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 effiency 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 langugage.

Aiming at strenghening 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 simpliest 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 inheritence 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, IOExcption 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 passport: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 passord, 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 Numeber, 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) Admininstrator Capability: booking room management, user management and room management

II. Related Works

1. PasswordDetermination.

```java
//Signature: String--->boolean
//Purpose: input a password then
//         determine it whether a password is valid or not.
//Example:
//        1234w--->false
//        12wwwwwwwwww+wwwwww--->false
//        1234wwwwwwwwwwwwwwww--->true
//Header:
public static boolean isValidPassword(String password) {
    if (password.length() < 16) {
        return false;
    }

    int digitCount = 0;
    for (int i = 0; i < password.length(); i++) {
        char ch = password.charAt(i);

        // Check if the character is not a letter, a digit, or a '+'
        if (!Character.isLetterOrDigit(ch) && ch != '+') {
            return false;
        }

        // Count the digits
        if (Character.isDigit(ch)) {
            digitCount++;
        }
    }

    // Check the count of digits
    return digitCount >= 3;
}
```

2.Use try and catch to deal with IOException.

```java
package Chapter12;

public class CircleWithCustomExceptionTest {

    public static void main(String[] args) {
        try {
            new CircleWithCustomException(5);
            new CircleWithCustomException(-5);
            new CircleWithCustomException(0);
            new CircleWithCustomException(7);
        }
        catch(InvalidRadiusException ex) {
            System.out.println(ex);
        }

        System.out.println("Number of objects created: " + CircleWithCustomException.getNumberOfObjects());

    }

}
```

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

```
public class RectangleFromSimpleGeometricObject extends SimpleGeometricObject {
    private double width;
    private double height;

    public RectangleFromSimpleGeometricObject() {
    }

    public RectangleFromSimpleGeometricObject(double width, double height) {
        this.width = width;
        this.height = height;
    }

    public RectangleFromSimpleGeometricObject(double width, double height, String color, boolean filled) {
        super(color,filled);//equal to line 19 20 step1
        this.width = width;//step2
        this.height = height;//step2
//      setColor(color);
//      setFilled(filled);
    }
```

4. Create a file and test a file.

```
package ChapterV12;

public class writeData {

    public static void main(String[] args) throws java.io.IOException {
         java.io.File file = new java.io.File("scores.txt");
            if (file.exists()) {
              System.out.println("File already exists");
              System.exit(1);
            }

            // Create a file
            java.io.PrintWriter output = new java.io.PrintWriter(file);

            // Write formatted output to the file
            output.print("John T. Perez ");
            output.println(90);
            output.print("Jamal K. Johnson ");
            output.println(85);

            // Close the file
            output.close();

    }

}
```

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("\"", 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.

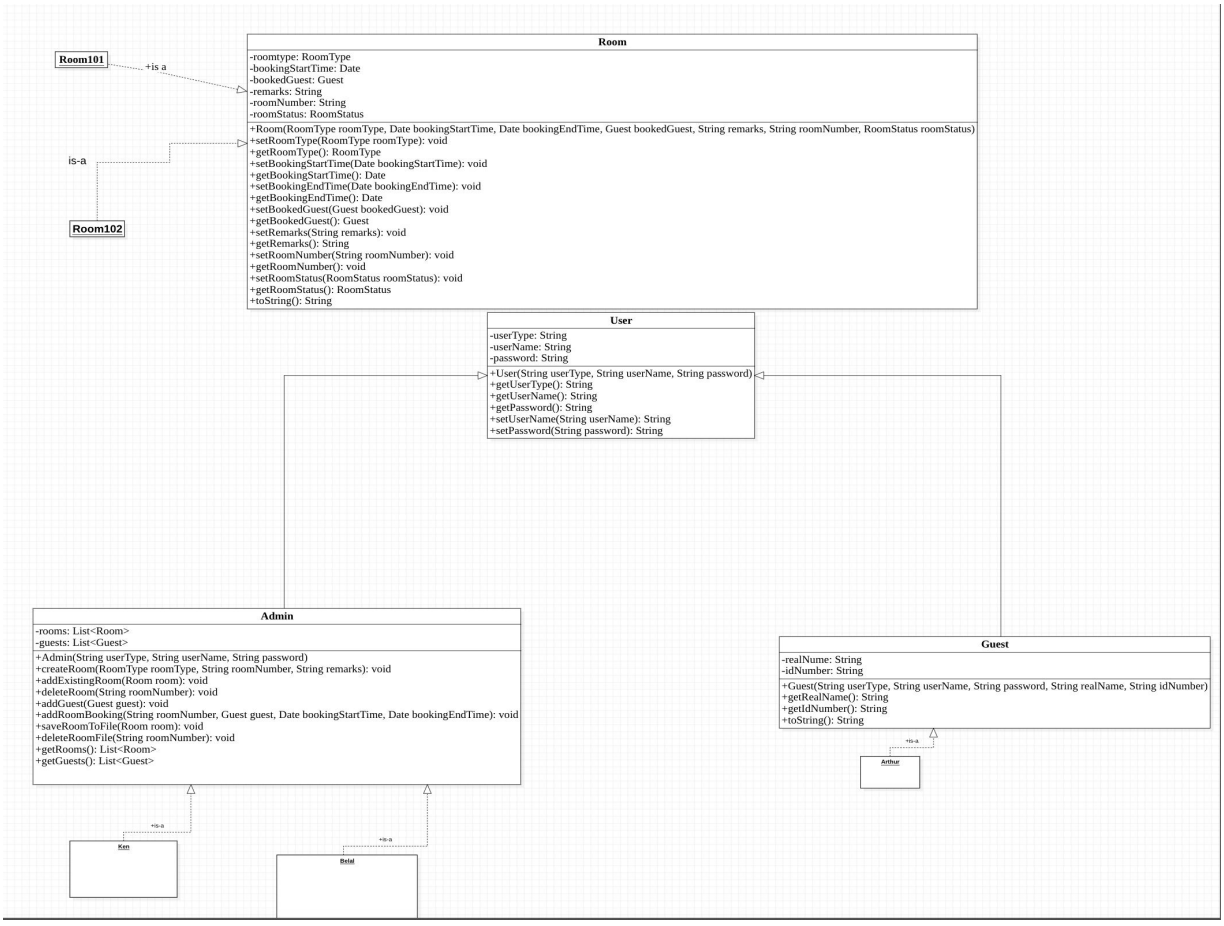
```
    private void saveUserToFile(Guest guest) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.txt", true))) {
            bw.write(String.format("%s,%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(),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

```java
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.

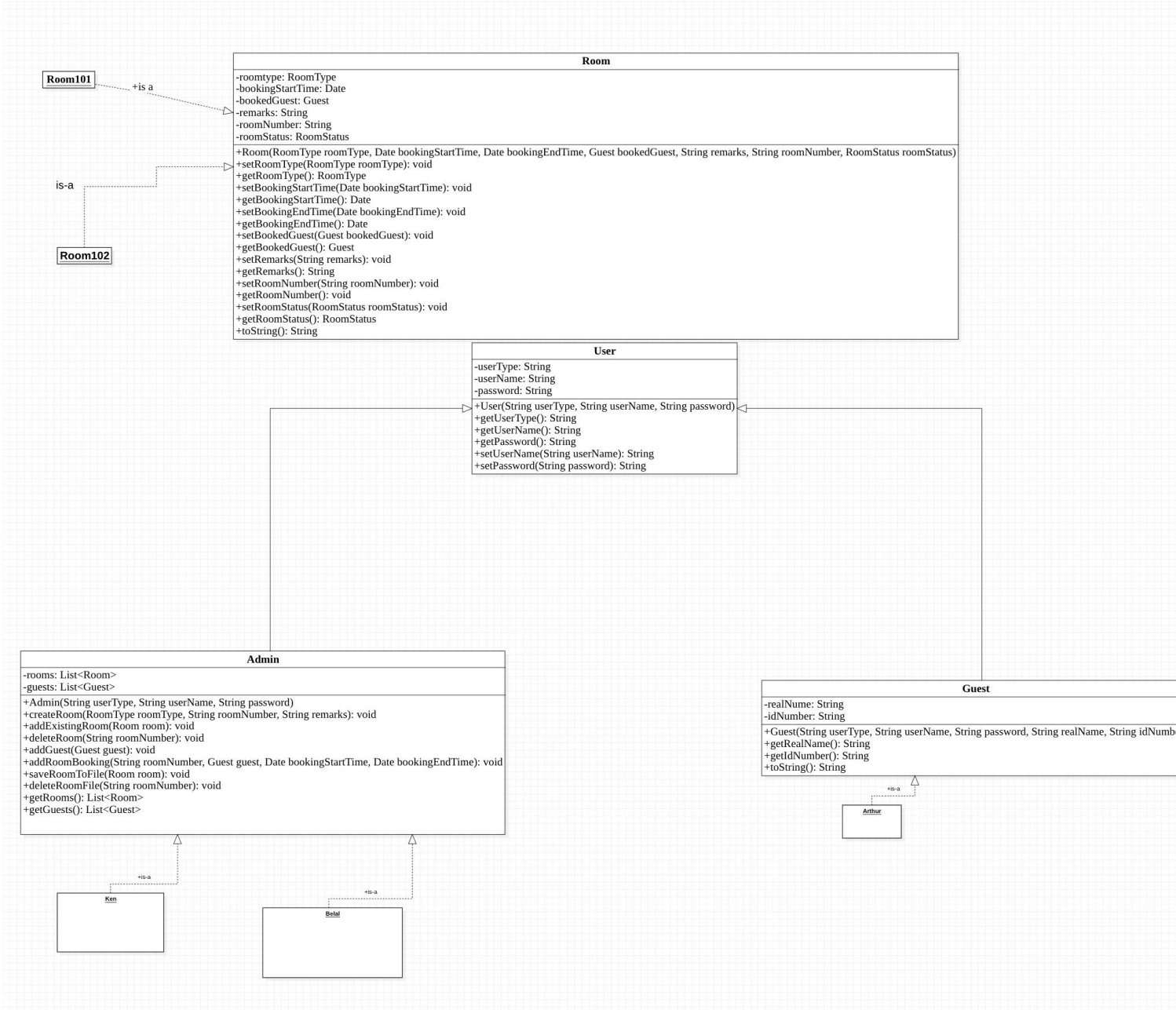
```
            if (updated) {
                saveUsersToFile();
                JOptionPane.showMessageDialog(this, "Password updated successfully!");
                this.setVisible(false);
                new HotelManagementSystem(users).setVisible(true);
            } else {
                JOptionPane.showMessageDialog(this, "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



Main
+HotelManagementFrame(): void
+UserLoader.loadUser
+RoomLoader.Loaduser
+HotelManagementSystem User
UserLoader
oadUser(String filePath): List<User>
RoomLoader
+loadRooms(): List<Room>
-loadRoomFromFile(File file): Room
+parseLine(String line): String
+hotelManagementSystem(List users): void
HotelManagementSystem
-users: List<User>
+rooms: List<Room>
+hotelManagementSystem(List users): void
+authenticate(String username, String password): boolean
+getUserByUsername(String username): User
+Register
+Forget Password
+LoginIsAInstanceOfAdmin
GuestRegistrationFrame
-users: List<User>
+GuestRegistrationFrame(List users): void
+isUsernameTaken(String username): boolean
+isPasswordValid(String password): boolean
+saveUserToFile(Guest guest): void
ForgetPasswordFrame
+users: List<User>
+ForgotPasswordFrame(List users): void
-isPasswordValid(String password): boolean
-saveUsersToFile(): void
AdminDashboardFrame
-rooms: List<Room>
-users: List<User>
-admin: Admin
+AdminDashboardFrame(Admin admin, List rooms, List users): void
-manageRooms(): void
-manageUsers(): void
+bookRoom(): void
-logout(): void
+RoomManagement
+Room Booking
+UserManagement
RoomManagementFrame
-admin: Admin
-rooms: List<Room>
-users: List<User>
+RoomManagementFrame(Admin admin, List rooms, List users)
+showAddRoomDialog(): void
+showDeleteRoomDialog(): void
BookingRoomFrame
+admin: Admin
+room: List<Room>
+users: List<User>
+BookingRoomFrame(Admin admin, List rooms, List users): void
+showBookingDialog(Room room): void
UserManagementFrame
-admin: Admin
-users: List<User>
+UserManagementFrame(Admin admin, List users)
+showAddUserDialog(): void
+showManageUsersDialog(): void
+showEditUserDialog(User user): void
+isPasswordValid(String password): boolean
+saveUserToFile(User user): void
+saveUsersToFile(): void

b) UML Code Generation

```java
import java.util.*;

/**
 *
 */
public class Admin extends User {

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

    /**
     *
     */
    private List<Room> rooms;

    /**
     *
     */
    private List<Guest> guests;

    /**
     * @param String userType
     * @param String userName
     * @param String password
     */
    public void Admin(void String userType, void String userName, void String password) {
        // TODO implement here
    }

    /**
     * @param RoomType roomType
     * @param String roomNumber
     * @param String remarks
     * @return
     */
    public void createRoom(void RoomType roomType, void String roomNumber, void String remarks) {
        // TODO implement here
        return null;
    }

    /**
     * @param Room room
     * @return
```

```java
    /**
     * @param Room room
     * @return
     */
    public void addExistingRoom(void Room room) {
        // TODO implement here
        return null;
    }

    /**
     * @param String roomNumber
     * @return
     */
    public void deleteRoom(void String roomNumber) {
        // TODO implement here
        return null;
    }

    /**
     * @param Guest guest
     * @return
     */
    public void addGuest(void Guest guest) {
        // TODO implement here
        return null;
    }

    /**
     * @param String roomNumber
     * @param Guest guest
     * @param Date bookingStartTime
     * @param Date bookingEndTime
     * @return
     */
    public void addRoomBooking(void String roomNumber, void Guest guest, void Date bookingStartTime, void Date bookingEndTime) {
        // TODO implement here
        return null;
    }

    /**
     * @param Room room
     * @return
     */
    public void saveRoomToFile(void Room room) {
        // TODO implement here
```

```
    public void saveRoomToFile(void Room room) {
        // TODO implement here
        return null;
    }

    /**
     * @param String roomNumber
     * @return
     */
    public void deleteRoomFile(void String roomNumber) {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public List<Room> getRooms() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public List<Guest> getGuests() {
        // TODO implement here
        return null;
    }

}
```

```
import java.util.*;

/**
 *
 */
public class AdminDashboardFrame {

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

    /**
     *
     */
    private List<Room> rooms;

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

    /**
     *
     */
    private Admin admin;

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

    /**
     * @return
     */
    private void manageRooms() {
        // TODO implement here
        return null;
    }
```

```
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;
    }
```

```
     * @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;
    }

    /**
     * @param Room room
     * @return
     */
    public void showBookingDialog(void Room room) {
        // TODO implement here
        return null;
    }

}
```

```
import java.util.*;

/**
 *
 */
public class ForgetPasswordFrame {

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

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

    /**
     * @param List users
     * @return
     */
    public void ForgotPasswordFrame(void List users) {
        // TODO implement here
        return null;
    }

    /**
     * @param String password
     * @return
     */
    private boolean isPasswordValid(void String password) {
        // TODO implement here
        return false;
    }

    /**
     * @return
     */
    private void saveUsersToFile() {
        // TODO implement here
        return null;
    }

}
```

```
     */
    public String getIdNumber() {
        // TODO implement here
        return "";
    }

    /**
     * @return
     */
    public String toString() {
        // TODO implement here
        return "";
    }

}
```

```
import java.util.*;

/**
 *
 */
public class GuestRegistrationFrane {

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

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

    /**
     * @param List users
     * @return
     */
    public void GuestRegistrationFrame(void List users) {
        // TODO implement here
        return null;
    }

    /**
     * @param String username
     * @return
     */
    public boolean isUsernameTaken(void String username) {
        // TODO implement here
        return false;
    }

    /**
     * @param String password
     * @return
     */
    public boolean isPasswordValid(void String password) {
        // TODO implement here
        return false;
    }

     * @param Guest guest
     * @return
     */
    public void saveUserToFile(void Guest guest) {
        // TODO implement here
        return null;
    }

}
```

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

```
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  }
```

```java
import java.util.*;

/**
 *
 */
public class Room {

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

    /**
     *
     */
    private RoomType roomtype;

    /**
     *
     */
    private Date bookingStartTime;

    /**
     *
     */
    private Guest bookedGuest;

    /**
     *
     */
    private String remarks;

    /**
     *
     */
    private String roomNumber;

    /**
     *
     */
    private RoomStatus roomStatus;

    /**
     * @param RoomType roomType
     * @param Date bookingStartTime
     * @param Date bookingEndTime
     * @param Guest bookedGuest
     * @param String remarks
     * @param String roomNumber
     * @param RoomStatus roomStatus
     */
    public void Room(void RoomType roomType, void Date bookingStartTime, void Date bookingEndTime, void Guest bookedGuest, void String remarks, void String roomNumber,
        // TODO implement here
    }

    /**
     * @param RoomType roomType
     * @return
     */
    public void setRoomType(void RoomType roomType) {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public RoomType getRoomType() {
        // TODO implement here
        return null;
    }

    /**
     * @param Date bookingStartTime
     * @return
     */
    public void setBookingStartTime(void Date bookingStartTime) {
        // TODO implement here
        return null;
    }

    /**
```

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

```
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     }
```

```
 *
 */
public class RoomLoader {

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

    /**
     * @return
     */
    public List<Room> loadRooms() {
        // TODO implement here
        return null;
    }

    /**
     * @param File file
     * @return
     */
    private Room loadRoomFromFile(void File file) {
        // TODO implement here
        return null;
    }

    /**
     * @param String line
     * @return
     */
    public String parseLine(void String line) {
        // TODO implement here
        return "";
    }

    /**
     * @param List users
     * @return
     */
    public void hotelManagementSystem(void List users) {
        // TODO implement here
        return null;
    }

}
```

```java
import java.util.*;

/**
 *
 */
public class RoomManagementFrame {

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

    /**
     *
     */
    private Admin admin;

    /**
     *
     */
    private List<Room> rooms;

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

    /**
     * @param Admin admin
     * @param List rooms
     * @param List users
     */
    public void RoomManagementFrame(void Admin admin, void List rooms, void List users) {
        // TODO implement here
    }

    /**
     * @return
     */
    public void showAddRoomDialog() {
        // TODO implement here
        return null;
    }
```

```java
    }

    /**
     * @return
     */
    public void showDeleteRoomDialog() {
        // TODO implement here
        return null;
    }

    /**
     *
     */
    public void Operation3() {
        // TODO implement here
    }

    /**
     *
     */
    public void Operation4() {
        // TODO implement here
    }

}
```

```
 */
public class User {

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

    /**
     *
     */
    private String userType;

    /**
     *
     */
    private String userName;

    /**
     *
     */
    private String password;

    /**
     * @param String userType
     * @param String userName
     * @param String password
     */
    public void User(void String userType, void String userName, void String password) {
        // TODO implement here
    }

    /**
     * @return
     */
    public String getUserType() {
        // TODO implement here
        return "";
    }

    /**
     * @return
     */
    public String getUserName() {
```

```
    public String getUserName() {
        // TODO implement here
        return "";
    }

    /**
     * @return
     */
    public String getPassword() {
        // TODO implement here
        return "";
    }

    /**
     * @param String userName
     * @return
     */
    public String setUserName(void String userName) {
        // TODO implement here
        return "";
    }

    /**
     * @param String password
     * @return
     */
    public String setPassword(void String password) {
        // TODO implement here
        return "";
    }

}
```

```java
import java.util.*;

/**
 *
 */
public class UserLoader {

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

    /**
     * @param String filePath
     * @return
     */
    public List<User> loadUser(void String filePath) {
        // TODO implement here
        return null;
    }

}
```

```java
import java.util.*;

/**
 *
 */
public class UserManagementFrame {

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

    /**
     *
     */
    private Admin admin;

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

    /**
     * @param Admin admin
     * @param List users
     */
    public void UserManagementFrame(void Admin admin, void List users) {
        // TODO implement here
    }

    /**
     * @return
     */
    public void showAddUserDialog() {
        // TODO implement here
        return null;
    }

    /**
     * @return
     */
    public void showManageUsersDialog() {
        // TODO implement here
        return null;
    }
```

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

```
        JLabel userLabel = new JLabel("Username:");
        JTextField userText = new JTextField(20);
        panel.add(userLabel);
        panel.add(userText);


        JLabel passwordLabel = new JLabel("Password:");
        JPasswordField passwordText = new JPasswordField(20);
        panel.add(passwordLabel);
        panel.add(passwordText);


        add(panel, BorderLayout.CENTER);


        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout());


        JButton loginButton = new JButton("Login");
        buttonPanel.add(loginButton);
        loginButton.addActionListener(e -> {
            String username = userText.getText();
            String password = new String(passwordText.getPassword());

           boolean authenticated = authenticate(username, password);
           if (authenticated) {
               User loggedInUser = getUserByUsername(username);
               if (loggedInUser instanceof Admin) {
                   this.setVisible(false);
                   new AdminDashboardFrame((Admin) loggedInUser, rooms, users).setVisible(true);
               }
               else {
                   JOptionPane.showMessageDialog(null, "Login successful");

               }
           }

           else {
               JOptionPane.showMessageDialog(null, "Invalid username or password");
           }
       });



       JButton registerButton = new JButton("Register");
       buttonPanel.add(registerButton);
       registerButton.addActionListener(e -> {
           this.setVisible(false);
           new GuestRegistrationFrame(users).setVisible(true);
       });

```

```
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
00
01      private boolean authenticate(String username, String password) {
02          for (User user : users) {
03              if (user.getUsername().equals(username) && user.getPassword().equals(password)) {
04                  return true;
05              }
06          }
07          return false;
08      }
09
10      private User getUserByUsername(String username) {
11          for (User user : users) {
12              if (user.getUsername().equals(username)) {
13                  return user;
14              }
15          }
16          return null;
17      }
18
19
20  }
21
 1
 2
 3  import javax.swing.*;
10
11  public class GuestRegistrationFrame extends JFrame {
12      private List<User> users;
13
14      public GuestRegistrationFrame(List<User> users) {
15          this.users = users;
16
17          // Set title
18          setTitle("Guest Registration");
19
20          // Set window size
21          setSize(400, 400);
22
23          // Set the default shutdown operation
24          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25
26          // Set window layout
27          setLayout(new GridLayout(6, 2, 10, 10));
28
29          // Create form labels and text fields
30          JLabel userTypeLabel = new JLabel("User Type:");
31          JTextField userTypeText = new JTextField("Guest");
32          userTypeText.setEnabled(false); // The userType cannot be changed.
33          add(userTypeLabel);
34          add(userTypeText);
35
36          JLabel usernameLabel = new JLabel("Username:");
37          JTextField usernameText = new JTextField();
38          add(usernameLabel);
39          add(usernameText);
40
```

```
        JLabel passwordLabel = new JLabel("Password:");
        JPasswordField passwordText = new JPasswordField();
        add(passwordLabel);
        add(passwordText);

        JLabel realNameLabel = new JLabel("Real Name:");
        JTextField realNameText = new JTextField();
        add(realNameLabel);
        add(realNameText);

        JLabel idNumberLabel = new JLabel("ID Number:");
        JTextField idNumberText = new JTextField();
        add(idNumberLabel);
        add(idNumberText);


        JButton registerButton = new JButton("Register");
        registerButton.addActionListener(e -> {
            String username = usernameText.getText();
            String password = new String(passwordText.getPassword());
            String realName = realNameText.getText();
            String idNumber = idNumberText.getText();

            if (isUsernameTaken(username)) {
                JOptionPane.showMessageDialog(this, "Username is already taken.");
                return;
            }

            if (isPasswordValid(password)) {
                Guest newGuest = new Guest("Guest", username, password, realName, idNumber);
                users.add(newGuest);
                saveUserToFile(newGuest);
                JOptionPane.showMessageDialog(this, "Registration successful!");
                this.setVisible(false);
                new HotelManagementSystem(users).setVisible(true);
            }
                JOptionPane.showMessageDialog(this, "Password must be at least 6 characters long and contain both letters and numbers.");
            }
        });
        add(registerButton);
    }

    private boolean isUsernameTaken(String username) {
        for (User user : users) {
            if (user.getUsername().equals(username)) {
                return true;
            }
        }
        return false;
    }

    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 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,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealName(), guest.getIdNum
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```java
package Login_System;


import javax.swing.*;

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

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


        setTitle("Forgot Password");


        setSize(400, 300);


        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        setLayout(new GridLayout(5, 2, 10, 10));


        JLabel usernameLabel = new JLabel("Username:");
        JTextField usernameText = new JTextField();
        add(usernameLabel);
        add(usernameText);

        JLabel realNameLabel = new JLabel("Real Name:");
        JTextField realNameText = new JTextField();
        add(realNameLabel);
        add(realNameText);

        JTextField idNumberText = new JTextField();
        add(idNumberLabel);
        add(idNumberText);

        JLabel newPasswordLabel = new JLabel("New Password:");
        JPasswordField newPasswordText = new JPasswordField();
        add(newPasswordLabel);
        add(newPasswordText);


        JButton submitButton = new JButton("Submit");
        submitButton.addActionListener(e -> {
            String username = usernameText.getText();
            String realName = realNameText.getText();
            String idNumber = idNumberText.getText();
            String newPassword = new String(newPasswordText.getPassword());

            if (!isPasswordValid(newPassword)) {
                JOptionPane.showMessageDialog(this, "Password must be at least 6 characters long and contain both letters and numbers.");
                return;
            }

            boolean updated = false;
            for (User user : users) {
                if (user instanceof Guest) {
                    Guest guest = (Guest) user;
                    if (guest.getUsername().equals(username) && guest.getRealName().equals(realName) && guest.getIdNumber().equals(idNumber))
                        guest.setPassword(newPassword);
                        updated = true;
                        break;
                    }
                }
            }
```

```
            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,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealName(), g
            } 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);
```

```
        JButton bookRoomButton = new JButton("Room Booking");
        bookRoomButton.addActionListener(e -> bookRoom());
        add(bookRoomButton);


        JButton logoutButton = new JButton("Log out");
        logoutButton.addActionListener(e -> logout());
        add(logoutButton);
    }

    private void manageRooms() {
        this.setVisible(false);
        new RoomManagementFrame(admin, rooms, users).setVisible(true);
    }

    private void manageUsers() {
        this.setVisible(false);
        new UserManagementFrame(admin, users).setVisible(true);
    }

    private void bookRoom() {
        this.setVisible(false);
        new BookingRoomFrame(admin, rooms, users).setVisible(true);
    }

    private void logout() {
        this.setVisible(false);
        new HotelManagementSystem(users).setVisible(true);
    }
}
```

```
import javax.swing.*;

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

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


        setTitle("Room Management");


        setSize(400, 300);


        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        setLayout(new GridLayout(3, 1, 10, 10));


        JButton addRoomButton = new JButton("Add Room");
        addRoomButton.addActionListener(e -> showAddRoomDialog());
        add(addRoomButton);


```

```
        deleteRoomButton.addActionListener(e -> showDeleteRoomDialog());
        add(deleteRoomButton);


        JButton returnButton = new JButton("Return");
        returnButton.addActionListener(e -> {
            this.setVisible(false);
            new AdminDashboardFrame(admin, rooms, users).setVisible(true);
        });
        add(returnButton);
    }

    private void showAddRoomDialog() {
        JDialog dialog = new JDialog(this, "Add Room", true);
        dialog.setSize(300, 200);
        dialog.setLayout(new GridLayout(3, 2, 10, 10));

        JLabel roomNumberLabel = new JLabel("Room Number:");
        JTextField roomNumberText = new JTextField();
        dialog.add(roomNumberLabel);
        dialog.add(roomNumberText);

        JLabel roomTypeLabel = new JLabel("Room Type:");
        JComboBox<RoomType> roomTypeComboBox = new JComboBox<>(RoomType.values());
        dialog.add(roomTypeLabel);
        dialog.add(roomTypeComboBox);

        JButton confirmButton = new JButton("Confirm");
        confirmButton.addActionListener(e -> {
            String roomNumber = roomNumberText.getText();
            RoomType roomType = (RoomType) roomTypeComboBox.getSelectedItem();
            admin.createRoom(roomType, roomNumber, ""); // 可根据需要添加备注
            rooms.add(new Room(roomType, null, null, null, "", roomNumber, Room.RoomStatus.vacantRoom));
            dialog.dispose();
            JOptionPane.showMessageDialog(this, "Room created successfully");
        });
        dialog.add(confirmButton);

        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(e -> dialog.dispose());
        dialog.add(cancelButton);

        dialog.setVisible(true);
    }

    private void showDeleteRoomDialog() {
        JDialog dialog = new JDialog(this, "Delete Room", true);
        dialog.setSize(300, 200);
        dialog.setLayout(new GridLayout(rooms.size() + 1, 1, 10, 10));

        for (Room room : rooms) {
            JButton roomButton = new JButton(room.getRoomNumber());
            roomButton.addActionListener(e -> {
                int response = JOptionPane.showConfirmDialog(this, "Confirm room deletion " + room.getRoomNumber() + " ? ", "Confirm",
                if (response == JOptionPane.YES_OPTION) {
                    admin.deleteRoom(room.getRoomNumber());
                    rooms.remove(room);
                    JOptionPane.showMessageDialog(this, "Room deletes successfully");
                    dialog.dispose();
                }
            });
            dialog.add(roomButton);
        }

        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(e -> dialog.dispose());
        dialog.add(cancelButton);
```

```java
import javax.swing.*;

public class UserManagementFrame extends JFrame {
    private Admin admin;
    private List<User> users;

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

        setTitle("User Management");

        setSize(400, 400);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new BorderLayout());

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 1, 10, 10));

        JButton addUserButton = new JButton("Add User");
        addUserButton.addActionListener(e -> showAddUserDialog());
        panel.add(addUserButton);

        JButton manageUsersButton = new JButton("Modify/Delete User");
        manageUsersButton.addActionListener(e -> showManageUsersDialog());
        panel.add(manageUsersButton);

        returnButton.addActionListener(e -> {
            this.setVisible(false);
            new AdminDashboardFrame(admin, null, users).setVisible(true);
        });
        panel.add(returnButton);

        add(panel, BorderLayout.CENTER);
    }

    private void showAddUserDialog() {
        JDialog dialog = new JDialog(this, "Add User", true);
        dialog.setSize(300, 300);
        dialog.setLayout(new GridLayout(6, 2, 10, 10));

        JLabel userTypeLabel = new JLabel("User Type:");
        JComboBox<String> userTypeComboBox = new JComboBox<>(new String[] {"Admin", "Guest"});
        dialog.add(userTypeLabel);
        dialog.add(userTypeComboBox);

        JLabel usernameLabel = new JLabel("Username:");
        JTextField usernameText = new JTextField();
        dialog.add(usernameLabel);
        dialog.add(usernameText);

        JLabel passwordLabel = new JLabel("Password:");
        JPasswordField passwordText = new JPasswordField();
        dialog.add(passwordLabel);
        dialog.add(passwordText);

        JLabel realNameLabel = new JLabel("Real Name:");
        JTextField realNameText = new JTextField();
        dialog.add(realNameLabel);
        dialog.add(realNameText);
```

```
        JLabel idNumberLabel = new JLabel("ID Number:");
        JTextField idNumberText = new JTextField();
        dialog.add(idNumberLabel);
        dialog.add(idNumberText);

        userTypeComboBox.addActionListener(e -> {
            if (userTypeComboBox.getSelectedItem().equals("Admin")) {
                realNameText.setEnabled(false);
                idNumberText.setEnabled(false);
            } else {
                realNameText.setEnabled(true);
                idNumberText.setEnabled(true);
            }
        });

        JButton confirmButton = new JButton("Confirm");
        confirmButton.addActionListener(e -> {
            String userType = (String) userTypeComboBox.getSelectedItem();
            String username = usernameText.getText();
            String password = new String(passwordText.getPassword());
            String realName = realNameText.getText();
            String idNumber = idNumberText.getText();

            if (isPasswordValid(password)) {
                User newUser;
                if ("Admin".equals(userType)) {
                    newUser = new Admin(userType, username, password);
                } else {
                    newUser = new Guest(userType, username, password, realName, idNumber);
                }
                users.add(newUser);
                saveUserToFile(newUser);
                dialog.dispose();
                JOptionPane.showMessageDialog(this, "User created successfully");
            } else {
                JOptionPane.showMessageDialog(this, "Passwords must be at least 6 characters long and contain letters and numbers.");
            }
        });
        dialog.add(confirmButton);

        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(e -> dialog.dispose());
        dialog.add(cancelButton);

        dialog.setVisible(true);
    }

    private void showManageUsersDialog() {
        JDialog dialog = new JDialog(this, "Modify/Delete User", true);
        dialog.setSize(300, 400);
        dialog.setLayout(new GridLayout(users.size() + 1, 1, 10, 10));

        for (User user : users) {
            JButton userButton = new JButton(user instanceof Guest ? ((Guest) user).getRealName() : user.getUsername());
            userButton.addActionListener(e -> showEditUserDialog(user));
            dialog.add(userButton);
        }

        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(e -> dialog.dispose());
        dialog.add(cancelButton);

        dialog.setVisible(true);
    }

    private void showEditUserDialog(User user) {
        JDialog dialog = new JDialog(this, "edit user", true);
        dialog.setSize(300, 200);
        dialog.setLayout(new GridLayout(4, 2, 10, 10));
```

```
    private void showEditUserDialog(User user) {
        JDialog dialog = new JDialog(this, "edit user", true);
        dialog.setSize(300, 200);
        dialog.setLayout(new GridLayout(4, 2, 10, 10));

        JLabel usernameLabel = new JLabel("Username:");
        JTextField usernameText = new JTextField(user.getUsername());
        dialog.add(usernameLabel);
        dialog.add(usernameText);

        JLabel passwordLabel = new JLabel("New Password:");
        JPasswordField passwordText = new JPasswordField();
        dialog.add(passwordLabel);
        dialog.add(passwordText);

        JButton confirmButton = new JButton("Confirm");
        confirmButton.addActionListener(e -> {
            String username = usernameText.getText();
            String password = new String(passwordText.getPassword());

            if (isPasswordValid(password)) {
                user.setUsername(username);
                user.setPassword(password);
                saveUsersToFile();
                dialog.dispose();
                JOptionPane.showMessageDialog(this, "The user information is updated successfully.");
            } else {
                JOptionPane.showMessageDialog(this, "Passwords must be at least 6 characters long and contain letters and numbers.");
            }
        });
        dialog.add(confirmButton);

        JButton deleteButton = new JButton("Delete User");
        JButton deleteButton = new JButton("Delete User");
        deleteButton.addActionListener(e -> {
            int response = JOptionPane.showConfirmDialog(this, "Confirm deletion of user " + user.getUsername() + "? ", "Confirm", JOpti
            if (response == JOptionPane.YES_OPTION) {
                users.remove(user);
                saveUsersToFile();
                dialog.dispose();
                JOptionPane.showMessageDialog(this, "User deletes successfully");
            }
        });
        dialog.add(deleteButton);

        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(e -> dialog.dispose());
        dialog.add(cancelButton);

        dialog.setVisible(true);
    }

    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 saveUserToFile(User user) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.tx
            if (user instanceof Guest) {
                Guest guest = (Guest) user;
                bw.write(String.format("%s,%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getRealN
            } else if (user instanceof Admin) {
                bw.write(String.format("%s,%s,%s%n", user.getUserType(), user.getUsername(), user.getPassword()));
```

```
198     private void saveUserToFile(User user) {
199         try (BufferedWriter bw = new BufferedWriter(new FileWriter("/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.t
200             if (user instanceof Guest) {
201                 Guest guest = (Guest) user;
202                 bw.write(String.format("%s,%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getReal
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.t
213             for (User user : users) {
214                 if (user instanceof Guest) {
215                     Guest guest = (Guest) user;
216                     bw.write(String.format("%s,%s,%s,%s,%s%n", guest.getUserType(), guest.getUsername(), guest.getPassword(), guest.getR
217                 }
218                 else if (user instanceof Admin) {
219                     bw.write(String.format("%s,%s,%s%n", user.getUserType(), user.getUsername(), user.getPassword()));
220                 }
221             }
222         } catch (IOException e) {
223             e.printStackTrace();
224         }
225     }
226 }
227
```

```
2
3  import javax.swing.*;
9
10 public class BookingRoomFrame extends JFrame {
11     private Admin admin;
12     private List<Room> rooms;
13     private List<User> users;
14     private static final SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
15
16     public BookingRoomFrame(Admin admin, List<Room> rooms, List<User> users) {
17         this.admin = admin;
18         this.rooms = rooms;
19         this.users = users;
20
21         setTitle("Booking Management");
22
23         setSize(600, 600);
24
25         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26
27         setLayout(new BorderLayout());
28
29         JPanel availableRoomsPanel = new JPanel();
30         availableRoomsPanel.setLayout(new GridLayout(rooms.size(), 1, 10, 10));
31
32         JPanel bookedRoomsPanel = new JPanel();
33         bookedRoomsPanel.setLayout(new GridLayout(rooms.size(), 1, 10, 10));
34
35         JLabel availableRoomsLabel = new JLabel("Available Rooms", SwingConstants.CENTER);
36         availableRoomsLabel.setFont(new Font("Serif", Font.BOLD, 20));
37         availableRoomsPanel.add(availableRoomsLabel);
38
```

```
    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 bookingStartDateLabel = new JLabel("Booking Start Date (yyyy-MM-dd):");
        JTextField bookingStartDateText = new JTextField();
        dialog.add(bookingStartDateLabel);
        dialog.add(bookingStartDateText);

        JLabel bookingEndDateLabel = new JLabel("Booking End Date (yyyy-MM-dd):");
        JTextField bookingEndDateText = new JTextField();
        dialog.add(bookingEndDateLabel);
        dialog.add(bookingEndDateText);

        JButton confirmButton = new JButton("Confirm");
        confirmButton.addActionListener(e -> {
            String guestName = guestNameText.getText();
            String guestId = guestIdText.getText();
            String bookingStartDateStr = bookingStartDateText.getText();
            String bookingEndDateStr = bookingEndDateText.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);


rivate 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:");
```

```
            try {
                Date bookingStartDate = dateFormat.parse(bookingStartDateStr);
                Date bookingEndDate = dateFormat.parse(bookingEndDateStr);

                Guest guest = new Guest("Guest", guestName, "", guestName, guestId);
                admin.addGuest(guest);
                admin.addRoomBooking(room.getRoomNumber(), guest, bookingStartDate, bookingEndDate);
                dialog.dispose();
                JOptionPane.showMessageDialog(this, "Room booked successfully!");
                this.setVisible(false);
                new BookingRoomFrame(admin, rooms, users).setVisible(true);
            } catch (ParseException ex) {
                JOptionPane.showMessageDialog(this, "Invalid date format. Please use yyyy-MM-dd.");
            }
        });
        dialog.add(confirmButton);

        JButton cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(e -> dialog.dispose());
        dialog.add(cancelButton);

        dialog.setVisible(true);
    }
}
```

```
package Login_System;

public class User {
    private String userType;
    private String username;
    private String password;

    public User(String userType, String username, String password) {
        this.userType = userType;
        this.username = username;
        this.password = password;
    }

    public String getUserType() {
        return userType;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

```java
import Login_System.User;

public class Admin extends User {
    private List<Room> rooms;
    private List<Guest> guests;
    private static final String ROOM_FOLDER_PATH = "/Users/matsumatsu/eclipse-workspace/testing/src/Login_System/User.txt";
    private static final SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

    public Admin(String userType, String username, String password) {
        super(userType, username, password);
        this.rooms = new ArrayList<>();
        this.guests = new ArrayList<>();
    }

    public void createRoom(RoomType roomType, String roomNumber, String remarks) {
        Room room = new Room(roomType, null, null, null, remarks, roomNumber, Room.RoomStatus.vacantRoom);
        rooms.add(room);
        saveRoomToFile(room);
    }

    public void addExistingRoom(Room room) {
        rooms.add(room);
        saveRoomToFile(room);
    }

    public void deleteRoom(String roomNumber) {
        rooms.removeIf(room -> room.getRoomNumber().equals(roomNumber));
        deleteRoomFile(roomNumber);
    }

    public void addGuest(Guest guest) {
        guests.add(guest);
    }

    public void addRoomBooking(String roomNumber, Guest guest, Date bookingStartTime, Date bookingEndTime) {
        for (Room room : rooms) {
            if (room.getRoomNumber().equals(roomNumber)) {
                room.setBookedGuest(guest);
                room.setBookingStartTime(bookingStartTime);
                room.setBookingEndTime(bookingEndTime);
                room.setRoomStatus(Room.RoomStatus.Booking);
                saveRoomToFile(room);
                break;
            }
        }
    }

    private void saveRoomToFile(Room room) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(ROOM_FOLDER_PATH + room.getRoomNumber() + ".txt"))) {
            bw.write("Room Type: " + room.getRoomType() + "\n");
            bw.write("Room Status: " + room.getRoomStatus() + "\n");
            bw.write("Remarks: " + room.getRemarks() + "\n");
            if (room.getRoomStatus() == Room.RoomStatus.Booking || room.getRoomStatus() == Room.RoomStatus.CheckingIn) {
                bw.write("Booked Guest: " + room.getBookedGuest().getUsername() + "\n");
                bw.write("Booking Start Time: " + dateFormat.format(room.getBookingStartTime()) + "\n");
                bw.write("Booking End Time: " + dateFormat.format(room.getBookingEndTime()) + "\n");
            } else {
                bw.write("Booked Guest: null\n");
                bw.write("Booking Start Time: null\n");
                bw.write("Booking End Time: null\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void deleteRoomFile(String roomNumber) {
        java.io.File file = new java.io.File(ROOM_FOLDER_PATH + roomNumber + ".txt");
        if (file.exists()) {
            file.delete();
        }
    }

    public List<Room> getRooms() {
        return rooms;
    }

    public List<Guest> getGuests() {
        return guests;
    }
}
```

```java
package Login_System;

public class Guest extends User {
    private String realName;
    private String idNumber;

    public Guest(String userType, String username, String password, String realName, String idNumber) {
        super(userType, username, password);
        this.realName = realName;
        this.idNumber = idNumber;
    }

    public String getRealName() {
        return realName;
    }

    public String getIdNumber() {
        return idNumber;
    }

    @Override
    public String toString() {
        return "Guest{" +
                "userType='" + getUserType() + '\'' +
                ", username='" + getUsername() + '\'' +
                ", realName='" + realName + '\'' +
                ", idNumber='" + idNumber + '\'' +
                '}';
    }
}
```

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 excute 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 fram and then jump into login fram. 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.