Cab.java

```
1 import javax.swing.ImageIcon;
2
3 public class Cab extends ImageIcon {
4 /**
5 * Cabs zijn objecten die in de gui worden weergegeven als
plaatjes en een
6 * realistische weergave zijn van rijdende treinen waarmee ons
algoritme
7 * moet werken.
8 */
9 private static final long serialVersionUID = 1L;
10
11 public boolean zichtbaar, inwissel = false , running = false;
12 public int id, x, y;
13 public Sensor sensor;
14 ReisBeheer rb;
15 private int rxnr, rxoff;
16 public double positie, angle;
17 private double pl, rx, ry;
18 TreinBeheer tb;
19 public int tpositie =8;
20
21
22 public Cab(int sid, ReisBeheer rb,TreinBeheer tb) {
23 super( "./cab.gif");
24 this .tb= tb;
25 id = sid;
26 this .rb = rb;
27 sensor = new Sensor( sid, this);
28 setPositie(0);
29 setZichtbaarheid(false);
30 }
31
32 public void setZichtbaarheid(boolean z) {
33
34 zichtbaar = z;
35 }
36
37 public void start() {
38 running = true;
39 }
40
41 public void stop() {
42 running = false;
43 }
44
45 public void run() {
46 setPositie(positie + 0.05);
47 }
48
49 public void setPositie(double p) {
50 positie = p;
51 reken();
52 }
53
54 // Voer een reeks berekeningen uit om de positie
55 // van het treintje op het scherm te bepalen
56 private void reken() {
57 // Zorgen dat positie altijd lager is als 100%
58 if (positie > 100)
59 positie = positie % 100;
60 // Voortgang berekenen
61 localiseer();
62
63 // Bochten berekenen
64 localiseer2();
65
66 // Poll sensor
67 psensor();
68
69 }
70
71 // Zoek de ware coordinaten uit
72 private void localiseer() {
73 if (positie >= 0 && positie < 37.5) {
74 // Cab is boven
75 rxoff = (int) (Math.round(rx) % 350);
76 if (rxoff > 177 && rxoff < 183) {
77 rxnr = (int) Math.round((rx - 180) / 350);
78 checkWissel(rxnr);
79 }
80 wOffset(1);
81 rx = 81 + ((1092 / 37.5) * positie);
82 ry = 26 + pl;
83 } else if (positie >= 37.5 && positie < 50) {
84 // Cab is rechts
85 if (ry > 117 && ry < 123) {
86 checkWissel(3);
87 }
88 wOffset(2);
89 rx = 1174 - pl;
90 ry = 26 + ((379 / 12.5) * (positie - 37.5));
91 } else if (positie >= 50 && positie < 87.5) {
92 // Cab is onder
93 rxoff = (int) (Math.round(rx) % 350);
94 if (rxoff > 47 && rxoff < 53) {
95 rxnr = (int) (7 - Math.round((rx - 180) / 350));
96 checkWissel(rxnr);
97 }
98 wOffset(3);
99 rx = 1174 - ((1092 / 37.5) * (positie - 50));
100 ry = 405 + pl;
101 } else if (positie >= 87.5 && positie < 100) {
102 // Cab is links
```

```
if (ry > 327 && ry < 333) {
checkWissel(7);
}
wOffset(4);
rx = 81 + pl;
ry = 405 - ((379 / 12.5) * (positie - 87.5));
}
}

// draai het plaatje in de bochten mbv ware coordinaten
private void localiseer2() {
int ux = (int) rx;
int uy = (int) ry;
if (ux < 151 && uy < 96) {
// Linksboven
angle = getAngle(ux - 151 , uy - 96) + Math .PI;
rx = 151 + Math.cos(angle) * 70;
ry = 96 + Math.sin(angle) * 70;
} else if (ux > 1103 && uy < 96) {
// Rechtsboven
angle = getAngle(ux - 1103 , uy - 96) + (Math .PI * 2);
rx = 1104 + Math.cos(angle) * 70;
ry = 96 + Math.sin(angle) * 70;
} else if (ux > 1103 && uy > 335) {
// Rechtsonder
angle = getAngle(ux - 1103 , uy - 335);
rx = 1104 + Math.cos(angle) * 70;
ry = 335 + Math.sin(angle) * 70;
} else if (ux < 151 && uy > 335) {
// Linksonder
angle = getAngle(ux - 151 , uy - 335) + (Math .PI * 1);
rx = 151 + Math.cos(angle) * 70;
ry = 335 + Math.sin(angle) * 70;
}

x = (int) rx;
y = (int) ry;
}

// Bereken de door openstaande wissels ontstane offset
private void wOffset(int pos) {
if (pos == 1) {
// Cabs zijn boven
if (!(rxoff >= 48 && rxoff < 178) && inwissel) {
double rxoff2 = rx - (180 + (rxnr * 350));
if ( rxoff2 < 80.0) {
pl = Math.sin( rxoff2 / 73) * 50;
} else if ( rxoff2 >= 80 && rxoff2 <= 125) {
pl = 43;
} else if ( rxoff2 > 125 && rxoff2 < 200) {
pl = Math.sin(1 - (( rxoff2 - 125) / 73)) * 50;
} else {
pl = 0;
}

} else {
inwissel = false;
pl = 0;
}
} else if (pos == 3) {
// Cabs zijn onder
if (!(rxoff >= 52 && rxoff < 178) && inwissel) {
double rxoff2 = rx - (180 + ((6 - rxnr) * 350));
if ( rxoff2 > 0 && rxoff2 < 80.0) {
pl = Math.sin( rxoff2 / 73) * 50;
} else if ( rxoff2 >= 80 && rxoff2 <= 125) {
pl = 43;
} else if ( rxoff2 > 125 && rxoff2 < 200) {
pl = Math.sin(1 - (( rxoff2 - 125) / 73)) * 50;
} else {
pl = 0;
}
} else {
inwissel = false;
pl = 0;
}
} else if (pos == 2 || pos == 4) {
// Cabs zijn links of rechts
if (inwissel) {
if (ry >= 115 && ry <= 205) {
pl = Math.sin((ry - 115) / 73) * 45;
} else if (ry >= 205 && ry <= 250) {
pl = 43;
} else if (ry >= 250 && ry <= 320) {
pl = Math.sin(1 - ((ry - 250) / 73)) * 45;
} else {
pl = 0;
}
} else {
inwissel = false;
pl = 0;
}
}
}

// Controleer of de trein een sensor passeert
int
sens[]={10,11,12,13,20,21,22,23,30,31,32,33,40,41,42,43,50,51,52,
53,60,61,62,63,70,71
2,73,80,81,82,83};

private void psensor() {
int scheck = sensor.check();
if (scheck > 0) {


```

```
206
207 tb.sensorBericht(this .id,sens[scheck-1]);
208
209 // hij is langs sensor scheck stuur een bericht naar
treinbeheer
210 }
211 }
212
213 // Controleer of de trein een station inrijd
214 private void checkWissel(int wid) {
215 if ( rb.haltes.get(wid) .wissel.status()) {
216 inwissel = true;
217 } else {
218 inwissel = false;
219 }
220 }
221
222 // bereken hoek in graden
223 private double getAngle(double com1, double com2) {
224 // System.out.println(Math.cos(com1 / com2));
225 return Math.atan( com2 / com1);
226 }
227
228 }
```

DatabaseGegevens.java

```
1
2 import java.sql.*;
3
4 public class DatabaseGegevens{
5
6 public static void getData (){
7
8 String driver = "com.mysql.jdbc.Driver";
9
10 /* url: JDBC URL; parts one and two are supplied by your
driver,
11 * and the third part specifies your data source. */
12 String url = "jdbc:mysql://localhost/HvA";
13
14 // userid: your login name or user name
15 String userid = "root";
16
17 // password: your password for the DBMS
18 String password = "";
19
20 Connection conn = null;
21 Statement stmt = null;
22 ResultSet rs = null;
23
24 try{
25 // Load driver
26 Class.forName( driver );
27
28 // Class.forName( "the class name supplied with your driver" )
```

```
29 System .out.println( "Driver has been loaded." );
30
31 } catch( java.lang.ClassNotFoundException e ) {
32 System .err.print( "ClassNotFoundException: " );
33 System .err.println( e.getMessage() );
34 }
35
36 try{
37 // Make connection with the DB
38 conn = DriverManager.getConnection( url ,userid ,password );
39 // Create statement
40 stmt = conn.createStatement();
41 // Create resultset
42 rs = stmt.executeQuery( "SELECT * from Rails" );
43
44 if ( rs != null ){
45 while ( rs.next() ) {
46
47 // Get all Integer values from resultset
48 int id = rs.getInt( "id");
49 int vertrek = rs.getInt( "vertrek" );
50 int bestemming = rs.getInt( "bestemming" );
51 int passagiers = rs.getInt( "passagiers" );
52 int uren = rs.getInt( "uren" );
53 int minuten = rs.getInt( "minuten" );
54
55 // Print all values
56
System .out.println("-----------------------------------------
-----------------------
---");
57 System .out.println( "id = " + id + ", " + "vertrek = " +
vertrek +
" + "bestemming = "
58 + bestemming + ", " + "passagiers = " + passagiers + ", "
"tijdstip = " + uren + ":" + minuten + "u" );
59 }
60 }
61
62 // Close SQL statement
63 stmt.close();
64 stmt = null;
65
66 // Close connection with the DB
67 conn.close();
68 conn = null;
69
70 } catch( SQLException ex1 ) {
71 System .err.println( "SQLException: " + "1: " +
ex1.getMessage() );
72 //ex1.printStackTrace();
73 }
74
75 /* close any JDBC instances that weren't explicitly closed
76 * during normal code path, so we don't 'leak' resources. */
77 if ( stmt != null ) {
78 try{
79 stmt.close();
80
81 } catch( SQLException ex2 ) {
82 System .err.println( "SQLException: " + "2:" +
ex2.getMessage() );
83 }
84 }
85
86 if ( conn != null ){
87 try{
88 conn.close();
89
90 } catch( SQLException ex3 ){
91 System .err.println( "SQLException: " + "3: " +
ex3.getMessage() );
92 }
93 }
94 }
95 }
```

Grafiek3.java

```
1 import java.awt.Color;
2 import java.awt.Dimension;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 import org.jfree.chart.ChartFactory;
7 import org.jfree.chart.ChartPanel;
8 import org.jfree.chart.JFreeChart;
9 import org.jfree.chart.axis.NumberAxis;
10 import org.jfree.chart.plot.PlotOrientation;
11 import org.jfree.chart.plot.XYPlot;
12 import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
13 import org.jfree.data.category.CategoryDataset;
14 import org.jfree.data.category.DefaultCategoryDataset;
15 import org.jfree.data.xy.XYDataset;
16 import org.jfree.data.xy.XYSeries;
17 import org.jfree.data.xy.XYSeriesCollection;
18 import org.jfree.ui.ApplicationFrame;
19 import org.jfree.ui.RefineryUtilities;
20 import javax.swing.*;
21
22 /**
23 * A simple demonstration application showing how to create a
line chart using
24 * data from an {@link XYDataset}.
25 *
26 */
27 public class Grafiek3 extends JPanel {
28 public XYDataset dataset;
29 public JFreeChart chart;
30 public ChartPanel chartPanel;
31 private ReisBeheer rb;
32 private TreinBeheer tb;
33 private int teller = 0;
34 XYSeries series1, series2, series3, series4;
35
36 /**
37 * Creates a new demo.
38 *
39 * @param title
40 * the frame title.
41 */
42 public Grafiek3( XYDataset data, String x, String y, String
name) {
43
44 dataset = data;
45 chart = createChart( dataset, x, y, name);
46 chartPanel = new ChartPanel(chart);
47 chartPanel.setPreferredSize(new Dimension(980, 280));
48 add(chartPanel);
49 series1 = new XYSeries( "reizen");
50 series2 = new XYSeries( "treinen");
51 series3 = new XYSeries( "gemiddelde wachttijd");
52 series4 = new XYSeries( "gemiddelde reistijd");
53
54 }
55
56 /**
57 * Creates a sample dataset.
58 *
59 * @return a sample dataset.
60 *
61 */
62
63 public XYSeriesCollection createDataset(XYSeriesCollection
dataset) {
64
65 series1.add( teller, rb.getActieveReizen().size());
66 series2.add( teller, tb.treinen.size());
67 series3.add( teller, rb.gemWachttijd());
68 series4.add( teller, rb.gemReisttijd());
69
70 teller++;
71 dataset.removeAllSeries();
72 dataset.addSeries( series1);
73 dataset.addSeries( series2);
74 dataset.addSeries( series3);
75 dataset.addSeries( series4);
76 return dataset;
77
78 }
79
80 /**
81 * Creates a chart.
82 *
83 * @param dataset
84 * the data for the chart.
85 *
86 * @return a chart.
87 */
88 private JFreeChart createChart(final XYDataset dataset, String
x, String y,
89 String name) {
90
91 // create the chart...
```

```
92 final JFreeChart chart = ChartFactory.createXYLineChart( name,
// chart
93 // title
94 x, // x axis label
95 y, // y axis label
96 dataset, // data
97 PlotOrientation.VERTICAL, true, // include legend
98 true, // tooltips
99 false // urls
100 );
101
102 // NOW DO SOME OPTIONAL CUSTOMISATION OF THE CHART...
103 chart.setBackgroundPaint(Color .white);
104
105 // final StandardLegend legend = (StandardLegend)
chart.getLegend();
106 // legend.setDisplaySeriesShapes(true);
107
108 // get a reference to the plot for further customisation...
109 final XYPlot plot = chart.getXYPlot();
110 plot.setBackgroundPaint(Color .white);
111 // plot.setAxisOffset(new Spacer(Spacer.ABSOLUTE, 5.0, 5.0,
5.0, 5.0));
112 plot.setDomainGridlinePaint(Color .white);
113 plot.setRangeGridlinePaint(Color .red);
114
115 final XYLineAndShapeRenderer renderer = new
XYLineAndShapeRenderer();
116 renderer.setSeriesLinesVisible(0, true);
117 renderer.setSeriesShapesVisible(1, false);
118 plot.setRenderer(renderer);
119
120 // change the auto tick unit selection to integer units
only...
121 final NumberAxis rangeAxis = (NumberAxis)
plot.getRangeAxis();
122 rangeAxis.setStandardTickUnits(
NumberAxis.createIntegerTickUnits());
123
124 // OPTIONAL CUSTOMISATION COMPLETED.
125
126 return chart;
127
128 }
129
130 }
131
```

Halte.java

```
1 import java.util.LinkedList;
2 public class Halte
3 {
4 Wissel wissel = new Wissel();
5
6 int id;
7 public LinkedList<Reis> reizen;
```

```
8 int tijd;
9 int wachtenden;
10
11 public Halte(int id ){
12
13 reizen = new LinkedList<Reis>();
14 this .id= id;
15 }
16
17
18
19 public int getId (){
20 return id;
21 }
22
23 void addReis(Reis reis ){
24 //reis.setStatus(1);
25 if ( reis.gereserveerd ()){
26 reizen.addFirst(reis);
27 }
28
29 else{
30 reizen.addLast(reis);
31 }
32 System .out.println( "reis aan halte "+id+ " toegevoegd");
33
34 }
35
36 public void open(boolean op ){
37 wissel.set(op);
38 }
39
40
41 public Reis getEersteReis (){
42 if( reizen.isEmpty()) return null;
43 if( reizen.size()==1) return (Reis) reizen.getFirst();
44 Reis r0=(Reis) reizen.getFirst();
45 Reis r1=(Reis) reizen.get(1);
46 if(! r0.gereserveerd() && r1.gereserveerd())
47 return r1;
48 return r0;
49
50 }
51
52 public int getAantalWachtenden (){
53 return reizen.size();
54 }
55
56
57
58
59
60
61
62
63 }
64
```

MultiLineLabelUI.java

```
1 import java.awt.*;
2 import java.util.StringTokenizer;
3
4 import javax.swing.*;
5 import javax.swing.plaf.basic.*;
6
7 class MultiLineLabelUI extends BasicLabelUI {
8 static {
9 labelUI = new MultiLineLabelUI();
10 }
11
12 protected String layoutCL(JLabel label, FontMetrics
fontMetrics,
13 String text, Icon icon, Rectangle viewR, Rectangle iconR,
14 Rectangle textR) {
15 String s = layoutCompoundLabel((JComponent) label,
fontMetrics,
```

```
16 splitStringByLines(text) , icon, label.getVerticalAlignment(),
17 label.getHorizontalAlignment(),
18 label.getVerticalTextPosition() , label
19 .getHorizontalTextPosition() , viewR, iconR, textR,
20 label.getIconTextGap());
21
22 if ( s.equals(""))
23 return text;
24 return s;
25 }
26
27 static final int LEADING = SwingConstants .LEADING;
28 static final int TRAILING = SwingConstants .TRAILING;
29 static final int LEFT = SwingConstants .LEFT;
30 static final int RIGHT = SwingConstants .RIGHT;
31 static final int TOP = SwingConstants .TOP;
32 static final int CENTER = SwingConstants .CENTER;
33
34 /**
35 * Compute and return the location of the icons origin, the
location of
36 * origin of the text baseline, and a possibly clipped version
of the
37 * compound labels string. Locations are computed relative to
the viewR
38 * rectangle. The JComponents orientation (LEADING/TRAILING)
will also be
39 * taken into account and translated into LEFT/RIGHT values
accordingly.
40 */
41 public static String layoutCompoundLabel(JComponent c,
FontMetrics fm,
42 String[] text, Icon icon, int verticalAlignment,
43 int horizontalAlignment, int verticalTextPosition,
44 int horizontalTextPosition, Rectangle viewR, Rectangle iconR,
45 Rectangle textR, int textIconGap) {
46 boolean orientationIsLeftToRight = true;
47 int hAlign = horizontalAlignment;
48 int hTextPos = horizontalTextPosition;
49
50 if (c != null) {
51 if (!( c.getComponentOrientation().isLeftToRight())) {
52 orientationIsLeftToRight = false;
53 }
54 }
55
56 // Translate LEADING/TRAILING values in horizontalAlignment
57 // to LEFT/RIGHT values depending on the components
orientation
58 switch (horizontalAlignment) {
59 case LEADING:
60 hAlign = (orientationIsLeftToRight) ? LEFT : RIGHT;
61 break;
62 case TRAILING:
63 hAlign = (orientationIsLeftToRight) ? RIGHT : LEFT;
64 break;
65 }
66
67 // Translate LEADING/TRAILING values in horizontalTextPosition
68 // to LEFT/RIGHT values depending on the components
orientation
69 switch (horizontalTextPosition) {
70 case LEADING:
71 hTextPos = (orientationIsLeftToRight) ? LEFT : RIGHT;
72 break;
73 case TRAILING:
74 hTextPos = (orientationIsLeftToRight) ? RIGHT : LEFT;
75 break;
76 }
77
78 return layoutCompoundLabel( fm, text, icon, verticalAlignment,
hAlign,
79 verticalTextPosition, hTextPos, viewR, iconR, textR,
80 textIconGap);
81 }
82
83 /**
84 * Compute and return the location of the icons origin, the
location of
85 * origin of the text baseline, and a possibly clipped version
of the
86 * compound labels string. Locations are computed relative to
the viewR
87 * rectangle. This layoutCompoundLabel() does not know how to
handle
88 * LEADING/TRAILING values in horizontalTextPosition (they will
default to
89 * RIGHT) and in horizontalAlignment (they will default to
CENTER). Use the
90 * other version of layoutCompoundLabel() instead.
91 */
92 public static String layoutCompoundLabel(FontMetrics fm,
String[] text,
93 Icon icon, int verticalAlignment, int horizontalAlignment,
94 int verticalTextPosition, int horizontalTextPosition,
95 Rectangle viewR, Rectangle iconR, Rectangle textR, int
textIconGap) {
96 /*
97 * Initialize the icon bounds rectangle iconR.
98 */
99
100 if (icon != null) {
101 iconR.width = icon.getIconWidth();
102 iconR.height = icon.getIconHeight();
103 } else {
104 iconR.width = iconR.height = 0;
```

```
105 }
106
107 /*
108 * Initialize the text bounds rectangle textR. If a null or
and empty
109 * String was specified we substitute "" here and use 0,0,0,0
for textR.
110 */
111
112 // Fix for textIsEmpty sent by Paulo Santos
113 boolean textIsEmpty = (text == null)
114 || ( text.length == 0)
115 || ( text.length == 1 && ((text[0] == null) || text[0]
116 .equals("")));
117
118 String rettext = "";
119 if (textIsEmpty) {
120 textR.width = textR.height = 0;
121 } else {
122 Dimension dim = computeMultiLineDimension( fm, text);
123 textR.width = dim.width;
124 textR.height = dim.height;
125 }
126
127 /*
128 * Unless both text and icon are non-null, we effectively
ignore the
129 * value of textIconGap. The code that follows uses the value
of gap
130 * instead of textIconGap.
131 */
132
133 int gap = (textIsEmpty || (icon == null)) ? 0 : textIconGap;
134
135 if (!textIsEmpty) {
136
137 /*
138 * If the label text string is too wide to fit within the
available
139 * space "..." and as many characters as will fit will be
displayed
140 * instead.
141 */
142
143 int availTextWidth;
144
145 if (horizontalTextPosition == CENTER) {
146 availTextWidth = viewR.width;
147 } else {
148 availTextWidth = viewR.width - ( iconR.width + gap);
149 }
150
151 if ( textR.width > availTextWidth && text.length == 1) {
152 String clipString = "...";
153 int totalWidth = SwingUtilities.computeStringWidth( fm,
154 clipString);
155 int nChars;
156 for (nChars = 0 ; nChars < text[0].length() ; nChars++) {
157 totalWidth += fm.charWidth(text[0].charAt(nChars));
158 if (totalWidth > availTextWidth) {
159 break;
160 }
161 }
162 rettext = text[0].substring(0 , nChars) + clipString;
163 textR.width = SwingUtilities.computeStringWidth( fm,
rettext);
164 }
165 }
166
167 /*
168 * Compute textR.x,y given the verticalTextPosition and
169 * horizontalTextPosition properties
170 */
171
172 if (verticalTextPosition == TOP) {
173 if (horizontalTextPosition != CENTER) {
174 textR.y = 0;
175 } else {
176 textR.y = -( textR.height + gap);
177 }
178 } else if (verticalTextPosition == CENTER) {
179 textR.y = ( iconR.height / 2) - ( textR.height / 2);
180 } else { // (verticalTextPosition == BOTTOM)
181 if (horizontalTextPosition != CENTER) {
182 textR.y = iconR.height - textR.height;
183 } else {
184 textR.y = ( iconR.height + gap);
185 }
186 }
187
188 if (horizontalTextPosition == LEFT) {
189 textR.x = -( textR.width + gap);
190 } else if (horizontalTextPosition == CENTER) {
191 textR.x = ( iconR.width / 2) - ( textR.width / 2);
192 } else { // (horizontalTextPosition == RIGHT)
193 textR.x = ( iconR.width + gap);
194 }
195
196 /*
197 * labelR is the rectangle that contains iconR and textR. Move
it to its
198 * proper position given the labelAlignment properties.
199 *
200 * To avoid actually allocating a Rectangle, Rectangle.union
has been
201 * inlined below.
```

```
202 */
203 int labelR_x = Math.min( iconR.x, textR.x);
204 int labelR_width = Math.max( iconR.x + iconR.width, textR.x
205 + textR.width)
206 - labelR_x;
207 int labelR_y = Math.min( iconR.y, textR.y);
208 int labelR_height = Math.max( iconR.y + iconR.height, textR.y
209 + textR.height)
210 - labelR_y;
211
212 int dx, dy;
213
214 if (verticalAlignment == TOP) {
215 dy = viewR.y - labelR_y;
216 } else if (verticalAlignment == CENTER) {
217 dy = ( viewR.y + ( viewR.height / 2))
218 - ( labelR_y + ( labelR_height / 2));
219 } else { // (verticalAlignment == BOTTOM)
220 dy = ( viewR.y + viewR.height) - ( labelR_y + labelR_height);
221 }
222
223 if (horizontalAlignment == LEFT) {
224 dx = viewR.x - labelR_x;
225 } else if (horizontalAlignment == RIGHT) {
226 dx = ( viewR.x + viewR.width) - ( labelR_x + labelR_width);
227 } else { // (horizontalAlignment == CENTER)
228 dx = ( viewR.x + ( viewR.width / 2))
229 - ( labelR_x + ( labelR_width / 2));
230 }
231
232 /*
233 * Translate textR and glypyR by dx,dy.
234 */
235
236 textR.x += dx;
237 textR.y += dy;
238
239 iconR.x += dx;
240 iconR.y += dy;
241
242 return rettext;
243 }
244
245 protected void paintEnabledText(JLabel l, Graphics g, String
s, int textX,
246 int textY) {
247 int accChar = l.getDisplayedMnemonic();
248 g.setColor( l.getForeground());
249 drawString( g, s, accChar, textX, textY);
250 }
251
252 protected void paintDisabledText(JLabel l, Graphics g, String
s, int textX,
253 int textY) {
254 int accChar = l.getDisplayedMnemonic();
255 g.setColor( l.getBackground());
256 drawString( g, s, accChar, textX, textY);
257 }
258
259 protected void drawString(Graphics g, String s, int accChar,
int textX,
260 int textY) {
261 if ( s.indexOf( '\n') == -1)
262 BasicGraphicsUtils.drawString( g, s, accChar, textX, textY);
263 else {
264 String[] strs = splitStringByLines(s);
265 int height = g.getFontMetrics().getHeight();
266 // Only the first line can have the accel char
267 BasicGraphicsUtils.drawString( g, strs[0] , accChar, textX,
textY);
268 for (int i = 1 ; i < strs.length; i++)
269 g.drawString(strs[i] , textX, textY + (height * i));
270 }
271 }
272
273 public static Dimension computeMultiLineDimension(FontMetrics
fm,
274 String[] strs) {
275 int i, c, width = 0;
276 for (i = 0 , c = strs.length; i < c; i++)
277 width = Math.max( width, SwingUtilities.computeStringWidth(
fm,
278 strs[i]));
279 return new Dimension( width, fm.getHeight() * strs.length);
280 }
281
282 protected String str;
283 protected String[] strs;
284
285 public String[] splitStringByLines(String str) {
286 if ( str.equals(this .str))
287 return strs;
288
289 this .str = str;
290
291 int lines = 1;
292 int i, c;
293 for (i = 0 , c = str.length() ; i < c; i++) {
294 if ( str.charAt(i) == '\n')
295 lines++;
296 }
297 strs = new String[lines];
298 StringTokenizer st = new StringTokenizer( str, "\n");
299
300 int line = 0;
301 while ( st.hasMoreTokens())
```

```
302 strs[line++] = st.nextToken();
303
304 return strs;
305 }
306 }
307
```

RailsApp.java

```
1 import java.awt.GraphicsDevice;
2 import java.awt.GraphicsEnvironment;
3 import java.awt.event.ActionEvent;
4 import java.awt.event.KeyEvent;
5
6 import javax.swing.AbstractAction;
7 import javax.swing.Action;
8 import javax.swing.JComponent;
9 import javax.swing.JFrame;
10 import javax.swing.JPanel;
11 import javax.swing.KeyStroke;
12
13 public class RailsApp extends JFrame {
14 /**
15 * De default klasse die alle controllers aanmaakt en de gui
aanmaakt.
16 * Tevens zorgt deze klasse ervoor dat het scherm fullscreen
gaat en dat de
17 * applicatie kan worden afgesloten met de ESC toets.
18 */
19 private static final long serialVersionUID = 1L;
20 private JPanel paneel;
21 public Station station;
22 private int time, timediv;
23 public static int uur;
24 public static int min;
25 private RailScherm railScherm;
26 private HoofdMenu hoofdMenu;
27 private Station2 stationTwee;
28 private GraphicsDevice device;
29 private ReisBeheer reisBeheer;
30 private TreinBeheer treinBeheer;
31 private boolean isFullScreen = false;
32
33 public RailsApp(GraphicsDevice device) {
34 super( device.getDefaultConfiguration());
35 reisBeheer = new ReisBeheer(treinBeheer);
36
37 // Initialiseer tekenscherm (fullscreen)
38
39 this .device = device;
40 // Applicatie titel
41 setTitle( "RailCab Team 6 Project");
42 // Controller objecten aanmaken
43
44 // Hoofdpaneel instellen
45 paneel = new JPanel();
46 paneel.setLayout(null);
47
48 // hoofdmenu instellen
49 hoofdMenu = new HoofdMenu();
50 hoofdMenu.setBounds(1130, 0, 150, 280);
51
52 // onderscherm instellen
53 railScherm = new RailScherm( treinBeheer, reisBeheer);
54 railScherm.setBounds(0, 280, 1280, 520);
55 treinBeheer = new TreinBeheer( reisBeheer, railScherm);
56
57 // scherm Linksboven instellen
58 station = new Station();
59 station.setBounds(0, 0, 150, 280);
60
61 // scherm boven instellen
```

```
62 stationTwee = new Station2();
63 stationTwee.setBounds(150, 0, 980, 280);
64
65 // Atributen toevoegen
66 paneel.add(railScherm);
67 paneel.add(station);
68 paneel.add(stationTwee);
69 paneel.add(hoofdMenu);
70 setContentPane(paneel);
71
72 // Close on escape listener maken
73 Action escape = new EscapeAction();
74
75 getRootPane().getInputMap(JComponent
.WHEN_IN_FOCUSED_WINDOW).put(
76 KeyStroke.getKeyStroke( KeyEvent.VK_ESCAPE, 0) , escape);
77 getRootPane().getActionMap().put( escape, escape);
78
79 begin();
80 }
81
82 public void begin() {
83 isFullScreen = device.isFullScreenSupported();
84 setUndecorated(isFullScreen);
85 setResizable(!isFullScreen);
86 if (isFullScreen) {
87 // Full-screen mode
88 device.setFullScreenWindow(this);
89 validate();
90 } else {
91 // Windowed mode
92 pack();
93 setVisible(true);
94 }
95 Statistiek.refresh();
96 loop();
97 }
98
99 public void loop() {
100 while (true) {
101 for ( Trein cab : treinBeheer.alleTreinen) {
102 if ( cab.running) {
103 cab.run();
104 }
105 }
106 timediv++;
107 if (timediv >= 100) {
108 time++;
109 Statistiek.refresh();
110 timediv = 0;
111 }
112 min = time % 60;
113 uur = (time - min) / 60;
114 if (uur > 23)uur = uur % 24;
115 railScherm.repaint();
116 try {
117 Thread.sleep(10);
118 } catch (Exception E1) {
119 System .out.println( "Kon niet slapen: " + E1);
120 }
121 }
122 // loop();
123 }
124
125 public static void main(String Args[]) {
126 GraphicsEnvironment env = GraphicsEnvironment
127 .getLocalGraphicsEnvironment();
128 GraphicsDevice myDevice = env.getDefaultScreenDevice();
129 JFrame frame = new RailsApp(myDevice);
130 frame.setDefaultCloseOperation(JFrame .EXIT_ON_CLOSE);
131 }
132
133 class EscapeAction extends AbstractAction {
134 /**
135 * Deze klasse is een actionlistener voor de escape toets
136 */
137 private static final long serialVersionUID = 1L;
138
139 public void actionPerformed(ActionEvent e) {
140 System.exit(getDefaultCloseOperation());
141 }
142 }
143
144 }
```

RailScherm.java

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import java.text.NumberFormat;
4 import java.util.ArrayList;
5 import javax.swing.*;
6 import javax.swing.border.LineBorder;
7
8 public class RailScherm extends JPanel {
9 /**
10 * Het scherm waarin alle cabs en de baan worden getekend
11 */
12 private static final long serialVersionUID = 1L;
13
14 // Achtergrond afbeelding
15 private ImageIcon back = new ImageIcon( "./railsscherm.gif");
16
17 private static JLabel klok = new JLabel("",JLabel .CENTER);
18
19
20 // TreinBeheer referentie
21 private TreinBeheer treinBeheer;
22 private ReisBeheer reisBeheer;
23
24 public int uur;
25 public int min;
26
27 // Object Lists
28 public static ArrayList<JButton> stat = new
ArrayList<JButton>(); // Station
29 // Buttons
30 public static ArrayList<JLabel> cinfo = new
ArrayList<JLabel>(); // Cab info
31 // labels
32 public static ArrayList<JLabel> sinfo = new
ArrayList<JLabel>(); // Station
33 // infolabels
34 private ArrayList<ImageIcon> switchicons = new
ArrayList<ImageIcon>(); // WisselIcons
35 private ArrayList<ImageIcon> sensoricons = new
ArrayList<ImageIcon>(); // SensorIcons
36
37 // Label Color
38 Color lcol = new Color(255, 255, 255, 125);
39
40 // Status
41 public boolean running = false; // Running status van de
simulatie
42
43 // Station JButton posities
44 private int[] stax = { 220, 580, 940, 960, 940, 580, 220, 200
};
45 private int[] stay = { 130, 130, 130, 230, 330, 330, 330, 230
};
46 // Sensor icon posities
47 private int[] senx = { 153, 257, 295, 397, 503, 607, 645, 747,
853, 957,
48 995, 1097, 1139, 1096, 1096, 1139, 1096, 995, 957, 853, 746,
645,
49 607, 503, 396, 295, 257, 153, 111, 153, 153, 111 };
50 private int[] seny = { 56, 98, 98, 56, 56, 98, 98, 56, 56, 98,
98, 56, 90,
51 195, 232, 335, 435, 477, 477, 435, 435, 477, 477, 435, 435,
477,
52 477, 435, 338, 232, 195, 92 };
53 // Wissel icon posities
54 private int[] wisx = { 179, 529, 879, 1139, 1070, 720, 370,
111 };
55 private int[] wisy = { 56, 56, 56, 116, 435, 435, 435, 312 };
56
57 public RailScherm( TreinBeheer treinBeheer, ReisBeheer
reisBeheer) {
58 this .treinBeheer = treinBeheer;
59 this .reisBeheer = reisBeheer;
60 klok.setBounds(1180, 470, 60, 30);
61 klok.setBackground(Color .white);
62 klok.setBorder(BorderFactory.createLineBorder(Color .black,
1));
63 klok.setText( "0:00");
64 klok.setOpaque(true);
65 add(klok);
66
67 // Voorgrond componenten aanmaken in een for lus
```

```
68 for (int i = 0 ; i < 16 ; i++) {
69 // Cab labels aanmaken
70 cinfo.add(new JLabel( "Cab #" + (i)));
71 add( cinfo.get(i));
72 }
73 // Achtergrond componenten aanmaken in een for lus
74 for (int i = 0 ; i <= 7 ; i++) {
75 // Stat buttons aanmaken
76 stat.add(new JButton( "Station " + (i + 1)));
77 sinfo.add(new JLabel( "Wachtenden: 0\nLngst Wachtend:"));
78
79 JButton init = stat.get(i);
80 init.setBounds(stax[i] , stay[i], 120, 20);
81 init.addActionListener(new statHandler( "Station " + (i +
1)));
82 add(init);
83 JLabel init2 = sinfo.get(i);
84 init2.setBounds(stax[i] , stay[i] + 20, 120, 40);
85 init2.setBackground(Color .white);
86 init2.setOpaque(true);
87 init2.setBorder(BorderFactory.createEtchedBorder());
88 init2.setUI(new MultiLineLabelUI());
89 add( init2);
90
91 // Switchicons aanmaken, boven, onder, links en recht allemaal
met
92 // een eigen
93 // afbeelding
94 if (i <= 2) {
95 switchicons.add(new ImageIcon( "./w_open_b.gif"));
96 } else if (i == 3) {
97 switchicons.add(new ImageIcon( "./w_open_r.gif"));
98 } else if (i >= 4 && i <= 6) {
99 switchicons.add(new ImageIcon( "./w_open_o.gif"));
100 } else if (i == 7) {
101 switchicons.add(new ImageIcon( "./w_open_l.gif"));
102 }
103 }
104
105 // Alle sensor icons aanmaken
106 for (int i = 0 ; i <= 32 ; i++) {
107 sensoricons.add(new ImageIcon( "./s_aan.gif"));
108 }
109
110 // Layout initialiseren
111 setLayout(null);
112 // Simulatie starten zolang running == true
113 running = true;
114 repaint();
115 }
116
117 // Teken het scherm
118 public void paintComponent(Graphics g) {
119 super.paintComponent(g);
120 // Achtergrond afbeelding weergeven
121 back.paintIcon(this , g, 0, 0);
122 int ti = 0;
123 for ( Cab ucab : TreinBeheer.alleTreinen) {
124 // Teken zichtbare treinen
125 if ( ucab.zichtbaar) {
126 cinfo.get(ti).setVisible(true);
127 ucab.paintIcon(this , g, ucab.x, ucab.y);
128 // Teken Cab info labels
129 JLabel lab = cinfo.get(ti);
130 lab.setBounds( ucab.x - 70 , ucab.y + 25, 100, 40);
131 lab.setBackground(lcol);
132 lab.setOpaque(true);
133 lab.setText( "Cab #" + ti + "\nBestemming: -");
134 lab.setUI(new MultiLineLabelUI());
135 lab.setBorder(new LineBorder(Color .gray, 1));
136 }else{
137 cinfo.get(ti).setVisible(false);
138 }
139
140 ti++;
141 } // Einde for-each cab lus
142
143 // Teken wissels
144 for (int i = 0 ; i <= 7 ; i++) {
145 if ( ReisBeheer.haltes.get(i) .wissel.status()) {
146 switchicons.get(i).paintIcon(this , g, wisx[i] , wisy[i]);
147 }
148 }
149 // Teken sensors
150 for ( Cab cab : TreinBeheer.treinen) {
151 int i = cab.sensor.check() - 1;
152 if (i != -1 && cab.zichtbaar) {
153 sensoricons.get(i).paintIcon(this , g, senx[i] , seny[i]);
154 }
155 }
156 setClock();
157 setTreinText();
158 setLabelText();
159 Tabel2.reload();
160 } // Einde repaint
161 public static void setClock() {
162 String add;
163 if ( RailsApp.min < 10 ){
164 add = "0";
165 }else{
166 add = "";
167 }
168 if ( RailsApp.min%2 == 1 ){
169 klok.setText( RailsApp.uur + ":" + add + RailsApp.min);
170 }else{
171 klok.setText( RailsApp.uur + " " + add + RailsApp.min);
```

```
172 }
173
174 }
175 public static void setLabelText() {
176 for (int i =0 ; i <= 7 ; i++) {
177 int rsize = ReisBeheer.haltes.get(i).getAantalWachtenden();
178 String lw;
179 if (rsize > 0 ){
180 lw = "" + ReisBeheer.haltes.get(i) .reizen.getLast()
.wachttijd;
181 }else{
182 lw = "-";
183 }
184 sinfo.get(i).setText(
185 "Wachtenden: " + rsize
186 + "\nLngst Wachtend: " + lw);
187 }
188 }
189 public static void setTreinText (){
190 int j = 0;
191 for ( Trein tr: TreinBeheer.treinen) {
192 if ( tr.bestemming == 0 ){
193 cinfo.get(j).setText( "Cab #" + j + "\nBestemming: -");
194 }else{
195 cinfo.get(j).setText( "Cab #" + j + "\nBestemming: #" +
tr.bestemming);
196 }
197 j++;
198 }
199 }
200
201 // ActionListeners van de Station Knoppen
202 class statHandler implements ActionListener {
203 private String station;
204
205 public statHandler(String naam) {
206 station = naam;
207 }
208
209 public void actionPerformed(ActionEvent e) {
210 System .out.println( "1");
211 Station.setStation( station, 0, 0);
212 }
213 }
214
215 }
216
```

Reis.java

```
1
2 import java.awt.event.ActionEvent;
3 import java.awt.event.ActionListener;
4 import java.util.Calendar;
5 import java.util.GregorianCalendar;
6
7 import javax.swing.Timer;
8
9
10 public class Reis
11 {
```

```
12
////////////////////////////////////////////////////////ATRIBUTEN/
//////////////////////
//////////////////////
13 //private Calendar cal = new GregorianCalendar();
14 private Timer timer;
15 private int aantal;
16 private Halte startHalte;
17 public Halte bestemmingHalte;
18 private int[] tijd={99,99};
19 public int id;
20
21 public int status;
22
23 //positie
24 //private Object positie;
25
26 private boolean gereserveerd;
27
28
29 //in seconden
30 public int wachttijd=0;
31 public int reistijd=0;
32
33
34
35
36 // tijd wordt ingevoer in de volgende formaat: {u,m} uren in
24fotmaat, dus niet
hoger dan 23uur en 59minuten! anders loopt die vast
37
38 public Reis(int aantal, Halte start, Halte eind, int[] tijd) {
39 this .aantal= aantal;
40 startHalte = start;
41 bestemmingHalte = eind;
42 this .tijd = tijd;
43 System .out.println( "nieuw reis aangemaakt\n"+aantal+ "
passagiers\n
starthalte="+ startHalte.getId()+
44 "\n eindhalte="+ bestemmingHalte.getId()+ "\n
starttijd="+getStartTijd());
45 timer = new Timer(1000,new TimerHandler());
46 gereserveerd=true;
47 startReis();
48
49 }
50
51 public Reis( int aantal, Halte start, Halte eind ){
52 //this.id=id;
53 this .aantal= aantal;
54 startHalte = start;
55 bestemmingHalte = eind;
56 status=1;
57
58 timer = new Timer(1000,new TimerHandler());
59 startReis();
60 gereserveerd= false;
61
62 }
63
64
65
////////////////////////////////////////////////////////GETTERS
//////////////////////
//////////////////////////////
66 public String getStartTijd (){
67 String t="";
68 if(tijd[0]<10) t=t+ "0";
69 t=t+tijd[0]+":";
70 if(tijd[1]<10) t=t+ "0";
71 t=t+tijd[1];
72 return t;
73 }
74 public boolean gereserveerd (){
75 return gereserveerd;
76 }
77
78 public int getAantal (){
79 return aantal;
80 }
81
82 public int getStatus (){
83 return status;
84 }
85
86 public Halte getStart (){
87 return startHalte;
88 }
89
90 public Halte getBestemming (){
91 return bestemmingHalte;
92 }
93
94 public int getWachtTijd (){
95 return wachttijd;
96 }
97
98 public int getReistijd (){
99 return reistijd;
100 }
101
102 public int getID (){
103 return id;
104 }
105
```

```
//////////////////////////////////////////////////////SETTERS
//////////////////////
////////////////////////////
106
107
108 public void setStatus(int st ){
109 status= st;
110 }
111
112
113 void startReis (){
114
115 timer.start();
116
117 }
118
119 void stopReis (){
120
121 timer.stop();
122 System .out.println( "reistijd="+reistijd);
123 System .out.println( "wachttijd="+wachttijd);
124 System .out.println( "Reis beeindigt");
125 }
126
127 public Reis getThis (){
128 return this;
129 }
130
//////////////////////////////////////////////////CONTROLE///////
//////////////////////
////////////////////////////
131
132 //TIMERHANDLER//
133 //om de seconde voert die de in houd hiervan in
134 class TimerHandler implements ActionListener{
135
136 public void actionPerformed(ActionEvent e ){
137 Calendar c=new GregorianCalendar();
138 int uur = RailsApp.uur;
139 int min = RailsApp.min;
140 if(uur==(tijd[0]) && (min==(tijd[1]))) {
141 if(status==0 ){
142 setStatus(1);
143 ReisBeheer.haltes.get( startHalte.id - 1).addReis(getThis());
144 }
145 }
146 //wachttijd en reistijd bijhouden
147 if(status==1)
148 wachttijd++;
149 if(status==2)
150 reistijd++;
151 if (status==3) stopReis();
152
153 }
154
155
156
157 }
158 }
```

ReisBeheer.java

```
1
2
3 import java.util.ArrayList;
4
5 public class ReisBeheer {
6 public static ArrayList<Reis> reizen;
7 public static ArrayList<Reis> treizen;
8 public static ArrayList<Halte> haltes;
9 int[] def = {99,99};
10 TreinBeheer treinbeheer;
11
12
13 public static int getWacht(int stat) { //int station
14 int wacht = 0;
15 for (int i = 0 ; i < reizen.size() ; i++) {
16 if (( reizen.get(i).getStart() .id) == (stat - 1)) {
17 if ( reizen.get(i).getStatus() == 0) {
18 wacht++;
19 }
20 }
21 }
22 return wacht;
23 }
24
25 public ReisBeheer(TreinBeheer tb) {
26 reizen=new ArrayList<Reis>();
27 treizen=new ArrayList<Reis>();
28 haltes=new ArrayList<Halte>();
29 for(int i=1 ;i<9 ;i++ ){
30 haltes.add(new Halte(i));
31 }
32
33
34 int[] tijd={99,99}; //add reizen voor test
```

```
35
36 /*for(int i=0;i<8;i++){
37 addReis(5,(8-i),i+1,tijd);
38 }
39 */
40 }
41
42 public static void addReis(int aantal, int start, int
bestemming, int[] tijd ){
43 Reis r;
44
45 if(tijd[0]==99&&tijd[1]==99 ){
46 r=new Reis( aantal, haltes.get(start) ,
haltes.get(bestemming));
47 reizen.add(r);
48 treizen.add(r);
49 haltes.get(start).addReis(r);
50
51 }
52 else{
53 r=new Reis( aantal,haltes.get(start) , haltes.get(bestemming)
,tijd);
54 reizen.add(r);
55 treizen.add(r);
56 }
57
58
59
60
61 }
62 public static double gemReisttijd (){
63 int rt=0;
64 int aantal=0;
65 for( Reis r:reizen ){
66 if( r.status>1 && r.status < 3 ){
67 rt=rt+ r.reistijd;
68 aantal++;
69 }
70 }
71 if(rt>0&&aantal>0)
72 return rt/ aantal;
73 else return 0;
74
75
76 }
77
78 public static double gemWachttijd (){
79 int wt=0;
80 int aantal=0;
81 for( Reis r:reizen ){
82 if( r.status>0 && r.status <3 ){
83 wt=wt+ r.wachttijd;
84 aantal++;
85 }
86 }
87 if(wt>0&&aantal>0 ){
88 return wt/ aantal;
89 }else{
90 return 0;
91 }
92 }
93
94 public Halte getHalte(int id ){
95 return haltes.get(id);
96 }
97
98
99 public void addHalte(int id ){
100 Halte h=new Halte(id);
101 haltes.add(h);
102 }
103
104
105 //reizen die wachten en in trein zitten
106 public static ArrayList<Reis> getActieveReizen (){
107 ArrayList<Reis> ar=new ArrayList<Reis>();
108 for( Reis r:reizen ){
109 if( r.getStatus()>0&& r.getStatus()<3)
110 ar.add(r);
111 }
112
113
114 return ar;
115 }
116
117
118
119 //geeft alle reizen met deze status
120 public static ArrayList<Reis> getReizen(int status ){
121 ArrayList<Reis> ar=new ArrayList<Reis>();
122
123 for(int i=0 ;i< reizen.size() ;i++ ){
124 if(( reizen.get(i)).getStatus()==status)
125 ar.add( reizen.get(i));
126
127
128 }
129 // System.out.println("reizen met status "+status+"
zijn:"+ar.size());
130
131 return ar;
132 }
133 //geeft de halte met de meeste wachtenden
134 public Halte druksteHalte (){
135 int aantal=0;
136 Halte drukste =(Halte) haltes.get(0);
```

```
137 for(int i=1 ;i< haltes.size() ;i++ ){
138 Halte h= haltes.get(i);
139 if( h.getAantalWachtenden()>aantal ){
140 drukste = h;
141 aantal = h.getAantalWachtenden();
142 }
143 }
144 if(aantal==0) return null;
145 //System.out.println("halte nr "+drukste.getId());
146 return drukste;
147 }
148
149 public Halte langsteWachttijdHalte() {
150 Halte h, langsteWachttijdHalte = null;
151 int wachttijd = 0;
152 for (int i = 0 ; i < haltes.size() ; i++) {
153 h = haltes.get(i);
154 if ( h.getEersteReis() != null)
155 if ( h.getEersteReis().getWachtTijd() > wachttijd) {
156 wachttijd = h.getEersteReis().getWachtTijd();
157 langsteWachttijdHalte = h;
158 }
159 }
160
161 if (langsteWachttijdHalte != null)
162 System .out.println( "halte nr " +
langsteWachttijdHalte.getId());
163 return langsteWachttijdHalte;
164 }
165
166
167 }
168
```

Sector.java

```
1
2
3 public class Sector {
4 public int id;
5 public Trein trein;
6
7
8 public Sector(int id ){
9 this .id= id;
10 trein=null;
11 }
12 public boolean vrij (){
13 if (trein==null)
14 return true;
15 else return false;
16 }
17
18
19
20 }
21
```

Sensor.java

```
1 public class Sensor {
2 // Posities
3 private double[] posities = { 2.5, 6, 7.5, 11, 14.5, 18, 19.5,
23, 26.5,
4 30, 31.5, 35, 39.5, 43, 44.5, 47.5, 52.5, 56, 57.5, 61, 64.5,
68,
5 69.5, 73, 76.5, 80, 81.5, 85, 89.5, 93, 94.5, 97.5 };
6 private int id;
7 private Cab cab;
8
9 public Sensor(int id, Cab cab) {
10 this .id = id;
```

```
this .cab = cab;
}

public int check() {
int sensorid = 0;
for (double sp : posities) {
if ( cab.positie >= sp - 0.15 && cab.positie <= sp + 0.15) {
// Boolean controleert of dit een positie is die in een
// wissel zit
boolean iswissel = (sensorid - 1) % 4 == 0
|| (sensorid - 2) % 4 == 0;
if ((iswissel && cab.inwissel) || !iswissel) {
return sensorid + 1;
}
}
sensorid++;
}
return 0;
}
}
```

Station2.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.EtchedBorder;

import java.util.ArrayList;
import java.util.Date;

public class Station2 extends JPanel {

private static final long serialVersionUID = 1L;

// maak de objecten aan.
private static JLabel tekst, t1, t2, t3, t4;
private static JComboBox bestemming, vertrek, tijdu, tijdm,
personen;
private static JButton oke, annul, verwijder, random;
public static JCheckBox ranj, nuu;
private static Tabel2 tabell;
public static Statistiek statt;
private static int zicht;

int[] trix = { 10, 220, 450, 680, 10, 220, 450, 680 };
int[] triy = { 10, 10, 10, 10, 120, 120, 120, 120 };
private static ArrayList<Train> treinen = new
ArrayList<Train>();

Font f1 = new Font( "SansSerif", Font .BOLD, 26);
Font f2 = new Font( "SansSerif", Font .BOLD, 14);
Font f3 = new Font( "SansSerif", Font .BOLD, 12);
Font f4 = new Font( "SansSerif", Font .PLAIN, 12);
final Border selectedBorder = new EtchedBorder();

public Station2() {
setBackground(Color .WHITE);
setLayout(null);

// stel tekst van paneel in.
tekst = new JLabel();
tekst.setFont( f1);
tekst.setBounds(30, 10, 600, 40);

// stel label bestemming in
t1 = new JLabel( "Bestemming:");
t1.setFont( f2);
t1.setBounds(40, 80, 100, 20);

// stel combobox bestemming in
bestemming = new JComboBox();
bestemming.setFont( f2);
bestemming.addItem(makeObj( "Selecteer Station"));
for (int x = 1 ; x < 9 ; x++) {
bestemming.addItem(makeObj( "Station " + x));
}
bestemming.setEditable(true);
bestemming.setBounds(150, 80, 300, 20);

// stel label vertrekstation in
t2 = new JLabel( "Vertrek: ");
t2.setFont( f2);
t2.setBounds(40, 50, 100, 20);

// stel combobox vertrekstation in
vertrek = new JComboBox();
vertrek.setFont( f2);
vertrek.addItem(makeObj( "Selecteer Station"));
for (int x = 1 ; x < 9 ; x++) {
vertrek.addItem(makeObj( "Station " + x));
}
vertrek.setBounds(150, 50, 300, 20);
```

```
70 vertrek.setEditable(true);
71 vertrek.setEnabled(true);
72
73 // stel label tijd in
74 t3 = new JLabel( "Tijd: ");
75 t3.setFont( f2);
76 t3.setBounds(40, 110, 100, 20);
77
78 // stel combobox tijdu(uur) in
79 tijdu = new JComboBox();
80 tijdu.setFont( f2);
81 tijdu.setBounds(150, 110, 80, 20);
82 tijdu.addItem(makeObj( "Uur"));
83 for (int i = 00 ; i < 24 ; i++) {
84 String uur;
85 if (i < 10) {
86 uur = ( "0" + i);
87 } else {
88 uur = ("" + i);
89 }
90 tijdu.addItem(makeObj("" + uur));
91 }
92 tijdu.setEditable(true);
93
94 // stel combobox tijdm(minuten) in
95 tijdm = new JComboBox();
96 tijdm.setFont( f2);
97 tijdm.setBounds(240, 110, 80, 20);
98 tijdm.addItem(makeObj( "Min"));
99 for (int j = 00 ; j < 60 ; j++) {
100 String min;
101 if (j < 10) {
102 min = ( "0" + j);
103 } else {
104 min = ("" + j);
105 }
106 tijdm.addItem(makeObj("" + min));
107 }
108 tijdm.setEditable(true);
109
110 // stel label passagiers in
111 t4 = new JLabel( "Passagiers");
112 t4.setFont( f2);
113 t4.setBounds(40, 140, 100, 20);
114
115 // stel combobox personen in
116 personen = new JComboBox();
117 personen.setFont( f2);
118 personen.setBounds(150, 140, 300, 20);
119 personen.addItem(makeObj( "Aantal"));
120 for (int i = 1 ; i <= 8 ; i++) {
121 personen.addItem(makeObj("" + i));
122 }
```

```
123 personen.setEditable(true);
124
125 // stel knop oke in
126 oke = new JButton();
127 oke.setFont( f2);
128 oke.setBounds(800, 40, 170, 30);
129 oke.setText( "OK");
130 oke.addActionListener(new knopOke());
131
132 // stel knop annuleren in
133 annul = new JButton();
134 annul.setFont( f2);
135 annul.setBounds(800, 80, 170, 30);
136 annul.setText( "Annuleren");
137 annul.addActionListener(new knopAnn());
138
139 // stel knop verwijder in
140 verwijder = new JButton();
141 verwijder.setFont( f2);
142 verwijder.setBounds(800, 120, 170, 30);
143 verwijder.setText( "Verwijderen");
144 verwijder.setVisible(false);
145 verwijder.addActionListener(new knopVerw());
146
147 // stel knop random in
148 random = new JButton();
149 random.setFont( f2);
150 random.setBounds(40, 190, 100, 30);
151 random.setText( "Random");
152 random.addActionListener(new knopRand());
153
154 // Checkbox voor random vanaf dit station instellen
155 ranj = new JCheckBox( "Random vanaf dit station?");
156 ranj.setToolTipText( "dat is : " +
Station.stationNaam.getText());
157 ranj.setSelected(true);
158 ranj.setBounds(40, 225, 200, 20);
159
160 // Checkbox voor tijd nu instellen
161 nuu = new JCheckBox( "Nu meteen vertrekken!");
162 nuu.setToolTipText( "(NB, dit is over 5 min!)");
163 nuu.setSelected(false);
164 nuu.setBounds(40, 245, 300, 30);
165
166 tabell = new Tabel2();
167 tabell.setBounds(10, 50, 900, 230);
168 tabell.setBorder(selectedBorder);
169 tabell.setFont( f4);
170
171 statt = new Statistiek();
172 statt.setBounds(0, 0, 980, 280);
173
174 for (int i = 0 ; i < 8 ; i++) {
```

```
175 Train init = new Train(i);
176 treinen.add(init);
177 init.setBounds(trix[i] , triy[i], 200, 100);
178 add(init);
179 }
180
181 // plaats de objecten op de panel
182 add(tekst);
183 add( t1);
184 add(bestemming);
185 add( t2);
186 add(vertrek);
187 add( t3);
188 add(tijdu);
189 add(tijdm);
190 add( t4);
191 add(personen);
192 add(oke);
193 add(annul);
194 add(verwijder);
195 add(random);
196 add(ranj);
197 add(nuu);
198 add(tabell);
199 add(statt);
200 maakZichtbaar(5);
201 zicht = 4;
202 }
203
204 public static void maakZichtbaar(int waar) {
205 switch (waar) {
206 case 1: {
207 // scherm toevoegen zichtbaar
208 zicht = 1;
209 tekst.setText( "reis toevoegen");
210 tekst.setVisible(true);
211 t1.setVisible(true);
212 bestemming.setVisible(true);
213 t2.setVisible(true);
214 vertrek.setVisible(true);
215 t3.setVisible(true);
216 tijdu.setVisible(true);
217 tijdm.setVisible(true);
218 t4.setVisible(true);
219 personen.setVisible(true);
220 oke.setVisible(true);
221 annul.setVisible(true);
222 verwijder.setVisible(false);
223 random.setVisible(true);
224 ranj.setVisible(true);
225 nuu.setVisible(true);
226 tabell.setVisible(false);
227 for ( Train trein : treinen) {
228 trein.setVisible(false);
229 }
230 statt.setVisible(false);
231 break;
232 }
233
234 case 2: {
235 // leeg
236 zicht = 2;
237 tekst.setVisible(false);
238 t1.setVisible(false);
239 bestemming.setVisible(false);
240 t2.setVisible(false);
241 vertrek.setVisible(false);
242 t3.setVisible(false);
243 tijdu.setVisible(false);
244 tijdm.setVisible(false);
245 t4.setVisible(false);
246 personen.setVisible(false);
247 oke.setVisible(false);
248 annul.setVisible(false);
249 verwijder.setVisible(false);
250 random.setVisible(false);
251 ranj.setVisible(false);
252 nuu.setVisible(false);
253 tabell.setVisible(false);
254 for ( Train trein : treinen) {
255 trein.setVisible(false);
256 }
257 statt.setVisible(false);
258 break;
259 }
260
261 case 3: {
262 // reistabel zichtbaar
263 zicht = 3;
264 tekst.setText( "Reizen Weergeven");
265 tekst.setVisible(true);
266 t1.setVisible(false);
267 bestemming.setVisible(false);
268 t2.setVisible(false);
269 vertrek.setVisible(false);
270 t3.setVisible(false);
271 tijdu.setVisible(false);
272 tijdm.setVisible(false);
273 t4.setVisible(false);
274 personen.setVisible(false);
275 oke.setVisible(false);
276 annul.setVisible(false);
277 verwijder.setVisible(false);
278 random.setVisible(false);
279 ranj.setVisible(false);
280 nuu.setVisible(false);
```

```
tabell.setVisible(true);
for ( Train trein : treinen) {
trein.setVisible(false);
}
statt.setVisible(false);
break;
}


case 5: {
// Treinen Zichtbaar
zicht = 5;
tekst.setText("");
tekst.setVisible(false);
t1.setVisible(false);
bestemming.setVisible(false);
t2.setVisible(false);
vertrek.setVisible(false);
t3.setVisible(false);
tijdu.setVisible(false);
tijdm.setVisible(false);
t4.setVisible(false);
personen.setVisible(false);
oke.setVisible(false);
annul.setVisible(false);
verwijder.setVisible(false);
random.setVisible(false);
ranj.setVisible(false);
nuu.setVisible(false);
tabell.setVisible(false);
for ( Train trein : treinen) {
trein.setVisible(false);
}
statt.setVisible(true);
break;
}

}
}




public int getBestemmingid() {
int best;
best = bestemming.getSelectedIndex();
return best;
}


public int getVertrekid() {
int vert;
vert = vertrek.getSelectedIndex();
return vert;
}

// geeft de tijd in een String
public String getTijd() {
String tijd;
tijd = ("" + tijdu.getSelectedItem() + ":" +
tijdm.getSelectedItem());
return tijd;
}

public int getUurID() {
int uur;

if ( tijdu.getSelectedIndex() == 0) {
uur = 100;
} else {
uur = tijdu.getSelectedIndex();
}
return uur;
}

public int getMinID() {
int min;

if ( tijdm.getSelectedIndex() == 0) {
min = 100;
} else {
min = tijdm.getSelectedIndex();
}
return min;
}

// geeft het aantal personen in een String
public String getPersonen() {
String pers;
pers = ("" + personen.getSelectedItem());
return pers;
}

public int getPersonenID() {
int pers = personen.getSelectedIndex();
return pers;
}

// handler voor de OKE knop
class knopOke implements ActionListener {
public void actionPerformed(ActionEvent e) {
// Dit stukje code zorgt ervoor dat in het scherm reis
toevoegen
// alle velden worden ingevuld!
```

```
385 if (zicht == 1) {
386 if (( bestemming.getSelectedIndex() == 0)
387 || ( vertrek.getSelectedIndex() == 0)
388 || ( personen.getSelectedIndex() == 0)) {
389 JOptionPane.showMessageDialog( personen,
390 "Alle velden moeten ingevuld worden", "FOUT",
391 JOptionPane .WARNING_MESSAGE);
392 }
393 if (( bestemming.getSelectedIndex()) == (vertrek
394 .getSelectedIndex())) {
395 JOptionPane.showMessageDialog( personen,
396 "Bestemming kan niet het vertrek station zijn!",
397 "FOUT", JOptionPane .WARNING_MESSAGE);
398 } else {
399 System .out.println( "Station "+getBestemmingid());
400 System .out.println( "Station "+getVertrekid());
401 System .out.println(getTijd());
402 System .out.println(getPersonen());
403
404 int size = ReisBeheer.reizen.size();
405
406 //
ReisBeheer.addReis(getPersonenID(),ReisBeheer.haltes.get(getVertr
ekid()),ReisBeheer.hal
tes.get(getBestemmingid()),new
407 // int[]{getUurID(),getMinID()});
408 ReisBeheer.addReis(getPersonenID(), getVertrekid() - 1,
409 getBestemmingid() - 1, new int[] { getUurID() - 1,
410 getMinID() - 1 });
411
412 System .out.println( "Aantal wachtenden station 5: "
413 + ReisBeheer.getWacht(5));
414 // System.out.println("Aantal mensen wachtend in station 5:
415 // "+ ReisBeheer.haltes.get(5).getAantalWachtenden());
416 RailScherm.setLabelText();
417 }
418
419 // dit stukje code word uitgevoerd als je op ok klikt in het
420 // reizen weergeven scherm.
421 if (zicht == 3) {
422 System .out.println( "Oke Opgeslagen");
423 }
424 }
425 }
426 }
427
428 // handler voor de annuleren knop
429 class knopAnn implements ActionListener {
430 public void actionPerformed(ActionEvent e) {
431 maakZichtbaar(2);
432 bestemming.setSelectedIndex(0);
433 vertrek.setSelectedIndex(0);
434 tijdu.setSelectedIndex(0);
435 tijdm.setSelectedIndex(0);
436 personen.setSelectedIndex(0);
437 }
438 }
439
440 class knopVerw implements ActionListener {
441 public void actionPerformed(ActionEvent e) {
442 tabell.removeRow();
443 RailScherm.setLabelText();
444 }
445 }
446
447
448
449 /** @author Willy
450 * Deze methode maakt een x aantal verschillende reizen aan
door middel van de
451 * methoden onder deze class.
452 *
453 */
454
455 class knopRand implements ActionListener {
456 public void actionPerformed(ActionEvent e) {
457 int x;
458 String s = (String) JOptionPane.showInputDialog( random,
459 "Hoeveel reizen wilt u aanmaken?\n", "Hoeveel?",
460 JOptionPane .YES_NO_CANCEL_OPTION, null, null, null);
461 try {
462 x = Integer.parseInt( s, 10);
463 if (x > 0 && x <= 200) {
464 for (int i = 0 ; i < x; i++) {
465 String vertre;
466 String tijd;
467 int ver = 1;
468 int tijdmm;
469 int tijduu;
470 String bestem = ("" +getRandomBestemmingID());
471 if ( ranj.isSelected()) {
472 vertre = ("" + Station.stationNaam.getText()
473 .charAt(8));
474 } else {
475 vertre = ("" + getRandomVertrekID());
476 }
477 if ( vertre.equals(bestem))
478 bestem = (""+getRandomBestemmingID());
479 if ( nuu.isSelected() == true) {
480 tijd = getComputerTijd(ver);
481 tijdmm = 99;
482 tijduu = 99;
483 } else {
484 tijd = (tijdu + ":" + tijdm);
485 tijdmm = getRandomMin();
486 tijduu = getRandomUur();
```

```
}
String pasa = getRandomPersonen();
System .out.println( "Bestemming = " + bestem);
System .out.println( "Vertrek = " + vertre);
System .out.println( "Tijd = " + tijd);
System .out.println( "Aantal pasagiers = " + pasa);
System .out.println("----------------------------");
ReisBeheer.addReis(Integer
.parseInt(getRandomPersonen()), Integer
.parseInt(vertre) - 1,
getRandomBestemmingID() - 1, new int[] {
tijduu, tijdmm });
RailScherm.setLabelText();
}
}
if (x > 200) {
JOptionPane.showMessageDialog( personen,
"U mag geen getal groter dan 200 invullen",
"Ongeldige invoer", JOptionPane .WARNING_MESSAGE);
}
if (x <= 0) {
JOptionPane.showMessageDialog( personen,
"U moet een getal groter dan 0 invoeren",
"Ongeldige invoer", JOptionPane .WARNING_MESSAGE);
}
} catch (NumberFormatException nFE) {
JOptionPane.showMessageDialog( personen,
"U moet een getal invoeren!", "Ongeldige invoer",
JOptionPane .WARNING_MESSAGE);
}
}
}


//Geeft de computertijd in een String
public String getComputerTijd(int min) {
Date nu = new Date();
String tijde;
String tijdu;
String tijdm;
int uur = nu.getHours();
int minuut = nu.getMinutes() + min;
if (minuut >= 60) {
minuut = (minuut - 60);
uur = (uur + 1);
if (uur > 23) {
uur = 0;
}
}
if (minuut < 10) {
tijdm = ( "0" + minuut);
} else {
tijdm = ("" + minuut);
}
if (uur < 10) {
tijdu = ( "0" + uur);
} else {
tijdu = ("" + uur);
}
tijde = (tijdu + ":" + (tijdm));
return tijde;
}



// maakt een willekeurig bestemming in een int
public int getRandomBestemmingID() {
int bestemming = ((int) (8 * Math.random()) + 1);
return bestemming;
}

//maakt een willekeurig vertrek punt in een int
public int getRandomVertrekID() {
int vertrek = ((int) (8 * Math.random()) + 1);
return vertrek;
}

//maakt een willekeurige Tijd in een String doormiddel van de
methoden getRandomUur
en getRandomMin
public String getRandomTijd() {
String tijd;

tijd = (getRandomUur() + ":" + getRandomMin());
return tijd;
}

//maakt een willekeurig Uur in een int
public int getRandomUur() {
String tijdu;
int uur;
int tijduu = ((int) (24 * Math.random()));
if (tijduu < 10)
tijdu = ( "0" + tijduu);
else
tijdu = ("" + tijduu);
uur = (0 + Integer.parseInt( tijdu, 10));
return uur;
}

//maakt een willekeurige minuten aan in een int
public int getRandomMin() {
int min;
String tijdm;
int tijdmm = ((int) (60 * Math.random()));
if (tijdmm < 10)
```

```
591 tijdm = ( "0" + tijdmm);
592 else
593 tijdm = ("" + tijdmm);
594 min = (0 + Integer.parseInt( tijdm, 10));
595
596 return min;
597 }
598
599 //maakt een willekeurig aantal personen aan in een String
600 public String getRandomPersonen() {
601 String pasa = ("" + ((int) (8 * Math.random()) + 1));
602 return pasa;
603 }
604
605 // zorgt ervoor dat er objecten in de comboboxen geplaatst
kunnen worden
606
607 private static Object makeObj(final String item) {
608 return new Object() {
609 public String toString() {
610 return item;
611 }
612 };
613 }
614
615 }
616
```

Station.java

```
1 import java.awt.*;
2 import java.awt.event.*;
3
4 import javax.swing.*;
5 import javax.swing.border.Border;
6 import javax.swing.border.EtchedBorder;
7
8 //import javax.swing.border.EmptyBorder;
9
10 public class Station extends JPanel {
11 public static JLabel stationNaam;
12 public static JLabel wachtenden;
13 public static JLabel grtBest;
14 private static JButton rtoevoegen;
15 private static JButton raanpassen;
16 private static JButton rweergeven;
17 public static Station2 stat;
18 Font f1 = new Font( "SansSerif", Font .BOLD, 26);
19 final Border selectedBorder = new EtchedBorder();
20
21 public Station() {
22 setBackground(Color .WHITE);
23 this.setLayout(null);
24
25 stationNaam = new JLabel();
26 stationNaam.setFont( f1);
27 stationNaam.setBorder(selectedBorder);
28 stationNaam.setBounds(0, 0, 150, 40);
29 stationNaam.setText( "Station 1");
30
31 wachtenden = new JLabel();
32 wachtenden.setText( "Aantal wachtenden: ");
33 wachtenden.setBounds(0, 45, 150, 20);
34
35 grtBest = new JLabel();
36 grtBest.setText( "Grootste bestemming: ");
37 grtBest.setBounds(0, 65, 150, 20);
38
39 rtoevoegen = new JButton( "Reis toevoegen");
40 rtoevoegen.setBounds(5, 140, 140, 20);
41 rtoevoegen.addActionListener(new reisToevoegen());
42
43 raanpassen = new JButton( "Statistieken");
44 raanpassen.setBounds(5, 180, 140, 20);
45 raanpassen.addActionListener(new reisAanpassen());
46 raanpassen.setEnabled(true);
47
48 rweergeven = new JButton( "Reizen weergeven");
49 rweergeven.setBounds(5, 160, 140, 20);
```

```
50 rweergeven.addActionListener(new reizenWeergeven());
51
52 add(stationNaam);
53 add(wachtenden);
54 add(grtBest);
55 add(rtoevoegen);
56 add(raanpassen);
57 add(rweergeven);
58
59 maakZichtbaar(1);
60 }
61
62 public void setNaam(int i) {
63 switch (i) {
64 case 1:
65 stationNaam.setText( "Station 1");
66 case 2:
67 stationNaam.setText( "Station 2");
68 case 3:
69 stationNaam.setText( "Station 3");
70 case 4:
71 stationNaam.setText( "Station 4");
72 case 5:
73 stationNaam.setText( "Station 5");
74 case 6:
75 stationNaam.setText( "Station 6");
76 case 7:
77 stationNaam.setText( "Station 7");
78 case 8:
79 stationNaam.setText( "Station 8");
80 default:
81 stationNaam.setText( "Station 1");
82 }
83 }
84
85 public static void setStation(String naam, int wacht, int
best) {
86 stationNaam.setText(naam);
87 wachtenden.setText( "Aantal wachtenden: " + wacht);
88 grtBest.setText( "Grootste bestemming: " + best);
89 System .out.println( "Stationnaam = " +
stationNaam.getText());
90 Station2.ranj.setToolTipText( "dat is : "
91 + Station.stationNaam.getText());
92 }
93
94 class reisToevoegen implements ActionListener {
95 public void actionPerformed(ActionEvent e) {
96 Station2.maakZichtbaar(1);
97 raanpassen.setEnabled(true);
98 }
99 }
100
```

```
101 class reisAanpassen implements ActionListener {
102 public void actionPerformed(ActionEvent e) {
103 Station2.maakZichtbaar(5);
104 }
105 }
106
107 class reizenWeergeven implements ActionListener {
108 public void actionPerformed(ActionEvent e) {
109 Station2.maakZichtbaar(3);
110 raanpassen.setEnabled(true);
111 }
112 }
113
114 public static void maakZichtbaar(int waar) {
115 switch (waar) {
116 case 1: {
117 // Station scherm zichtbaar
118 stationNaam.setVisible(true);
119 wachtenden.setVisible(true);
120 grtBest.setVisible(true);
121 rtoevoegen.setVisible(true);
122 raanpassen.setVisible(true);
123 rweergeven.setVisible(true);
124 }
125 break;
126 case 2: {
127 // leeg
128 stationNaam.setVisible(false);
129 wachtenden.setVisible(false);
130 grtBest.setVisible(false);
131 rtoevoegen.setVisible(false);
132 raanpassen.setVisible(true);
133 rweergeven.setVisible(false);
134
135 }
136 break;
137 case 3: {
138 // trein scherm zichtbaar
139 stationNaam.setVisible(true);
140 wachtenden.setVisible(false);
141 grtBest.setVisible(false);
142 rtoevoegen.setVisible(false);
143 raanpassen.setVisible(false);
144 rweergeven.setVisible(false);
145 }
146 break;
147 }
148 }
149 }
150
```

Statistiek.java

```
1 import java.awt.*;
2 import java.awt.event.ActionEvent;
3 import java.awt.event.ActionListener;
4 import org.jfree.data.xy.XYDataset;
5 import org.jfree.data.xy.XYSeries;
6 import org.jfree.data.xy.XYSeriesCollection;
7
8 import javax.swing.*;
9
10 import org.jfree.data.category.DefaultCategoryDataset;
11
12 import javax.swing.Timer;
13
14 public class Statistiek extends JPanel {
15
16 public static JComboBox cmbx, cmby, cmbGrap;
17 public static JLabel lblx, lbly, lblGrap;
18 public static JButton oke;
19 // public static Grafiek1 stat1;
20 public static Grafiek3 stat2;
21 public static DefaultCategoryDataset dataset;
22 public static XYSeriesCollection form;
23
24 public Statistiek() {
25 setLayout(null);
26 form = new XYSeriesCollection();
27 stat2 = new Grafiek3( form, "Tijd", "Aantal", "Voortgang");
28 stat2.setBounds(0, 0, 980, 280);
29 stat2.setVisible(true);
30 add( stat2);
31 }
32
33 public static void refresh() {
34 form = stat2.createDataset(form);
35 stat2.dataset = form;
36 stat2.setBounds(0, 0, 980, 280);
37 }
38
39 }
40
```

Tabel2.java

```
1 import javax.swing.*;
2 import javax.swing.event.TableModelEvent;
3 import javax.swing.event.TableModelListener;
4 import javax.swing.table.AbstractTableModel;
5
6 import java.awt.*;
7 import java.awt.event.ActionEvent;
8 import java.util.*;
9
10 public class Tabel2 extends JPanel {
11
12 private static final long serialVersionUID = 1L;
13 private JPanel thePanel;
14 public static JTable theTable;
15 public static LocalTableModel theTableModel;
16 private AbstractAction theRemoveRowAction;
17
18 private static int actief, cvar;
19 private JToolBar toolBar;
20 public static ArrayList<Reis> theRows = ReisBeheer.reizen;
21 private static ArrayList<Reis> sRows = new ArrayList<Reis>();
22
23 private class Row {
24 private int ID;
25 private String bestemming;
26 private String vertrek;
27 private String tijd;
```

```
28 private int pers;
29
30 public Row(int aID, String aBestemming, String aVertrek,
String aTijd,
31 int aPers) {
32 ID = aID;
33 bestemming = aBestemming;
34 vertrek = aVertrek;
35 tijd = aTijd;
36 pers = aPers;
37 }
38
39 public int getID() {
40 return ID;
41 }
42
43 public String getBest() {
44 return bestemming;
45 }
46
47 public String getVertrek() {
48 return vertrek;
49 }
50
51 public String getTijd() {
52 return tijd;
53 }
54
55 public int getPers() {
56 return pers;
57 }
58 }
59
60 public class LocalTableModel extends AbstractTableModel {
61 public String getColumnName(int column) {
62 if (column == 0) {
63 return "Status";
64 }
65 if (column == 1) {
66 return "Vertrek Punt";
67 }
68 if (column == 2) {
69 return "Bestemming";
70 }
71 if (column == 3) {
72 return "Vertrek Tijd";
73 }
74 if (column == 4 ){
75 return "Personen";
76 }
77 if (column == 5 ){
78 return "Wachttijd";
79 }else{
80 return "Reistijd";
81 }
82 }
83
84 public Class getColumnClass(int columnIndex) {
85 if (columnIndex == 0) {
86 return Integer.class;
87 }
88 if (columnIndex == 1) {
89 return String.class;
90 }
91 if (columnIndex == 2) {
92 return String.class;
93 }
94 if (columnIndex == 3) {
95 return String.class;
96 }
97 if (columnIndex == 3) {
98 return Integer.class;
99 }
100 if (columnIndex == 3) {
101 return Integer.class;
102 }else {
103 return Integer.class;
104 }
105 }
106
107 public boolean isCellEditable(int rowIndex, int columnIndex)
{
108 return false;
109 }
110
111 public int getRowCount() {
112 return sRows.size();
113 }
114
115 public int getColumnCount() {
116 return 7;
117 }
118
119 public Object getValueAt(int rowIndex, int columnIndex) {
120 Reis row = sRows.get(rowIndex);
121 if (columnIndex == 0) {
122 int st = row.getStatus();
123 if (st == 0) {
124 return ( "Gereserveerd");
125 } else if (st == 1) {
126 return ( "Wachtend");
127 } else if (st == 2) {
128 return ( "Reizend");
129 } else if (st == 3) {
130 return ( "Beëindigd");
131 }
```

```
132 }
133 if (columnIndex == 1) {
134 return ( "Station " + ( row.getStart() .id));
135 }
136 if (columnIndex == 2) {
137 return ( "Station " + ( row.getBestemming() .id));
138 }
139 if (columnIndex == 3) {
140 if ( row.getStartTijd().equals( "99:99" )){
141 return ( "Direct Vertrekken");
142 }else{
143 return ("" + row.getStartTijd());
144 }
145 }
146 if (columnIndex == 4) {
147 return ("" + row.getAantal());
148 }
149 if (columnIndex == 5) {
150 return ("" + row.getWachtTijd());
151 }else {
152 return ("" + row.getReistijd());
153 }
154 }
155 }
156
157 public Tabel2() {
158 setLayout(null);
159 actief = 0;
160 thePanel = new JPanel(new BorderLayout());
161 // thePanel.setLayout(null);
162
163 toolBar = new JToolBar();
164 toolBar.setFloatable(false);
165 toolBar.add(new AbstractAction( "Alle reizen") {
166 public void actionPerformed(ActionEvent e) {
167 loadAll();
168 }
169 });
170 theRemoveRowAction = new AbstractAction( "Actieve Reizen") {
171 public void actionPerformed(ActionEvent e) {
172 loadAct();
173 }
174 };
175 toolBar.add(theRemoveRowAction);
176 toolBar.add(new AbstractAction( "Afgehandelde reizen") {
177 public void actionPerformed(ActionEvent e) {
178 loadAf();
179 }
180 });
181 thePanel.add( toolBar, BorderLayout .NORTH);
182
183 theTableModel = new LocalTableModel();
184 theTable = new JTable(theTableModel) {
185 public void refresh() {
186 repaint();
187 }
188 };
189 theTable.setAutoCreateRowSorter(true);
190
191 thePanel.add(new JScrollPane(theTable), BorderLayout
.CENTER);
192 // validateSelection();
193 // loadAll();
194 reload();
195 thePanel.setBounds(0, 0, 900, 230);
196 add(thePanel);
197 }
198
199 public void removeRow() {
200 int index = theTable.getSelectedRow();
201 if (index == -1)
202 return;
203 ReisBeheer.treizen.remove(index);
204 theTableModel.fireTableRowsDeleted( index, index);
205 index = ( ReisBeheer.treizen.size() - 1);
206 if (index >= 0) {
207 theTable.getSelectionModel().setSelectionInterval( index,
index);
208 }
209 }
210
211 // Herlaad de lijst met reizen die moeten worden weergegeven
212 public static void reload() {
213 sRows.clear();
214 if (actief == 1) { // Inactieve reizen
215 for ( Reis reis : theRows) {
216 if ( reis.getStatus() == 3)
217 sRows.add(reis);
218 }
219 } else if (actief == 2) { // Actieve reizen
220 for ( Reis reis : theRows) {
221 if ( reis.getStatus() < 3)
222 sRows.add(reis);
223 }
224 } else { // Alle reizen
225 for ( Reis reis : theRows) {
226 sRows.add(reis);
227 }
228 }
229
230 refresh();
231 }
232
233 public void loadAll() {
234 actief = 0;
235 reload();
```

```
236 }
237
238 public void loadAf() {
239 actief = 1;
240 reload();
241 }
242
243 public void loadAct() {
244 actief = 2;
245 reload();
246 }
247
248 // Vernieuw de tabel
249 public static void refresh() {
250 System .out.println( "Refresh||Rows: " +
theTableModel.getRowCount() + "|| LSize
sRows.size());
251 if ( theTableModel.getRowCount() > 0) {
252 cvar = 1;
253 theTableModel.fireTableRowsInserted(0,
254 theTableModel.getRowCount() - 1);
255 } else {
256 if (cvar == 1)
257 theTableModel.fireTableRowsDeleted(0, 0);
258 cvar = 0;
259 }
260 }
261
262 }
263
```

Trein.java

```
1 import java.awt.event.ActionEvent;
2 import java.awt.event.ActionListener;
3
4 import javax.swing.Timer;
5
6
7 /**
8 * Write a description of class Trein here.
9 *
10 * @author (your name)
11 * @version (a version number or a date)
12 */
13
14 public class Trein extends Cab {
15 /**
16 *
17 */
18 private static final long serialVersionUID = 1L;
19 public Reis reis;
20 boolean rijdt, inHalte;
21 public int wtijd;
22 public Timer timer;
23 public int id;
24 public int positie;
25 public byte status=0;
26 //public int snelheid;
27
28
29 public int bestemming;
30
31
32
33 //public int reserveringen;
34
35 TreinBeheer tb;
36
37 public Trein(int sid, ReisBeheer rb,TreinBeheer tb) {
38 super( sid,rb,tb);
39 this .id= sid;
40
41 this .tb= tb;
42 timer = new Timer(300,new TimerHandler());
43 //timer.start();
44 inHalte=false;
45 rijdt=super .running;
46 reis=null;
47
48 //this.id=sid
49 // TODO Auto-generated constructor stub
50 }
51
52
53
54 public int getId (){
55 return id;
56 }
57
58 ////////////////////algoritme//////////////////////////////
59 public void sen0( Trein trein,int sid ){
60 //System.out.println("halte "+sid/10+" heeft
"+reisBeheer.haltes.get((sid/10)-1).reizen.size()+" reizen");
61 //als de halte de bestemming is of hij is vrij en er zijn
wachtenden in halte:
```

```
62 if(this .bestemming*10==sid||(
tb.reisBeheer.haltes.get((sid/10)-1) .reizen.size()
&& this .reis==null )){
63 //test sector of die vrij is
64 if( tb.sector[sid] .trein==null ){
65
66 tb.reisBeheer.haltes.get((sid/10)-1).open(true);
67 this .positie= sid;
68 tb.sector[sid] .trein=this;
69 this.rij(true);
70 //als het de eerste sensor is moet de laatste tb.sector
vrijgemaakt
worden
71 if(sid==10&& tb.sector[84] .trein==this ){
72 tb.sector[84] .trein=null;
73 //tb.sector[83].trein=null;
74 }
75 else if( tb.sector[sid-6] .trein==this ){
76 tb.sector[sid-6] .trein=null;
77 //tb.sector[sid-7].trein=null;
78 }
79
80 }
81 //als de sector niet vrij is stopt die
82 else if(this .positie!=sid ){
83 //System.out.println("trein "+this.id+" stopt omdat tb.sector
"+sid+"
bezet en sector nn "+this.positie+" is geblokkeert");
84 this.rij(false);
85 this.wacht();
86 }
87 }
88 //als de halte niet de bestemming is:
89 else{
90 //getest of de parallelsector aan de halte vrij is
91 if( tb.sector[sid+3] .trein==null ){
92 this.rij(true);
93 this .positie=sid+3;
94 tb.sector[sid+3] .trein=this;
95 tb.reisBeheer.haltes.get((sid/10)-1).open(false);
96 if(sid==10 && tb.sector[84] .trein==this ){
97 tb.sector[84] .trein=null;
98
99 }
100 else if( tb.sector[sid-6] .trein==this) {
101 tb.sector[sid-6] .trein=null;
102
103 }
104
105 }
106
107 else if(this .positie!=sid+3 ){
108 this.rij(false);
```

```
109 this.wacht();
110 //System.out.println("trein "+this.id+"stopt omdat sector
"+(sid+3)+"
bezet en sector ss "+this.positie+" is geblokkeert");
111 }
112
113
114
115 }
116 }
117
118 //als een this bij de uitstaphalte is:
119 public void sen1( Trein trein,int sid ){
120 //System.out.println("sen1 methode");
121 //tb.reisBeheer.haltes.get((sid/10)-1).open(false);
122 //heeft de trein een reis in zich
123 if(this .reis!=null ){
124 // System.out.println("er zit een reis in");
125
126 this.rij(false);
127
128 this .status=0;
129
130 this .reis.status=3;
131 this .reis.stopReis();
132 this .reis=null;
133 this .bestemming=0;
134 // System.out.println("trein wacht");
135 this.wacht();
136
137
138 }
139 else if( tb.sector[sid] .trein==null ){
140 this.rij(true);
141 this .positie= sid;
142 tb.sector[sid-1] .trein=null;
143 tb.sector[sid] .trein=this;
144 }
145 else if(this .positie!=sid ){
146 this.rij(false);
147 this.wacht();
148 // System.out.println("trein "+this.id+"stopt omdat sector
"+sid+" is bezet en
sector "+this.positie+" is geblokkeert");
149 }
150
151
152
153
154
155
156
157 }
```

```
158 public void sen2( Trein trein,int sid ){
159
160 if( tb.reisBeheer.haltes.get(((sid-2)/10)-1) .reizen.size()>0
&& this .reis==null
161 //oppikken
162 this .reis= tb.reisBeheer.haltes.get(((sid-2)/10)-1)
.reizen.removeLast();
163 this .reis.status=2;
164 this.rij(false);
165 this .bestemming=this .reis.bestemmingHalte.id;
166 //System.out.println("wactt"+this.inHalte);
167 this.wacht();
168 } else if( tb.sector[sid+1] .trein==null && tb.sector[sid]
.trein==null ){
169 this.rij(true);
170 this .positie= sid;
171 tb.sector[sid-1] .trein=null;
172 tb.sector[sid] .trein=this;
173 tb.sector[sid+1] .trein=this;
174 //System.out.println("wactt"+this.inHalte);
175 } else if(this .positie!=sid ){
176 this.rij(false);
177 this.wacht();
178 // System.out.println("trein "+this.id+"stopt omdat sector
"+sid+" is bezet en
sector "+this.positie+" is geblokkeert");
179 }
180 }
181 public void sen3( Trein trein,int sid ){
182 if( tb.sector[sid+1] .trein==null ){
183 tb.sector[sid] .trein=null;
184 tb.sector[sid-1] .trein=null;
185 this.rij(true);
186 tb.sector[sid+1] .trein=this;
187 this .positie=sid+1;
188
189
190 }
191 // deze om te voorkomen dat bij eventueel snelle 2e aanroep
de trein niet gestopt
worden, want hij is net daarin gezet
192 else if(this .positie!=(sid+1) )
193 {
194 this.rij(false);
195 //System.out.println("trein "+this.id+"stopt omdat sector
"+(sid+1)+" is
bezet en sector "+this.positie+" is geblokkeert");
196 this.wacht();
197 }
198
199 }
200 ////////////////////einde algoritme//////////////////////
201
```

```
202 public void wacht (){
203 timer.start();
204 wtijd=2;
205 inHalte=true;
206 }
207
208
209 public Trein getThis (){
210 return this;
211
212 }
213
214 public void addReis(Reis reis ){
215 this .reis= reis;
216 this .bestemming= rb.haltes.indexOf( reis.getBestemming());
217 // this.bestemming.addFirst(bestemming);
218 }
219
220 public void rij(boolean r ){
221 //System.out.println("trein "+id+" rijdt "+r);
222 if (r ){
223 super.start();
224 rijdt=true;
225 }
226 else{
227 super.stop();
228 rijdt=false;
229 }
230 }
231
232
233 public void setPos(int pos ){
234 positie = pos;
235 }
236
237 public Reis getReis (){
238 return reis;
239
240 }
241 public void laadUit (){
242
243 }
244 public void removeReis (){
245 reis=null;
246 status=0;
247 }
248 class TimerHandler implements ActionListener{
249
250
251 public void actionPerformed(ActionEvent e ){
252
253 if(wtijd>0 ){
254 wtijd--;
```

```
255 }
256 if((wtijd==0&&inHalte) && getThis() .positie%10==0 ){
257 inHalte=false;
258 // System.out.println("bij de timer is de pos: "+positie+"
trein
"+id);
259 getThis() .timer.stop();
260 sen1(getThis() ,positie+1);
261
262 }
263 else if((wtijd==0&&inHalte) && (getThis() .positie%10==1 )){
264 inHalte=false;
265 // System.out.println("bij de timer is de pos: "+positie+"
trein
"+id);
266 getThis() .timer.stop();
267 sen2(getThis() ,positie+1);
268 }
269
270 else if(getThis() .positie%10==2 ){
271
272 getThis() .timer.stop();
273 sen3(getThis() ,positie+1);
274 }
275 else if(getThis() .positie%10==3 ){
276
277 getThis() .timer.stop();
278
279 sen3(getThis() ,positie);
280 }
281 else if(getThis() .positie%10==4 ){
282
283 getThis() .timer.stop();
284 if(positie==84 ){
285 sen0(getThis(),10);
286 }
287 else{
288 sen0(getThis() ,positie+6);
289 }
290 }
291 }
292 }
293 }
```

TreinBeheer.java

```
1 import java.util.ArrayList;
2
3 import javax.swing.Timer;
4
5
6
7
8 public class TreinBeheer{
9 public ReisBeheer reisBeheer;
10 //actieve treinen
11 public static ArrayList<Trein> treinen = new
ArrayList<Trein>();
12
13
14 public static ArrayList<Trein> alleTreinen = new
ArrayList<Trein>();
15 RailScherm rs;
16
17 public Sector[] sector = new Sector[85];
18 Timer timer;
19
20
21 public TreinBeheer( ReisBeheer reisBeheer,RailScherm rs) {
22
23 //super.start();
24 this .rs= rs;
25 this .reisBeheer = reisBeheer;
26 for (int i = 0 ; i <10 ; i++ ){
27 // Cabs aanmaken
28 alleTreinen.add(new Trein( i,reisBeheer,this));
29 alleTreinen.get(i) .status=-1;
30 alleTreinen.get(i) .zichtbaar =false;
31 //alleTreinen.get(i).start();
32 }
33 for (int i = 0 ; i < 85 ; i++ ){
34 sector[i]=new Sector(i);
35 }
36
37 treinBij();
38
39 }
40
41 Trein trein;
42
43 int ttid,ssid;
44 public void sensorBericht(int tid, int sid ){
45
46 // hierdoor wordt voorkomen dat bij identieke berichten achter
elkaar actie wordt
genomen
47 if (!(this .ssid==sid&&this .ttid==tid)) {
```

```
48 this .ssid= sid;
49 this .ttid= tid;
50 trein= treinen.get(tid);
51
52 for(int i=0 ;i< sector.length;i++ ){
53 if(sector[i] .trein!=null)
54 System .out.println( "in sector "+i+ " rijdt trein "+sector[i]
.trein.id);
55 }
56 //als een trein bij een wissel is word hier gecheckt of dat
zijn bestemming
is, bij ja wordt de wissel geopent
57 if((sid%10)==0 ){
58 treinen.get(tid). sen0( trein,sid);
59 }
60
61 //als een trein de eerste sensor in een halte triggert, wordt
de wissel
meteen gesloten
62 if(sid%10==1 ){
63 treinen.get(tid). sen1( trein,sid);
64 }
65 if(sid%10==2 ){
66 treinen.get(tid). sen2( trein,sid);
67 }
68
69 if(sid%10==3 ){
70
71
72 treinen.get(tid). sen3( trein,sid);
73 }
74 //System.out.println("in sector "+trein.positie+" rijdt trein
"+trein.id);
75 //System.out.println("*****************");
76 //trein.positie=sid;
77 System .out.println( "sensor: "+sid+ " trein: "+tid);
78 for(int i=0 ;i< sector.length;i++ ){
79 //if(sector[i].trein!=null)
80 //System.out.println("in sector "+i+" rijdt trein
"+sector[i].trein.id)
81 }
82 //System.out.println("////////////////////");
83 if(( reisBeheer.getReizen(1).size() >
( treinen.size()*3))&& treinen.size()< alleTreinen.size() ) {
84 treinBij();
85 }
86 if(sid==83 && reisBeheer.getReizen(1).size() <
treinen.size()*3 &&
trein.reis==null&& trein.equals( treinen.get( treinen.size()-1))
&& treinen.size()>1 ){
87 treinAf();
88 }
89
90 }
91 //als de volgende sector bezet is, stop
92
93
94 }
95 public void treinBij (){
96 if(sector[84] .trein==null ){
97 System .out.println( "trein toegevoed");
98 System .out.println( "er zijn "+ alleTreinen.size()+ " alle
treinen");
99 Trein t= alleTreinen.get( treinen.size());
100 treinen.add(t);
101 sector[84] .trein= t;
102 t.positie=84;
103 t.status=0;
104 treinen.get( t.id) .zichtbaar=true;
105 treinen.get( t.id).rij(true);
106
107 }
108 }
109 public void treinAf (){
110 System .out.println( "trein verwijderd");
111 Trein t= treinen.get( treinen.size()-1);
112 t.rij(false);
113 t.positie=0;
114 trein.status=-1;
115 treinen.remove(t);
116 t.setZichtbaarheid(false);
117 sector[84] .trein=null;
118 System .out.println( "--treinAf");
119 }
120
121 }
```

Wissel.java

```
1 public class Wissel {
2
3 public boolean open;
4
5 public Wissel() {
6
7 }
8
9 public void set(boolean open) {
10 this .open = open;
11 }
12
13 public boolean status() {
14 return open;
15 }
16
```

```
17 }
18
```