```
1  import java.awt.BorderLayout;
2  import java.awt.GridLayout;
3  import java.awt.Dimension;
4  import java.awt.FlowLayout;
5  import java.awt.Color;
6  import java.awt.event.ActionListener;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ItemEvent;
9  import java.awt.event.ItemListener;
10 import java.awt.event.MouseEvent;
11 import java.awt.event.MouseMotionListener;
12 import javax.swing.BorderFactory;
13 import javax.swing.border.*;
14 import javax.swing.ImageIcon;
15 import javax.swing.JButton;
16 import javax.swing.JComboBox;
17 import javax.swing.JFrame;
18 import javax.swing.JLabel;
19 import javax.swing.JPanel;
20 import javax.swing.JOptionPane;
21 import javax.swing.JCheckBox;
22 import javax.swing.JScrollPane;
23 import javax.swing.JTable;
24 import javax.swing.JTextArea;
25 import javax.swing.JTextField;
26 import javax.swing.JPasswordField;
27 import java.util.ArrayList;
28 import javax.swing.table.DefaultTableModel;
29
30
31 import javafx.application.Platform;
32
33 public class RunUI implements ActionListener
34 {
35
36    //Made by Paul
37    JFrame frame = new JFrame();
38    InterfaceILoader fileLoader = new FileLoader();
39    JButton
button_1,button_2,button_3,button_4,button_5,button_6;
40    JPanel panel_1, panel_2, panel_3, panel_4, panel_5,
panel_6, panel_7;
41    JComboBox box_1, box_2, box_3;
42    JTextField inputField_1, inputField_2, inputField_3,
inputField_4, inputField_5, inputField_6, inputField_7,
inputField_8, inputField_9;
43    JTextField inputField[] = new JTextField[15];
44    JCheckBox checkBox[] = new JCheckBox[15];
45    JButton button[] = new JButton[15];
46
47    String chosenAirport;
48    boolean justDraw;
```

```
49    int screen = 1;          //Which screen is currently
displayed to user.
50    int currentWish = 0;     //Which 'wish' is currently
open.
51
52
53
54
55    public static void main(String[] args)
56    {
57       RunUI initialize = new RunUI();
58    }
59
60    public RunUI()
61    {
62       fileLoader.load();       //Load all data
63
64       chosenAirport = "CPH";
65
66       inputField = new JTextField[15];
67       for (int i = 0; i < 15; i++)
68       {
69          inputField[i] = new JTextField(15);
70          checkBox[i] = new JCheckBox();
71          button[i] = new JButton();
72       }
73
74
75
76       frame.setVisible(true);
//Make the frame visible.
77
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//center the frame
78
79       Update();
80    }
81
82
83    public void Update()
84    {
85     //There goes the drawing stuff
86
87
//////////////////////////////////////////////MAIN
SCREEN//////////////////////////////////////////////
88         if (screen == 1)
89          {
90             frame.setTitle("TravelClub5050");
91             button_1 = new JButton("Login");
92             button_2 = new JButton("Register");
93             button_3 = new JButton("About");
```

```
 94              button_4 = new JButton("Help");
 95              button_5 = new JButton("Exit");
 96              button_1.addActionListener(this);
 97              button_2.addActionListener(this);
 98              button_3.addActionListener(this);
 99              button_4.addActionListener(this);
100              button_5.addActionListener(this);
101              inputField_1 = new JTextField(10);
102              inputField_2 = new JTextField(10);
103
104
105              frame.getContentPane().removeAll(); //Delete
everything on frame
106              panel_1 = new JPanel();
107              panel_2 = new JPanel();
108              panel_3 = new JPanel();
109
110              //Add stuff
111              panel_1.add(new JLabel("Welcome to
Travel5050. The best travel agency with a number in it's
name"));
112              panel_2.add(new JLabel("Username:"));
113              panel_2.add(inputField_1);
114              panel_2.add(new JLabel("Password:"));
115              panel_2.add(inputField_2);
116              panel_2.add(button_1);
117              panel_3.add(button_2);
118              panel_3.add(button_3);
119              panel_3.add(button_4);
120              panel_3.add(button_5);
121
122              frame.setLayout(new BorderLayout()); //Border
layout on frame.
123              frame.add(panel_1, BorderLayout.NORTH);
124              frame.add(panel_2, BorderLayout.CENTER);
125              frame.add(panel_3, BorderLayout.SOUTH);
126              frame.setLocationRelativeTo(null); //centers
on screen
127              frame.pack();                          //Display
it
128          }
129
130
131
132
133
134
////////////////////////////////////////////Create a new
account////////////////////////////////////////////
135          if (screen == 2)
136          {
137              frame.setTitle("Create new account");
```

```
138             button_1 = new JButton("Create account");
139             button_2 = new JButton("Back");
140             button_1.addActionListener(this);
141             button_2.addActionListener(this);
142             inputField_1 = new JTextField(15);
143             inputField_2 = new JTextField(15);
144             inputField_3 = new JTextField(15);
145             inputField_4 = new JTextField(15);
146             inputField_5 = new JTextField(15);
147             inputField_6 = new JTextField(15);
148             inputField_7 = new JTextField(15);
149             inputField_8 = new JTextField(15);
150             inputField_9 = new JTextField(15);
151
152             frame.getContentPane().removeAll(); //Delete
everything on frame
153             panel_1 = new JPanel();
154             panel_2 = new JPanel();
155             panel_3 = new JPanel();
156
157             //Add stuff
158             panel_1.setLayout(new GridLayout(10,1));
159             panel_1.add(new JLabel("First Name:  "));
160             panel_1.add(new JLabel("Last Name:  "));
161             panel_1.add(new JLabel("Email address:
"));
162             panel_1.add(new JLabel("Phone number:  "));
163             panel_1.add(new JLabel("Nationality:  "));
164             panel_1.add(new JLabel("Password:  "));
165             panel_1.add(new JLabel("Address:  "));
166             panel_1.add(new JLabel("Age:  "));
167             panel_1.add(new JLabel("Username:  "));
168             panel_2.setLayout(new GridLayout(10,1));
169             panel_2.add(inputField_1);
170             panel_2.add(inputField_2);
171             panel_2.add(inputField_3);
172             panel_2.add(inputField_4);
173             panel_2.add(inputField_5);
174             panel_2.add(inputField_6);
175             panel_2.add(inputField_7);
176             panel_2.add(inputField_8);
177             panel_2.add(inputField_9);
178             panel_3.add(button_1);
179             panel_3.add(button_2);
180
181             frame.setLayout(new BorderLayout()); //Border
layout on frame.
182             frame.add(panel_1, BorderLayout.WEST);
183             frame.add(panel_2, BorderLayout.EAST);
184             frame.add(panel_3, BorderLayout.SOUTH);
185             frame.pack();                       //Display
it
```

```
186          }
187
188
189
190
191
192
193
//////////////////////////////////////////////Help
screen//////////////////////////////////////////////
194          if (screen == 3)
195          {
196              frame.setTitle("Help screen");
197
198              frame.getContentPane().removeAll(); //Delete
everything on frame
199              panel_1 = new JPanel();
200              panel_2 = new JPanel();
201              panel_3 = new JPanel();
202              button_1 = new JButton("Back");
203              button_1.addActionListener(this);
204
205              //Add stuff
206              panel_1.setLayout(new GridLayout(10,1));
207              panel_1.add(new JLabel("Some advices to test
the system"));
208              panel_1.add(new JLabel("1. Create an
account"));
209              panel_1.add(new JLabel("2. Log in using the
username and password"));
210              panel_1.add(new JLabel("3. Create a new
wish"));
211              panel_1.add(new JLabel("4. Check the offer
that the system made on new wish"));
212              panel_1.add(new JLabel("It's possible to log
in as an admin using username: admin  password: admin"));
213              panel_2.add(button_1);
214
215              frame.setLayout(new BorderLayout()); //Border
layout on frame.
216              frame.add(panel_1, BorderLayout.CENTER);
217              frame.add(panel_2, BorderLayout.SOUTH);
218              frame.pack();                          //Display
it
219          }
220
221
222
223
224
225
226
```

```
//////////////////////////////////////////////About screen//////////////////////////////////////////////
            if (screen == 4)
            {
                frame.setTitle("About us");

                frame.getContentPane().removeAll(); //Delete everything on frame
                panel_1 = new JPanel();
                panel_2 = new JPanel();
                panel_3 = new JPanel();
                button_1 = new JButton("Back");
                button_1.addActionListener(this);


                //Add stuff
                panel_1.setLayout(new GridLayout(10,1));
                panel_1.add(new JLabel("Travelclub5050 early prototype"));
                panel_1.add(new JLabel("Paul - User interface"));
                panel_1.add(new JLabel("Oliver - Users, wishes"));
                panel_1.add(new JLabel("Adam - location and extras"));
                panel_1.add(new JLabel("Kaspars - saving, loading"));
                panel_2.add(button_1);


                frame.setLayout(new BorderLayout()); //Border layout on frame.
                frame.add(panel_1, BorderLayout.CENTER);
                frame.add(panel_2, BorderLayout.SOUTH);
                frame.pack();                        //Display it
            }




//////////////////////////////////////////////User - main screen//////////////////////////////////////////////
            if (screen == 5)
            {
                frame.setTitle("User main menu");
                button[0] = new JButton("New Wish");
                button[1] = new JButton("Log out");
                button[0].addActionListener(this);
                button[1].addActionListener(this);
```

```
 267
 268
 269             frame.getContentPane().removeAll(); //Delete
everything on frame
 270             panel_1 = new JPanel();
 271             panel_2 = new JPanel();
 272             panel_3 = new JPanel();
 273
 274             //Set up buttons for wishes
 275             for (int i=0;i<10;i++)
 276             {
 277                button[2+i] = new JButton("See offer");
 278                button[2+i].addActionListener(this);
 279             }
 280
 281             //Add stuff
 282             String[] wishNameList =
fileLoader.getWishesName();
 283             for (int i=0; i < wishNameList.length ;i++)
 284             {
 285                if (i > 10)
 286                   break;
 287
 288                panel_1.setLayout(new GridLayout(10,1));
 289                panel_1.add(new JLabel(wishNameList[i]));
 290
 291                panel_2.setLayout(new GridLayout(10,1));
 292                panel_2.add(button[2+i]);
 293             }
 294
 295             panel_3.add(button[0]);
 296             panel_3.add(button[1]);
 297
 298             frame.setLayout(new BorderLayout()); //Border
layout on frame.
 299             frame.add(panel_1, BorderLayout.WEST);
 300             frame.add(panel_2, BorderLayout.EAST);
 301             frame.add(panel_3, BorderLayout.SOUTH);
 302             frame.pack();
 303          }
 304
 305
 306
 307
 308
 309
/////////////////////////////////////////////Admin - main
screen/////////////////////////////////////////////
 310          if (screen == 6)
 311          {
 312             frame.setTitle("Admin");
 313             button[0] = new JButton("See Users");
```

```
 314             button[1] = new JButton("Edit addons");
 315             button[2] = new JButton("Backup");
 316             button[3] = new JButton("Log out");
 317             button[4] = new JButton("Inbox");
 318             button[5] = new JButton("Financial Report");
 319             button[0].addActionListener(this);
 320             button[1].addActionListener(this);
 321             button[2].addActionListener(this);
 322             button[3].addActionListener(this);
 323             button[4].addActionListener(this);
 324             button[5].addActionListener(this);
 325
 326
 327             frame.getContentPane().removeAll(); //Delete
everything on frame
 328             panel_1 = new JPanel();
 329             panel_2 = new JPanel();
 330             panel_3 = new JPanel();
 331             panel_4 = new JPanel();
 332
 333             //Add stuff
 334             panel_1.setLayout(new GridLayout(2,1));
 335             panel_1.add(new JLabel("     You have 0
urgent messages"));
 336             panel_2.add(button[4]);
 337             panel_2.add(button[5]);
 338             panel_3.add(button[0]);
 339             panel_3.add(button[1]);
 340             panel_3.add(button[2]);
 341             panel_3.add(button[3]);
 342             frame.setLayout(new BorderLayout()); //Border
layout on frame.
 343             frame.add(panel_1, BorderLayout.WEST);
 344             frame.add(panel_2, BorderLayout.EAST);
 345             frame.add(panel_3, BorderLayout.SOUTH);
 346             frame.pack();                        //Display
it
 347             frame.setSize(400, 150); // sets frame size
 348             frame.setLocationRelativeTo(null); //centers
on screen
 349         }
 350
 351
 352
 353
 354
 355
/////////////////////////////////////////////User - New
wish/////////////////////////////////////////////
 356         if (screen == 7)
 357         {
 358             justDraw = true;
```

```
 359
 360             frame.setTitle("New Wish");
 361             button_1 = new JButton("Create wish");
 362             button_2 = new JButton("Back");
 363             button_1.addActionListener(this);
 364             button_2.addActionListener(this);
 365             inputField_1 = new JTextField(15);
 366             inputField_2 = new JTextField(15);
 367             inputField_3 = new JTextField(15);
 368             inputField_4 = new JTextField(15);
 369             inputField_5 = new JTextField(15);
 370             box_1 = new JComboBox();
 371             box_2 = new JComboBox();
 372             box_2.addActionListener(this);
 373
 374             for (int i=0; i<15;i++)
 375             {
 376                inputField[i].setText("");
 377             }
 378
 379             String[] locationsNames =
fileLoader.getLocations();
 380             for (int i = 0; i < locationsNames.length;
i++)
 381             {
 382                box_1.addItem(locationsNames[i]);
 383                box_2.addItem(locationsNames[i]);
 384             }
 385
 386             frame.getContentPane().removeAll(); //Delete
everything on frame
 387             panel_1 = new JPanel();
 388             panel_2 = new JPanel();
 389             panel_3 = new JPanel();
 390             panel_4 = new JPanel();
 391             panel_5 = new JPanel();
 392             panel_6 = new JPanel();
 393             panel_7 = new JPanel();
 394
 395             //Add stuff
 396             panel_1.setLayout(new GridLayout(10,1));
 397             panel_1.add(new JLabel("From:  "));
 398             panel_1.add(new JLabel("To:  "));
 399             panel_1.add(new JLabel("Departure date:
"));
 400             panel_1.add(new JLabel("Return date:
"));
 401             panel_1.add(new JLabel("Max Price:  "));
 402             panel_1.add(new JLabel("Number of people:
"));
 403             panel_1.add(new JLabel("+/- days for
departure:  "));
```

```
404             panel_1.add(new JLabel("+/- days for return:
"));
405
406             panel_2.setLayout(new GridLayout(10,1));
407             panel_2.add(box_1);
408             panel_2.add(box_2);
409             panel_2.add(inputField[0]);
410             panel_2.add(inputField[1]);
411             panel_2.add(inputField[2]);
412             panel_2.add(inputField[3]);
413             panel_2.add(inputField[4]);
414             panel_2.add(inputField[5]);
415             panel_2.add(inputField[6]);
416
417
418             panel_3.setLayout(new GridLayout(10,1));
419             panel_4.setLayout(new GridLayout(10,1));
420             panel_5.setLayout(new GridLayout(10,1));
421             panel_6.setLayout(new GridLayout(1,5));
422
423             String[] extrasName =
fileLoader.getExtras(chosenAirport);
424
425             panel_3.add(new JLabel("Add-on:  "));
426             panel_4.add(new JLabel("Your price:  "));
427             panel_5.add(new JLabel("Do you want it:  "));
428             for (int i = 0; i < extrasName.length; i++)
429             {
430                panel_3.add(new JLabel(extrasName[i]));
431                panel_4.add(inputField[6+i]);
432                panel_5.add(checkBox[i]);
433             }
434
435
436             panel_6.add(button_1);
437             panel_6.add(button_2);
438
439
440
441             panel_7.setLayout(new BorderLayout());
442             panel_7.add(panel_3, BorderLayout.WEST);
443             panel_7.add(panel_4, BorderLayout.CENTER);
444             panel_7.add(panel_5, BorderLayout.EAST);
445             panel_7.add(panel_6, BorderLayout.SOUTH);
446
447             frame.setLayout(new BorderLayout()); //Border
layout on frame.
448             frame.add(panel_1, BorderLayout.WEST);
449             frame.add(panel_2, BorderLayout.EAST);
450             frame.add(panel_7, BorderLayout.SOUTH);
451             frame.pack();
452             justDraw = false;
```

```
453         }
454
455
456
457
458
459
//////////////////////////////////////////////User - Edit
wish//////////////////////////////////////////////
460         if (screen == 8)
461         {
462              frame.setTitle("Wish details");
463
464             button[0] = new JButton("Delete wish");
465             button[1] = new JButton("Back");
466             button[2] = new JButton("Pay");
467             button[0].addActionListener(this);
468             button[1].addActionListener(this);
469             button[2].addActionListener(this);
470
471
472             frame.getContentPane().removeAll(); //Delete
everything on frame
473             panel_1 = new JPanel();
474             panel_2 = new JPanel();
475             panel_3 = new JPanel();
476             panel_4 = new JPanel();
477             panel_5 = new JPanel();
478             panel_6 = new JPanel();
479             panel_7 = new JPanel();
480
481
482
483             String[] extrasName =
fileLoader.getChosenExtras(currentWish);
484
485             panel_3.add(new JLabel("Add-ons:  "));
486             for (int i = 0; i < extrasName.length; i++)
487             {
488                System.out.println(extrasName[i]);
489                panel_3.add(new JLabel(extrasName[i]));
490             }
491
492             //Add stuff
493             panel_1.setLayout(new GridLayout(10,1));
494             panel_1.add(new JLabel("From:  "));
495             panel_1.add(new JLabel("To:  "));
496             panel_1.add(new JLabel("Departure date:
"));
497             panel_1.add(new JLabel("Return date:
"));
498             panel_1.add(new JLabel("Max Price:  "));
```

```
 499                panel_1.add(new JLabel("Number of people:
"));
 500                panel_1.add(new JLabel("+/- days for
departure:  "));
 501                panel_1.add(new JLabel("+/- days for return:
"));
 502
 503                panel_2.setLayout(new GridLayout(10,1));
 504                panel_2.add(new
JLabel(fileLoader.getWishFrom(currentWish)));
 505                panel_2.add(new
JLabel(fileLoader.getWishTo(currentWish)));
 506                panel_2.add(new
JLabel(fileLoader.getWishDeparture(currentWish)));
 507                panel_2.add(new
JLabel(fileLoader.getWishReturn(currentWish)));
 508                panel_2.add(new
JLabel(fileLoader.getWishMaxPrice(currentWish)));
 509                panel_2.add(new
JLabel(fileLoader.getWishNumberOfPeople(currentWish)));
 510                panel_2.add(new
JLabel(fileLoader.getWishDaysOfDeparture(currentWish)));
 511                panel_2.add(new
JLabel(fileLoader.getWishDaysOfReturn(currentWish)));
 512
 513
 514                panel_3.setLayout(new GridLayout(10,1));
 515                panel_4.setLayout(new GridLayout(10,1));
 516                panel_5.setLayout(new GridLayout(10,1));
 517                panel_6.setLayout(new BorderLayout());
 518
 519
 520
 521                //add pay total sum  borderlayour -
center/panel6
 522
System.out.println(fileLoader.getWishPrice(currentWish));
 523
System.out.println(fileLoader.getWishExpectedTotal(currentWish
));
 524                if
(Integer.parseInt(fileLoader.getWishPrice(currentWish)) >
Integer.parseInt(fileLoader.getWishExpectedTotal(currentWish))
)
 525                {
 526                   panel_6.add(new JLabel("Sorry, we couldn't
get the price you wanted. But we can offer the trip for: " +
fileLoader.getWishPrice(currentWish) +
"Kr"),BorderLayout.NORTH);
 527                }
 528                else
 529                {
```

```
 530                 panel_6.add(new JLabel("Great! The trip
came out " +
(Integer.parseInt(fileLoader.getWishExpectedTotal(currentWish)
) - Integer.parseInt(fileLoader.getWishPrice(currentWish))) +
" Kr. cheaper than expected. Total sum to pay: " +
fileLoader.getWishPrice(currentWish) + "
Kr."),BorderLayout.NORTH);
 531             }
 532
 533             //
 534             panel_6.add(button[0], BorderLayout.WEST);
 535             panel_6.add(button[1], BorderLayout.CENTER);
 536             panel_6.add(button[2], BorderLayout.EAST);
 537
 538
 539             panel_7.setLayout(new BorderLayout());
 540             panel_7.add(panel_3, BorderLayout.WEST);
 541             panel_7.add(panel_4, BorderLayout.CENTER);
 542             panel_7.add(panel_6, BorderLayout.SOUTH);
 543
 544             frame.setLayout(new BorderLayout()); //Border
layout on frame.
 545             frame.add(panel_1, BorderLayout.WEST);
 546             frame.add(panel_2, BorderLayout.EAST);
 547             frame.add(panel_7, BorderLayout.SOUTH);
 548             frame.pack();
 549         }
 550
 551
 552
 553
////////////////////////////////////////////Payment - by
Adam/////////////////////////////////////////////
 554         if (screen == 9)
 555         {
 556             frame.getContentPane().removeAll(); //Delete
everything on frame
 557             frame.setTitle("Payment");
 558
 559
 560             JTextField txtField = new JTextField(40);
 561             JTextField txt1Field = new JTextField(40);
 562             JTextField txt2Field = new JTextField(40);
 563             button[0] = new JButton("Back");
 564             button[1] = new JButton("Pay");
 565
 566
 567             button[0].addActionListener(this);
 568             button[1].addActionListener(this);
 569
 570             JPanel north = new JPanel(new GridLayout(3,
2)); //panel with gridlayout to go into north section of
```

```
frames borderlayout.
 571             north.add(new JLabel("Type your credit card
number:"));
 572             north.add(txtField);
 573             north.add(new JLabel("Security number:"));
 574             north.add(txt1Field);
 575             north.add(new JLabel("Owner name:"));
 576             north.add(txt2Field);
 577
 578             JPanel south = new JPanel(new
GridLayout(1,2));
 579             south.add(button[0]);
 580             south.add(button[1]);
 581
 582
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 583             frame.setLayout(new BorderLayout());
//Borderlayout on frame.
 584             frame.add(north, BorderLayout.NORTH);
//previously built panel in north section of frames
borderlayout.
 585             frame.add(south, BorderLayout.SOUTH);
//button in south section.
 586             frame.pack(); //sets frames size depending on
components in it.
 587             frame.setVisible(true); //make frame visible
 588
 589         }
 590
 591
 592     }
 593
 594
 595
 596
 597
 598
 599
 600
 601
 602
 603
 604
 605
 606
 607     public void actionPerformed(ActionEvent event){
// ACTION PART
 608           if (screen == 1)  //Main screen
 609           {
 610              if (event.getSource() == button_1)
 611              {
 612                 boolean isOK = true;
```

```
613                 if (isString(inputField_1.getText()) ==
false)
614                 {
615                    isOK = false;
616                 }
617                 if (isString(inputField_2.getText()) ==
false)
618                 {
619                    isOK = false;
620                 }
621 
622                 if (isOK == true)
623                 {
624                    //try loggin in
625                    if
(inputField_1.getText().equals("admin") == true)
626                    {
627                       if
(inputField_2.getText().equals("admin") == true)
628                       {
629                          screen = 6;
630                          Update();
631                       }
632                    }
633                    if
(fileLoader.tryLogin(inputField_1.getText(),
inputField_2.getText()) == 1)
634                    {
635                       //Change to login screen
636                       screen = 5; //User - main screen
637                       Update();
638                    }
639                    else
640                    {
641                       //Cannot log in
642                    }
643                 }
644 
645 
646              }
647              if (event.getSource() == button_2)
648              {
649                 //Change to register screen
650                 screen = 2; //Register
651                 Update();
652              }
653              if (event.getSource() == button_3)
654              {
655                 //Change to about screen
656                 screen = 4; //About screen
657                 Update();
658              }
```

```
659             if (event.getSource() == button_4)
660             {
661                //Change to help screem
662                screen = 3; //Help screen
663                Update();
664             }
665             if (event.getSource() == button_5)
666             {
667                //Exit
668                fileLoader.save();
669                System.exit(0);
670             }
671          }
672
673
674
675
676          if (screen == 2)  //Create new user
677          {
678             if (event.getSource() == button_1)
679             {
680                boolean isOK = true;
681                boolean somethingIsEmpty = false;
682
683
684
685                if (isString(inputField_1.getText()) ==
false)
686                {
687                   somethingIsEmpty = true;
688                   isOK = false;
689                }
690                if
(stringNoNumbers(inputField_1.getText()) == false)
691                {
692                   isOK = false;
693
JOptionPane.showMessageDialog(null,"Your name can't contain
numbers!","Error",JOptionPane.WARNING_MESSAGE);
694                }
695
696                if (isString(inputField_2.getText()) ==
false)
697                {
698                   somethingIsEmpty = true;
699                   isOK = false;
700                }
701                if
(stringNoNumbers(inputField_2.getText()) == false)
702                {
703                   isOK = false;
704
```

```
JOptionPane.showMessageDialog(null,"Your last name can't
contain numbers!","Error",JOptionPane.WARNING_MESSAGE);
 705                }
 706
 707                if (isString(inputField_3.getText()) ==
false)
 708                {
 709                   somethingIsEmpty = true;
 710                   isOK = false;
 711                }
 712
 713                if (isString(inputField_4.getText()) ==
false)
 714                {
 715                   somethingIsEmpty = true;
 716                   isOK = false;
 717                }
 718                if
(stringNoLetters(inputField_4.getText()) == false)
 719                {
 720                   isOK = false;
 721
JOptionPane.showMessageDialog(null,"Your phone number can't
contain letters!","Error",JOptionPane.WARNING_MESSAGE);
 722                }
 723
 724                if (isString(inputField_5.getText()) ==
false)
 725                {
 726                   somethingIsEmpty = true;
 727                   isOK = false;
 728                }
 729                if
(stringNoNumbers(inputField_5.getText()) == false)
 730                {
 731                   isOK = false;
 732
JOptionPane.showMessageDialog(null,"Your nationality can't
contain numbers","Error",JOptionPane.WARNING_MESSAGE);
 733                }
 734
 735                if (inputField_6.getText().length() < 8)
 736                {
 737
JOptionPane.showMessageDialog(null,"Password has to be atleast
8 characters long!","Error",JOptionPane.WARNING_MESSAGE);
 738                   isOK = false;
 739                }
 740
 741                if (inputField_7.getText().length() < 5)
 742                {
 743
```

```
JOptionPane.showMessageDialog(null,"Please enter your full
address","Error",JOptionPane.WARNING_MESSAGE);
 744                    isOK = false;
 745                }
 746
 747                if (inputField_8.getText().length() < 1)
 748                {
 749                    somethingIsEmpty = true;
 750                    isOK = false;
 751                }
 752                if
(stringNoLetters(inputField_8.getText()) == false)
 753                {
 754                    JOptionPane.showMessageDialog(null,"Age
can only be in numbers!","Error",JOptionPane.WARNING_MESSAGE);
 755                    isOK = false;
 756                }
 757
 758                if (inputField_9.getText().length() < 1)
 759                {
 760                    somethingIsEmpty = true;
 761                    isOK = false;
 762                }
 763                if
(fileLoader.checkUsername(inputField_9.getText()) == 1)
 764                {
 765                    isOK = false;
 766
JOptionPane.showMessageDialog(null,"Username already
taken","Error",JOptionPane.WARNING_MESSAGE);
 767                }
 768
 769
 770                if (somethingIsEmpty == true)
 771                {
 772                    JOptionPane.showMessageDialog(null,"You
must fill out all
fields!","Error",JOptionPane.WARNING_MESSAGE);
 773                }
 774
 775                if (isOK == true)
 776                {
 777                    //Save the user
 778
fileLoader.makeNewUser(inputField_1.getText(),
inputField_2.getText(),  inputField_6.getText(),
inputField_3.getText(),  inputField_4.getText(),
inputField_5.getText(), inputField_7.getText(),
inputField_8.getText(), inputField_9.getText());
 779
 780                    //Change to main screem
 781                    screen = 1; //Main screen
```

```
782                     Update();
783                 }
784             }
785             if (event.getSource() == button_2)
786             {
787                 //Change to main screem
788                 screen = 1; //Main screen
789                 Update();
790             }
791         }
792 
793 
794 
795 
796         if (screen == 3)  //Help screen
797         {
798             if (event.getSource() == button_1)
799             {
800                 //Change to main screem
801                 screen = 1; //Main screen
802                 Update();
803             }
804         }
805 
806 
807         if (screen == 4)  //About screen
808         {
809             if (event.getSource() == button_1)
810             {
811                 //Change to main screem
812                 screen = 1; //Main screen
813                 Update();
814             }
815         }
816 
817 
818 
819 
820         if (screen == 5)  //User - main screen
821         {
822           if (event.getSource() == button[0])
823           {
824               //Change to new wish screem
825               screen = 7;
826               Update();
827           }
828           if (event.getSource() == button[1])
829           {
830               //Change to main menu screem
831               screen = 1;
832               Update();
833           }
```

```
834
835             //Go on individual wish window.
836             for (int i=0;i<10;i++)
837             {
838                 if (event.getSource() == button[2+i])
839                 {
840                    currentWish = i;
841                    screen = 8;
842                    Update();
843                 }
844             }
845          }
846
847
848
849
850
851
852          if (screen == 6)  //Admin - main screen
853          {
854             if (event.getSource() == button[3])
855                {
856                   screen = 1;
857                   Update();
858                }
859          }
860          if (screen == 7)  //User - new wish
861          {
862            if (event.getSource() == button_1)   //Try to
create new wish
863             {
864              boolean isOK = true;
865              boolean somethingIsEmpty = false;
866
867              if
(box_1.getSelectedItem().toString().substring(0,3).equals(box_
2.getSelectedItem().toString().substring(0,3)) == true)
868              {
869                 isOK = false;
870                 JOptionPane.showMessageDialog(null,"You
can't fly without
flying!","Error",JOptionPane.WARNING_MESSAGE);
871              }
872              //check if inputField[0] == date
873              //check if inputField[1] == date
874              if (isInteger(inputField[2].getText()) ==
false)
875                 {
876                    isOK = false;
877                    JOptionPane.showMessageDialog(null,"You
can't use letters for max
price","Error",JOptionPane.WARNING_MESSAGE);
```

```
878                 }
879              if (inputField[2].getText().length() < 1)
880                 {
881                    somethingIsEmpty = true;
882                    isOK = false;
883                 }
884
885              if (isInteger(inputField[3].getText()) ==
false)
886                 {
887                    isOK = false;
888                    JOptionPane.showMessageDialog(null,"You
can't use letters for number of
people","Error",JOptionPane.WARNING_MESSAGE);
889                 }
890              if (inputField[3].getText().length() < 1)
891                 {
892                    somethingIsEmpty = true;
893                    isOK = false;
894                 }
895
896              if (isInteger(inputField[4].getText()) ==
false)
897                 {
898                    isOK = false;
899                    JOptionPane.showMessageDialog(null,"You
can't use letters for days of
departure","Error",JOptionPane.WARNING_MESSAGE);
900                 }
901              if (inputField[4].getText().length() < 1)
902                 {
903                    somethingIsEmpty = true;
904                    isOK = false;
905                 }
906
907              if (isInteger(inputField[5].getText()) ==
false)
908                 {
909                    isOK = false;
910                    JOptionPane.showMessageDialog(null,"You
can't use letters for days of
return","Error",JOptionPane.WARNING_MESSAGE);
911                 }
912              if (inputField[5].getText().length() < 1)
913                 {
914                    somethingIsEmpty = true;
915                    isOK = false;
916                 }
917
918
919                 ArrayList<Extra> extrasList = new
ArrayList<Extra>();
```

```
 920                 String[] extrasPriceList =
fileLoader.getLocationPrices(chosenAirport);
 921
 922                 String[] extrasName =
fileLoader.getExtras(chosenAirport);
 923                 for (int i = 0; i < extrasName.length;
i++)
 924                   {
 925                       if (checkBox[i].isSelected() ==
true)
 926                       {
 927
 928
 929                          //Check if the inputBox contains
something
 930                          if (inputField[6 +
i].getText().length() < 1)
 931                          {
 932                             somethingIsEmpty = true;
 933                             isOK = false;
 934
 935                          }
 936                          else
 937                          {
 938                             //Check if the inputBox
contains only numbers. Letters not allowed.
 939                             if (inputField[6 +
i].getText().matches("^-?\\d+$") == false)
 940                             {
 941
JOptionPane.showMessageDialog(null,"You must enter only
numbers for the price for Add-
on","Error",JOptionPane.WARNING_MESSAGE);
 942                                isOK = false;
 943                             }
 944                             else
 945                             {
 946                                //List of chosen extras and
their prices.
 947                                extrasList.add(new
Extra(chosenAirport, extrasName[i],
Integer.parseInt(inputField[6 + i].getText()),
Integer.parseInt(extrasPriceList[i])));
 948
System.out.println(chosenAirport + " " + extrasName[i] + " " +
inputField[6 + i].getText() + " " + extrasPriceList[i]);
 949                             }
 950
 951                          }
 952                       }
 953                   }
 954
```

```
955
956
957
958
959                 if (somethingIsEmpty == true)
960                    {
961                       JOptionPane.showMessageDialog(null,"You
must fill out all
fields!","Error",JOptionPane.WARNING_MESSAGE);
962                    }
963
964                 if (isOK == true)
965                    {
966                       //Change to user main screen. Try to
create a new wish
967                       //String airportFrom, String airportTo,
String dateDeparture, String dateReturn, int numberOfPersons
, int plusMinusDateDeparture, int plusMinusDateReturn, boolean
contactBeforeBooking, boolean reserveWithoutPaying, boolean
paid, int maxPrice, ArrayList<Extra> extrasList
968                       System.out.println("Trying to create
wish");
969
fileLoader.makeNewWish(box_1.getSelectedItem().toString().subs
tring(0,3), box_2.getSelectedItem().toString().substring(0,3),
inputField[0].getText(), inputField[1].getText(),
Integer.parseInt(inputField[3].getText()),
Integer.parseInt(inputField[4].getText()),
Integer.parseInt(inputField[5].getText()), false, false,
false, Integer.parseInt(inputField[2].getText()),extrasList);
970                       fileLoader.save();
971
972                       //fileLoader.makeNewWish();
973                       screen = 5; //new wish screen
974                       Update();
975                    }
976                 }
977
978              if (event.getSource() == button_2)
979              {
980                  //Change to new wish screem
981                  screen = 5; //new wish screen
982                  Update();
983                 }
984                 if (event.getSource() == box_2)
985                 {
986                    if (justDraw == false)
987                    {
988                      //System.out.println("changed");
989                       String selectedItem =
box_2.getSelectedItem().toString();//box_2.getSelectedValue().
toString();
```

```
 990                    chosenAirport =
selectedItem.substring(0,3);
 991                    System.out.println(chosenAirport);
 992
 993
 994
 995                    panel_7.removeAll();
 996
 997                    panel_3 = new JPanel();
 998                    panel_4 = new JPanel();
 999                    panel_5 = new JPanel();
1000                    panel_6 = new JPanel();
1001
1002                    panel_3.setLayout(new
GridLayout(10,1));
1003                    panel_4.setLayout(new
GridLayout(10,1));
1004                    panel_5.setLayout(new
GridLayout(10,1));
1005                    panel_6.setLayout(new GridLayout(1,5));
1006
1007                   // panel_3.add();
1008                    String[] extrasName =
fileLoader.getExtras(chosenAirport);
1009
1010                    panel_3.add(new JLabel("Add-on:  "));
1011                    panel_4.add(new JLabel("Your price:
"));
1012                    panel_5.add(new JLabel("Do you want it:
"));
1013                    for (int i = 0; i < extrasName.length;
i++)
1014                    {
1015                       panel_3.add(new
JLabel(extrasName[i]));
1016                       panel_4.add(inputField[6+i]);
1017                       panel_5.add(checkBox[i]);
1018                    }
1019
1020                    panel_6.add(button_1);
1021                    panel_6.add(button_2);
1022
1023                    panel_7.add(panel_3,
BorderLayout.WEST);
1024                    panel_7.add(panel_4,
BorderLayout.CENTER);
1025                    panel_7.add(panel_5,
BorderLayout.EAST);
1026                    panel_7.add(panel_6,
BorderLayout.SOUTH);
1027
1028                    panel_7.revalidate();
```

```
1029                    frame.repaint();
1030                }
1031             }
1032          }
1033          if (screen == 8)  //User - My wishes
1034          {
1035            if (event.getSource() == button[0])
//DELETE
1036            {
1037                screen = 5;
1038                fileLoader.deleteWish(currentWish);
1039                fileLoader.save();
1040                Update();
1041            }
1042            if (event.getSource() == button[1])
//BACK
1043            {
1044                //Change to main menu screem
1045                screen = 5;
1046                Update();
1047            }
1048            if (event.getSource() == button[2])
//PAY
1049            {
1050                //Change to main menu screem
1051                screen = 9;
1052                Update();
1053            }
1054          }
1055
1056          if (screen == 9)  //PAY
1057          {
1058            if (event.getSource() == button[0])
1059            {
1060                //Change to main menu screem
1061                screen = 5;
1062                Update();
1063            }
1064            if (event.getSource() == button[1])
1065            {
1066                //Change to main menu screem
1067
JOptionPane.showMessageDialog(button[0],"Thank you for
payment, you will recive message with confirmation of
succesfull payment.");
1068            }
1069          }
1070
1071          if (screen == 10)  //Admin - edit wish
1072          {
1073
1074          }
```

```
1075    }
1076
1077
1078
1079
1080
1081    public boolean isString(String x)
1082    {
1083       //Check if string is NOT empty
1084       if (x.length() >= 2){
1085          return true;
1086       } else{
1087          return false;
1088       }
1089    }
1090    public boolean stringNoNumbers(String x)
1091    {
1092       if (x.length() > 0)
1093       {
1094          //Things you can find on google....
1095          if(x.matches(".*\\d.*")){
1096             return false;      //Does contain a number
1097          } else{
1098             return true;       //Does not contain a
number
1099          }
1100       }
1101       else
1102       {
1103          return true;
1104       }
1105    }
1106    public boolean stringNoLetters(String x)
1107    {
1108       if (x.length() > 0)
1109       {
1110          if(x.matches(".*\\d.*")){
1111             return true;      //Does contain a letter
1112          } else{
1113             return false;       //Does not contain a
letter
1114          }
1115       }
1116       else
1117       {
1118          return true;
1119       }
1120    }
1121
1122
1123     public static boolean isInteger(String s) {
1124       return isInteger(s,10);
```

```
1125     }
1126
1127     public static boolean isInteger(String s, int radix)
{
1128           if(s.isEmpty()) return false;
1129           for(int i = 0; i < s.length(); i++) {
1130               if(i == 0 && s.charAt(i) == '-') {
1131                   if(s.length() == 1) return false;
1132                   else continue;
1133               }
1134               if(Character.digit(s.charAt(i),radix) < 0)
return false;
1135           }
1136           return true;
1137      }
1138 }
1139
```

```
1 import java.util.ArrayList;
 2 import java.io.*;
 3
 4 public interface InterfaceILoader
 5 {
 6    //Made by Kaspars
 7    public void load();     //This loads up all the data
 8    public void save();     //This saves all the data
 9
10    //User management
11    public int tryLogin(String username, String password);
12    public int getUserType(String username);
13    public void makeNewUser(String firstName, String
lastName, String password, String email, String phone, String
nationality, String address, String age, String username);
14    public int checkUsername(String username);
15
16    //Functions to get information about particular wish
17    public String[] getExtras(String location);
18    public String[] getLocations();
19    public String[] getLocationPrices(String location);
20    public String[] getWishesName();
21    public String[] getChosenExtras(int wish);
22    public String getWishFrom(int wish);
23    public String getWishTo(int wish);
24    public String getWishDeparture(int wish);
25    public String getWishReturn(int wish);
26    public String getWishMaxPrice(int wish);
27    public String getWishNumberOfPeople(int wish);
```

```
28    public String getWishDaysOfDeparture(int wish);
29    public String getWishDaysOfReturn(int wish);
30    public String getWishPrice(int wish);
31    public String getWishExpectedTotal(int wish);
32
33
34
35    //Wish management.
36    public void makeNewWish(String airportFrom, String
airportTo, String dateDeparture, String dateReturn, int
numberOfPersons, int plusMinusDateDeparture, int
plusMinusDateReturn, boolean contactBeforeBooking, boolean
reserveWithoutPaying, boolean paid, int maxPrice,
ArrayList<Extra> extrasList);
37    public void deleteWish(int numberOfWish);
38 }
```

```
 1 import java.util.ArrayList;
 2 import java.io.*;
 3
 4 public class FileLoader implements InterfaceILoader
 5 {
 6    //Made by Kaspars
 7
 8    //Variables
 9    private ArrayList<User> users = new ArrayList<User>();
10    private ArrayList<Location> locations = new
ArrayList<Location>();
11    private User loggedUser = new User();
12
13    public void load()
14    {
15          //Users
16          try {
17             FileInputStream fileInput = new
FileInputStream("users.txt");
18             ObjectInputStream objectInput = new
ObjectInputStream(fileInput);
19
20             users =
(ArrayList<User>)objectInput.readObject();
21          }
22          catch (Exception e){
23             e.printStackTrace();
24             System.out.println("Couldn't load users");
25          }
26
```

```
27
28
29          //Locations & Add-ons
30          try {
31             FileInputStream fileInput = new
FileInputStream("locations.txt");
32             ObjectInputStream objectInput = new
ObjectInputStream(fileInput);
33
34             locations =
(ArrayList<Location>)objectInput.readObject();
35          }
36          catch (Exception e){
37             e.printStackTrace();
38
39             //Locations made by Adam
40             System.out.println("Couldn't load locations");
41             Location loc1 = new Location("CPH",
"Copenhagen");
42             Location loc2 = new Location("BKK",
"Bangkok");
43             Location loc3 = new Location("CIA", "Rome");
44             Location loc4 = new Location("WAW", "Warsaw");
45             Location loc5 = new Location("CDG", "Paris");
46             Location loc6 = new Location("HND", "Tokyo");
47
48             loc1.addExtra("Pick-up from airport to hotel",
350);
49             loc1.addExtra("Pick-up from hotel to airport",
350);
50             loc1.addExtra("Hotel in Copenhagen for time of
tour", 1100);
51             loc1.addExtra("One day tour of Copenhagen",
333);
52             loc1.addExtra("Rent a bicycle", 99);
53             loc1.addExtra("Sightseeing museums", 450);
54
55             loc2.addExtra("Elephant tour for a day in
Chiang Mai", 782);
56             loc2.addExtra("Pick-up from airport to hotel",
350);
57             loc2.addExtra("Hotel in Bangkok for time of
tour", 1100);
58             loc2.addExtra("Pick-up from hotel to airport",
350);
59
60             loc3.addExtra("Pick-up from airport to hotel",
350);
61             loc3.addExtra("Hotel in Roma for time of
tour", 999);
62             loc3.addExtra("Pick-up from hotel to airport",
350);
```

```
 63              loc3.addExtra("Sightseeing of the Vatican
City", 45);
 64              loc3.addExtra("One day tour of Roma", 420);
 65              loc3.addExtra("Gladiator fight", 876);
 66
 67              loc5.addExtra("Pick-up from airport to hotel",
350);
 68              loc5.addExtra("Hotel in Paris for time of
tour", 870);
 69              loc5.addExtra("Pick-up from hotel to airport",
350);
 70              loc5.addExtra("One day tour of Paris", 569);
 71              loc5.addExtra("Sightseeing museums ", 150);
 72              loc5.addExtra("See Mona Lise", 1860);
 73
 74              loc4.addExtra("Pick-up from airport to hotel",
230);
 75              loc4.addExtra("Hotel in Warsaw for time of
tour", 870);
 76              loc4.addExtra("One day tour of Warsaw", 555);
 77              loc4.addExtra("Sightseeing museums ", 420);
 78              loc4.addExtra("Enjoy national foods", 421);
 79              loc4.addExtra("Horse ride through city", 422);
 80
 81              loc6.addExtra("Pick-up from airport to hotel",
230);
 82              loc6.addExtra("Pick-up from hotel to airport",
230);
 83              loc6.addExtra("Hotel in Tokyo", 111);
 84              loc6.addExtra("Get a robot assistant", 560);
 85              loc6.addExtra("Eat live fish", 90);
 86              loc6.addExtra("Eat ramen", 20);
 87
 88              locations.add(loc1);
 89              locations.add(loc2);
 90              locations.add(loc3);
 91              locations.add(loc4);
 92              locations.add(loc5);
 93              locations.add(loc6);
 94          }
 95
 96      }
 97
 98
 99
100
101     public void save()
102     {
103        //Add logged in user to the list of users
104        for (User u : users)
105        {
106           if
```

```
(u.getUsername().equals(loggedUser.getUsername()))
                    {
                        //Check password
                        if
(u.getPassword().equals(loggedUser.getPassword()))
                        {
                           u = loggedUser;
                        }
                    }
         }


        //Users
        try {
            File file = new File("users.txt");
            file.createNewFile(); //create the file, if it
does not exist.

            FileOutputStream fileStream = new
FileOutputStream("users.txt", false);
            ObjectOutputStream outStream = new
ObjectOutputStream(fileStream);

            outStream.writeObject(users);

            fileStream.close();
            outStream.close();
        }
        catch (IOException e)
        {
            System.out.println("Can't save users");
        }



        //Locations & Add-ons
        try {
            File file = new File("locations.txt");
            file.createNewFile(); //create the file, if it
does not exist.

            FileOutputStream fileStream = new
FileOutputStream("locations.txt", false);
            ObjectOutputStream outStream = new
ObjectOutputStream(fileStream);

            outStream.writeObject(locations);

            fileStream.close();
            outStream.close();
        }
        catch (IOException e)
```

```
151         {
152             System.out.println("Can't save locations");
153         }
154 
155 
156 
157     }
158 
159     public int tryLogin(String username, String password){
160         //boolean Found = false;
161         for (User u : users)
162           {
163            System.out.println(u.getUsername());
164            System.out.println(username);
165            System.out.println(u.getPassword());
166            System.out.println(password);
167             //Find the username
168              if (u.getUsername().equals(username))
169              {
170                  //Check password
171                  if (u.getPassword().equals(password))
172                  {
173                     //Successfuly logged in
174                     System.out.println("Logged in");
175                     loggedUser = u;
176                     return 1;
177                     //Found = true;
178                  }
179              }
180           }
181 
182         System.out.println("couldn't log in");
183         return 0;
184     }
185 
186     public int checkUsername(String username)
187     {
188         boolean Found = false;
189          for (User u : users)
190           {
191            if (u.getUsername().equals(username))
192            {
193               Found = true;
194            }
195           }
196 
197           if (Found == true)
198            {
199               return 1;
200            } else{
201               return 0;
202            }
```

```
203     }
204
205     public int getUserType(String username){
206         System.out.println("It's a normal user");
207         return 0;
208     }
209     public void makeNewUser(String firstName, String
lastName, String password, String email, String phone, String
nationality, String address, String age, String username){
210         users.add(new User(firstName, lastName, password,
email, phone, nationality, address, age, username));
211         save();
212     }
213
214
215     public String[] getExtras(String location)
216     {
217         ArrayList<Extras> extrasArrayList = new
ArrayList<Extras>();
218         ArrayList<String> extrasNameList = new
ArrayList<String>();
219
220         for (Location l : locations)
221           {
222            if (l.getShortLoc().equals(location))
223            {
224               extrasArrayList = l.getExtras();
225                for (Extras e : extrasArrayList)
226                 {
227                     extrasNameList.add(e.getName());
228                 }
229            }
230           }
231         return extrasNameList.toArray(new
String[extrasNameList.size()]); //This came from google!
232     }
233
234     public String[] getLocations()
235     {
236         ArrayList<String> locationNamesArrayList = new
ArrayList<String>();
237
238
239         for (Location l : locations)
240           {
241               locationNamesArrayList.add(l.getShortLoc() + "
- " + l.getFullLoc());
242           }
243
244           String[] locationNamesArray =
locationNamesArrayList.toArray(new
String[locationNamesArrayList.size()]);
```

```
245         return locationNamesArray;
246     }
247 
248 
249 
250     public String[] getLocationPrices(String location)
251     {
252 
253         for (Location l : locations)
254         {
255             if (l.getShortLoc().equals(location))
256             {
257                 String[] locationPriceArray =
l.getPriceList();
258                 return locationPriceArray;
259             }
260         }
261         String[] locationPriceArray = new String[1];
262         return locationPriceArray;
263     }
264 
265 
266 
267     public void makeNewWish(String airportFrom, String
airportTo, String dateDeparture, String dateReturn, int
numberOfPersons, int plusMinusDateDeparture, int
plusMinusDateReturn, boolean contactBeforeBooking, boolean
reserveWithoutPaying, boolean paid, int maxPrice,
ArrayList<Extra> extrasList)
268     {
269         loggedUser.makeNewWish(airportFrom, airportTo,
dateDeparture, dateReturn, numberOfPersons,
plusMinusDateDeparture, plusMinusDateReturn,
contactBeforeBooking, reserveWithoutPaying, paid, maxPrice,
extrasList);
270     }
271 
272     public String[] getWishesName()
273     {
274         ArrayList<String> wishNamesArrayList = new
ArrayList<String>();
275 
276         ArrayList<Wish> wishesList = loggedUser.getWishes();
277 
278         for (Wish w : wishesList)
279           {
280               wishNamesArrayList.add(w.getAirportFrom() + "
- " + w.getAirportTo());
281         }
282 
283 
284         String[] wishNameArray =
```

```
wishNamesArrayList.toArray(new
String[wishNamesArrayList.size()]);
285         return wishNameArray;
286
287     }
288
289     public String[] getChosenExtras(int wish)
290     {
291         return loggedUser.getExtrasName(wish);
292     }
293
294     public String getWishFrom(int wish)
295     {
296         return loggedUser.getWishFrom();
297     }
298     public String getWishTo(int wish)
299     {
300         return loggedUser.getWishTo();
301     }
302     public String getWishDeparture(int wish)
303     {
304         return loggedUser.getWishDeparture();
305     }
306     public String getWishReturn(int wish)
307     {
308         return loggedUser.getWishReturn();
309     }
310     public String getWishMaxPrice(int wish)
311     {
312         return loggedUser.getWishMaxPrice();
313     }
314     public String getWishNumberOfPeople(int wish)
315     {
316         return loggedUser.getWishNumberOfPeople();
317     }
318     public String getWishDaysOfDeparture(int wish)
319     {
320         return loggedUser.getWishDaysOfDeparture();
321     }
322     public String getWishDaysOfReturn(int wish)
323     {
324         return loggedUser.getWishDaysOfReturn();
325     }
326     public String getWishPrice(int wish)
327     {
328         return loggedUser.getWishPrice();
329     }
330     public String getWishExpectedTotal(int wish)
331     {
332         return loggedUser.getMaxPrice();
333     }
334
```

```
335     //Wish management.
336     public void deleteWish(int numberOfWish)
337     {
338        loggedUser.deleteWish(numberOfWish);
339     }
340 }
```

```
  1 import java.util.ArrayList;
  2 import java.io.*;
  3
  4 public class User implements Serializable
  5 {
  6     //Made by Oliver
  7
  8
  9     private String firstname;
 10     private String lastname;
 11     private String password;
 12     private String email;
 13     private String phone;
 14     private String nationality;
 15     private String address;
 16     private String age;
 17     private String username;
 18     private boolean isAdmin;
 19
 20     private ArrayList<Wish> wishesList = new
ArrayList<Wish>();
 21     private Wish chosenWish;
 22
 23     public User()
 24     {
 25        firstname = "";
 26        lastname = "";
 27        password = "";
 28        email = "";
 29        phone = "";
 30        nationality = "";
 31        address = "";
 32        age = "";
 33        username = "";
 34        isAdmin = false;
 35     }
 36
 37     public User(String firstname, String lastname, String
password, String email, String phone, String nationality,
```

```
String address, String age, String username)
    {
       this.firstname = firstname;
       this.lastname = lastname;
       this.password = password;
       this.email = email;
       this.phone = phone;
       this.nationality = nationality;
       this.address = address;
       this.age = age;
       this.username = username;
       isAdmin = false;
    }

    public String getFirstName()
    {
       return firstname;
    }
    public String getLastName()
    {
       return lastname;
    }
    public String getPassword()
    {
       return password;
    }
    public String getEmail()
    {
       return email;
    }
    public String getPhone()
    {
       return phone;
    }
    public String getNationality()
    {
       return nationality;
    }
    public String getAddress()
    {
       return address;
    }
    public String getAge()
    {
       return age;
    }
    public String getUsername()
    {
       return username;
    }
```

```
89     public void setFirstName(String firstname)
90     {
91        this.firstname = firstname;
92     }
93     public void setLastName(String lastname)
94     {
95        this.lastname = lastname;
96     }
97     public void setPassword(String password)
98     {
99        this.password = password;
100    }
101    public void setEmail(String email)
102    {
103       this.email = email;
104    }
105    public void setPhone(String phone)
106    {
107       this.phone = phone;
108    }
109    public void setNationality(String nationality)
110    {
111       this.nationality = nationality;
112    }
113    public void setAddress(String address)
114    {
115       this.address = address;
116    }
117    public void setAge(String age)
118    {
119       this.age = age;
120    }
121
122    public void makeNewWish(String airportFrom, String
airportTo, String dateDeparture, String dateReturn, int
numberOfPersons, int plusMinusDateDeparture, int
plusMinusDateReturn, boolean contactBeforeBooking, boolean
reserveWithoutPaying, boolean paid, int maxPrice,
ArrayList<Extra> extrasList)
123    {
124       wishesList.add(new Wish(airportFrom, airportTo,
dateDeparture, dateReturn, numberOfPersons,
plusMinusDateDeparture, plusMinusDateReturn,
contactBeforeBooking, reserveWithoutPaying, paid, maxPrice,
extrasList));
125    }
126
127    public ArrayList<Wish> getWishes()
128    {
129       return wishesList;
130    }
131
```

```
132    public String[] getExtrasName(int wish)
133    {
134    int i = 0;
135       for (Wish w : wishesList)
136         {
137            if (i == wish)
138            {
139             //this is the current displayed wish
140             chosenWish = w;
141             return w.getAddOnNames();
142            }
143
144            i += 1;
145         }
146
147      System.out.println("Something wasn't found");
148      String[] locationPriceArray = new String[1];
149      return locationPriceArray;
150    }
151
152    public String getWishFrom()
153    {
154       return chosenWish.getFrom();
155    }
156    public String getWishTo()
157    {
158       return chosenWish.getTo();
159    }
160    public String getWishDeparture()
161    {
162       return chosenWish.getDeparture();
163    }
164    public String getWishReturn()
165    {
166       return chosenWish.getReturn();
167    }
168    public String getWishMaxPrice()
169    {
170       return chosenWish.getMaxPrice();
171    }
172    public String getWishNumberOfPeople()
173    {
174       return chosenWish.getNumberOfPeople();
175    }
176    public String getWishDaysOfDeparture()
177    {
178       return chosenWish.getDaysOfDeparture();
179    }
180    public String getWishDaysOfReturn()
181    {
182       return chosenWish.getDaysOfReturn();
183    }
```

```
184     public String getWishPrice()
185     {
186        return chosenWish.calculatePrice();
187     }
188     public String getMaxPrice()
189     {
190        return chosenWish.calculateTotalPrice();
191     }
192     public void deleteWish(int wish)
193     {
194        wishesList.remove(wish);
195     }
196 }
```

```
  1 import java.util.ArrayList;
  2 import java.io.*;
  3
  4 public class Wish implements Serializable
  5 {
  6     //Made by Oliver
  7
  8     private String airportFrom;
  9     private String airportTo;
 10     private String dateDeparture;
 11     private String dateReturn;
 12     private int numberOfPersons;
 13     private int maxPrice;
 14     private int realPrice;
 15     private int plusMinusDateDeparture;
 16     private int plusMinusDateReturn;
 17     private boolean contactBeforeBooking;
 18     private boolean reserveWithoutPaying;
 19     private boolean paid;
 20
 21     private ArrayList<Extra> extrasList = new
ArrayList<Extra>();
 22
 23     public Wish()
 24     {
 25
 26     }
 27
 28     public Wish(String airportFrom, String airportTo,
String dateDeparture, String dateReturn, int numberOfPersons,
int plusMinusDateDeparture, int plusMinusDateReturn, boolean
contactBeforeBooking, boolean reserveWithoutPaying, boolean
paid, int maxPrice, ArrayList<Extra> extrasList)
```

```
29    {
30       this.airportFrom = airportFrom;
31       this.airportTo = airportTo;
32       this.dateDeparture = dateDeparture;
33       this.dateReturn = dateReturn;
34       this.numberOfPersons = numberOfPersons;
35       this.plusMinusDateDeparture =
plusMinusDateDeparture;
36       this.plusMinusDateReturn = plusMinusDateReturn;
37       this.contactBeforeBooking = contactBeforeBooking;
38       this.reserveWithoutPaying = reserveWithoutPaying;
39       this.paid = paid;
40       this.maxPrice = maxPrice;
41       this.realPrice = 400 + (int)(Math.random()*2000);
42       this.extrasList = extrasList;
43
44    }
45
46    public String getAirportFrom()
47    {
48       return airportFrom;
49    }
50    public String getAirportTo()
51    {
52       return airportTo;
53    }
54
55    public int getRealPrice()
56    {
57       //Calculate
58       return 1;
59    }
60
61    public String[] getAddOnNames()
62    {
63       ArrayList<String> extrasArrayList = new
ArrayList<String>();
64
65       for (Extra e : extrasList)
66         {
67             extrasArrayList.add(e.getName());
68         }
69
70       String[] extrasArray = extrasArrayList.toArray(new
String[extrasArrayList.size()]);
71       return extrasArray;
72    }
73
74    public String getFrom()
75    {
76       return airportFrom;
77    }
```

```
78     public String getTo()
79     {
80        return airportTo;
81     }
82     public String getDeparture()
83     {
84        return dateDeparture;
85     }
86     public String getReturn()
87     {
88        return dateReturn;
89     }
90     public String getMaxPrice()
91     {
92        return Integer.toString(maxPrice);
93     }
94     public String getNumberOfPeople()
95     {
96        return Integer.toString(numberOfPersons);
97     }
98     public String getDaysOfDeparture()
99     {
100       return Integer.toString(plusMinusDateDeparture);
101    }
102    public String getDaysOfReturn()
103    {
104       return Integer.toString(plusMinusDateReturn);
105    }
106    public String calculatePrice()
107    {
108       int totalPrice = 0;
109       if (maxPrice > realPrice)
110          {
111             totalPrice = realPrice + (maxPrice -
realPrice)/2;
112
113          }
114       else
115          {
116             totalPrice = realPrice;
117          }
118
119       for (Extra e : extrasList)
120         {
121             if (e.getMaxPrice() > e.getRealPrice())
122             {
123                totalPrice += e.getRealPrice() +
(e.getMaxPrice() - e.getRealPrice())/2;
124             }
125             else
126             {
127                totalPrice += e.getRealPrice();
```

```
            }
        }


        return Integer.toString(totalPrice);
    }
    public String calculateTotalPrice()
    {
       int totalPrice = 0;
       totalPrice = maxPrice;

       for (Extra e : extrasList)
         {
                totalPrice += e.getMaxPrice();
         }


         return Integer.toString(totalPrice);
    }
}
```

```
import java.util.ArrayList;
import java.io.*;

public class Extra implements Serializable
{
    //Made by Oliver


    private String airport;
    private String name;
    private int maximumPrice;
    private int realPrice;



    //Constructor
    public Extra()
    {

    }

    public Extra(String airport, String name, int
maximumPrice, int realPrice)
    {
       this.airport = airport;
```

```
25        this.name = name;
26        this.maximumPrice = maximumPrice;
27        this.realPrice = realPrice;
28     }
29
30     public String getName()
31     {
32        return name;
33     }
34
35     public int getPriceDiff()
36     {
37        return (maximumPrice - realPrice);
38     }
39     public int getRealPrice()
40     {
41        return realPrice;
42     }
43     public int getMaxPrice()
44     {
45        return maximumPrice;
46     }
47
48  }
```

```
1  import java.awt.BorderLayout;
2  import java.awt.GridLayout;
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JLabel;
8  import javax.swing.JPanel;
9  import javax.swing.JTextField;
10 import javax.swing.JOptionPane;
11
12 public class Payment implements ActionListener
13 {
14    public static void main(String[] args)
15    {
16       new Payment();
17    }
18
19    JFrame frame;
20    JTextField txtField;
21    JTextField txt1Field;
22    JTextField txt2Field;
```

```
23      JButton btn;
24      JButton btn1;
25
26
27      public Payment()
28      {
29         txtField = new JTextField(40);
30         txt1Field = new JTextField(40);
31         txt2Field = new JTextField(40);
32         btn = new JButton("Cancel");
33         btn1 = new JButton("Pay");
34
35
36         btn.addActionListener(this);
37         btn1.addActionListener(this);
38
39         JPanel north = new JPanel(new GridLayout(3, 2));
//panel with gridlayout to go into north section of frames
borderlayout.
40         north.add(new JLabel("Type your credit card
number:"));
41         north.add(txtField);
42         north.add(new JLabel("Security number:"));
43         north.add(txt1Field);
44         north.add(new JLabel("Owner name:"));
45         north.add(txt2Field);
46
47         JPanel south = new JPanel(new GridLayout(1,2));
48         south.add(btn);
49         south.add(btn1);
50
51         frame = new JFrame("Payment");
52         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
53         frame.setLayout(new BorderLayout()); //Borderlayout
on frame.
54         frame.add(north, BorderLayout.NORTH); //previously
built panel in north section of frames borderlayout.
55         frame.add(south, BorderLayout.SOUTH); //button in
south section.
56         frame.pack(); //sets frames size depending on
components in it.
57         frame.setVisible(true); //make frame visible
58      }
59
60      public void actionPerformed(ActionEvent event)
61      {
62         if (event.getSource() == btn)
63         {
64            //code to return to screen 2
65         }
66
67         else if (event.getSource() == btn1)
```

```
68        {
69           JOptionPane.showMessageDialog(btn1,"Thank you for
payment, you will recive message with confirmation of
succesfull payment.");
70        }
71     }
72  }
73
```

```
1  import java.util.ArrayList;
 2  import java.io.*;
 3
 4  public class Location implements Serializable
 5  {
 6     //Made by Adam
 7     private String shortLocation;
 8     private String fullLocation;
 9     private ArrayList<Extras> extraList = new
ArrayList<Extras>();
10
11     //Constructor
12     public Location()
13     {
14
15     }
16     public Location(String shortLocation, String
fullLocation)
17     {
18        this.shortLocation = shortLocation;
19        this.fullLocation = fullLocation;
20     }
21
22
23
24
25
26     //Methods
27     public void addExtra(String name, int price)
28     {
29        extraList.add(new Extras(name, price));
30     }
31
32     public String getShortLoc()
33     {
34        return shortLocation;
35     }
36     public String getFullLoc()
37     {
38        return fullLocation;
```

```
39     }
40
41     public String[] getPriceList()
42     {
43        ArrayList<String> priceArrayList = new
ArrayList<String>();
44
45        for (Extras e : extraList)
46          {
47                //Collect prices from all add-ons
48                priceArrayList.add(e.getPrice());
49          }
50
51
52        String[] locationPriceArray =
priceArrayList.toArray(new String[priceArrayList.size()]);
53
54        return locationPriceArray;
55     }
56
57     public ArrayList<Extras> getExtras()
58     {
59        return extraList;
60     }
61  }
```

```
1  import java.util.ArrayList;
 2  import java.io.*;
 3  import java.util.*;
 4
 5  public class Extras implements Serializable
 6  {
 7     //Made by Adam
 8
 9     private String name;
10     private int price;
11
12     //Constructor
13     public Extras()
14     {
15
16     }
17
18     public Extras(String name, int price)
19     {
20        this.name = name;
21        this.price = price;
```

```
22     }
23
24     public String getName()
25     {
26        return name;
27     }
28
29     public String getPrice()
30     {
31        return Integer.toString(price);
32     }
33 }
```