Object Oriented Programming
Programming report
Assignment 4: Library

Corradini Matteo Berke Atac
S3051390 S3075168

May 30, 2016

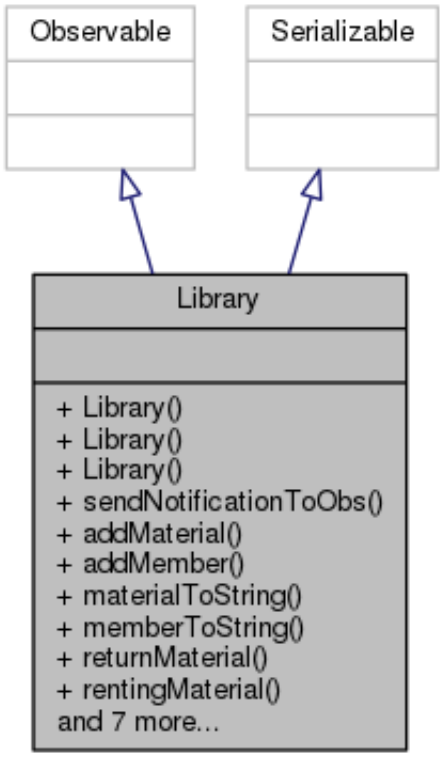
1 Problem description

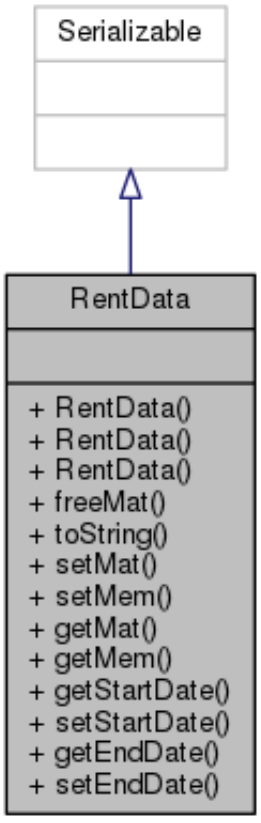
The problem was to provide a graphical program in order to manage the information of a general library.

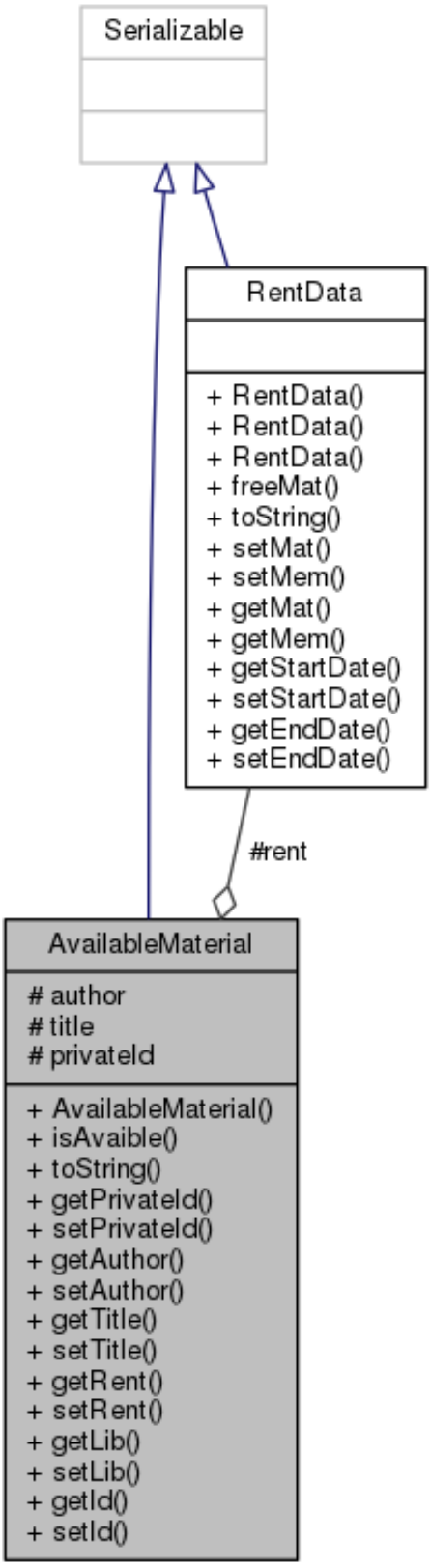
The information are the following: members, material (books or multimedia) and the actual situation of the rents. We have to implement an user interface, with the common and basic features (i.e., add information or make a rent), and furthermore some extra features (i.e., save the library situation and restore it from a file). The final project will be a management software for a library, in order to simplify the daily operations.

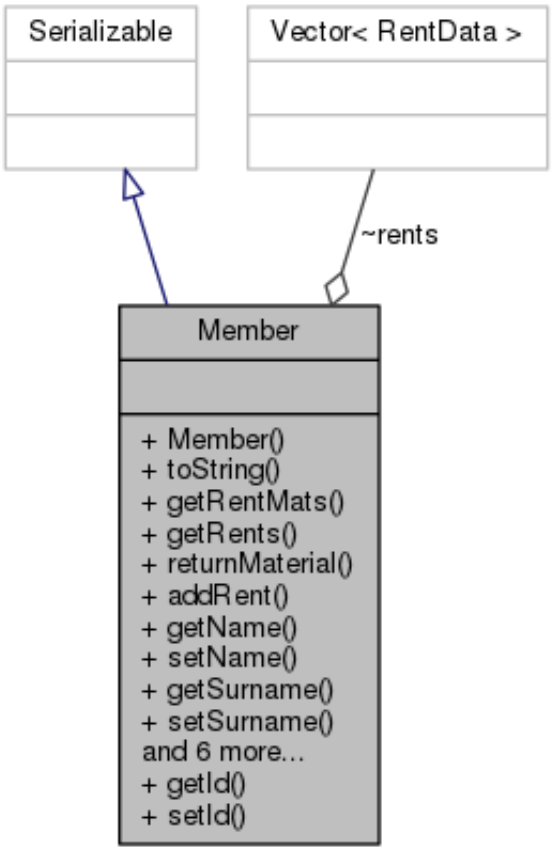
2 Problem analysis

This time we had to write the code from the beginning, creating a model with the relation between the crucial information: members and materials
Both of this information are managed by the library, which also manages an archive of the past rents. However, it has not directly the information about the actual situation of the renting, because it is required that it has to be a relation between the member and the rented book. We choose to keep this information only once, without an array of rents in library class, but looking in the members to find a rent. In this way, the information in our database appear only once, without problem about redundancy and inconsistency.
The key of the relation between the members and the materials is the following: a member could rent one or more material, thus it is necessary to have an array of contents in each member object. Instead, a material could be rented by one member at a time, this is way we need one field for the renter.
For the members we store the basic information (surname, name and address), for the material we store the title and the author, and for the rents we store the start and the end date. For every one, we also store a reference of the object (i.e. if a book is rented, we store in the book the information about the rent, which stores a reference to the renter). The members and the materials have to keep a reference to the library in which they are.
Below the UML of the relation class.


Observable
Serializable
Library
+ Library()
+ Library()
+ Library()
+ sendNotificationToObs()
+ addMaterial()
+ addMember()
+ materialToString()
+ memberToString()
+ returnMaterial()
+ rentingMaterial()
and 7 more...


Serializable
RentData
+ RentData()
+ RentData()
+ RentData()
+ freeMat()
+ toString()
+ setMat()
+ setMem()
+ getMat()
+ getMem()
+ getStartDate()
+ setStartDate()
+ getEndDate()
+ setEndDate()


Serializable
RentData
+ RentData()
+ RentData()
+ RentData()
+ freeMat()
+ toString()
+ setMat()
+ setMem()
+ getMat()
+ getMem()
+ getStartDate()
+ setStartDate()
+ getEndDate()
+ setEndDate()
#rent
AvailableMaterial
author
title
privateId
+ AvailableMaterial()
+ isAvaible()
+ toString()
+ getPrivateId()
+ setPrivateId()
+ getAuthor()
+ setAuthor()
+ getTitle()
+ setTitle()
+ getRent()
+ setRent()
+ getLib()
+ setLib()
+ getId()
+ setId()

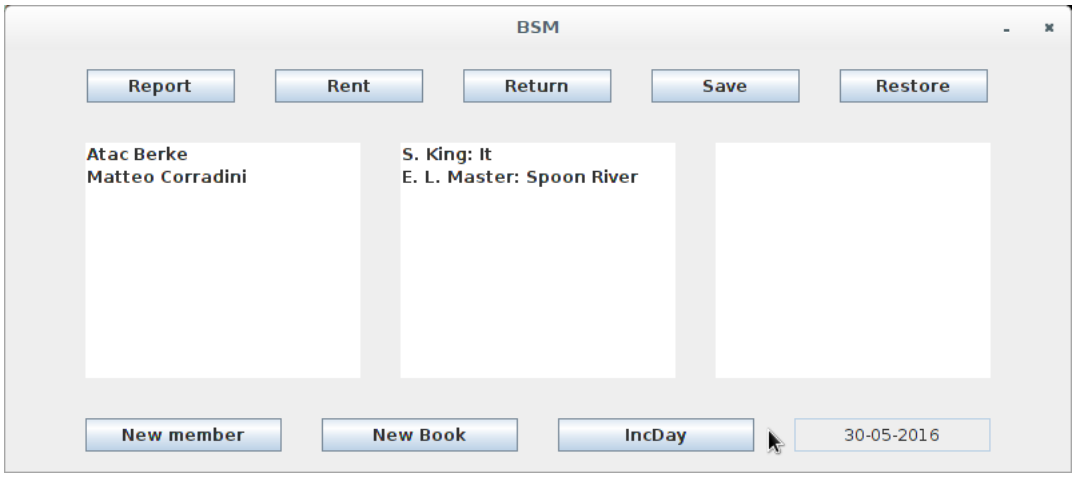


3 Program design

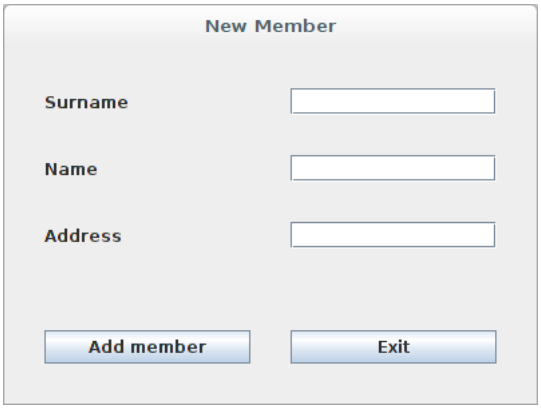
In the program, we developed the relations that we spoke about above.
`Library.java` represents the library, which keeps a `Vector` of materials, members and an archive. We implement the method to add a member and a material, sending a notification to the observer when one of this operation is done. We have also created a method to print this information from the `Vector`, and a method to get the free materials. `Library.java` manages also the renting and the return of the materials, settings these information in each involved object.
`Member.java` represents a member, with the setters and getters for the general information and the `toString` method to print these information. To create the ID, we use a static field `id` and static method to get it or set it. In this way, the value is unique in the class, and it is available whenever it is needed to create a new member.
The `AvaibleMaterials.java` is an abstract class in order to be extended in more specific subclasses as book or CDs. It has a static field `id`, which the function is the same of the one in `Member.java`. We implemented a method to know if the material is free in the library or is not.
In `RentData.java` we created 2 different constructors, to have a different type of input date (Date object or String object).
Each class that stores information implements `Serializable` in order to serialize the information on file, and restore it later.
In `LibraryWindow.java` we created the user interface with all the required features, using Swing library. We created methods for draw the interface, methods to implements the features, and another utility methods (i.e. `refreshLists()` in order to update the JLists).

4 Evaluation of the program

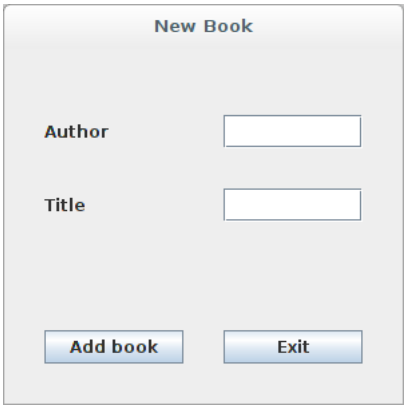
We obtain the following windows.



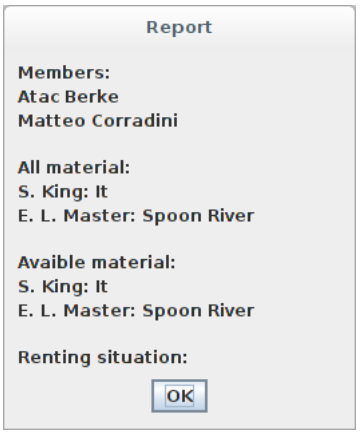
This is the main window, where is possible to have a report (see the picture below), make a rent after with the selected member and book or return a material (with the same logic). It is also possible increment the date, and add members or books with the dedicate window.



This is the window where it is possible to insert a member. If the operation is correct, it shows a dialog with the inserted information, otherwise it shows a dialog with the errors.



This is the window where it is possible to add a new book. The control of the error and the feedback of the operation are the same of above.



This is the window report, with the information about the members, the materials and the renting situation.

5 Extension of the program

We use Java `Serializable` interface in order to store the information about the Library in a file. We use the same interface to restore this information from the same file, and recreate the same environment.
We checked the input in the `JTextField`, to send appropriate error dialogs or confirm dialogs.

6 Conclusions

Our program solves the assignment requests implementing also the third iteration.
With this assignment we learned to draw a user interface using Swing with different type of layout (`BorderLayout` and `GridLayout`). We learned also how the events work in Swing, and the InnerClass to manage these events.
We learned how `Oberserver` works, in order to debug information when the content changes.
We learned how to serialize an object and its own fields.

7 Appendix: program text

Listing 1: `Library`

```
1  import java.io.Serializable;
2  import java.util.Date;
3  import java.util.Observable;
4  import java.util.Vector;
5
6  public class Library extends Observable implements Serializable{
7
8      private Vector<Member> members;
9      private Vector<AvailableMaterial> materials;
10     private Vector<RentData> rentHistory;
11
12     public Library(){
13         members = new Vector<Member>();
14         materials = new Vector<AvailableMaterial>();
```

```
15          rentHistory = new Vector<RentData>();
16      }
17
18      public Library(Vector<Member> members, Vector<AvailableMaterial> materials
            ,
19                      Vector<RentData> rentHistory){
20          this.setMembers(members);
21          this.setMaterials(materials);
22          this.setRentHistory(rentHistory);
23      }
24
25      public Library(Library _lib){
26          this(_lib.getMembers(), _lib.getMaterials(), _lib.getRentHistory());
27      }
28
29      public void sendNotificationToObs(String message){
30          setChanged();
31          this.notifyObservers(message);
32      }
33
34      public void addMaterial(AvailableMaterial mat){
35          materials.add(mat.getPrivateId(), mat);
36          this.sendNotificationToObs(this.materialToString());
37      }
38
39      public void addMember(Member mem){
40          members.add(mem.getMemberId(), mem);
41          this.sendNotificationToObs(this.memberToString());
42      }
43
44      public String materialToString(){
45          String buff = "";
46          int count = 1;
47          for (AvailableMaterial mat : this.getMaterials()){
48              buff+= count+")␣"+mat+"\n";
49              count++;
50          }
51          return buff;
52      }
53
54      public String memberToString(){
55          String buff = "";
56          int count = 1;
57          for (Member mem : this.getMembers()){
58              buff+= count+")␣"+mem+"\n";
59              count++;
60          }
61          return buff;
62      }
63
64      public void returnMaterial(RentData rent){
65          rent.setEndDate(new Date());
66          (rent.getMem()).returnMaterial(rent);
67          rent.freeMat();
68          rentHistory.add(rent);
69      }
70
```

```
71      public boolean rentingMaterial(Member mem, AvailableMaterial mat, String
            startDate){
72          if (!mat.isAvaible()) return false;
73          RentData rent = new RentData(mem, mat, startDate);
74          mem.addRent(rent);
75          mat.setRent(rent);
76          return true;
77      }
78
79      public Vector<AvailableMaterial> getFreeMaterial(){
80          Vector<AvailableMaterial> freeMat = new Vector<AvailableMaterial> ();
81          for (AvailableMaterial mat : this.getMaterials()){
82              if (mat.isAvaible()) freeMat.addElement(mat);
83          }
84          return freeMat;
85      }
86
87      public Vector<Member> getMembers() {
88          return members;
89      }
90      public Vector<AvailableMaterial> getMaterials() {
91          return materials;
92      }
93
94      public Vector<RentData> getRentHistory() {
95          return rentHistory;
96      }
97
98      public void setRentHistory(Vector<RentData> rentHistory) {
99          this.rentHistory = rentHistory;
100     }
101
102     public void setMembers(Vector<Member> members) {
103         this.members = members;
104     }
105
106     public void setMaterials(Vector<AvailableMaterial> materials) {
107         this.materials = materials;
108     }
109 }
```

Listing 2: `AvailableMaterial`

```
1  import java.io.Serializable;
2
3  public abstract class AvailableMaterial implements Serializable{
4
5      protected String author;
6      protected String title;
7      protected RentData rent;
8      protected int privateId;
9      private Library lib;
10
11     private static int id = 0;
12
13     public AvailableMaterial(String author, String title, Library lib){
14         this.setAuthor(author);
15         this.setTitle(title);
16         this.setLib(lib);
```

```
17          this.setPrivateId(AvailableMaterial.getId());
18          AvailableMaterial.setId(AvailableMaterial.getId()+1);
19      }
20
21      public static int getId(){
22          return id;
23      }
24
25      public static void setId(int new_id){
26          id = new_id;
27      }
28
29      public boolean isAvaible(){
30          return rent == null ? true : false;
31      }
32
33      @Override
34      public String toString(){
35          return this.getAuthor()+ ": "+this.getTitle();
36      }
37
38      /* Begin getters and setters*/
39      public int getPrivateId() {
40          return privateId;
41      }
42      public void setPrivateId(int privateId) {
43          this.privateId = privateId;
44      }
45      public String getAuthor() {
46          return author;
47      }
48      public void setAuthor(String author) {
49          this.author = author;
50      }
51      public String getTitle() {
52          return title;
53      }
54      public void setTitle(String title) {
55          this.title = title;
56      }
57      public RentData getRent() {
58          return rent;
59      }
60      public void setRent(RentData rent) {
61          this.rent = rent;
62      }
63      public Library getLib() {
64          return lib;
65      }
66      public void setLib(Library lib) {
67          this.lib = lib;
68      }
69      /*End getters and setters*/
70  }
```

Listing 3: `Member`

```
1   import java.io.Serializable;
2   import java.util.Vector;
```

```
3
4  public class Member implements Serializable{
5
6      private String name;
7      private String surname;
8      private int memberId;
9      private Library lib;
10     private String address;
11
12     private static int id = 0;
13
14     Vector<RentData> rents;
15
16     public Member(String name, String surname, String address, Library lib) {
17         this.setName(name);
18         this.setSurname(surname);
19         this.setLib(lib);
20         this.setAddress(address);
21         this.rents = new Vector<RentData>(lib.getMaterials().size());
22         this.setMemberId(Member.getId());
23         Member.setId(Member.getId()+1);
24     }
25
26     @Override
27     public String toString(){
28         return this.getSurname() + "␣" + this.getName();
29     }
30
31     public Vector<AvailableMaterial> getRentMats(){
32         Vector<AvailableMaterial> rentMat = new Vector<AvailableMaterial>();
33         for (RentData rent : this.getRents())
34             rentMat.addElement(rent.getMat());
35         return rentMat;
36     }
37
38     public Vector<RentData> getRents() {
39         return rents;
40     }
41
42     public void returnMaterial(RentData rent){
43         this.rents.remove(rent);
44     }
45
46     public void addRent(RentData rent){
47         this.rents.add(rent);
48     }
49
50     public String getName() {
51         return name;
52     }
53     public void setName(String name) {
54         this.name = name;
55     }
56     public String getSurname() {
57         return surname;
58     }
59     public void setSurname(String surname) {
60         this.surname = surname;
```

```
    }
    public int getMemberId() {
        return memberId;
    }
    public void setMemberId(int memberId) {
        this.memberId = memberId;
    }
    public static int getId() {
        return id;
    }
    public static void setId(int new_id){
        id = new_id;
    }
    public Library getLib() {
        return lib;
    }
    public void setLib(Library lib) {
        this.lib = lib;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }

}
```

Listing 4: `RentData`

```
import java.io.Serializable;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class RentData implements Serializable{

    private AvailableMaterial mat;
    private Member mem;

    private Date startDate;
    private Date endDate;

    public RentData(){
        this.setStartDate(null);
    }

    public RentData(Member mem, AvailableMaterial mat, Date startDate){
        this.setMat(mat);
        this.setMem(mem);
        this.setStartDate(startDate);
    }

    public RentData(Member mem, AvailableMaterial mat, String startDate){
        this.setMat(mat);
        this.setMem(mem);
        DateFormat format = new SimpleDateFormat("dd-MM-yyyy");
        Date date;
```

```
30          try {
31              date = format.parse(startDate);
32              this.setStartDate(date);
33          } catch (ParseException e) {
34              e.printStackTrace();
35          }
36      }
37
38      public void freeMat(){
39          this.mat.setRent(null);
40      }
41
42      @Override
43      public String toString(){
44          DateFormat format = new SimpleDateFormat("dd-MM-yyyy");
45          return this.getMem() + "␣rents␣" + this.getMat()
46                  + "␣from:␣" + format.format(this.getStartDate());
47      }
48      public void setMat(AvailableMaterial mat) {
49          this.mat = mat;
50      }
51      public void setMem(Member mem) {
52          this.mem = mem;
53      }
54      public AvailableMaterial getMat() {
55          return mat;
56      }
57      public Member getMem() {
58          return mem;
59      }
60      public Date getStartDate() {
61          return startDate;
62      }
63      public void setStartDate(Date startDate) {
64          this.startDate = startDate;
65      }
66      public Date getEndDate() {
67          return endDate;
68      }
69      public void setEndDate(Date endDate) {
70          this.endDate = endDate;
71      }
72  }
```

Listing 5: `Book`

```
1
2   public class Book extends AvailableMaterial{
3
4       public Book(String author, String title, Library lib){
5           super(author, title, lib);
6       }
7
8   }
```

Listing 6: `LibraryWindow`

```
1   import java.awt.BorderLayout;
2   import java.awt.Dimension;
```

```
3  import java.awt.GridLayout;
4  import java.awt.Toolkit;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import java.awt.event.WindowAdapter;
8  import java.awt.event.WindowEvent;
9  import java.io.FileInputStream;
10 import java.io.FileOutputStream;
11 import java.io.IOException;
12 import java.io.ObjectInputStream;
13 import java.io.ObjectOutputStream;
14 import java.io.Serializable;
15 import java.text.DateFormat;
16 import java.text.ParseException;
17 import java.text.SimpleDateFormat;
18 import java.util.Calendar;
19 import java.util.Date;
20 import java.util.Vector;
21
22 import javax.swing.JButton;
23 import javax.swing.JFrame;
24 import javax.swing.JList;
25 import javax.swing.JOptionPane;
26 import javax.swing.JPanel;
27 import javax.swing.JTextField;
28 import javax.swing.border.EmptyBorder;
29 import javax.swing.event.ListSelectionEvent;
30 import javax.swing.event.ListSelectionListener;
31
32 public class LibraryWindow extends JFrame{
33
34     private final int NUMBER_BUTTONS = 8;
35
36     private final int WINDOW_WIDTH = 800;
37     private final int WINDOW_HEIGHT = 350;
38
39     private final int WINDOW_PADDING = 30;
40     private final int SPACE_BETWEEN_BUTTONS = 30;
41
42
43     private String ButtonName[] = {"Report", "Rent", "Return", "Save", "
           Restore",
44                                    "New␣member", "New␣Book", "IncDay"};
45
46     private JButton buttons[];
47
48     private JList<Member> memberJlist;
49     private JList<AvailableMaterial> freeMatJlist;
50     private JList<AvailableMaterial> rentedMatJlist;
51
52     JPanel panel;
53
54     private JTextField date;
55     private JPanel northPanel;
56     private JPanel southPanel;
57     private JPanel centerPanel;
58
59     private Library lib;
```

```
60
61      public LibraryWindow(Library _lib){
62          this.lib = new Library(_lib);
63          this.setWindowProperites();
64
65          this.drawPanels();
66          this.drawButtons();
67          this.drawTextField();
68
69          /*Add scrollbar*/
70          this.drawLists(lib);
71
72          /*Show rented book by clicked member*/
73          this.memberJlist.addListSelectionListener(new ListSelectionListener(){
74              @Override
75              public void valueChanged(ListSelectionEvent event){
76                  Member selectedMember = LibraryWindow.this.memberJlist.
                        getSelectedValue();
77                  if (selectedMember != null)
78                      LibraryWindow.this.rentedMatJlist.setListData(
                            selectedMember.getRentMats());
79              }
80          });
81
82          /*Report button*/
83          this.buttons[0].addActionListener(new ActionListener() {
84              @Override
85              public void actionPerformed(ActionEvent e) {
86                  LibraryWindow.this.printReport();
87              }
88          });
89          /*Rent button*/
90          this.buttons[1].addActionListener(new ActionListener() {
91              @Override
92              public void actionPerformed(ActionEvent e) {
93                  LibraryWindow.this.makeRent();
94              }
95          });
96          /*Return button*/
97          this.buttons[2].addActionListener(new ActionListener() {
98              @Override
99              public void actionPerformed(ActionEvent e) {
100                 LibraryWindow.this.makeReturn();
101             }
102         });
103         /*Save button*/
104         this.buttons[3].addActionListener(new ActionListener() {
105             @Override
106             public void actionPerformed(ActionEvent e) {
107                 LibraryWindow.this.saveLibrary();
108             }
109         });
110         /*Restore button*/
111         this.buttons[4].addActionListener(new ActionListener() {
112             @Override
113             public void actionPerformed(ActionEvent e) {
114                 LibraryWindow.this.restoreLibrary();
115             }
```

```
116         });
117         /*New Member button*/
118         this.buttons[5].addActionListener(new ActionListener() {
119             @Override
120             public void actionPerformed(ActionEvent e) {
121                 LibraryWindow.this.addNewMember();
122             }
123         });
124         /*New Book button*/
125         this.buttons[6].addActionListener(new ActionListener() {
126             @Override
127             public void actionPerformed(ActionEvent e) {
128                 LibraryWindow.this.addNewBook();
129             }
130         });
131 
132         /*IncDay button*/
133         this.buttons[7].addActionListener(new ActionListener() {
134             @Override
135             public void actionPerformed(ActionEvent e) {
136                 LibraryWindow.this.incDate();
137             }
138         });
139 
140         this.drawPanel();
141     }
142 
143     private void restoreLibrary(){
144           try
145           {
146              FileInputStream fileIn = new FileInputStream("lib.obj");
147              ObjectInputStream in = new ObjectInputStream(fileIn);
148              this.lib = (Library) in.readObject();
149              in.close();
150              fileIn.close();
151              this.refreshLists();
152              JOptionPane.showMessageDialog(null, "Library␣restored␣from␣lib.
                     obj");
153           }catch(IOException i)
154           {
155               System.out.println(i);
156               JOptionPane.showMessageDialog(null, "Something␣went␣wrong...");
157           } catch (ClassNotFoundException e) {
158             // TODO Auto-generated catch block
159               System.out.println(e);
160               JOptionPane.showMessageDialog(null, "Something␣went␣wrong...");
161         }
162     }
163 
164     private void saveLibrary(){
165         try{
166              FileOutputStream fileOut =
167              new FileOutputStream("lib.obj");
168              ObjectOutputStream out = new ObjectOutputStream(fileOut);
169              out.writeObject(this.lib);
170              out.close();
171              fileOut.close();
172              JOptionPane.showMessageDialog(null, "Library␣saved␣in␣lib.obj");
```

```
173           }catch(IOException i)
174           {
175               System.out.println(i);
176              JOptionPane.showMessageDialog(null, "Something␣went␣wrong...");
177           }
178       }
179
180       private void addNewBook(){
181           NewMaterialDialog matDialog = new NewMaterialDialog(this.lib);
182           matDialog.setVisible(true);
183           this.disableButtons();
184           matDialog.addWindowListener(new WindowAdapter() {
185               @Override
186               public void windowClosed(WindowEvent e) {
187                   LibraryWindow.this.refreshLists();
188                   LibraryWindow.this.enableButtons();
189               }
190           });
191       }
192
193       private void addNewMember(){
194           NewMemberDialog memberDialog = new NewMemberDialog(this.lib);
195           memberDialog.setVisible(true);
196           this.disableButtons();
197           memberDialog.addWindowListener(new WindowAdapter() {
198               @Override
199               public void windowClosed(WindowEvent e) {
200                   LibraryWindow.this.refreshLists();
201                   LibraryWindow.this.enableButtons();
202               }
203           });
204       }
205
206       private void printReport(){
207           String report = "Members:\n";
208           for (Member mem : this.lib.getMembers()){
209               report += mem+"\n";
210           }
211           report += "\nAll␣material:\n";
212           for (AvailableMaterial mat : this.lib.getMaterials()){
213               report += mat+"\n";
214           }
215           report += "\nAvaible␣material:\n";
216           for (AvailableMaterial mat : this.lib.getFreeMaterial()){
217               report += mat+"\n";
218           }
219           report += "\nRenting␣situation:\n";
220           for (Member mem : this.lib.getMembers()){
221               for (RentData rent : mem.getRents())
222                   report += rent+"\n";
223           }
224           this.disableButtons();
225           JOptionPane.showMessageDialog(this, report, "Report", JOptionPane.
                  PLAIN_MESSAGE);
226           this.enableButtons();
227       }
228
229       private void makeRent(){
```

```
230         if (this.memberJlist.getSelectedValue() != null && this.freeMatJlist.
                getSelectedValue() != null){
231             if (this.lib.rentingMaterial(this.memberJlist.getSelectedValue(),
232                                          this.freeMatJlist.getSelectedValue(),
233                                          this.date.getText())){
234                 this.refreshLists();
235             }
236         }
237     }
238
239     private void makeReturn(){
240         Member mem = this.memberJlist.getSelectedValue();
241         AvailableMaterial mat = this.rentedMatJlist.getSelectedValue();
242         if (mem != null && mat  != null){
243             RentData rentToRemove = mat.getRent();
244             if (rentToRemove != null){
245                 this.lib.returnMaterial(rentToRemove);
246                 this.refreshLists();
247             }
248         }
249     }
250
251     private void incDate(){
252         DateFormat format = new SimpleDateFormat("dd-MM-yyyy");
253         Date date;
254         try {
255             date = format.parse(this.date.getText());
256             Calendar c = Calendar.getInstance();
257             c.setTime(date);
258             c.add(Calendar.DATE, 1);
259             this.date.setText(format.format(c.getTime()));
260         } catch (ParseException e) {
261             e.printStackTrace();
262         }
263     }
264
265     private void setWindowProperites(){
266         this.setTitle("BSM");
267         this.setResizable(false);
268         this.setSize(new Dimension(WINDOW_WIDTH, WINDOW_HEIGHT));
269         this.getContentPane().setLayout(new BorderLayout());
270         Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
271         this.setLocation(dim.width/2-this.getSize().width/2, dim.height/2-this
                .getSize().height/2);
272     }
273
274     private void drawPanels(){
275         this.panel = new JPanel (new BorderLayout()) ;
276         panel.setBorder(new EmptyBorder(WINDOW_PADDING/2, 2*WINDOW_PADDING,
                WINDOW_PADDING/2, 2*WINDOW_PADDING));
277         this.northPanel = new JPanel (new GridLayout(0, 5,
                SPACE_BETWEEN_BUTTONS, 0));
278         this.southPanel = new JPanel (new GridLayout(0, 4,
                SPACE_BETWEEN_BUTTONS, 0));
279         this.centerPanel = new JPanel (new GridLayout(0, 3,
                SPACE_BETWEEN_BUTTONS, 0));
280         centerPanel.setBorder(new EmptyBorder(WINDOW_PADDING, 0,
                WINDOW_PADDING, 0));
```

```
281     }
282
283     private void drawButtons(){
284         this.buttons = new JButton[NUMBER_BUTTONS];
285         for (int i = 0; i < NUMBER_BUTTONS; i++){
286             this.buttons[i] = new JButton(ButtonName[i]);
287             if (i < 5)  northPanel.add(buttons[i]);
288             else        this.southPanel.add(buttons[i]);
289         }
290     }
291
292     private void drawTextField(){
293         Date date = new Date();
294         System.out.println(date.toString());
295         SimpleDateFormat dt = new SimpleDateFormat("dd-MM-yyyy");
296         this.date = new JTextField(dt.format(date));
297         this.date.setHorizontalAlignment(JTextField.CENTER);
298         this.date.setEditable(false);
299         this.southPanel.add(this.date);
300     }
301
302     private void drawPanel(){
303         panel.add(centerPanel, BorderLayout.CENTER);
304         panel.add(northPanel, BorderLayout.NORTH);
305         panel.add(southPanel, BorderLayout.SOUTH);
306         this.add(panel);
307     }
308
309     private void drawLists(Library lib){
310         this.memberJlist = new JList<Member>(lib.getMembers());
311         this.centerPanel.add(memberJlist);
312
313         this.freeMatJlist = new JList<AvailableMaterial>(this.lib.
                getFreeMaterial());
314         this.centerPanel.add(freeMatJlist);
315
316         this.rentedMatJlist = new JList<AvailableMaterial>();
317         this.centerPanel.add(rentedMatJlist);
318     }
319
320     public void refreshLists(){
321         this.freeMatJlist.setListData(this.lib.getFreeMaterial());
322         if (this.memberJlist.getSelectedValue() != null)
323             this.rentedMatJlist.setListData(this.memberJlist.getSelectedValue
                    ().getRentMats());
324         else{
325             this.rentedMatJlist.setListData(new Vector<AvailableMaterial>());
326         }
327         this.memberJlist.setListData(this.lib.getMembers());
328     }
329
330     private void disableButtons(){
331         for (JButton button : this.buttons)
332             button.setEnabled(false);
333     }
334
335     private void enableButtons(){
336         for (JButton button : this.buttons)
```

```
            button.setEnabled(true);
    }

}
```

Listing 7: NewMaterialDialog

```
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

public class NewMaterialDialog extends JDialog {
    final int WIDTH_WINDOW = 300;
    final int HEIGHT_WINDOW = 300;

    private JPanel centerPanel;
    private JPanel southPanel;

    private JTextField txtAuthor;
    private JTextField txtTitle;

    private JButton btnSubmit;
    private JButton btnExit;

    private Library lib;

    public NewMaterialDialog(Library lib){
        this.lib = lib;
        this.setWindow();
        this.setPanel();
        this.drawFields();

        this.btnSubmit = new JButton("Add␣book");
        this.btnSubmit.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                NewMaterialDialog.this.addMaterial();
                NewMaterialDialog.this.clearTextField();
            }
        });

        this.btnExit = new JButton("Exit");
        this.btnExit.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                NewMaterialDialog.this.dispose();
            }
        });

```

```
53          this.southPanel.add(btnSubmit);
54          this.southPanel.add(btnExit);
55          this.add(centerPanel, BorderLayout.CENTER);
56          this.add(southPanel, BorderLayout.SOUTH);
57      }
58
59      private void addMaterial(){
60          String author = this.txtAuthor.getText();
61          String title = this.txtTitle.getText();
62          if (author != "" && title != ""){
63              AvailableMaterial newMat = new Book(author, title, this.lib);
64              this.lib.addMaterial(newMat);
65              JOptionPane.showMessageDialog(null, "Book␣added,␣author:␣" +
                    author + ";␣title:␣" + title);
66          }else{
67              String error = "The␣fields␣" + author == "" ? "author" : "" +
68                             title == "" ? "title" : "" +
69                             "are␣compulsory";
70              JOptionPane.showMessageDialog(null, error);
71          }
72
73      }
74
75      private void clearTextField(){
76          this.txtAuthor.setText("");
77          this.txtTitle.setText("");
78      }
79
80      private void setPanel(){
81          this.centerPanel = new JPanel();
82          this.centerPanel.setLayout(new GridLayout(2,2,30,30));
83          this.centerPanel.setBorder(new EmptyBorder(50, 30, 50, 30));
84          this.southPanel = new JPanel();
85          this.southPanel.setLayout(new GridLayout(0,2,30,30));
86          this.southPanel.setBorder(new EmptyBorder(30, 30, 30, 30));
87      }
88
89      private void setWindow(){
90          this.setTitle("New␣Book");
91          this.setResizable(false);
92          this.setSize(WIDTH_WINDOW, HEIGHT_WINDOW);
93          this.getContentPane().setLayout(new BorderLayout());
94          Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
95          this.setLocation(dim.width/2-this.getSize().width/2, dim.height/2-this
                .getSize().height/2);
96      }
97
98      private void drawFields(){
99          JLabel lblAuthor = new JLabel("Author");
100         this.centerPanel.add(lblAuthor);
101         this.txtAuthor = new JTextField();
102         this.centerPanel.add(txtAuthor);
103         JLabel lblTitle = new JLabel("Title");
104         this.centerPanel.add(lblTitle);
105         this.txtTitle = new JTextField();
106         this.centerPanel.add(txtTitle);
107     }
108 }
```

Listing 8: NewMemberDialog

```
1  import java.awt.BorderLayout;
2  import java.awt.Dimension;
3  import java.awt.GridLayout;
4  import java.awt.Toolkit;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import javax.swing.JButton;
8  import javax.swing.JDialog;
9  import javax.swing.JLabel;
10 import javax.swing.JOptionPane;
11 import javax.swing.JPanel;
12 import javax.swing.JTextField;
13 import javax.swing.border.EmptyBorder;
14
15
16 public class NewMemberDialog extends JDialog{
17
18     final int WIDTH_WINDOW = 400;
19     final int HEIGHT_WINDOW = 300;
20
21     private JPanel centerPanel;
22     private JPanel southPanel;
23
24     private JTextField txtSurname;
25     private JTextField txtName;
26     private JTextField txtAddress;
27
28     private JButton btnSubmit;
29     private JButton btnExit;
30
31     private Library lib;
32
33     public NewMemberDialog(Library lib){
34         this.lib = lib;
35         this.setWindow();
36         this.setPanel();
37         this.drawFields();
38
39         this.btnSubmit = new JButton("Add␣member");
40         this.btnSubmit.addActionListener(new ActionListener() {
41             @Override
42             public void actionPerformed(ActionEvent e) {
43                 NewMemberDialog.this.addMember();
44                 NewMemberDialog.this.clearTextField();
45             }
46         });
47
48         this.btnExit = new JButton("Exit");
49         this.btnExit.addActionListener(new ActionListener() {
50             @Override
51             public void actionPerformed(ActionEvent e) {
52                 NewMemberDialog.this.dispose();
53             }
54         });
55
56         this.southPanel.add(btnSubmit);
57         this.southPanel.add(btnExit);
```

```
58          this.add(centerPanel, BorderLayout.CENTER);
59          this.add(southPanel, BorderLayout.SOUTH);
60      }
61
62      private void addMember(){
63          String name = this.txtName.getText();
64          String surname = this.txtSurname.getText();
65          String address = this.txtAddress.getText();
66          if (surname != "" && name != "" && address != ""){
67              Member newMember = new Member(surname, name, address, this.lib);
68              this.lib.addMember(newMember);
69              JOptionPane.showMessageDialog(null, "Member␣added,␣surname:␣" +
                    surname + ";␣name:␣"+ name + ";␣address:␣"+ address);
70          }else{
71              String error = "The␣fields␣" + surname == "" ? "surname" : "" +
72                             name == "" ? "name" : "" +
73                             address == "" ? "address" : "" +
74                             "are␣compulsory";
75              JOptionPane.showMessageDialog(null, error);
76          }
77      }
78
79      private void clearTextField(){
80          this.txtSurname.setText("");
81          this.txtName.setText("");
82          this.txtAddress.setText("");
83      }
84
85      private void setPanel(){
86          this.centerPanel = new JPanel();
87          this.centerPanel.setLayout(new GridLayout(3, 2, 30, 30));
88          this.centerPanel.setBorder(new EmptyBorder(30, 30, 30, 30));
89          this.southPanel = new JPanel();
90          this.southPanel.setLayout(new GridLayout(0,2,30,30));
91          this.southPanel.setBorder(new EmptyBorder(30, 30, 30, 30));
92      }
93
94      private void setWindow(){
95          this.setTitle("New␣Member");
96          this.setResizable(false);
97          this.setSize(WIDTH_WINDOW, HEIGHT_WINDOW);
98          this.getContentPane().setLayout(new BorderLayout());
99          Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
100         this.setLocation(dim.width/2-this.getSize().width/2, dim.height/2-this
                .getSize().height/2);
101     }
102
103     private void drawFields(){
104         JLabel lblSurname = new JLabel("Surname");
105         this.centerPanel.add(lblSurname);
106         txtSurname = new JTextField();
107         this.centerPanel.add(txtSurname);
108         JLabel lblName = new JLabel("Name");
109         this.centerPanel.add(lblName);
110         this.txtName = new JTextField();
111         this.centerPanel.add(txtName);
112         JLabel lblAddress = new JLabel("Address");
113         this.centerPanel.add(lblAddress);
```

```
114         this.txtAddress = new JTextField();
115         this.centerPanel.add(txtAddress);
116     }
117 }
```

Listing 9: `LibraryTest`

```
1  import java.util.Observable;
2  import java.util.Observer;
3
4  import javax.swing.JFrame;
5
6  public class LibraryTest implements Observer{
7
8       public static void main (String[] argv){
9
10         Library lib = new Library();
11         lib.addObserver(new LibraryTest());
12
13         lib.addMaterial(new Book("S. King", "It", lib));
14         lib.addMaterial(new Book("E. L. Master", "Spoon River", lib));
15         lib.addMember(new Member("Berke", "Atac", "Van Houtenlaan 27", lib));
16         lib.addMember(new Member("Corradini", "Matteo", "Van Houtenlaan 27",
               lib));
17
18         LibraryWindow window = new LibraryWindow(lib);
19         window.setVisible(true);
20         window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
21
22     }
23
24     @Override
25     public void update(Observable lib, Object arg1) {
26         // TODO Auto-generated method stub
27         System.out.println(arg1);
28     }
29
30 }
```