen Filtereinstellungen durchführen
667                         // Abfrage #1: alle geprüften Leiterplatten abfragen (Zeitraum bzw. Einzelauftrag)
668                         // "UNION"
669                         // Abfrage #2: alle Leiterplatten ohne Fehler abfragen (Zeitraum bzw. Einzelauftrag)
670                         // "UNION"
671                         // Abfrage #3: alle Leiterplatten mit echten Fehlern abfragen (Zeitraum bzw. Einzelauftrag)
672                         SQL_Handling.query("SELECT COUNT(DISTINCT Card_Bar_Code) AS 'Counter', Face, 'all' AS Code, "
673                             + "CONVERT (varchar(10), Card_Bar_Code) AS Order_Nr "
674                             + "FROM TESTED_OBJECT "
675                             + "RIGHT JOIN CARDS ON CARDS.Card_Id = TESTED_OBJECT.Card_Id "
676                             + "INNER JOIN PANELS ON PANELS.Panel_Id = CARDS.Panel_Id "
677                             + "INNER JOIN MACHINE ON MACHINE.MACHINE_ID = PANELS.MACHINE_ID "
678                             + "INNER JOIN RECIPE ON RECIPE.RECIPE_ID = PANELS.RECIPE_ID "
679                             + "WHERE " + var + (rdbtnOrder.isSelected() ? "" : (machine.equals("--- ALLE AOIs ---") ? "" : " "
680                             + "AND MACHINE_NAME LIKE '" + machine + "'") + " "
```

```
681                     + (product.equals("--- ALLE PRODUKTE ---") ? "" : " AND File_Name LIKE '" + product + "'")) + " "
682                     + "GROUP BY Face, CONVERT (varchar(10), Card_Bar_Code) "
683                     + "UNION "
684                     + "SELECT COUNT(DISTINCT Card_Bar_Code), Face, Repair_Error_Comment AS Code, "
685                     + "CONVERT (varchar(10), Card_Bar_Code) AS Order_Nr "
686                     + "FROM TESTED_OBJECT "
687                     + "RIGHT JOIN CARDS ON CARDS.Card_Id = TESTED_OBJECT.Card_Id "
688                     + "INNER JOIN PANELS ON PANELS.Panel_Id = CARDS.Panel_Id "
689                     + "INNER JOIN MACHINE ON MACHINE.MACHINE_ID = PANELS.MACHINE_ID "
690                     + "INNER JOIN RECIPE ON RECIPE.RECIPE_ID = PANELS.RECIPE_ID "
691                     + "WHERE " + var + " AND Repair_Error_Comment IS NULL "
692                     + (rdbtnOrder.isSelected() ? "" : (machine.equals("--- ALLE AOIs ---") ? "" : " "
693                     + "AND MACHINE_NAME LIKE '" + machine + "'") + " "
694                     + (product.equals("--- ALLE PRODUKTE ---") ? "" : " AND File_Name LIKE '" + product + "'")) + " "
695                     + "AND Card_Status Like '1' "
696                     + "GROUP BY Face, Repair_Error_Comment, CONVERT (varchar(10), Card_Bar_Code) "
697                     + "UNION "
698                     + "SELECT COUNT(DISTINCT Card_Bar_Code), Face, '_real' AS Code, "
699                     + "CONVERT (varchar(10), Card_Bar_Code) AS Order_Nr "
700                     + "FROM TESTED_OBJECT "
701                     + "RIGHT JOIN CARDS ON CARDS.Card_Id = TESTED_OBJECT.Card_Id "
702                     + "INNER JOIN PANELS ON PANELS.Panel_Id = CARDS.Panel_Id "
703                     + "INNER JOIN MACHINE ON MACHINE.MACHINE_ID = PANELS.MACHINE_ID "
704                     + "INNER JOIN RECIPE ON RECIPE.RECIPE_ID = PANELS.RECIPE_ID "
705                     + "WHERE " + var +  (rdbtnOrder.isSelected() ? "" : (machine.equals("--- ALLE AOIs ---") ? "" : " "
706                     + "AND MACHINE_NAME LIKE '" + machine + "'") + " "
707                     + (product.equals("--- ALLE PRODUKTE ---") ? "" : " AND File_Name LIKE '" + product + "'")) + " "
708                     + "AND Repair_Error_Comment NOT LIKE '810' AND Repair_Error_Comment IS NOT NULL "
709                     + "GROUP BY Face, CONVERT (varchar(10), Card_Bar_Code) "
710                     + "ORDER BY Order_Nr ASC, Face DESC, Code ASC");
711
712                 // First Pass Yield(s) über die Klasse FPY_Handling berechnen
713                 FPY_Handling.calcFPY();
714             }
715             // SQL-Fehler
716             // a) Dialogfeld anzeigen
717             // b) Fehlerdetails in der Konsole ausgeben
718             catch (SQLException e_fpy1) {
719                 JOptionPane.showMessageDialog(rootPane, "Verbindungsfehler");
720                 e_fpy1.printStackTrace();
```

```
721                 }
722             }
723             // allgemeiner Fehler
724             // a) Dialogfeld anzeigen
725             // b) Fehlerdetails in der Konsole ausgeben
726             catch (Exception e_fpy2)
727             {
728                 JOptionPane.showMessageDialog(rootPane, "ungültige Selektion");
729                 e_fpy2.printStackTrace();
730             }
731         }
732     });
733 }
734 
735 // Main-Methode => macht die Klasse "ausführbar"
736 public static void main(String[] args)
737 {
738     EventQueue.invokeLater(new Runnable()
739     {
740         public void run()
741         {
742             try
743             {
744                 //  Optik ans Betriebssystem anpassen
745                 UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
746 
747                 // neue Instanz der Klasse _Main_Window erstellen
748                 // Konstruktor aufrufen
749                 _Main_Window frame = new _Main_Window();
750 
751                 // Fensterparameter anpassen
752                 // Fenstergroesse beim Start
753                 // Fenster zentriert
754                 // Fenster sichtbar
755                 frame.setSize(1024, 640);
756                 frame.setLocationRelativeTo(null);
757                 frame.setVisible(true);
758             }
759 
760             // Bei Fehler(n), Dialog anzeigen
```

```
761                 catch (Exception e)
762                 {
763                     JOptionPane.showMessageDialog(null, "Verbindungsfehler");
764                 }
765             }
766         });
767 
768     }
769 
770     // Getter-Methode für das Tabfenster zum Zugriff für andere Klassen
771     public static JTabbedPane getTabPane()
772     {
773         return tabbedPane;
774     }
775 }
776 
```

```java
// ------------------------------------------------------------------------------------
// ---------- Diplomarbeit: Fehlerdatenauswertung in Java ------------------------------
// ---------- Hochschule Mittweida, 2017 -----------------------------------------------
// ---------- Autor: Strohmaier Markus -------------------------------------------------
// ------------------------------------------------------------------------------------

// ------- Klasse Chart_Handling.java -----------------------------------------------

// Name des Packages
package aoi_java;

// Imports aus Standard-Bibliotheken
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.sql.SQLException;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JPanel;
import javax.swing.filechooser.FileNameExtensionFilter;

//Imports der externen JFreeChart-Bibliothek
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.labels.ItemLabelAnchor;
import org.jfree.chart.labels.ItemLabelPosition;
import org.jfree.chart.labels.StandardCategoryItemLabelGenerator;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.CategoryItemRenderer;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.TextAnchor;

//Klassenkopf (Start der Klasse)
public class Chart_Handling {
```

```
41
42      // Methode zum Generieren der Diagrammdaten
43      // Übergabe des Filtermerkmals an die Methode
44      private static DefaultCategoryDataset createChartData(String filter) throws SQLException {
45
46          // Variable zum Speichern des Datensatzes
47          DefaultCategoryDataset data = new DefaultCategoryDataset();
48
49          // Hilfsvariable als Zähler
50          int i = 0;
51
52          // Solange neue Daten aus der SQL-Abfrage verfügbar sind
53          while(SQL_Handling.getResultSet().next())
54          {
55              // Maximal 10 Werte anzeigen (aus Übersichtsgründen)
56              if (i < 10)
57
58                  // Wert als Integer zum Datensatz hinzufügen
59                  data.setValue((Integer) SQL_Handling.getResultSet().getObject(1),
60
61                          // Legendeneintrag im Diagramm festlegen
62                          "Anzahl / " + filter,
63
64                          // Merkmal an den Datensatz übergeben
65                          SQL_Handling.getResultSet().getObject(2).toString());
66
67              // Zählvariable erhöhen
68              i++;
69          }
70
71          // Datensatz zurückgeben
72          return data;
73      }
74
75      // Methode zum Zeichnen des Diagramms
76      // Übergabe des Filtermerkmals an die Methode
77      public static void createChart(String filter) throws SQLException  {
78
79          // Daten über die lokale Methode crateChartData abrufen
80          DefaultCategoryDataset dataset = createChartData(filter);
```

```
81
82          // Diagramm erstellen (Säulendiagramm)
83          JFreeChart chart = ChartFactory.createBarChart(
84
85                  // Diagrammtitel
86                  "Top 10 - Fehler nach " + filter,
87
88                  // Bezeichnung der x- und y-Achse (leer)
89                  "", "",
90
91                  // Angabe des Datensatzes
92                  dataset,
93
94                  // Orientierung der Balken im Diagramm
95                  PlotOrientation.VERTICAL,
96
97                  // Legende anzeigen, Tooltips deaktivieren, URLs deaktivieren
98                  true, false, false
99              );
100
101         // Diagrammoptimierungen einfügen
102         // Diagramm-Renderer abfragen und bearbeiten
103         CategoryItemRenderer renderer = ((CategoryPlot)chart.getPlot()).getRenderer();
104
105         // Balkenbeschriftungen erstellen und anzeigen
106         renderer.setBaseItemLabelGenerator(new StandardCategoryItemLabelGenerator());
107         renderer.setBaseItemLabelsVisible(true);
108
109         // Position der Balkenbeschriftung bearbeiten (ans Ende des Balkens stellen)
110         ItemLabelPosition position = new ItemLabelPosition(ItemLabelAnchor.OUTSIDE12,
111                 TextAnchor.TOP_CENTER);
112         renderer.setBasePositiveItemLabelPosition(position);
113
114         // neues Panel für das Diagramm erstellen
115         JPanel panel = new JPanel();
116         panel.setLayout(new BorderLayout(0, 0));
117
118         // Diagramm in ein vordefiniertes Chart-Panel und dann ins Hauptpanel einfügen
119         ChartPanel chartPanel = new ChartPanel(chart);
120         panel.add(chartPanel, BorderLayout.CENTER);
```

```
121
122         // Schaltfläche zum Diagrammexport einfügen
123         JButton btnPic = new JButton("Als Bild Speichern");
124         btnPic.setFont(new Font("Tahoma", Font.PLAIN, 20));
125         panel.add(btnPic, BorderLayout.SOUTH);
126
127         // Action-Listener für die Export-Schaltfläche
128         btnPic.addActionListener(new ActionListener() {
129             public void actionPerformed(ActionEvent e) {
130                 try
131                 {
132                     // Datei-Auswahldialog öffnen
133                     JFileChooser chooser = new JFileChooser();
134
135                     // Datei-Filter generieren (JPEG, PNG)
136                     FileNameExtensionFilter filter_JPG = new FileNameExtensionFilter("JPEG-Datei", "jpg", "jpeg");
137                     FileNameExtensionFilter filter_PNG = new FileNameExtensionFilter("PNG-Datei", "png");
138
139                     // Filter des Auswahldialogs anpassen
140                     // Filter "alle Dateien" ausblenden
141                     chooser.setAcceptAllFileFilterUsed(false);
142
143                     // Standardfilter JPG
144                     chooser.setFileFilter(filter_JPG);
145
146                     // Optionaler Filter PNG
147                     chooser.addChoosableFileFilter(filter_PNG);
148
149                     // Wenn der Auswahldialog mit OK bestätigt wird
150                     if (chooser.showSaveDialog(null) == JFileChooser.APPROVE_OPTION) {
151
152                         // Eingegebenen Dateinamen abfragen
153                         String name = chooser.getSelectedFile().toString();
154
155                         // Ausgewählte Filter abfragen
156                         // Wenn JPEG
157                         if(chooser.getFileFilter().equals(filter_JPG))
158                         {
159                             // Dateiendung abfragen und ggf. ergänzen
160                             if(!(name.endsWith(".jpg") || name.endsWith(".jpeg")))
```

```
161                             name += ".jpg";
162
163                         // Diagramm speichern (Dateiname, Diagramm, Auflösung 1280x800)
164                         ChartUtilities.saveChartAsJPEG(new File(name), chart, 1280, 800);
165                     }
166                     // Wenn PNG
167                     else
168                     {
169                         // Dateiendung abfragen und ggf. ergänzen
170                         if(!name.endsWith(".png"))
171                             name += ".png";
172
173                         // Diagramm speichern (Dateiname, Diagramm, Auflösung 1280x800)
174                         ChartUtilities.saveChartAsPNG(new File(name), chart, 1280, 800);
175                     }
176                 }
177             }
178             // Bei Input- Output-Fehler
179             catch (IOException e_io)
180             {
181                 // Fehlerdetails in der Konsole ausgeben
182                 e_io.printStackTrace();
183             }
184
185         }
186     });
187
188     // Panel mit enthaltenem Diagramm als neuen Tab im Hauptfenster hinzufügen
189     _Main_Window.getTabPane().addTab("Diagramm", panel);
190
191     // direkt zum neu hinzugefügten Tab springen
192     _Main_Window.getTabPane().setSelectedIndex(_Main_Window.getTabPane().getTabCount() - 1);
193 }
194 }
195
```

```java
// ------------------------------------------------------------------------------------
// ---------- Diplomarbeit: Fehlerdatenauswertung in Java ------------------------------
// ---------- Hochschule Mittweida, 2017 -----------------------------------------------
// ---------- Autor: Strohmaier Markus -------------------------------------------------
// ------------------------------------------------------------------------------------

// ------- Klasse Chart_Options.java -----------------------------------------------------

// Name des Packages
package aoi_java;

//Imports aus Standard-Bibliotheken
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

//Klassenkopf (Start der Klasse)
public class Chart_Options extends JDialog
{
    // Globale Variablen deklarieren
    // Filter-Variablen
    private static JComboBox<String> cbFace;
    private static JComboBox<String> cbError;
    private static JComboBox<String> cbFilter;

    // Statusvariable des Dialogfelds
    private static boolean abort;

    // Konstruktor der Klasse
    public Chart_Options()
    {
        // Statusvariable initialisieren
        // ist solange true, bis der Benutzer den Dialog mit ok bestätigt
```

```
41          abort = true;
42
43          // Neues modales Dialogfenster erstellen
44          JDialog dialog = new JDialog(this, "Diagrammoptionen", true);
45
46          // Größe und Position anpassen
47          dialog.setSize(360, 180);
48          dialog.setLocationRelativeTo(null);
49          dialog.setLayout(new BorderLayout(0, 0));
50
51          // Größe nicht veränderbar
52          dialog.setResizable(false);
53
54          // Neues Panel für die Filter in den Dialog einfügen
55          JPanel panel = new JPanel();
56          panel.setLayout(new GridLayout(0, 2, 10, 20));
57          panel.setBorder(new EmptyBorder(10, 10, 10, 10));
58
59          // Neues Panel für die Schaltflächen einfügen
60          JPanel panel_btn = new JPanel();
61          panel_btn.setLayout(new GridLayout(0, 4, 0, 4));
62          panel_btn.setBorder(new EmptyBorder(6, 6, 6, 6));
63
64          // "OK"- und "Abbrechen"-Schaltflächen hinzufügen
65          JButton btn_OK = new JButton("OK");
66          JButton btn_Cancel = new JButton("Abbrechen");
67          panel_btn.add(new JLabel(""));
68          panel_btn.add(btn_OK);
69          panel_btn.add(btn_Cancel);
70          panel_btn.add(new JLabel(""));
71
72          // Bezeichner und Drop-Down-Box für die Seitenauswahl einfügen
73          panel.add(new JLabel ("Seite(n): "));
74          cbFace = new JComboBox<String>();
75          cbFace.addItem("ALLE");
76          cbFace.addItem("nur Top");
77          cbFace.addItem("nur Bottom");
78          panel.add(cbFace);
79
80          // Bezeichner und Drop-Down-Box für die Fehlertype einfügen
```

```
81          panel.add(new JLabel ("Fehlertype(n): "));
82          cbError = new JComboBox<String>();
83          cbError.addItem("ALLE");
84          cbError.addItem("nur Pseudo");
85          cbError.addItem("nur echte");
86          cbError.setSelectedIndex(2);
87          panel.add(cbError);
88
89          // Bezeichner und Drop-Down-Box für das Fehlermerkmal einfügen
90          panel.add(new JLabel ("Filtern nach: "));
91          cbFilter = new JComboBox<String>();
92          cbFilter.addItem("Position");
93          cbFilter.addItem("Bauteil");
94          cbFilter.addItem("Gehäuseform");
95          cbFilter.addItem("Art");
96          panel.add(cbFilter);
97
98          dialog.add(panel, BorderLayout.CENTER);
99          dialog.add(panel_btn, BorderLayout.SOUTH);
100
101         // Action-Listener für die "Abbrechen"-Schaltfläche
102         btn_Cancel.addActionListener(new ActionListener() {
103             public void actionPerformed(ActionEvent e) {
104
105                 // Dialog schließen
106                 dialog.dispose();
107             }
108         });
109
110         // Action-Listener für die "OK"-Schaltfläche
111         btn_OK.addActionListener(new ActionListener() {
112             public void actionPerformed(ActionEvent e) {
113
114                 // Statusvariable umschalten
115                 abort = false;
116
117                 // Dialog schließen
118                 dialog.dispose();
119             }
120         });
```

```
121
122         // Dialog sichtbar machen
123         dialog.setVisible(true);
124     }
125
126     // Getter-Methoden
127     // Gewählte Seite zurückgeben
128     public static String getFace()
129     {
130         return cbFace.getSelectedItem().toString();
131     }
132
133     // Gewählte Fehlertype zurückgeben
134     public static String getError()
135     {
136         return cbError.getSelectedItem().toString();
137     }
138
139     // Gewähltes Fehlermerkmal zurückgeben
140     public static String getFilter()
141     {
142         return cbFilter.getSelectedItem().toString();
143     }
144
145     // Statusvariable zurückgeben
146     public static boolean getAbort()
147     {
148         return abort;
149     }
150 }
151
```

```java
// ------------------------------------------------------------------------------------
// ---------- Diplomarbeit: Fehlerdatenauswertung in Java ------------------------------
// ---------- Hochschule Mittweida, 2017 -----------------------------------------------
// ---------- Autor: Strohmaier Markus -------------------------------------------------
// ------------------------------------------------------------------------------------

// ------- Klasse Date_Handling.java -----------------------------------------------------

// Name des Packages
package aoi_java;

// Imports aus Standard-Bibliotheken
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Properties;
import javax.swing.JFormattedTextField.AbstractFormatter;

// Imports der externen JDatePicker-Bibliothek
import org.jdatepicker.impl.JDatePanelImpl;
import org.jdatepicker.impl.JDatePickerImpl;
import org.jdatepicker.impl.UtilDateModel;

// Klassenkopf (Start der Klasse)
public class Date_Handling {

    // Globale Variable für die Datumsformatierung
    private static SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

    // Methode zum Erstellen eines neuen DatePickers
    public static JDatePickerImpl createDatePicker()
    {
        // Parameter zum Erstellen des JDatePickers
        UtilDateModel udm = new UtilDateModel();
        Properties p = new Properties();
        p.put("text.today", "Heute");

        // Generierung eines neuen DatePanels und DatePickers
        JDatePanelImpl datePanel = new JDatePanelImpl(udm, p);
        JDatePickerImpl datePicker = new JDatePickerImpl(datePanel, new AbstractFormatter() {
```

```
41              @Override
42              // Überschreiben der Methode stringToValue des Standard-AbstractFormatters
43              public Object stringToValue(String text) throws ParseException {
44
45                  // Umwandlung des Textes in das gewünschte Datumsformat und Rückgabe
46                  return dateFormat.parseObject(text);
47              }
48              @Override
49              // Überschreiben der Methode valueToString des Standard-AbstractFormatters
50              public String valueToString(Object value) throws ParseException {
51
52                  // Umwandlung nur wenn ein Wert entahlten ist
53                  if (value != null) {
54
55                      // Wert ins "Calendar"-Format umspeichern
56                      Calendar cal = (Calendar) value;
57
58                   // Umwandlung des Calendar-Wertes in das gewünschte Datumsformat und Rückgabe
59                      return dateFormat.format(cal.getTime());
60                  }
61
62                  // bei leerem Wert => leere Rückgabe
63                  return "";
64              }
65          });
66
67          // Rückgabe des Date-Pickers
68          return datePicker;
69      }
70
71      // Methode zum Ermitteln des heutigen Tages
72      public static String getToday()
73      {
74          // neue Calendar-Instanz erstellen
75          Calendar cal = Calendar.getInstance();
76
77          // Umwandlung des Calendar-Wertes in das gewünschte Datumsformat und Rückgabe
78          return dateFormat.format(cal.getTime());
79      }
80
```

```java
    // Methode zum Ermitteln der letzten Kalenderwoche (Mo-So)
    public static String[] getLastYearWeek()
    {
        // neue Calendar-Instanz erstellen
        Calendar cal = Calendar.getInstance();

        // Feld mit 2 leeren Stringwerten erstellen
        String[] dates = {"", ""};

        // aktuelle Kalenderwoche abfrgaen
        int yearWeek = cal.get(Calendar.WEEK_OF_YEAR);

        // erstellte Kalenderinstanz manipluieren
        // Woche um 1 verringern
        cal.set(Calendar.WEEK_OF_YEAR, yearWeek-1);

        // Tag auf Montag stellen
        cal.set(Calendar.DAY_OF_WEEK, Calendar.MONDAY);

        // Datum und Zeit abfragen, formatieren und in das erste Feld (Startdatum) schreiben
        dates[0] = dateFormat.format(cal.getTime());

        // Tag auf Sonntag stellen
        cal.set(Calendar.DAY_OF_WEEK, Calendar.SUNDAY);

        // Datum und Zeit abfragen, formatieren und in das zweite Feld (Enddatum) schreiben
        dates[1] = dateFormat.format(cal.getTime());

        // Datumswerte rückgeben
        return dates;
    }

    // Methode zum Ermitteln des letzten Monats
    public static String[] getLastMonth()
    {
        // neue Calendar-Instanz erstellen
        Calendar cal = Calendar.getInstance();

        // Feld mit 2 leeren Stringwerten erstellen
        String[] dates = {"", ""};
```

```java

        // aktuelles Monat ermitteln
        // Jänner = 0, Dezember = 11
        int month = cal.get(Calendar.MONTH);

        // Wenn das Monat Jänner ist
        if(month == 0)
        {
            // Monatsvariable auf "Dezember+1" setzen
            month = 12;

            // Jahr um 1 verringern
            cal.set(Calendar.YEAR, cal.get(Calendar.YEAR)-1);
        }

        // Monat um 1 verringern
        cal.set(Calendar.MONTH, month-1);

        // Tag auf den 1. des Monats stellen
        cal.set(Calendar.DAY_OF_MONTH, 1);

        // Datum und Zeit abfragen, formatieren und in das erste Feld (Startdatum) schreiben
        dates[0] = dateFormat.format(cal.getTime());

        // Tag auf den letzten des Monats stellen
        cal.set(Calendar.DAY_OF_MONTH, cal.getActualMaximum(Calendar.DAY_OF_MONTH));

        // Datum und Zeit abfragen, formatieren und in das zweite Feld (Enddatum) schreiben
        dates[1] = dateFormat.format(cal.getTime());

        // Datumswerte rückgeben
        return dates;
    }

    // Methode zum Ermitteln des vorletzten Monats
    public static String[] getMonthBeforeLast()
    {
        // neue Calendar-Instanz erstellen
        Calendar cal = Calendar.getInstance();

```

```java
        // Feld mit 2 leeren Stringwerten erstellen
        String[] dates = {"", ""};

        // aktuelles Monat ermitteln
        // Jänner = 0, Dezember = 11
        int month = cal.get(Calendar.MONTH);

        // Wenn das Monat Jänner ist
        if(month == 0)
        {
            // Monatsvariable auf "Dezember+1" setzen
            month = 12;

            // Jahr um 1 verringern
            cal.set(Calendar.YEAR, cal.get(Calendar.YEAR)-1);
        }

        // Wenn das Monat Februar ist
        if(month == 1)
        {
            // Monatsvariable auf "Dezember+2" setzen
            month = 13;

            // Jahr um 1 verringern
            cal.set(Calendar.YEAR, cal.get(Calendar.YEAR)-1);
        }

        // Monat um 2 verringern
        cal.set(Calendar.MONTH, month-2);

        // Tag auf den 1. des Monats stellen
        cal.set(Calendar.DAY_OF_MONTH, 1);

        // Datum und Zeit abfragen, formatieren und in das erste Feld (Startdatum) schreiben
        dates[0] = dateFormat.format(cal.getTime());

        // Tag auf den letzten des Monats stellen
        cal.set(Calendar.DAY_OF_MONTH, cal.getActualMaximum(Calendar.DAY_OF_MONTH));

        // Datum und Zeit abfragen, formatieren und in das zweite Feld (Enddatum) schreiben
```

```
201         dates[1] = dateFormat.format(cal.getTime());
202
203         // Datumswerte rückgeben
204         return dates;
205     }
206 }
207
```

```java
// ------------------------------------------------------------------------------------
// ---------- Diplomarbeit: Fehlerdatenauswertung in Java ------------------------------
// ---------- Hochschule Mittweida, 2017 -----------------------------------------------
// ---------- Autor: Strohmaier Markus -------------------------------------------------
// ------------------------------------------------------------------------------------

// ---------- Klasse FPY_Handling.java -------------------------------------------------

// Name des Packages
package aoi_java;

// Imports aus Standard-Bibliotheken
import java.awt.BorderLayout;
import java.sql.SQLException;
import java.text.DecimalFormat;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

// Klassenkopf (Start der Klasse)
public class FPY_Handling {

    // Methode zur Berechnung des First Pass Yields
    public static void calcFPY() throws SQLException
    {
        // Variablen deklarieren und initialisieren
        boolean botFlag = false;
        int quantity = 0, ok = 0, pseudo = 0, real = 0;
        int qtySum = 0, okSum = 0, pseudoSum = 0, realSum = 0;
        double fpy1 = 0, fpy2 = 0,  fpy1Sum = 0, fpy2Sum = 0;

        // neues Standard-Tabellenmodell erzeugen
        DefaultTableModel dtmFPY = new DefaultTableModel() {

            // Zellen sollen nicht editierbar sein
            //override cell editable
            public boolean isCellEditable(int row, int column)
            {
```

```
41                  return false;
42              }
43          };
44
45          // Neue Tabelle mit dem Tabellenmodell anlegen
46          JTable tableFPY = new JTable(dtmFPY);
47
48          // Spalten einfügen
49          dtmFPY.addColumn("Production Order");
50          dtmFPY.addColumn("Face");
51          dtmFPY.addColumn("Inspected Cards");
52          dtmFPY.addColumn("Cards OK");
53          dtmFPY.addColumn("False Failure Cards");
54          dtmFPY.addColumn("FPY False Failure");
55          dtmFPY.addColumn("Real Failure Cards");
56          dtmFPY.addColumn("FPY Real Failures");
57
58          // Automatische Formatierung für die FPYs in Prozent mit 2 Nachkommstellen
59          DecimalFormat df = new DecimalFormat("0.00%");
60
61          // Solange neue Daten aus der Datenbankabfragae vorhanden sind...
62          while(SQL_Handling.getResultSet().next())
63          {
64              // Abfrage, ob es sich um die Bottom-Seite handelt
65              if(SQL_Handling.getResultSet().getObject(2).toString().equals("BOTTOM"))
66                  botFlag = true;
67
68              // Abfrage der Leiterplatten ohne Fehler
69              if(SQL_Handling.getResultSet().getObject(3) == null)
70                  ok = (int) SQL_Handling.getResultSet().getObject(1);
71
72              // Abfrage der Leiterplatten mit echten Fehlern
73              else if (SQL_Handling.getResultSet().getObject(3).toString().equals("_real"))
74                  real = (int) SQL_Handling.getResultSet().getObject(1);
75
76              // Abfrage der gesamten geprüften Leiterplatten
77              else
78                  quantity = (int) SQL_Handling.getResultSet().getObject(1);
79
80              // Wenn geprüfte Leiterplatten vorhanden sind
```

```
81              if (quantity > 0)
82              {
83                  // Berechnung der Pseudfehler => "alle Leiterplatten" - "Leiterplatten mit echten Fehlern" - "Leiteplatten ohne Fehler"
84                  pseudo = quantity - real - ok;
85
86                  // Berechnung der FPYs
87                  // FPY1 = Fehlerratenberechnung alle Fehler (inkl. Pseudofehler)
88                  fpy1 = (double) ok/quantity;
89
90                  // FPY2 = Fehlerratenberechnung nur mit echten Fehlern
91                  fpy2 = (double) real/quantity;
92
93                  // neue Zeile mit den berechneten Daten zum Tabellenmodell hinzufügen
94                  dtmFPY.addRow(new Object[] {SQL_Handling.getResultSet().getObject(4).toString(),
95                          (botFlag ? "BOTTOM" : "TOP"), quantity, ok, pseudo, df.format(fpy1), real, df.format(fpy2)});
96
97                  // Fehler-Werte aufsummieren
98                  qtySum += quantity;
99                  okSum += ok;
100                 pseudoSum += pseudo;
101                 realSum += real;
102
103                 // Initialisierungszustand der Variablen wiederherstellen
104                 quantity = 0;
105                 ok = 0;
106                 real = 0;
107                 pseudo = 0;
108                 fpy1 = 0;
109                 fpy2 = 0;
110                 botFlag = false;
111             }
112         }
113
114         // neues Panel für die Tabellenanzeige erstellen
115         JPanel panelFPY = new JPanel();
116         panelFPY.setLayout(new BorderLayout(0, 0));
117
118         // Wenn Daten vorhanden
119         if (qtySum > 0)
120         {
```

```
121             // Durchschnittswert der FPYs berechnen
122             fpy1Sum = (double) okSum/qtySum;
123             fpy2Sum = (double) (okSum+pseudoSum)/qtySum;
124 
125             // neue Zeile mit den summierten Werten zum Tabellenmodell hinzufügen
126             dtmFPY.addRow(new Object[] {"SUMME", "", qtySum, okSum, pseudoSum,
127                     df.format(fpy1Sum), realSum, df.format(fpy2Sum)});
128 
129             // Scrollbalken-Panel und die Tabelle einfügen
130             JScrollPane scrollPane = new JScrollPane(tableFPY);
131             panelFPY.add(scrollPane, BorderLayout.CENTER);
132 
133             // Panel mit der enthaltenem Tabelle als neuen Tab im Hauptfenster hinzufügen
134             _Main_Window.getTabPane().addTab("FPY", null, panelFPY, null);
135 
136             // direkt zum neu hinzugefügten Tab springen
137             _Main_Window.getTabPane().setSelectedIndex(_Main_Window.getTabPane().getTabCount() - 1);
138         }
139         // Wenn keine Daten vorhanden - Dialogfenster öffnen
140         else
141             JOptionPane.showMessageDialog(panelFPY, "keine Daten vorhanden");
142     }
143 }
144 
```

```java
// ------------------------------------------------------------------------------------
// ---------- Diplomarbeit: Fehlerdatenauswertung in Java ---------------------------
// ---------- Hochschule Mittweida, 2017 --------------------------------------------
// ---------- Autor: Strohmaier Markus ----------------------------------------------
// ------------------------------------------------------------------------------------

// ---------- Klasse SQL_Handling.java ----------------------------------------------

// Name des Packages
package aoi_java;

// Imports aus Standard-Bibliotheken
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

// Klassenkopf (Start der Klasse)
public class SQL_Handling {

    // Globale Variablen initialisieren
    private static Connection conn = null;
    private static ResultSet resultSet = null;

    // Methode zum Verbindungsaufbau mit der Datenbank
    public static void connect() throws SQLException
    {
        // Angabe der Parameter (Server, Datenbankname, integratedSecurity => Windows Authentication)
        String server = "seidel-db03";
        String database = "AOI_VISION20_11_2016";
        String dbURL = "jdbc:sqlserver://" + server + ";databaseName=" + database + ";integratedSecurity=true";

        // Verbindung über die URL anfordern
        conn = DriverManager.getConnection(dbURL);

        // Wenn Verbindung erfolgreich
        if (conn != null)
        {
```

```
41            // Ausgabe der Verbindungsdetails in der Konsole
42            DatabaseMetaData dm = (DatabaseMetaData) conn.getMetaData();
43            System.out.println("Driver name: " + dm.getDriverName());
44            System.out.println("Driver version: " + dm.getDriverVersion());
45            System.out.println("Product name: " + dm.getDatabaseProductName());
46            System.out.println("Product version: " + dm.getDatabaseProductVersion());
47            System.out.println("\nConnection to " + database + " on " + server + " established");
48        }
49    }
50
51    // Methode zum Trennen der Datenbankverbindung
52    public static void disconnect() throws SQLException
53    {
54        // Wenn eine Verbindung besteht
55        if (conn != null && !conn.isClosed())
56        {
57            // Verbindung schließen und Info in der Konsole anzeigen
58            conn.close();
59            conn = null;
60            System.out.println("\n---\nDisconnected from Server\n---");
61        }
62    }
63
64    // Methode zur Abfrage von Daten (Übergabewert ist die Abfrage)
65    public static void query(String query) throws SQLException
66    {
67        // Statement erstellen
68        Statement stmt = conn.createStatement();
69
70        // Abfrage durchführen und Daten in "resultSet" speichern
71        resultSet = stmt.executeQuery(query);
72    }
73
74    // Getter-Methode für das ResultSet (zum Abfragen der Daten)
75    public static ResultSet getResultSet() {
76        return resultSet;
77    }
78
79    // Getter-Methode für die Verbindung (zum Abfragen des Verbindungsstatus)
80    public static Connection getConnection() {
```

```
81          return conn;
82      }
83  }
84
```

```java
// ------------------------------------------------------------------------------------
// ---------- Diplomarbeit: Fehlerdatenauswertung in Java ---------------------------
// ---------- Hochschule Mittweida, 2017 --------------------------------------------
// ---------- Autor: Strohmaier Markus ----------------------------------------------
// ------------------------------------------------------------------------------------

// ---------- Klasse Table_Handling.java --------------------------------------------

// Name des Packages
package aoi_java;

//Imports aus Standard-Bibliotheken
import java.awt.BorderLayout;
import java.awt.Desktop;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.sql.SQLException;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.ListSelectionModel;
import javax.swing.RowFilter;
import javax.swing.SwingConstants;
import javax.swing.border.EmptyBorder;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;
```

```java
import javax.swing.table.TableRowSorter;

//Klassenkopf (Start der Klasse)
public class Table_Handling
{
    // Methode zum Exportieren der Tabelle als *.csv-Datei
    public static void exportToCSV(DefaultTableModel dtm) throws IOException
    {
        // Datei-Auswahldialog öffnen
        JFileChooser chooser = new JFileChooser();

        // CSV-Filter anlegen und zuweisen
        FileNameExtensionFilter filter = new FileNameExtensionFilter("CSV-Datei", "csv");
        chooser.setFileFilter(filter);

        // Wenn der Auswahldialog mit OK bestätigt wird
        if (chooser.showSaveDialog(null) == JFileChooser.APPROVE_OPTION)
        {
            // Eingegebenen Dateinamen abfragen
            String name = chooser.getSelectedFile().toString();

            // Dateiendung abfragen und ggf. ergänzen
            if(!name.endsWith(".csv"))
                name += ".csv";

            // neue Datei und Bufferspeicher erstellen
            File file = new File(name);
            BufferedWriter bw = new BufferedWriter(new FileWriter(file));

            // Spaltennamen abfragen und mit Trennzeichen ';' in den Buffer schreiben
            for (int h = 0 ; h < dtm.getColumnCount();h++)
            {
                bw.write(dtm.getColumnName(h).toString());
                if (h+1 != dtm.getColumnCount())
                  bw.write(";");
            }
            // in nächste Zeile wechseln
            bw.newLine();

            // Reihen nacheinander abfragen und mit Trennzeichen ';' in Buffer schreiben
```

```
81              for (int i = 0; i < dtm.getRowCount(); i++)
82              {
83                  for (int j = 0; j < dtm.getColumnCount(); j++)
84                  {
85                      String value = dtm.getValueAt(i, j).toString();
86                      bw.write(value);
87                      bw.write(";");
88                  }
89                  // in nächste Zeile wechseln
90                  bw.newLine();
91              }
92              // gepufferte Daten in die Datei schreiben und Bufferspeicher schließen
93              bw.flush();
94              bw.close();
95          }
96      }
97
98      // Methode zum Öffnen der Fehlerbilder
99      public static void openImageFolder(JTable table, JPanel panel) throws IOException
100     {
101         // Ausgewählte Zeile ermitteln
102         int selRow = table.getSelectedRow();
103
104         // Wenn eine Zeile ausgewählt, dann...
105         if (selRow >= 0)
106         {
107             // Erforderliche Zeilendaten in lokale Variablen speichern
108             String panelID = table.getValueAt(selRow, 0).toString();
109             String face = table.getValueAt(selRow, 6).toString();
110             String jedec = table.getValueAt(selRow, 12).toString();
111             String topology = table.getValueAt(selRow, 10).toString();
112             String error = table.getValueAt(selRow, 13).toString();
113             String file = "";
114
115             // Fehlertype der selektierten Zeile ermitteln
116             switch(error)
117             {
118                 // '-' => keine Fehler am Board
119                 case "-": JOptionPane.showMessageDialog(panel, "keine Fehler am ausgewählten Board");break;
120
```

```
121                 // '810' => Pseudofehler
122                 case "810": error = "Gut\\"; break;
123 
124                 // alles andere => echter Fehler
125                 default: error = "Repariert\\"; break;
126             }
127 
128             // Wenn Pseudofehler oder echter Fehler, dann...
129             if(!error.equals("-"))
130             {
131                 // Pfad zum Fehlerbild angeben
132                 String directory = "\\\\see0004651\\D\\RepPlatz-Bilder\\" + error +
133                         panelID + "_" + face + "\\";
134 
135                 // Alle Dateien im Pfad ermitteln
136                 File[] filesInDir = new File(directory).listFiles();
137 
138                 // Alle Dateinamen im Pfad durchsuchen
139                 for(int i = 0; i < filesInDir.length; i++)
140 
141                     // Wenn Dateiname mit Fehlerinformationen übereinstimmt, dann...
142                     if(filesInDir[i].toString().startsWith(directory + jedec + "_" + topology))
143 
144                         // Dateiname zwischenspeichern
145                         file = filesInDir[i].toString();
146 
147                 // Datei öffnen
148                 Desktop.getDesktop().open(new File(file));
149             }
150         }
151     }
152 
153     // Tabellen-GUI erzeugen
154     public static void createTable (long order) throws SQLException {
155 
156         // Standard-Tabellenmodel anlegen mit nicht veränderbaren Zellen
157         DefaultTableModel dtm = new DefaultTableModel()
158         {
159             // Zellen sollen nicht editierbar sein
160             //override cell editable
```

```
161             public boolean isCellEditable(int row, int column)
162             {
163                 return false;
164             }
165         };
166 
167         // Drop-Down-Box für den Filtererstellen
168         JComboBox<String> filterBox = new JComboBox<String>();
169 
170         // Daten für die Tabelle generieren
171         createTableData(dtm, filterBox);
172 
173         // Tabelle mit angepassten Parametern erstellen
174         // Auto-Sortierung einschalten (beim Klick auf den Spaltennamen)
175         // Nach Spalte 2 sortieren (=Board_ID_Code)
176         // Nur Einzelselektion erlauben (für die Fehlerbild-Anzeige notwendig)
177         JTable table = new JTable(dtm);
178         table.setAutoCreateRowSorter(true);
179         table.getRowSorter().toggleSortOrder(1);
180         table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
181 
182         // neues Panel erstellen
183         JPanel panel = new JPanel();
184         panel.setLayout(new BorderLayout(0, 0));
185 
186         // Panel in neuem Tab einfügen
187         _Main_Window.getTabPane().addTab("Tabelle " + order, null, panel, null);
188 
189         // direkt zum neu hinzugefügten Tab springen
190         _Main_Window.getTabPane().setSelectedIndex(_Main_Window.getTabPane().getTabCount() - 1);
191 
192         // Scrollbalken-Panel und die Tabelle einfügen
193         JScrollPane scrollPane = new JScrollPane(table);
194         panel.add(scrollPane, BorderLayout.CENTER);
195 
196         // Neues Panel für den Filter (North = Oben) einfügen
197         JPanel panelFilter = new JPanel();
198         panelFilter.setLayout(new BorderLayout(0, 0));
199         panelFilter.setBorder(new EmptyBorder(3, 3, 5, 3));
200         panel.add(panelFilter, BorderLayout.NORTH);
```

```
201
202         // Drop-Down-Box zur Filterauswahl einfügen
203         panelFilter.add(filterBox, BorderLayout.EAST);
204
205         // Bezeichner einfügen
206         JLabel lblFilter = new JLabel("Tabellenfilter: ");
207         panelFilter.add(lblFilter, BorderLayout.WEST);
208
209         // Textfeld zur Filtereingabe einfügen
210         JTextField txtFilter = new JTextField();
211         panelFilter.add(txtFilter, BorderLayout.CENTER);
212         txtFilter.setToolTipText("Filtern");
213         txtFilter.setHorizontalAlignment(SwingConstants.CENTER);
214         txtFilter.setColumns(10);
215
216         // Listener für das Textfeld erstellen (wird bei Tastatureingaben aktiv)
217         txtFilter.addKeyListener(new KeyAdapter()
218         {
219             public void keyReleased(KeyEvent arg0)
220             {
221                 // Benutzerdefinierter Sortierer zum Standard-Tabellenmodel hinzufügen
222                 TableRowSorter<DefaultTableModel> sorter = new TableRowSorter<DefaultTableModel>(dtm);
223
224                 // Filter zum Sortierer hinzufügen
225                 // Regex Filter = Zeichenfilter (?i = Groß-/Kleinschreibung ignorieren)
226                 // zu filternden Text übergeben und zu filternde Tabellenspalte übergeben
227                 sorter.setRowFilter(RowFilter.regexFilter("(?i)" +
228                 txtFilter.getText(), filterBox.getSelectedIndex()));
229
230                 // Sortierer bzw. Filter auf die Tabelle anwenden
231                 table.setRowSorter(sorter);
232             }
233         });
234
235         // Neues Panel für die Schaltflächen (South = Unten) einfügen
236         JPanel panelButtons = new JPanel();
237         panelButtons.setLayout(new GridLayout(0, 1, 0, 5));
238         panel.add(panelButtons, BorderLayout.SOUTH);
239         panelButtons.setBorder(new EmptyBorder(5, 0, 5, 0));
240
```

```
241         // Schaltfläche für die Fehlerbildanzeige einfügen
242         JButton btnImage = new JButton("Fehlerbilder anzeigen");
243         btnImage.setFont(new Font("Tahoma", Font.PLAIN, 20));
244         panelButtons.add(btnImage);
245 
246         // Action-Listener für die Fehlerbild-Schaltfläche
247         btnImage.addActionListener(new ActionListener() {
248             public void actionPerformed(ActionEvent e) {
249                 try
250                 {
251                     // Lokale Methode zum Öffnen der Bilder aufrufen
252                     openImageFolder(table, panel);
253                 }
254                 // Bei Fehler (keine Bilder vorhanden)
255                 catch (Exception e_image)
256                 {
257                     // Dialog mit unten stehender Information anzeigen
258                     JOptionPane.showMessageDialog(panel, "Fehlerbilder wurden bereits archiviert");
259                 }
260             }
261         });
262 
263 
264         // Schaltfläche für CSV-Export einfügen
265         JButton btnExport = new JButton("Tabelle exportieren");
266         btnExport.setFont(new Font("Tahoma", Font.PLAIN, 20));
267         panelButtons.add(btnExport);
268 
269         // Action-Listener für die Export-Schaltfläche
270         btnExport.addActionListener(new ActionListener()
271         {
272             public void actionPerformed(ActionEvent e)
273             {
274                 // Neuen Thread mit den folgenden Informationen erstellen
275                 new Thread()
276                 {
277                     public void run()
278                     {
279                         try
280                         {
```

```
281                         // Lokale Methode zum Exportieren aufrufen
282                         exportToCSV(dtm);
283                     }
284                     // Bei Fehler(n)
285                     catch (IOException e)
286                     {
287                         // Fehlerdetails in der Konsole ausgeben
288                         e.printStackTrace();
289                     }
290                 }
291             // Thread starten
292             }.start();
293         }
294     });
295 }
296
297 // Methode zum Einlesen und Aufbereiten der Tabellendaten
298 public static void createTableData (DefaultTableModel dtm, JComboBox<String> filterBox) throws SQLException
299 {
300     // Anzahl der Spalten über den Ergebnissatz der SQL-Abfrage ermitteln
301     int colCount = SQL_Handling.getResultSet().getMetaData().getColumnCount();
302
303     // Spaltengenerierung der Tabelle
304     for (int colIndex = 0; colIndex < colCount; colIndex++)
305     {
306         // Spalten schreiben
307         dtm.addColumn(SQL_Handling.getResultSet().getMetaData().getColumnLabel(colIndex+1));
308
309         // Filterbox-Einträge ebenfalls schreiben
310         filterBox.addItem(SQL_Handling.getResultSet().getMetaData().getColumnLabel(colIndex+1));
311     }
312
313     // Reihengenerierung der Tabelle
314     // Eine Reihe wird als ein Objekt mit gespeichert
315     // Anzahl der Feldelemente ist abhängig von der Spaltenanzahl
316     Object[] row = new Object[colCount];
317
318     // Solange neue Ergbissätze anstehen
319     while(SQL_Handling.getResultSet().next())
320     {
```

```
321             for (int i = 0; i < colCount; i++)
322             {
323                 // NULL-Werte und leere Werte mit einem '-' überschreiben
324                 if (SQL_Handling.getResultSet().getObject(i+1) == null ||
325                         SQL_Handling.getResultSet().getObject(i+1).toString().equals(""))
326                     row[i] = "-";
327 
328                 // alle anderen Werte werden noch einmal separat abgefragt
329                 // NULL-Werte können nicht mit Switch/Case abgefragt werden
330                 else switch(SQL_Handling.getResultSet().getObject(i+1).toString())
331                 {
332                     // 0 = Code für 'OK'
333                     case "0": row[i] = "OK"; break;
334 
335                     // 32 = Code für 'NICHT OK'
336                     case "32": row[i] = "NOT OK"; break;
337 
338                     // alle anderen Werte so schreiben, wie sie in der Datenbank enthalten sind
339                     default: row[i] = SQL_Handling.getResultSet().getObject(i+1); break;
340                 }
341             }
342             // Reihe zur Tabelle hinzufuegen
343             dtm.addRow(row);
344         }
345     }
346 }
347 
```