```java
package programm;

import java.awt.BorderLayout;

public class ProgrammGUI extends JFrame implements ActionListener {

    private static final long serialVersionUID = -2654122148608276635L;
    JPanel p;
    JPanel p1 = new JPanel();
    JPanel p2 = new JPanel();
    JPanel p3 = new JPanel();
//===Die ganzen Knoepfe und so...=================================================================
    JLabel kontonummerlbl = new JLabel("KontoNr.:");
    JTextField kontonummertxt = new JTextField();
    JLabel betraglbl = new JLabel("Betrag:");
    JTextField betragtxt = new JTextField();
    JLabel dispolimitlbl = new JLabel("Dispolimit:");
    JTextField dispolimittxt = new JTextField();
    JButton ein = new JButton("Einzahlung");
    JButton aus = new JButton("Auszahlung");
    JButton addkonto = new JButton("add Konto");
    JButton delkonto = new JButton("del Konto");
    JMenuBar menu = new JMenuBar();

    JLabel tablbl1 = new JLabel("Konto");
    JLabel tablbl2 = new JLabel("Betrag");
    JLabel tablbl3 = new JLabel("Dispolimit");

    JTextArea kontolistetxtar = new JTextArea();
//===Tabelle=====================================================================================
    String[][] inhalt;
    JTable tabelle;
//===Menu========================================================================================
    JMenu menu1 = new JMenu("Datei");
    JMenuItem beendenItem = new JMenuItem("Beenden");
//===Map=========================================================================================
    Map<Integer, Konto> konten = new TreeMap<Integer, Konto>();

//=== Konstruktor================================================================================
    public ProgrammGUI() {
        readkonten();
        AbstractTableModel atm = new AbstractTableModel() {
            private static final long serialVersionUID = -4406771747484807059L;
            public String getColumnName(int col) {
                return null; // todo
            }
            public int getRowCount() {
                return inhalt.length;
            }
            public int getColumnCount() {
                return 3;
            }
            public Object getValueAt(int row, int col) {
                return inhalt[row][col];
            }
            public boolean isCellEditable(int row, int col) {
                return false;
            }
            public void setValueAt(Object value, int row, int col) {
                fireTableCellUpdated(row, col);
            }
        };

        map2array();
        tabelle = new JTable(atm);
        this.setTitle("Kontenverwaltung");
        initGUI();
    }

    public void initGUI() {
```

```
 99
100         setDefaultCloseOperation(DISPOSE_ON_CLOSE); // echtes Schliessen (mit d. Kreuz)!
101         // addWindowListener(new MyWindowCloser()); //Schliessen mit Bestaetigung...
102
103         addkonto.addActionListener(this);
104         delkonto.addActionListener(this);
105         ein.addActionListener(this);
106         aus.addActionListener(this);
107         beendenItem.addActionListener(this);
108
109         tabelle.addMouseListener(new MouseAdapter() {
110             public void mouseClicked(final MouseEvent e) {
111                 // Klick auf einen Tabelleneintrag
112                 if (e.getClickCount() == 1) {
113                     int i = tabelle.getSelectedRow();
114                     Konto k = konten.get(Integer.parseInt(inhalt[i][0]));
115                     kontonummertxt.setText(k.getKontonummerS());
116                     betragtxt.setText(k.getBetragS());
117                     dispolimittxt.setText(k.getDispolimitS());
118                 }
119             }
120         });
121
122         initMenu();
123         this.setJMenuBar(menu);
124         initP1();
125         initP2();
126         initP3();
127         p = (JPanel) this.getContentPane();
128         p.setLayout(new BorderLayout());
129         p.add(p1, BorderLayout.SOUTH);
130         p.add(p2, BorderLayout.CENTER);
131         this.pack();
132     }
133
134     private void initP3() {
135         p3.setLayout(new GridLayout(1, 3));
136         p3.add(tablbl1);
137         p3.add(tablbl2);
138         p3.add(tablbl3);
139     }
140
141     private void initP2() {
142         p2.setPreferredSize(new Dimension(0, 200));
143         p2.setBorder(BorderFactory.createLineBorder(Color.blue, 2));
144         p2.setLayout(new BorderLayout());
145         p2.add(p3, BorderLayout.NORTH);
146         p2.add(tabelle, BorderLayout.CENTER);
147     }
148
149     private void initP1() {
150         p1.setLayout(new GridLayout(5, 2));
151         p1.add(kontonummerlbl);
152         p1.add(kontonummertxt);
153         p1.add(betraglbl);
154         p1.add(betragtxt);
155         p1.add(dispolimitlbl);
156         p1.add(dispolimittxt);
157         p1.add(ein);
158         p1.add(aus);
159         p1.add(addkonto);
160         p1.add(delkonto);
161     }
162
163     private void initMenu() {
164         menu.add(menu1);
165         menu1.add(beendenItem);
166     }
167 //===Map in Array umwandeln=========================================================
168     private void map2array() {
```

```
169         Collection<Konto> alleKonten = konten.values();
170         inhalt = new String[konten.size()][3];
171
172         int i = 0;
173         for (Konto k : alleKonten) {
174             inhalt[i][0] = k.getKontonummerS();
175             inhalt[i][1] = k.getBetragS();
176             inhalt[i][2] = k.getDispolimitS();
177             i++;
178         }
179     }
180
181     @Override
182     public void actionPerformed(ActionEvent ereignis) {
183         final int ERROR = -1;
184         Object source = ereignis.getSource();
185
186         int kontonummer, betrag, dispolimit;
187
188         try {
189             kontonummer = Integer.parseInt(kontonummertxt.getText());
190             betrag = Integer.parseInt(betragtxt.getText());
191             dispolimit = Integer.parseInt(dispolimittxt.getText());
192
193         } catch (NumberFormatException e) {
194             kontonummer = ERROR;
195             betrag = ERROR;
196             dispolimit = ERROR;
197         }
198
199         if (source == addkonto && kontonummer != ERROR && betrag != ERROR && dispolimit != ERROR) {
200             if (kontonummer < 1 || betrag < 1) {
201                 JOptionPane.showMessageDialog(this, "Bitte geben Sie positive Zahl ein!", "Fehler!",
    JOptionPane.ERROR_MESSAGE);
202             }
203
204             else {
205                 Konto konto = new Konto(kontonummer, betrag, dispolimit);
206                 konten.put(kontonummer, konto);
207             }
208         }
209
210         if (source == delkonto && kontonummer != ERROR && betrag != ERROR && dispolimit != ERROR) {
211
212             if (kontonummer < 1 || betrag < 1) {
213                 JOptionPane.showMessageDialog(this, "Bitte geben Sie positive Zahl ein!", "Fehler!",
    JOptionPane.ERROR_MESSAGE);
214             }
215
216             else {
217             konten.remove(kontonummer);
218             }
219         }
220
221         if (source == ein && kontonummer != ERROR && betrag != ERROR && dispolimit != ERROR) {
222
223             if (kontonummer < 1 || betrag < 1) {
224                 JOptionPane.showMessageDialog(this, "Bitte geben Sie positive Zahl ein!", "Fehler!",
    JOptionPane.ERROR_MESSAGE);
225             }
226
227             else {
228                 Konto k = konten.get(kontonummer);
229                 if (k == null) {
230                     JOptionPane.showMessageDialog(this, "Konto wurde nicht gefunden!", "Fehler!",
    JOptionPane.ERROR_MESSAGE);
231                 }
232
233                 else {
234                     int neuerbetrag = betrag + k.getBetrag();
```

```
235                 k.betrag = neuerbetrag;
236             }
237         }
238     }
239
240     if (source == aus && kontonummer != ERROR && betrag != ERROR && dispolimit != ERROR) {
241
242         if (kontonummer < 1 || betrag < 1) {
243             JOptionPane.showMessageDialog(this, "Bitte geben Sie positive Zahl ein!", "Fehler!",
    JOptionPane.ERROR_MESSAGE);
244         }
245
246         else {
247             Konto k = konten.get(kontonummer);
248             if (k == null) {
249                 JOptionPane.showMessageDialog(this, "Konto wurde nicht gefunden!", "Fehler!",
    JOptionPane.ERROR_MESSAGE);
250             } else {
251                 int neuerbetrag = k.getBetrag() - betrag;
252                 if (neuerbetrag < dispolimit) {
253                     JOptionPane.showMessageDialog(this, "Sie habet d. Limit erreicht!", "Fehler!",
    JOptionPane.ERROR_MESSAGE);
254                 } else {
255                     k.betrag = neuerbetrag;
256                 }
257             }
258         }
259     }
260
261     if (kontonummer == ERROR && source != beendenItem || betrag == ERROR && source != beendenItem || dispolimit ==
    ERROR && source != beendenItem){
262         JOptionPane.showMessageDialog(this, "Falsche Eingabe!", "Fehler!", JOptionPane.ERROR_MESSAGE);
263     }
264
265     if (source == beendenItem) {
266         writekonten();
267         System.exit(0);
268     }
269
270     map2array();
271     tabelle.repaint();
272     writekonten();
273   }
274
275   @SuppressWarnings("unchecked")
276   private void readkonten() {// ===Aus einer Datei lesen!
    ==================================================
277     try {
278         ObjectInputStream input = new ObjectInputStream(new FileInputStream("konten.dat"));
279         konten = (Map<Integer, Konto>) input.readObject();
280         input.close();
281     } catch (FileNotFoundException e) {
282         JOptionPane.showMessageDialog(this, "Die Datei wurde nicht gefunden!", "Fehler!",
    JOptionPane.ERROR_MESSAGE);
283     } catch (Exception e) {
284         e.printStackTrace();
285     }
286   }
287   private void writekonten() {// ===In eine Datei schreiben!!!
    ==================================================
288     ObjectOutputStream output;
289     try {
290         output = new ObjectOutputStream(new FileOutputStream("konten.dat"));
291         output.writeObject(konten);
292         output.close();
293     } catch (Exception e) {
294         e.printStackTrace();
295     }
296   }
297 }
```