



Project Title:

Course code:2231

Course Name: Computer Programming

Section:W03

Student Group List:

Student Name: Shang Songlin Student Number: 1308184

Student Name: Ye Cong Student Number: 1306248

Student Name: Zhu kailiang..... Student Number: 1306289

...

Professor: Dr Ken Ehimwenma, Ph.D.

Date:

5 / 26 / 2024

Abstract

Nowadays, the demand for efficiency and versatility for management is increasing and manual processing of large amounts of data is no longer adequate. However, the operation and recording efficiency of customer information in traditional hotels is limited and the fault tolerance rate is low. Under this circumstance, we designed "Sign-on system" in Java programming to improve the efficiency of customer management. In order to realize the demands of it, we use Object-Oriented Programming methods including inheritance, encapsulation, and polymorphism to handle the operation and management of administrators and users. In this way, multiple functions of the object can be selected and realized. We also use methods from IOException, Interface, File class and ArrayList to manage, process and protect data and form UI as well to optimize the user's usage experience. This system can not only realize the basic function of user operation own account such as logging in the system and modifying their password, but also add the administrator part to control the user's operation and data, which can effectively prevent users' loss from misoperation or intentional damage to the system.

I. Introduction

Through the syllabus of CPS2231, we design the "Sign-on System Project" with the tool of the Java programming language.

Aiming at strengthening the ability of problem solving and designing the algorithms with using a high-level knowledge, we focus on the case of optimizing traditional hotel's sign-on system and invoke a simplest and intuitive User Interface to visualize our outcomes. It is found that traditional hotel's occupancy management system has the problems of unfriendly user interface, data integration issues, lack of automation and Limited Scalability. Accordingly, our project tries to simplify the interface and give direct instruction for each step to avoid the gulf of execution and evaluation using GUI, combine the guest sign-on system and administrator sign-on system together into user object to improve the consistence of data and raise the reusing rate of the code, procreating a text file in Eclipse to store the data of users' and administrators' identity information and furtherly expanding the capabilities for administrators. It is worth mentioning that for visualizing the outcome, we extends the JFrame to display a User Interface.

To carry out the principle of object-oriented programming thinking, we utilize the idea of class inheritance like defining the User superclass as well as create subclasses Admin and Guest, introducing GUI and BufferedWriter to realize the File Input and Output respectively, IOException and proper use of class abstraction and encapsulation.

The primary steps for running this system are:

1. "Run" the Main class in Eclipse to launch the Sign-on interface
2. Input the User Name and Password, there are three options: (1) Forget the password: come to the 3rd step (2) Register: Come to the 4rd step (3) Log in: Come to the 5rd step
3. If you forget the password, you can reset it by entering the User Name, User ID number, New Password, then back to the home page, and follow the 5th step to log in
4. Register: Enter the User Name, User ID Number, and click the Finished button
5. Setting the PasswordThe log-in operation will be divided into two cases to consider:
 - (1) Guest Capability: undeveloped further
 - (2) Administrator Capability: booking room management, user management and room management

II. Related Works

1. Password Determination.

```
18     //Signature: String--->boolean
19     //Purpose: input a password then
20     //          determine it whether a password is valid or not.
21     //Example:
22     //          1234w--->false
23     //          12wwwwwwww+wwwwww--->false
24     //          1234wwwwwwwwwwww--->true
25     //Header:
26     public static boolean isValidPassword(String password) {
27         if (password.length() < 16) {
28             return false;
29         }
30
31         int digitCount = 0;
32         for (int i = 0; i < password.length(); i++) {
33             char ch = password.charAt(i);
34
35             // Check if the character is not a letter, a digit, or a '+'
36             if (!Character.isLetterOrDigit(ch) && ch != '+') {
37                 return false;
38             }
39
40             // Count the digits
41             if (Character.isDigit(ch)) {
42                 digitCount++;
43             }
44         }
45
46         // Check the count of digits
47         return digitCount >= 3;
48     }
```

2. Use try and catch to deal with IOException.

```
1 package chapter12;
2
3 public class CircleWithCustomExceptionTest {
4
5     public static void main(String[] args) {
6         try {
7             new CircleWithCustomException(5);
8             new CircleWithCustomException(-5);
9             new CircleWithCustomException(0);
10            new CircleWithCustomException(7);
11        }
12        catch(InvalidRadiusException ex) {
13            System.out.println(ex);
14        }
15
16        System.out.println("Number of objects created: " + CircleWithCustomException.getNumberOfObjects());
17    }
18
19 }
```

3. Use Inheritance to make relationship with different class and to implement the method excuted.

```
3 public class RectangleFromSimpleGeometricObject extends SimpleGeometricObject {
4     private double width;
5     private double height;
6
7     public RectangleFromSimpleGeometricObject() {
8
9
10    public RectangleFromSimpleGeometricObject(double width, double height) {
11        this.width = width;
12        this.height = height;
13    }
14
15    public RectangleFromSimpleGeometricObject(double width, double height, String color, boolean filled) {
16        super(color,filled); //equal to line 19 20 step1
17        this.width = width; //step2
18        this.height = height; //step2
19        setColor(color);
20        setFilled(filled);
21    }
22
23
24
25
26
27
```

4. Create a file and test a file.

```
1 package ChapterV12;
2
3 public class writeData {
4
5     public static void main(String[] args) throws java.io.IOException {
6         java.io.File file = new java.io.File("scores.txt");
7         if (file.exists()) {
8             System.out.println("File already exists");
9             System.exit(1);
10        }
11
12        // Create a file
13        java.io.PrintWriter output = new java.io.PrintWriter(file);
14
15        // Write formatted output to the file
16        output.print("John T. Perez ");
17        output.println(90);
18        output.print("Jamal K. Johnson ");
19        output.println(85);
20
21        // Close the file
22        output.close();
23
24    }
25
26 }
27
```

5. List<User> and List<Room>related to ArrayList<String>

```
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter a URL: ");
    String url = input.nextLine();
    crawler(url); // Traverse the Web from the a starting url
}

public static void crawler(String startingURL) {
    ArrayList<String> listOfPendingURLs = new ArrayList<>();
    ArrayList<String> listOfTraversedURLs = new ArrayList<>();

    listOfPendingURLs.add(startingURL);
    while (!listOfPendingURLs.isEmpty() &&
           listOfTraversedURLs.size() <= 100) {
        String urlString = listOfPendingURLs.remove(0);
        listOfTraversedURLs.add(urlString);
        System.out.println("Crawl " + urlString);

        for (String s: getSubURLs(urlString)) {
            if (!listOfTraversedURLs.contains(s) &&
                !listOfPendingURLs.contains(s))
                listOfPendingURLs.add(s);
        }
    }
}

public static ArrayList<String> getSubURLs(String urlString) {
    ArrayList<String> list = new ArrayList<>();

    try {
        java.net.URL url = new java.net.URL(urlString);
        Scanner input = new Scanner(url.openStream());
        int current = 0;
        while (input.hasNext()) {
            String line = input.nextLine();
            current = line.indexOf("http:", current);
            while (current > 0) {
                int endIndex = line.indexOf("\n", current);
                if (endIndex > 0) { // Ensure that a correct URL is found
                    list.add(line.substring(current, endIndex));
                    current = line.indexOf("http:", endIndex);
                }
                else
                    current = -1;
            }
        }
    } catch (Exception ex) {
```

III. Methodology

1.To implement the Hotel Management Login System. The first problem is that we need to store user data in a database.

In our program, we use file to store data. In the course, we learned PrintWriter to store data. But in this system, we extraly learned the BufferedWriter to store data rather than PrintWriter. The reason for using BufferedWriter is that its function is to provide buffering so that it could have better performance to store data during a mass store in txt operation. It is suitable for this system because the hotel management system need to store a large amount of data. If we use bufferedWriter to store data in the file, it would more effect and not easily get system wrong.

```
) private void saveUserToFile(Guest guest) {
    try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.txt", true))) {
        bw.write(String.format("%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealName(), guest.getIdNumber()));
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}
```

2. Inheritance widely use in our program.We define method in different classes and make them connected by class relationship:Inheritance. Therefore, we could excute method code in some related different class. For instance1: we inherit JFrame class and them define the UI,which could let user input in the frame.

```

public class GuestRegistrationFrame extends JFrame {
    private List<User> users;

    public GuestRegistrationFrame(List<User> users) {
        this.users = users;

        // Set title
        setTitle("Guest Registration");

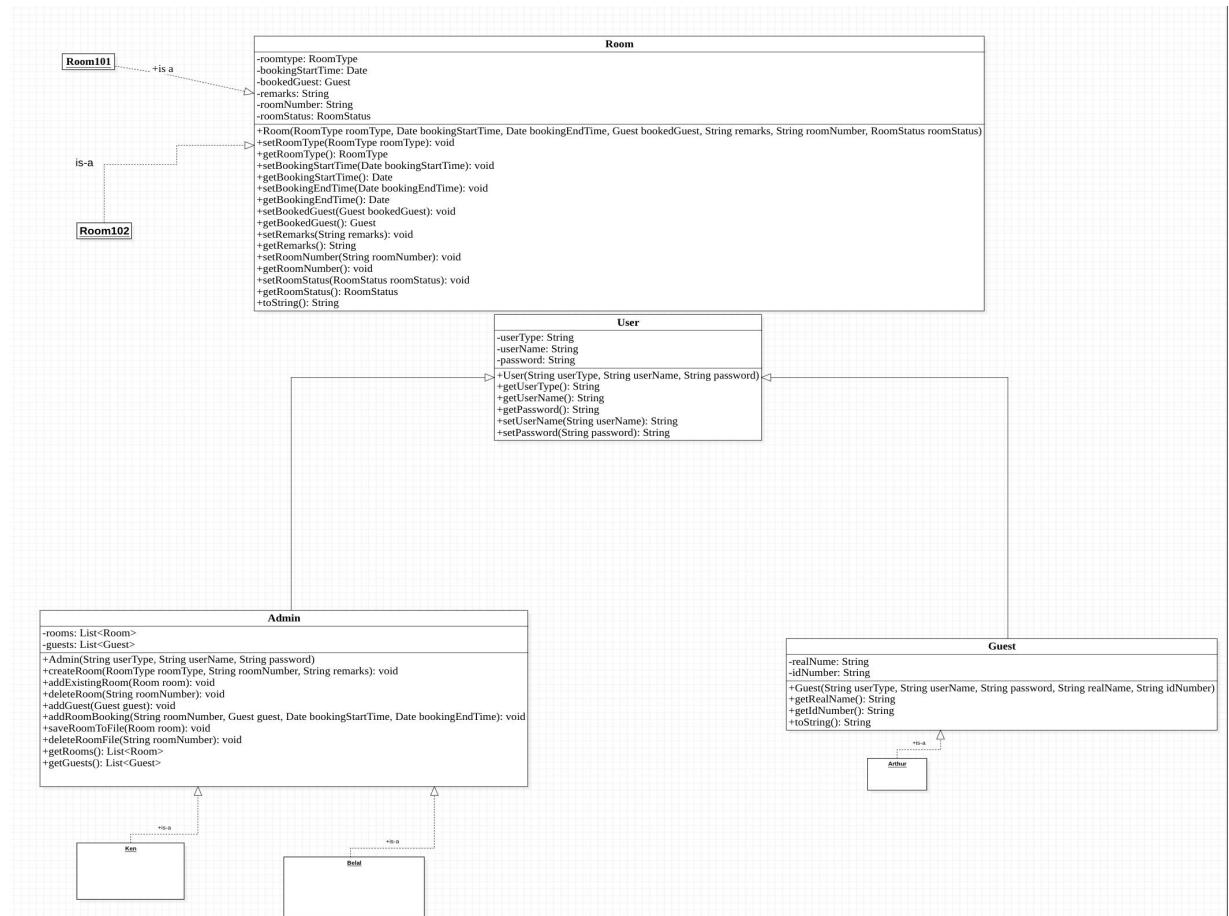
        // Set window size
        setSize(400, 400);

        // Set the default shutdown operation
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Set window layout
        setLayout(new GridLayout(6, 2, 10, 10));
    }
}

```

For instance2: The Guest and Admin inherit User. The Guest and Admin could use method which was defined in the superclass User. Then when the Frame come into Guest or Admin, They can implement their respective target functions according to the methods defined by the superclass.



3.We use OOP(Object Oriented Programming) to cause the object to carry the method. And we develop it unlike the course mentioned because of login system rule.

As we all know that when we login in and input the information in the frame, if our input is invalid so we do not have to create the space(memory address) for this object, just like Anonymous Array.If it created and used it by reference,then after being used it will be deleted directly to save the system space to make system more stable.

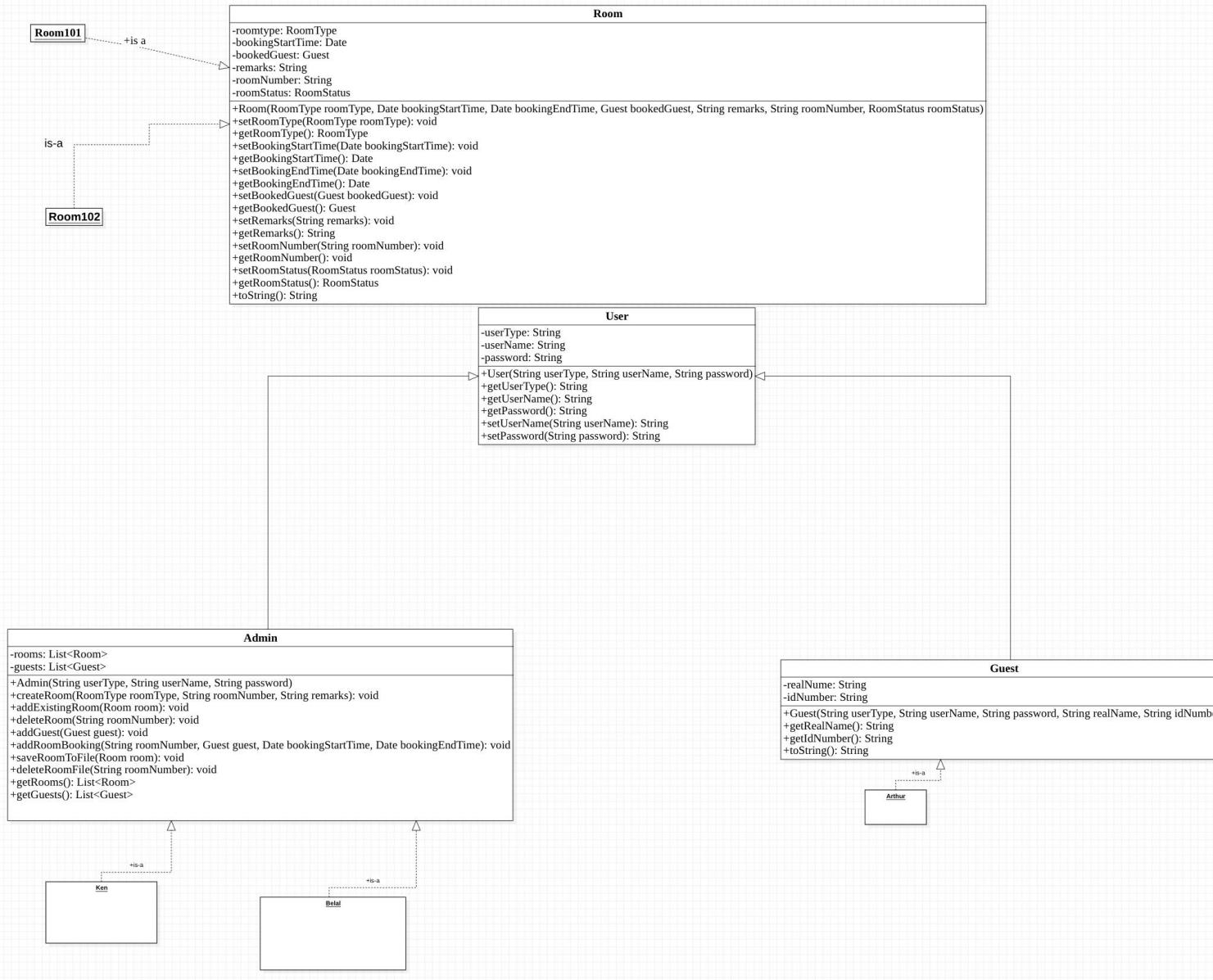
```
if (updated) {  
    saveUsersToFile();  
    JOptionPane.showMessageDialog(this, "Password updated successfully!");  
    this.setVisible(false);  
    new HotelManagementSystem(users).setVisible(true);  
} else {  
    JOptionPane.showMessageDialog(this, "User not found or information does not match.");  
}  
});  
add(submitButton);  
}
```

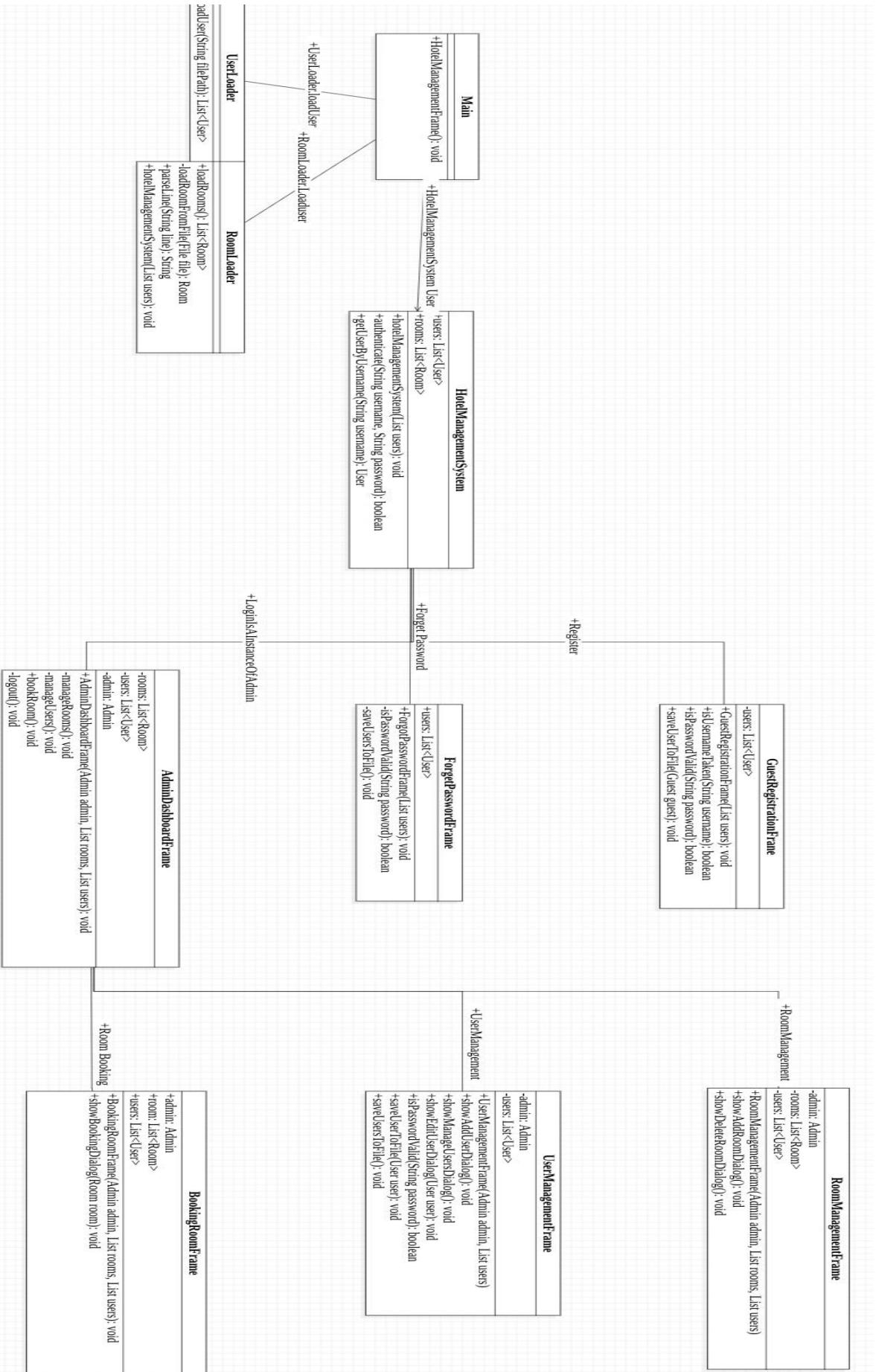
Analysis example:

```
new HotelManagementSystem(users).setVisible(true);
```

do not need to define objects and store space in memory address. If this statement execute, it will come into a new frame and then delete the space when the keyword new is created. This way is more effect and saved when we need to use the method carried by objects.It would let system more stable.

a) UML Diagrams





b) UML Code Generation

```
1 import java.util.*;
2
3 /**
4  * 
5  */
6 public class Admin extends User {
7
8     /**
9      * Default constructor
10     */
11    public Admin() {
12    }
13
14    /**
15     * 
16     */
17    private List<Room> rooms;
18
19    /**
20     * 
21     */
22    private List<Guest> guests;
23
24    /**
25     * @param String userType
26     * @param String userName
27     * @param String password
28     */
29    public void Admin(void String userType, void String userName, void String password) {
30        // TODO implement here
31    }
32
33    /**
34     * @param RoomType roomType
35     * @param String roomNumber
36     * @param String remarks
37     * @return
38     */
39    public void createRoom(void RoomType roomType, void String roomNumber, void String remarks) {
40        // TODO implement here
41        return null;
42    }
43
44    /**
45     * @param Room room
46     * @return
47     */
48    public void addExistingRoom(void Room room) {
49        // TODO implement here
50        return null;
51    }
52
53    /**
54     * @param String roomNumber
55     * @return
56     */
57    public void deleteRoom(void String roomNumber) {
58        // TODO implement here
59        return null;
60    }
61
62    /**
63     * @param Guest guest
64     * @return
65     */
66    public void addGuest(void Guest guest) {
67        // TODO implement here
68        return null;
69    }
70
71    /**
72     * @param String roomNumber
73     * @param Guest guest
74     * @param Date bookingStartTime
75     * @param Date bookingEndTime
76     * @return
77     */
78    public void addRoomBooking(void String roomNumber, void Guest guest, void Date bookingStartTime, void Date bookingEndTime) {
79        // TODO implement here
80        return null;
81    }
82
83    /**
84     * @param Room room
85     * @return
86     */
87    public void saveRoomToFile(void Room room) {
88        // TODO implement here
89    }
```



```
44 /**
45  * @param Room room
46  * @return
47  */
48 public void addExistingRoom(void Room room) {
49     // TODO implement here
50     return null;
51 }
52
53 /**
54  * @param String roomNumber
55  * @return
56  */
57 public void deleteRoom(void String roomNumber) {
58     // TODO implement here
59     return null;
60 }
61
62 /**
63  * @param Guest guest
64  * @return
65  */
66 public void addGuest(void Guest guest) {
67     // TODO implement here
68     return null;
69 }
70
71 /**
72  * @param String roomNumber
73  * @param Guest guest
74  * @param Date bookingStartTime
75  * @param Date bookingEndTime
76  * @return
77  */
78 public void addRoomBooking(void String roomNumber, void Guest guest, void Date bookingStartTime, void Date bookingEndTime) {
79     // TODO implement here
80     return null;
81 }
82
83 /**
84  * @param Room room
85  * @return
86  */
87 public void saveRoomToFile(void Room room) {
88     // TODO implement here
89 }
```

```

88     public void saveRoomToFile(void Room room) {
89         // TODO implement here
90         return null;
91     }
92
93     /**
94      * @param String roomNumber
95      * @return
96      */
97     public void deleteRoomFile(void String roomNumber) {
98         // TODO implement here
99         return null;
100    }
101
102    /**
103     * @return
104     */
105    public List<Room> getRooms() {
106        // TODO implement here
107        return null;
108    }
109
110    /**
111     * @return
112     */
113    public List<Guest> getGuests() {
114        // TODO implement here
115        return null;
116    }
117
118}

```

```

2     import java.util.*;
3
4     /**
5      *
6      */
7     public class AdminDashboardFrame {
8
9         /**
10          * Default constructor
11          */
12         public AdminDashboardFrame() {
13
14
15         /**
16          *
17          */
18         private List<Room> rooms;
19
20         /**
21          *
22          */
23         private List<User> users;
24
25         /**
26          *
27          */
28         private Admin admin;
29
30         /**
31          * @param Admin admin
32          * @param List rooms
33          * @param List users
34          * @return
35          */
36         public void AdminDashboardFrame(void Admin admin, void List rooms, void List users) {
37             // TODO implement here
38             return null;
39         }
40
41         /**
42          * @return
43          */
44         private void manageRooms() {
45             // TODO implement here
46             return null;
47         }

```

```
48
49     /**
50      * @return
51      */
52     private void manageUsers() {
53         // TODO implement here
54         return null;
55     }
56
57     /**
58      * @return
59      */
60     public void bookRoom() {
61         // TODO implement here
62         return null;
63     }
64
65     /**
66      * @return
67      */
68     private void logout() {
69         // TODO implement here
70         return null;
71     }
72
73 }
```

```
import java.util.*;

/**
 *
 */
public class BookingRoomFrame {

    /**
     * Default constructor
     */
    public BookingRoomFrame() {
    }

    /**
     *
     */
    public Admin admin;

    /**
     *
     */
    public List<Room> room;

    /**
     *
     */
    public List<User> users;

    /**
     * @param Admin admin
     * @param List rooms
     * @param List users
     * @return
     */
    public void BookingRoomFrame(void Admin admin, void List rooms, void List users) {
        // TODO implement here
        return null;
    }
}
```

```
31      * @param Admin admin
32      * @param List rooms
33      * @param List users
34      * @return
35      */
36     public void BookingRoomFrame(void Admin admin, void List rooms, void List users) {
37         // TODO implement here
38         return null;
39     }
40
41     /**
42      * @param Room room
43      * @return
44      */
45     public void showBookingDialog(void Room room) {
46         // TODO implement here
47         return null;
48     }
49
50 }
```

```
2     import java.util.*;
3
4     /**
5      *
6      */
7     public class ForgetPasswordFrame {
8
9         /**
10          * Default constructor
11          */
12         public ForgetPasswordFrame() {
13     }
14
15         /**
16          *
17          */
18         public List<User> users;
19
20         /**
21          * @param List users
22          * @return
23          */
24         public void ForgotPasswordFrame(void List users) {
25             // TODO implement here
26             return null;
27         }
28
29         /**
30          * @param String password
31          * @return
32          */
33         private boolean isPasswordValid(void String password) {
34             // TODO implement here
35             return false;
36         }
37
38         /**
39          * @return
40          */
41         private void saveUsersToFile() {
42             // TODO implement here
43             return null;
44         }
45
46 }
```

```

46     */
47     public String getIdNumber() {
48         // TODO implement here
49         return "";
50     }
51
52     /**
53      * @return
54      */
55     public String toString() {
56         // TODO implement here
57         return "";
58     }
59
60 }

1 import java.util.*;
2
3 /**
4  *
5  */
6 public class GuestRegistrationFrame {
7
8     /**
9      * Default constructor
10     */
11    public GuestRegistrationFrame() {
12    }
13
14    /**
15     *
16     */
17    private List<User> users;
18
19
20    /**
21     * @param List users
22     * @return
23     */
24    public void GuestRegistrationFrame(void List users) {
25        // TODO implement here
26        return null;
27    }
28
29    /**
30     * @param String username
31     * @return
32     */
33    public boolean isUsernameTaken(void String username) {
34        // TODO implement here
35        return false;
36    }
37
38    /**
39     * @param String password
40     * @return
41     */
42    public boolean isPasswordValid(void String password) {
43        // TODO implement here
44        return false;
45    }
46
47
48     /**
49      * @param Guest guest
50      * @return
51      */
52    public void saveUserToFile(void Guest guest) {
53        // TODO implement here
54        return null;
55    }
56

```

```
45     * @param String username
46     * @return
47     */
48     public User getUserByUsername(void String username) {
49         // TODO implement here
50         return null;
51     }
52 }
53 }
```

```
1
2     import java.util.*;
3
4     /**
5      *
6      */
7     public class Main {
8
9         /**
10          * Default constructor
11         */
12         public Main() {
13
14
15         /**
16          * @return
17          */
18         public void HotelManagementFrame() {
19             // TODO implement here
20             return null;
21
22
23     }
```

```

1   import java.util.*;
2
3   /**
4    * 
5    */
6   public class Room {
7
8       /**
9        * Default constructor
10       */
11      public Room() {
12
13      }
14
15      /**
16       * 
17       */
18      private RoomType roomtype;
19
20      /**
21       * 
22       */
23      private Date bookingStartTime;
24
25      /**
26       * 
27       */
28      private Guest bookedGuest;
29
30      /**
31       * 
32       */
33      private String remarks;
34
35      /**
36       * 
37       */
38      private String roomNumber;
39
40      /**
41       * 
42       */
43      private RoomStatus roomStatus;
44
45      /**
46       * @param RoomType roomType
47       * @param Date bookingStartTime
48       * @param Date bookingEndTime
49       * @param Guest bookedGuest
50       * @param String remarks
51       * @param String roomNumber
52       * @param RoomStatus roomStatus
53      */
54      public void Room(void RoomType roomType, void Date bookingStartTime, void Date bookingEndTime, void Guest bookedGuest, void String remarks, void String roomNumber,
55      // TODO implement here
56      )
57
58      /**
59       * @param RoomType roomType
60       * @return
61       */
62      public void setRoomType(void RoomType roomType) {
63      // TODO implement here
64      return null;
65      }
66
67      /**
68       * @return
69       */
70      public RoomType getRoomType() {
71      // TODO implement here
72      return null;
73      }
74
75      /**
76       * @param Date bookingStartTime
77       * @return
78       */
79      public void setBookingStartTime(void Date bookingStartTime) {
80      // TODO implement here
81      return null;
82      }
83
84      /**

```

```

84 /**
85 * @return
86 */
87 public Date getBookingStartTime() {
88     // TODO implement here
89     return null;
90 }
91 /**
92 * @param Date bookingEndTime
93 * @return
94 */
95 public void setBookingEndTime(void Date bookingEndTime) {
96     // TODO implement here
97     return null;
98 }
99 /**
100 * @return
101 */
102 public Date getBookingEndTime() {
103     // TODO implement here
104     return null;
105 }
106 /**
107 * @param Guest bookedGuest
108 * @return
109 */
110 public void setBookedGuest(void Guest bookedGuest) {
111     // TODO implement here
112     return null;
113 }
114 /**
115 * @return
116 */
117 public Guest getBookedGuest() {
118     // TODO implement here
119     return null;
120 }
121 /**
122 * @param String remarks
123 * @return
124 */
125 public void setRemarks(void String remarks) {
126     // TODO implement here
127     return null;
128 }
129 /**
130 * @param String remarks
131 * @return
132 */
133 public String getRemarks() {
134     // TODO implement here
135     return "";
136 }
137 /**
138 * @param String roomNumber
139 * @return
140 */
141 public void setRoomNumber(void String roomNumber) {
142     // TODO implement here
143     return null;
144 }
145 /**
146 * @return
147 */
148 public String getRoomNumber() {
149     // TODO implement here
150     return null;
151 }
152 /**
153 * @return
154 */
155 public void getRoomNumber() {
156     // TODO implement here
157     return null;
158 }
159 /**
160 * @param RoomStatus roomStatus
161 * @return
162 */
163 public void setRoomStatus(void RoomStatus roomStatus) {
164     // TODO implement here
165     return null;
166 }
167 /**
168 * @return
169 */
170 */

```

```
169      /**
170      * @return
171      */
172     public RoomStatus getRoomStatus() {
173         // TODO implement here
174         return null;
175     }
176
177     /**
178      * @return
179      */
180     public String toString() {
181         // TODO implement here
182         return "";
183     }
184
185 }
```

```
5      *  
6      */  
7  public class RoomLoader {  
8  
9      /**  
10      * Default constructor  
11      */  
12     public RoomLoader() {  
13    }  
14  
15     /**  
16     * @return  
17     */  
18     public List<Room> loadRooms() {  
19         // TODO implement here  
20         return null;  
21    }  
22  
23     /**  
24     * @param File file  
25     * @return  
26     */  
27     private Room loadRoomFromFile(void File file) {  
28         // TODO implement here  
29         return null;  
30    }  
31  
32     /**  
33     * @param String line  
34     * @return  
35     */  
36     public String parseLine(void String line) {  
37         // TODO implement here  
38         return "";  
39    }  
40  
41     /**  
42     * @param List users  
43     * @return  
44     */  
45     public void hotelManagementSystem(void List users) {  
46         // TODO implement here  
47         return null;  
48    }  
49  
50 }
```

```
1 import java.util.*;
2
3 /**
4  * 
5  */
6 public class RoomManagementFrame {
7
8     /**
9      * Default constructor
10     */
11    public RoomManagementFrame() {
12    }
13
14    /**
15     * 
16     */
17    private Admin admin;
18
19    /**
20     * 
21     */
22    private List<Room> rooms;
23
24    /**
25     * 
26     */
27    private List<User> users;
28
29    /**
30     * @param Admin admin
31     * @param List rooms
32     * @param List users
33     */
34    public void RoomManagementFrame(void Admin admin, void List rooms, void List users) {
35        // TODO implement here
36    }
37
38    /**
39     * @return
40     */
41    public void showAddRoomDialog() {
42        // TODO implement here
43        return null;
44    }
45
46    /**
47     * @return
48     */
49    public void showDeleteRoomDialog() {
50        // TODO implement here
51        return null;
52    }
53
54    /**
55     * 
56     */
57    public void Operation3() {
58        // TODO implement here
59    }
60
61    /**
62     * 
63     */
64    public void Operation4() {
65        // TODO implement here
66    }
67
68
69 }
```

```
5   */
6   public class User {
7
8     /**
9      * Default constructor
10     */
11    public User() {
12    }
13
14    /**
15     *
16     */
17    private String userType;
18
19    /**
20     *
21     */
22    private String userName;
23
24    /**
25     *
26     */
27    private String password;
28
29    /**
30     * @param String userType
31     * @param String userName
32     * @param String password
33     */
34    public void User(void String userType, void String userName, void String password) {
35      // TODO implement here
36    }
37
38    /**
39     * @return
40     */
41    public String getUserType() {
42      // TODO implement here
43      return "";
44    }
45
46    /**
47     * @return
48     */
49    public String getUserName() {
50
```

```
50      /**
51       * @return
52       */
53      public String getUserName() {
54        // TODO implement here
55        return "";
56      }
57
58      /**
59       * @return
60       */
61      public String getPassword() {
62        // TODO implement here
63        return "";
64      }
65
66      /**
67       * @param String userName
68       * @return
69       */
70      public String setUserName(void String userName) {
71        // TODO implement here
72        return "";
73      }
74
75      /**
76       * @param String password
77       * @return
78       */
79      public String setPassword(void String password) {
80        // TODO implement here
81        return "";
82      }
83
```

```
2 import java.util.*;
3
4 /**
5  * 
6  */
7 public class UserLoader {
8
9     /**
10      * Default constructor
11      */
12     public UserLoader() {
13
14     }
15
16     /**
17      * @param String filePath
18      * @return
19      */
20     public List<User> loadUser(void String filePath) {
21         // TODO implement here
22         return null;
23     }
24 }
```

```
2 import java.util.*;
3
4 /**
5  * 
6  */
7 public class UserManagementFrame {
8
9     /**
10      * Default constructor
11      */
12     public UserManagementFrame() {
13
14     }
15
16     /**
17      * 
18      */
19     private Admin admin;
20
21     /**
22      * 
23      */
24     private List<User> users;
25
26     /**
27      * @param Admin admin
28      * @param List users
29      */
30     public void UserManagementFrame(void Admin admin, void List users) {
31         // TODO implement here
32     }
33
34     /**
35      * @return
36      */
37     public void showAddUserDialog() {
38         // TODO implement here
39         return null;
40     }
41
42     /**
43      * @return
44      */
45     public void showManageUsersDialog() {
46         // TODO implement here
47         return null;
48     }
49 }
```

```
47     }
48
49     /**
50      * @param User user
51      * @return
52      */
53     public void showEditUserDialog(void User user) {
54         // TODO implement here
55         return null;
56     }
57
58     /**
59      * @param String password
60      * @return
61      */
62     public boolean isPasswordValid(void String password) {
63         // TODO implement here
64         return false;
65     }
66
67     /**
68      * @param User user
69      * @return
70      */
71     public void saveUserToFile(void User user) {
72         // TODO implement here
73         return null;
74     }
75
76     /**
77      * @return
78      */
79     public void saveUsersToFile() {
80         // TODO implement here
81         return null;
82     }
83
84 }
```

c) Full code implantation (and Screenshots)

```
1  -----
2
3 import java.util.List;
4
5 public class Main {
6     public static void main(String[] args) {
7         String filePath = "/Users/matsu/matsu/eclipse-workspace/testing/src/Login_System/User.txt";
8         List<User> users = UserLoader.loadUsers(filePath);
9         List<Room> rooms = RoomLoader.loadRooms();
10
11        HotelManagementSystem frame = new HotelManagementSystem(users);
12        frame.setVisible(true);
13    }
14}
15
```

```
2
3④ import javax.swing.*;⑤
4
5 public class HotelManagementSystem extends JFrame {
6     private List<User> users;
7     private List<Room> rooms;
8
9
10    public HotelManagementSystem(List<User> users) {
11        this.users = users;
12        this.rooms = RoomLoader.loadRooms();
13
14
15        setTitle("Hotel Management System");
16
17
18        setSize(400, 300);
19
20
21        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22
23
24        setLayout(new BorderLayout());
25
26
27        JLabel titleLabel = new JLabel("Hotel Management System", SwingConstants.CENTER);
28        titleLabel.setFont(new Font("Serif", Font.BOLD, 20));
29        add(titleLabel, BorderLayout.NORTH);
30
31
32        JPanel panel = new JPanel();
33        panel.setLayout(new GridLayout(3, 2, 10, 10));
34
35
36
```

```

37     JLabel userLabel = new JLabel("Username:");
38     JTextField userText = new JTextField(20);
39     panel.add(userLabel);
40     panel.add(userText);
41
42     JLabel passwordLabel = new JLabel("Password:");
43     JPasswordField passwordText = new JPasswordField(20);
44     panel.add(passwordLabel);
45     panel.add(passwordText);
46
47
48     add(panel, BorderLayout.CENTER);
49
50
51     JPanel buttonPanel = new JPanel();
52     buttonPanel.setLayout(new FlowLayout());
53
54
55     JButton loginButton = new JButton("Login");
56     buttonPanel.add(loginButton);
57     loginButton.addActionListener(e -> {
58         String username = userText.getText();
59         String password = new String(passwordText.getPassword());
60
61         boolean authenticated = authenticate(username, password);
62         if (authenticated) {
63             User loggedInUser = getUserByUsername(username);
64             if (loggedInUser instanceof Admin) {
65                 this.setVisible(false);
66                 new AdminDashboardFrame((Admin) loggedInUser, rooms, users).setVisible(true);
67             }
68             else {
69                 JOptionPane.showMessageDialog(null, "Login successful");
70             }
71         }
72     });
73
74     else {
75         JOptionPane.showMessageDialog(null, "Invalid username or password");
76     }
77 });
78
79
80
81     JButton registerButton = new JButton("Register");
82     buttonPanel.add(registerButton);
83     registerButton.addActionListener(e -> {
84         this.setVisible(false);
85         new GuestRegistrationFrame(users).setVisible(true);
86     });
87
88

```

```
90
91     JButton forgotPasswordButton = new JButton("Forgot Password");
92     buttonPanel.add(forgotPasswordButton);
93     forgotPasswordButton.addActionListener(e -> {
94         this.setVisible(false);
95         new ForgotPasswordFrame(users).setVisible(true);
96     });
97     add(buttonPanel, BorderLayout.SOUTH);
98 }
99
100
101     private boolean authenticate(String username, String password) {
102         for (User user : users) {
103             if (user.getUsername().equals(username) && user.getPassword().equals(password)) {
104                 return true;
105             }
106         }
107         return false;
108     }
109
110     private User getUserByUsername(String username) {
111         for (User user : users) {
112             if (user.getUsername().equals(username)) {
113                 return user;
114             }
115         }
116         return null;
117     }
118
119 }
120 }
121
122 import javax.swing.*;
123
124 public class GuestRegistrationFrame extends JFrame {
125     private List<User> users;
126
127     public GuestRegistrationFrame(List<User> users) {
128         this.users = users;
129
130         // Set title
131         setTitle("Guest Registration");
132
133         // Set window size
134         setSize(400, 400);
135
136         // Set the default shutdown operation
137         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
138
139         // Set window layout
140         setLayout(new GridLayout(6, 2, 10, 10));
141
142         // Create form labels and text fields
143         JLabel userTypeLabel = new JLabel("User Type:");
144         JTextField userTypeText = new JTextField("Guest");
145         userTypeText.setEnabled(false); // The userType cannot be changed.
146         add(userTypeLabel);
147         add(userTypeText);
148
149         JLabel nameLabel = new JLabel("Name:");
150         JTextField nameText = new JTextField();
151         add(nameLabel);
152         add(nameText);
153
154         JLabel addressLabel = new JLabel("Address:");
155         JTextField addressText = new JTextField();
156         add(addressLabel);
157         add(addressText);
158
159         JLabel cityLabel = new JLabel("City:");
160         JTextField cityText = new JTextField();
161         add(cityLabel);
162         add(cityText);
163
164         JLabel stateLabel = new JLabel("State:");
165         JTextField stateText = new JTextField();
166         add(stateLabel);
167         add(stateText);
168
169         JLabel zipLabel = new JLabel("Zip Code:");
170         JTextField zipText = new JTextField();
171         add(zipLabel);
172         add(zipText);
173
174         JButton registerButton = new JButton("Register");
175         JButton cancelButton = new JButton("Cancel");
176         registerButton.addActionListener(e -> {
177             String name = nameText.getText();
178             String address = addressText.getText();
179             String city = cityText.getText();
180             String state = stateText.getText();
181             String zip = zipText.getText();
182
183             User user = new User(name, address, city, state, zip, "Guest");
184             users.add(user);
185             JOptionPane.showMessageDialog(this, "Registration successful!");
186         });
187         cancelButton.addActionListener(e -> {
188             System.exit(0);
189         });
190
191         registerButton.setBounds(100, 100, 200, 50);
192         cancelButton.setBounds(100, 150, 200, 50);
193
194         add(registerButton);
195         add(cancelButton);
196     }
197 }
```

```

41     JLabel passwordLabel = new JLabel("Password:");
42     JPasswordField passwordText = new JPasswordField();
43     add(passwordLabel);
44     add(passwordText);
45
46     JLabel realNameLabel = new JLabel("Real Name:");
47     JTextField realNameText = new JTextField();
48     add(realNameLabel);
49     add(realNameText);
50
51     JLabel idNumberLabel = new JLabel("ID Number:");
52     JTextField idNumberText = new JTextField();
53     add(idNumberLabel);
54     add(idNumberText);
55
56
57     JButton registerButton = new JButton("Register");
58     registerButton.addActionListener(e -> {
59         String username = usernameText.getText();
60         String password = new String(passwordText.getPassword());
61         String realName = realNameText.getText();
62         String idNumber = idNumberText.getText();
63
64         if (isUsernameTaken(username)) {
65             JOptionPane.showMessageDialog(this, "Username is already taken.");
66             return;
67         }
68
69         if (isValidPassword(password)) {
70             Guest newGuest = new Guest("Guest", username, password, realName, idNumber);
71             users.add(newGuest);
72             saveUserToFile(newGuest);
73             JOptionPane.showMessageDialog(this, "Registration successful!");
74             this.setVisible(false);
75             new HotelManagementSystem(users).setVisible(true);
76         }
77         JOptionPane.showMessageDialog(this, "Password must be at least 6 characters long and contain both letters and numbers.");
78     });
79     add(registerButton);
80 }
81
82
83
84     private boolean isUsernameTaken(String username) {
85         for (User user : users) {
86             if (user.getUsername().equals(username)) {
87                 return true;
88             }
89         }
90         return false;
91     }
92
93     private boolean isValidPassword(String password) {
94         if (password.length() < 6) {
95             return false;
96         }
97         Pattern letter = Pattern.compile("[a-zA-Z]");
98         Pattern digit = Pattern.compile("[0-9]");
99         return letter.matcher(password).find() && digit.matcher(password).find();
100 }
101
102     private void saveUserToFile(Guest guest) {
103         try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.txt", true))) {
104             bw.write(String.format("%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealName(), guest.getIdNum));
105         }
106         catch (IOException e) {
107             e.printStackTrace();
108         }
109     }
110 }
111

```

```

1 package login_system;
2
3
4④ import javax.swing.*;④
5
6
7
8
9
10
11 public class ForgotPasswordFrame extends JFrame {
12     private List<User> users;
13
14
15④     public ForgotPasswordFrame(List<User> users) {
16         this.users = users;
17
18
19         setTitle("Forgot Password");
20
21         setSize(400, 300);
22
23
24
25         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26
27
28         setLayout(new GridLayout(5, 2, 10, 10));
29
30
31         JLabel usernameLabel = new JLabel("Username:");
32         JTextField usernameText = new JTextField();
33         add(usernameLabel);
34         add(usernameText);
35
36         JLabel realNameLabel = new JLabel("Real Name:");
37         JTextField realNameText = new JTextField();
38         add(realNameLabel);
39         add(realNameText);
40
41         JLabel idNumberLabel = new JLabel("ID Number:");
42         JTextField idNumberText = new JTextField();
43         add(idNumberLabel);
44         add(idNumberText);
45
46         JLabel newPasswordLabel = new JLabel("New Password:");
47         JPasswordField newPasswordText = new JPasswordField();
48         add(newPasswordLabel);
49         add(newPasswordText);
50
51         JButton submitButton = new JButton("Submit");
52         submitButton.addActionListener(e -> {
53             String username = usernameText.getText();
54             String realName = realNameText.getText();
55             String idNumber = idNumberText.getText();
56             String newPassword = new String(newPasswordText.getPassword());
57
58             if (!isPasswordValid(newPassword)) {
59                 JOptionPane.showMessageDialog(this, "Password must be at least 6 characters long and contain both letters and numbers.");
60                 return;
61             }
62
63             boolean updated = false;
64             for (User user : users) {
65                 if (user instanceof Guest) {
66                     Guest guest = (Guest) user;
67                     if (guest.getUsername().equals(username) && guest.getRealName().equals(realName) && guest.getIdNumber().equals(idNumber))
68                         guest.setPassword(newPassword);
69                     updated = true;
70                     break;
71                 }
72             }
73         });
74

```

```

    ...
    saveUsersToFile();
    JOptionPane.showMessageDialog(this, "Password updated successfully!");
    this.setVisible(false);
    new HotelManagementSystem(users).setVisible(true);
} else {
    JOptionPane.showMessageDialog(this, "User not found or information does not match.");
}
});
add(submitButton);
}

private boolean isPasswordValid(String password) {
    if (password.length() < 6) {
        return false;
    }
    Pattern letter = Pattern.compile("[a-zA-Z]");
    Pattern digit = Pattern.compile("[0-9]");
    return letter.matcher(password).find() && digit.matcher(password).find();
}

private void saveUsersToFile() {
    try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.txt"))) {
        for (User user : users) {
            if (user instanceof Guest) {
                Guest guest = (Guest) user;
                bw.write(String.format("%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealName(), guest.getPhone()));
            } else {
                bw.write(String.format("%s,%s,%s%n", user.getUserType(), user.getUsername(), user.getPassword()));
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

import javax.swing.*;

public class AdminDashboardFrame extends JFrame {
    private List<Room> rooms;
    private List<User> users;
    private Admin admin;

    public AdminDashboardFrame(Admin admin, List<Room> rooms, List<User> users) {
        this.admin = admin;
        this.rooms = rooms;
        this.users = users;

        setTitle("Admin Dashboard");

        setSize(500, 400);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new GridLayout(4, 1, 10, 10));

        JButton manageRoomsButton = new JButton("Room Management ");
        manageRoomsButton.addActionListener(e -> manageRooms());
        add(manageRoomsButton);

        JButton manageUsersButton = new JButton("User Management");
        manageUsersButton.addActionListener(e -> manageUsers());
        add(manageUsersButton);
    }
}

```

```

0    JButton bookRoomButton = new JButton("Room Booking");
1    bookRoomButton.addActionListener(e -> bookRoom());
2    add(bookRoomButton);
3
4
5    JButton logoutButton = new JButton("Log out");
6    logoutButton.addActionListener(e -> logout());
7    add(logoutButton);
8 }
9
0⊕ private void manageRooms() {
1    this.setVisible(false);
2    new RoomManagementFrame(admin, rooms, users).setVisible(true);
3 }
4
5⊕ private void manageUsers() {
6    this.setVisible(false);
7    new UserManagementFrame(admin, users).setVisible(true);
8 }
9
0⊕ private void bookRoom() {
1    this.setVisible(false);
2    new BookingRoomFrame(admin, rooms, users).setVisible(true);
3 }
4
5⊕ private void logout() {
6    this.setVisible(false);
7    new HotelManagementSystem(users).setVisible(true);
8 }
9 }

1 -----
2
3⊕ import javax.swing.*;
6
7 public class RoomManagementFrame extends JFrame {
8     private Admin admin;
9     private List<Room> rooms;
10    private List<User> users;
11
12⊕ public RoomManagementFrame(Admin admin, List<Room> rooms, List<User> users) {
13     this.admin = admin;
14     this.rooms = rooms;
15     this.users = users;
16
17     setTitle("Room Management");
18
19
20     setSize(400, 300);
21
22
23     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24
25
26     setLayout(new GridLayout(3, 1, 10, 10));
27
28
29
30     JButton addRoomButton = new JButton("Add Room");
31     addRoomButton.addActionListener(e -> showAddRoomDialog());
32     add(addRoomButton);
33
34

```

```

36     deleteRoomButton.addActionListener(e -> showDeleteRoomDialog());
37     add(deleteRoomButton);
38
39
40     JButton returnButton = new JButton("Return");
41     returnButton.addActionListener(e -> {
42         this.setVisible(false);
43         new AdminDashboardFrame(admin, rooms, users).setVisible(true);
44     });
45     add(returnButton);
46 }
47
48 private void showAddRoomDialog() {
49     JDialog dialog = new JDialog(this, "Add Room", true);
50     dialog.setSize(300, 200);
51     dialog.setLayout(new GridLayout(3, 2, 10, 10));
52
53     JLabel roomNumberLabel = new JLabel("Room Number:");
54     JTextField roomNumberText = new JTextField();
55     dialog.add(roomNumberLabel);
56     dialog.add(roomNumberText);
57
58     JLabel roomTypeLabel = new JLabel("Room Type:");
59     JComboBox<RoomType> roomTypeComboBox = new JComboBox<>(RoomType.values());
60     dialog.add(roomTypeLabel);
61     dialog.add(roomTypeComboBox);
62
63     JButton confirmButton = new JButton("Confirm");
64     confirmButton.addActionListener(e -> {
65         String roomNumber = roomNumberText.getText();
66         RoomType roomType = (RoomType) roomTypeComboBox.getSelectedItem();
67         admin.createRoom(roomType, roomNumber, ""); // 可根据需要添加备注
68         rooms.add(new Room(roomType, null, null, null, "", roomNumber, Room.RoomStatus.vacantRoom));
69         dialog.dispose();
70         JOptionPane.showMessageDialog(this, "Room created successfully");
71     });
72     dialog.add(confirmButton);
73
74     JButton cancelButton = new JButton("Cancel");
75     cancelButton.addActionListener(e -> dialog.dispose());
76     dialog.add(cancelButton);
77
78     dialog.setVisible(true);
79 }
80
81 private void showDeleteRoomDialog() {
82     JDialog dialog = new JDialog(this, "Delete Room", true);
83     dialog.setSize(300, 200);
84     dialog.setLayout(new GridLayout(rooms.size() + 1, 1, 10, 10));
85
86     for (Room room : rooms) {
87         JButton roomButton = new JButton(room.getRoomNumber());
88         roomButton.addActionListener(e -> {
89             int response = JOptionPane.showConfirmDialog(this, "Confirm room deletion " + room.getRoomNumber() + " ? ", "Confirm",
90                 if (response == JOptionPane.YES_OPTION) {
91                     admin.deleteRoom(room.getRoomNumber());
92                     rooms.remove(room);
93                     JOptionPane.showMessageDialog(this, "Room deletes successfully");
94                     dialog.dispose();
95                 }
96             });
97             dialog.add(roomButton);
98     }
99
100    JButton cancelButton = new JButton("Cancel");
101    cancelButton.addActionListener(e -> dialog.dispose());
102    dialog.add(cancelButton);

```

```

2
3@ import javax.swing.*;[]
10
11 public class UserManagementFrame extends JFrame {
12     private Admin admin;
13     private List<User> users;
14
15@     public UserManagementFrame(Admin admin, List<User> users) {
16         this.admin = admin;
17         this.users = users;
18
19         setTitle("User Management");
20
21         setSize(400, 400);
22
23         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24
25         setLayout(new BorderLayout());
26
27         JPanel panel = new JPanel();
28         panel.setLayout(new GridLayout(3, 1, 10, 10));
29
30         JButton addUserButton = new JButton("Add User");
31         addUserButton.addActionListener(e -> showAddUserDialog());
32         panel.add(addUserButton);
33
34         JButton manageUsersButton = new JButton("Modify/Delete User");
35         manageUsersButton.addActionListener(e -> showManageUsersDialog());
36         panel.add(manageUsersButton);
37
38         JButton returnButton = new JButton("Return to Dashboard");
39         returnButton.addActionListener(e -> {
40             this.setVisible(false);
41             new AdminDashboardFrame(admin, null, users).setVisible(true);
42         });
43         panel.add(returnButton);
44
45         add(panel, BorderLayout.CENTER);
46     }
47
48@     private void showAddUserDialog() {
49         JDialog dialog = new JDialog(this, "Add User", true);
50         dialog.setSize(300, 300);
51         dialog.setLayout(new GridLayout(6, 2, 10, 10));
52
53         JLabel userTypeLabel = new JLabel("User Type:");
54         JComboBox<String> userTypeComboBox = new JComboBox<>(new String[] {"Admin", "Guest"});
55         dialog.add(userTypeLabel);
56         dialog.add(userTypeComboBox);
57
58         JLabel usernameLabel = new JLabel("Username:");
59         JTextField usernameText = new JTextField();
60         dialog.add(usernameLabel);
61         dialog.add(usernameText);
62
63         JLabel passwordLabel = new JLabel("Password:");
64         JPasswordField passwordText = new JPasswordField();
65         dialog.add(passwordLabel);
66         dialog.add(passwordText);
67
68         JLabel realNameLabel = new JLabel("Real Name:");
69         JTextField realNameText = new JTextField();
70         dialog.add(realNameLabel);
71         dialog.add(realNameText);
72

```

```
15     JLabel idNumberLabel = new JLabel("ID Number: ");
16     JTextField idNumberText = new JTextField();
17     dialog.add(idNumberLabel);
18     dialog.add(idNumberText);
19
20     userTypeComboBox.addActionListener(e -> {
21         if (userTypeComboBox.getSelectedItem().equals("Admin")) {
22             realNameText.setEnabled(false);
23             idNumberText.setEnabled(false);
24         } else {
25             realNameText.setEnabled(true);
26             idNumberText.setEnabled(true);
27         }
28     });
29
30     JButton confirmButton = new JButton("Confirm");
31     confirmButton.addActionListener(e -> {
32         String userType = (String) userTypeComboBox.getSelectedItem();
33         String username = usernameText.getText();
34         String password = new String(passwordText.getPassword());
35         String realName = realNameText.getText();
36         String idNumber = idNumberText.getText();
37
38         if (isPasswordValid(password)) {
39             User newUser;
40             if ("Admin".equals(userType)) {
41                 newUser = new Admin(userType, username, password);
42             } else {
43                 newUser = new Guest(userType, username, password, realName, idNumber);
44             }
45             users.add(newUser);
46             saveUserToFile(newUser);
47             dialog.dispose();
48             JOptionPane.showMessageDialog(this, "User created successfully");
49         } else {
50             JOptionPane.showMessageDialog(this, "Passwords must be at least 6 characters long and contain letters and numbers.");
51         }
52     });
53     dialog.add(confirmButton);
54
55     JButton cancelButton = new JButton("Cancel");
56     cancelButton.addActionListener(e -> dialog.dispose());
57     dialog.add(cancelButton);
58
59     dialog.setVisible(true);
60 }
61
62 private void showManageUsersDialog() {
63     JDialog dialog = new JDialog(this, "Modify/Delete User", true);
64     dialog.setSize(300, 400);
65     dialog.setLayout(new GridLayout(users.size() + 1, 1, 10, 10));
66
67     for (User user : users) {
68         JButton userButton = new JButton(user instanceof Guest ? ((Guest) user).getRealName() : user.getUsername());
69         userButton.addActionListener(e -> showEditUserDialog(user));
70         dialog.add(userButton);
71     }
72
73     JButton cancelButton = new JButton("Cancel");
74     cancelButton.addActionListener(e -> dialog.dispose());
75     dialog.add(cancelButton);
76
77     dialog.setVisible(true);
78 }
79
80 private void showEditUserDialog(User user) {
81     JDialog dialog = new JDialog(this, "edit user", true);
82     dialog.setSize(300, 200);
83     dialog.setLayout(new GridLayout(4, 2, 10, 10));
84 }
```

```

150~     private void showEditUserDialog(User user) {
151~         JPanel panel = new JPanel();
152~         JButton cancelButton = new JButton("Cancel");
153~         JButton deleteButton = new JButton("Delete User");
154~         JButton confirmButton = new JButton("Confirm");
155~         JTextField usernameText = new JTextField(user.getUsername());
156~         JPasswordField passwordText = new JPasswordField();
157~         JLabel usernameLabel = new JLabel("Username:");
158~         JLabel passwordLabel = new JLabel("New Password:");
159~         dialog = new JPanel();
160~         dialog.setLayout(new GridLayout(4, 2, 10, 10));
161~         dialog.add(usernameLabel);
162~         dialog.add(usernameText);
163~         dialog.add(passwordLabel);
164~         dialog.add(passwordText);
165~         dialog.add(confirmButton);
166~         dialog.add(deleteButton);
167~         dialog.add(cancelButton);
168~         dialog.setVisible(true);
169~     }
170~ 
171~     private boolean isPasswordValid(String password) {
172~         if (password.length() < 6) {
173~             return false;
174~         }
175~         Pattern letter = Pattern.compile("[a-zA-Z]");
176~         Pattern digit = Pattern.compile("[0-9]");
177~         return letter.matcher(password).find() && digit.matcher(password).find();
178~     }
179~ 
180~     private void saveUserToFile(User user) {
181~         try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsu/Downloads/eclipse-workspace/testing/src/Login_System/User.txt"))) {
182~             if (user instanceof Guest) {
183~                 Guest guest = (Guest) user;
184~                 bw.write(String.format("%s,%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealName(), guest.getAge()));
185~             } else if (user instanceof Admin) {
186~                 bw.write(String.format("%s,%s,%s%n", user.getUserType(), user.getUsername(), user.getPassword()));
187~             }
188~         }
189~     }

```

```

198     private void saveUserToFile(User user) {
199         try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.txt"))) {
200             if (user instanceof Guest) {
201                 Guest guest = (Guest) user;
202                 bw.write(String.format("%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealName()));
203             } else if (user instanceof Admin) {
204                 bw.write(String.format("%s,%s,%s%n", user.getUserType(), user.getUsername(), user.getPassword()));
205             }
206         } catch (IOException e) {
207             e.printStackTrace();
208         }
209     }
210
211     private void saveUsersToFile() {
212         try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.txt"))) {
213             for (User user : users) {
214                 if (user instanceof Guest) {
215                     Guest guest = (Guest) user;
216                     bw.write(String.format("%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealName()));
217                 } else if (user instanceof Admin) {
218                     bw.write(String.format("%s,%s,%s%n", user.getUserType(), user.getUsername(), user.getPassword()));
219                 }
220             }
221         } catch (IOException e) {
222             e.printStackTrace();
223         }
224     }
225 }
226 package Login_System;
227
228 import javax.swing.*;
229
230 public class BookingRoomFrame extends JFrame {
231     private Admin admin;
232     private List<Room> rooms;
233     private List<User> users;
234     private static final SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
235
236     public BookingRoomFrame(Admin admin, List<Room> rooms, List<User> users) {
237         this.admin = admin;
238         this.rooms = rooms;
239         this.users = users;
240
241         setTitle("Booking Management");
242
243         setSize(600, 600);
244
245         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
246
247         setLayout(new BorderLayout());
248
249         JPanel availableRoomsPanel = new JPanel();
250         availableRoomsPanel.setLayout(new GridLayout(rooms.size(), 1, 10, 10));
251
252         JPanel bookedRoomsPanel = new JPanel();
253         bookedRoomsPanel.setLayout(new GridLayout(rooms.size(), 1, 10, 10));
254
255         JLabel availableRoomsLabel = new JLabel("Available Rooms", SwingConstants.CENTER);
256         availableRoomsLabel.setFont(new Font("Serif", Font.BOLD, 20));
257         availableRoomsPanel.add(availableRoomsLabel);
258
259     }
260
261     public void updateAvailableRooms() {
262         availableRoomsPanel.removeAll();
263         availableRoomsPanel.setLayout(new GridLayout(rooms.size(), 1, 10, 10));
264
265         for (Room room : rooms) {
266             JButton button = new JButton(room.getName());
267             button.addActionListener(new ActionListener() {
268                 @Override
269                 public void actionPerformed(ActionEvent e) {
270                     Room selectedRoom = room;
271                     Room bookedRoom = bookedRoomsPanel.getComponent(0);
272
273                     if (selectedRoom != null && bookedRoom != null) {
274                         String[] options = {"Book", "Cancel", "Exit"};
275                         int option = JOptionPane.showOptionDialog(null, "Select an action for " + selectedRoom.getName() + ":",
276                             "Booking Management", JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, null, options, options[0]);
277
278                         if (option == 0) {
279                             bookRoom(selectedRoom);
280                         } else if (option == 1) {
281                             cancelRoom(selectedRoom);
282                         }
283                     }
284                 }
285             });
286             availableRoomsPanel.add(button);
287         }
288
289         availableRoomsPanel.revalidate();
290     }
291
292     private void bookRoom(Room room) {
293         User user = users.get(0);
294
295         Room bookedRoom = bookedRoomsPanel.getComponent(0);
296
297         if (bookedRoom != null) {
298             bookedRoom.setText(user.getName());
299         } else {
300             bookedRoom = new JButton(room.getName());
301             bookedRoomsPanel.add(bookedRoom);
302         }
303
304         room.setBooked(true);
305     }
306
307     private void cancelRoom(Room room) {
308         User user = users.get(0);
309
310         Room bookedRoom = bookedRoomsPanel.getComponent(0);
311
312         if (bookedRoom != null) {
313             bookedRoom.setText(user.getName());
314         } else {
315             bookedRoom = new JButton(room.getName());
316             bookedRoomsPanel.add(bookedRoom);
317         }
318
319         room.setBooked(false);
320     }
321
322     public void updateBookedRooms() {
323         bookedRoomsPanel.removeAll();
324         bookedRoomsPanel.setLayout(new GridLayout(rooms.size(), 1, 10, 10));
325
326         for (Room room : rooms) {
327             JButton button = new JButton(room.getName());
328             button.addActionListener(new ActionListener() {
329                 @Override
330                 public void actionPerformed(ActionEvent e) {
331                     Room selectedRoom = room;
332                     Room availableRoom = availableRoomsPanel.getComponent(0);
333
334                     if (selectedRoom != null && availableRoom != null) {
335                         String[] options = {"Book", "Cancel", "Exit"};
336                         int option = JOptionPane.showOptionDialog(null, "Select an action for " + selectedRoom.getName() + ":",
337                             "Booking Management", JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, null, options, options[0]);
338
339                         if (option == 0) {
340                             bookRoom(selectedRoom);
341                         } else if (option == 1) {
342                             cancelRoom(selectedRoom);
343                         }
344                     }
345                 }
346             });
347             bookedRoomsPanel.add(button);
348         }
349
350         bookedRoomsPanel.revalidate();
351     }
352
353     public void updateAvailableRooms() {
354         availableRoomsPanel.removeAll();
355         availableRoomsPanel.setLayout(new GridLayout(rooms.size(), 1, 10, 10));
356
357         for (Room room : rooms) {
358             JButton button = new JButton(room.getName());
359             button.addActionListener(new ActionListener() {
360                 @Override
361                 public void actionPerformed(ActionEvent e) {
362                     Room selectedRoom = room;
363                     Room bookedRoom = bookedRoomsPanel.getComponent(0);
364
365                     if (selectedRoom != null && bookedRoom != null) {
366                         String[] options = {"Book", "Cancel", "Exit"};
367                         int option = JOptionPane.showOptionDialog(null, "Select an action for " + selectedRoom.getName() + ":",
368                             "Booking Management", JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE, null, options, options[0]);
369
370                         if (option == 0) {
371                             bookRoom(selectedRoom);
372                         } else if (option == 1) {
373                             cancelRoom(selectedRoom);
374                         }
375                     }
376                 }
377             });
378             availableRoomsPanel.add(button);
379         }
380
381         availableRoomsPanel.revalidate();
382     }
383 }
```

```

  ...
  private void showBookingDialog(Room room) {
    JDIALOG dialog = new JDIALOG(this, "Book Room", true);
    dialog.setSize(400, 400);
    dialog.setLayout(new GridLayout(7, 2, 10, 10));
    ...
    JLabel guestNameLabel = new JLabel("Guest Name:");
    JTextField guestNameText = new JTextField();
    dialog.add(guestNameLabel);
    dialog.add(guestNameText);
    ...
    JLabel guestIdLabel = new JLabel("Guest ID:");
    JTextField guestIdText = new JTextField();
    dialog.add(guestIdLabel);
    dialog.add(guestIdText);
    ...
    JLabel bookingStartLabel = new JLabel("Booking Start Date (yyyy-MM-dd):");
    JTextField bookingStartText = new JTextField();
    dialog.add(bookingStartLabel);
    dialog.add(bookingStartText);
    ...
    JLabel bookingEndLabel = new JLabel("Booking End Date (yyyy-MM-dd):");
    JTextField bookingEndText = new JTextField();
    dialog.add(bookingEndLabel);
    dialog.add(bookingEndText);
    ...
    JButton confirmButton = new JButton("Confirm");
    confirmButton.addActionListener(e -> {
      String guestName = guestNameText.getText();
      String guestId = guestIdText.getText();
      String bookingStartStr = bookingStartText.getText();
      String bookingEndStr = bookingEndText.getText();
      ...
      JLabel bookedRoomsLabel = new JLabel("BOOKED ROOMS", SwingConstants.CENTER);
      bookedRoomsLabel.setFont(new Font("Serif", Font.BOLD, 20));
      bookedRoomsPanel.add(bookedRoomsLabel);
      ...
      for (Room room : rooms) {
        if (room.getRoomStatus() == Room.RoomStatus.vacantRoom) {
          JButton roomButton = new JButton(room.getRoomNumber() + ": " + room.getRoomType());
          roomButton.addActionListener(e -> showBookingDialog(room));
          availableRoomsPanel.add(roomButton);
        } else {
          JLabel bookedRoomLabel = new JLabel(room.getRoomNumber() + ": " + room.getRoomType() + " (Booked by " + room.getBookedGuest);
          bookedRoomsPanel.add(bookedRoomLabel);
        }
      }
      ...
      JPanel centerPanel = new JPanel(new GridLayout(2, 1, 10, 10));
      centerPanel.add(availableRoomsPanel);
      centerPanel.add(bookedRoomsPanel);
      add(centerPanel, BorderLayout.CENTER);
      ...
      JButton returnButton = new JButton("Return");
      returnButton.addActionListener(e -> {
        this.setVisible(false);
        new AdminDashboardFrame(admin, rooms, users).setVisible(true);
      });
      add(returnButton, BorderLayout.SOUTH);
      ...
    });
    ...
    private void showBookingDialog(Room room) {
    JDIALOG dialog = new JDIALOG(this, "Book Room", true);
    dialog.setSize(400, 400);
    dialog.setLayout(new GridLayout(7, 2, 10, 10));
    ...
    JLabel guestNameLabel = new JLabel("Guest Name:");

```

```

99         try {
100             Date bookingStartDate = dateFormat.parse(bookingStartDateStr);
101             Date bookingEndDate = dateFormat.parse(bookingEndDateStr);
102
103             Guest guest = new Guest("Guest", guestName, "", guestName, guestId);
104             admin.addGuest(guest);
105             admin.addRoomBooking(room.getRoomNumber(), guest, bookingStartDate, bookingEndDate);
106             dialog.dispose();
107             JOptionPane.showMessageDialog(this, "Room booked successfully!");
108             this.setVisible(false);
109             new BookingRoomFrame(admin, rooms, users).setVisible(true);
110         } catch (ParseException ex) {
111             JOptionPane.showMessageDialog(this, "Invalid date format. Please use yyyy-MM-dd.");
112         }
113     });
114     dialog.add(confirmButton);
115
116     JButton cancelButton = new JButton("Cancel");
117     cancelButton.addActionListener(e -> dialog.dispose());
118     dialog.add(cancelButton);
119
120     dialog.setVisible(true);
121 }
122 }
123

```

```

1 package Login_System;
2
3 public class User {
4     private String userType;
5     private String username;
6     private String password;
7
8     public User(String userType, String username, String password) {
9         this.userType = userType;
10        this.username = username;
11        this.password = password;
12    }
13
14    public String getUserType() {
15        return userType;
16    }
17
18    public String getUsername() {
19        return username;
20    }
21
22    public String getPassword() {
23        return password;
24    }
25
26    public void setUsername(String username) {
27        this.username = username;
28    }
29
30    public void setPassword(String password) {
31        this.password = password;
32    }
33 }
34

```

```

9 import Login_System.User;
10
11 public class Admin extends User {
12     private List<Room> rooms;
13     private List<Guest> guests;
14     private static final String ROOM_FOLDER_PATH = "/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.txt";
15     private static final SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
16
17     public Admin(String userType, String username, String password) {
18         super(userType, username, password);
19         this.rooms = new ArrayList<>();
20         this.guests = new ArrayList<>();
21     }
22
23     public void createRoom(RoomType roomType, String roomNumber, String remarks) {
24         Room room = new Room(roomType, null, null, null, remarks, roomNumber, Room.RoomStatus.vacantRoom);
25         rooms.add(room);
26         saveRoomToFile(room);
27     }
28
29     public void addExistingRoom(Room room) {
30         rooms.add(room);
31         saveRoomToFile(room);
32     }
33
34     public void deleteRoom(String roomNumber) {
35         rooms.removeIf(room -> room.getRoomNumber().equals(roomNumber));
36         deleteRoomFile(roomNumber);
37     }
38
39     public void addGuest(Guest guest) {
40         guests.add(guest);
41     }
42
43
44     public void updateBooking(String roomNumber, Guest guest, Date bookingStartTime, Date bookingEndTime, String remarks) {
45         for (Room room : rooms) {
46             if (room.getRoomNumber().equals(roomNumber)) {
47                 room.setBookedGuest(guest);
48                 room.setBookingStartTime(bookingStartTime);
49                 room.setBookingEndTime(bookingEndTime);
50                 room.setRoomStatus(Room.RoomStatus.Booking);
51                 saveRoomToFile(room);
52                 break;
53             }
54         }
55
56     private void saveRoomToFile(Room room) {
57         try (BufferedWriter bw = new BufferedWriter(new FileWriter(ROOM_FOLDER_PATH + room.getRoomNumber() + ".txt"))) {
58             bw.write("Room Type: " + room.getRoomType() + "\n");
59             bw.write("Room Status: " + room.getRoomStatus() + "\n");
60             bw.write("Remarks: " + room.getRemarks() + "\n");
61             if (room.getRoomStatus() == Room.RoomStatus.Booking || room.getRoomStatus() == Room.RoomStatus.CheckingIn) {
62                 bw.write("Booked Guest: " + room.getBookedGuest().getUsername() + "\n");
63                 bw.write("Booking Start Time: " + dateFormat.format(room.getBookingStartTime()) + "\n");
64                 bw.write("Booking End Time: " + dateFormat.format(room.getBookingEndTime()) + "\n");
65             } else {
66                 bw.write("Booked Guest: null\n");
67                 bw.write("Booking Start Time: null\n");
68                 bw.write("Booking End Time: null\n");
69             }
70         } catch (IOException e) {
71             e.printStackTrace();
72         }
73     }
74
75     private void deleteRoomFile(String roomNumber) {
76         java.io.File file = new java.io.File(ROOM_FOLDER_PATH + roomNumber + ".txt");
77         if (file.exists()) {
78             file.delete();
79         }
80     }
81
82     public List<Room> getRooms() {
83         return rooms;
84     }
85
86     public List<Guest> getGuests() {
87         return guests;
88     }
89 }

```

```
 1  package com.bogdanov;
 2
 3  public class Guest extends User {
 4      private String realName;
 5      private String idNumber;
 6
 7      public Guest(String userType, String username, String password, String realName, String idNumber) {
 8          super(userType, username, password);
 9          this.realName = realName;
10          this.idNumber = idNumber;
11      }
12
13      public String getRealName() {
14          return realName;
15      }
16
17      public String getIdNumber() {
18          return idNumber;
19      }
20
21      @Override
22      public String toString() {
23          return "Guest{" +
24              "userType='" + getUserType() + '\n' +
25              ", username='" + getUsername() + '\n' +
26              ", realName='" + realName + '\n' +
27              ", idNumber='" + idNumber + '\n' +
28              '}';
29      }
30  }
```

IV. Reports and Analysis

a) Discussions

PS: Knowlegde we use.

We use knowledge from the course CPS2231 and then develop it. We use OOP(Object Oriented Programming) to execute method carried by object. Inheritance class relationship is also much used during our program.

This project involves User interface frame and text storage. To store data by ButteredWriter Class which are the knowledge PrintWriter we learned based on CPS2231 and extra learned from textbook and internet then we develop it.

1. Reports:

Our group program could be used by hotel managers and guests. Managers could manage or adjust room and user information. Guests could book rooms and see the room information.

This Login System could let guests to register their account in the register frame and then jump into login frame. If guests are forget the account, then we could let guests to find back their account by guests' ID number.

If the System examine your account and make sure you are admin. Then the login frame will jump into admin dashboard frame. Then you can adjust rooms information, add or delete new admin/guest and booking a new room for guest. Remember that we could not create admin in the first login frame, only admin manage frame could create admin account.

This Hotel management Login System is suitable for the actual situation for the hotel. It would reduce labor costs and make guest occupancy more efficient.

Analysis:

Advantage:

We highly take the security of user data very seriously, so many of our methods and variables are defined as private. And use it by getter and setter method.

Disadvantage:

Some functions still need to be developed to make the whole system more perfect. For example, customer-related functions can continue to be developed to increase the usefulness of the system.

V. Conclusions

The purpose of this system is to achieve the functions of users to operate the account and administrators to manage the user account. This system uses object-oriented programming to manage users, give users rights of operation and available functions, uses user interface to realize human-computer interaction, and uses Java basic programming to realize account function, which can effectively improve the efficiency of user data processing to adapt to the generation of a large number of user data combining with manpower. In terms of limitations, the users and administrators of this system have few functions and UI is relatively simple. Therefore, we can add the functions of users to and optimize code to improve this system in the future.

References

Betre.,K. (2020). Customizing the title bar of an application window. Medium.

<https://medium.com/swlh/customizing-the-title-bar-of-an-application-window-50a4ac3ed27e>

Yu, Y. (2022). Design and Implementation of Hotel Management Information System Based on Intelligent Terminal. In: Xu, Z., Alrabaee, S., Loyola-González, O., Zhang, X., Cahyani, N.D.W., Ab Rahman, N.H. (eds) Cyber Security Intelligence and Analytics. CSIA 2022. Lecture Notes on Data Engineering and Communications Technologies, vol 123. Springer, Cham.

https://doi.org/10.1007/978-3-030-96908-0_64

Appendix:

Contribution of Group Members

Student Name:Songlin Shang Student Number:1308184 ...

Logical Structure: Code structure dessign(UML), Implement the code, PPT writing, Report Writing

Student Name:Cong Ye Student Number:1306248.....

Implement the code, PPT writing, Report Writing

Student Name:Kailiang Zhu Student Number:1306289.....

Implement the code, PPT writing, Report Writing

NB: There shall be presentation by each member of a group for 5 mins.

Dr. Ken Ehimwenma, Ph.D.