Pelin Zeynep Kaya

220303008

Computer Engineering

Algorithms & Programming II

Purpose

This project is made for a hypothetical hotel to provide the employees and the general manager with the power to manage the rooms and employees with a user-friendly interface.

System Overview

The purpose is to provide a system to control operations as receptionist, manager, and general manager. Receptionist can view all the rooms, do check-in and check-out operations, managers can add new rooms and update the room information and general manager can add new employees i.e. receptionist and manager.

This program is designed to be a desktop program and works with files to store data.

Design Objectives

All changes made using this system is updated for all employees.

Receptionist

Receptionist can view the rooms, check in and check out guests.

Manager

Manager can view the rooms, add rooms, and update room information.

General Manager

There is only one general manager. General manager can add receptionists and managers.

Design Overview

This program uses Object-Oriented Programming (OOP) principles. Files are used for data storage. For creating graphical user interface, Java Swing and IntelliJ was used.

Environment Overview

This program is designed to be a desktop application.

System Architecture

Login Page

Receptionist Login

Manager Login

General Manager Login

The system compromise three main user type.

Users are not allowed to create new accounts as the program does not include a register page. The only way to get a username and password is by the general manager adding a new employee i.e. receptionist and manager. Password is generated randomly and assigned to the users.

File Handling

The File handling in the application is very sophisticated but efficient and the Files communicate with each other flawlessly. The available files inside the application include the following: rooms.txt, checkInRooms.txt, takenRooms.txt, employees.txt and finally the customers.txt. The text files related to rooms store the information of the rooms which then get loaded into their corresponding table models when the application starts, furthermore the customers.txt stores all the residents' information when they check in. The employee.txt is used in two key parts of the application, firstly it is used to check if the information inputted by the user is correct in the LoginPage class, secondly it is used by the General Manager Class to add employees and remove them from the txt file. Most of the communication with the files is done through methods and in the methods using buffered readers and writers.

**Classes**

Employee

The abstract Employee Class is used primarily for overwriting its' toString function, this class has two inheritors(Management,Reception) these two classes overwrite and define the abstract toString method inside the Employee Class which is used to write to files in the correct format for each inheritor.

General Manager

The General manager class utilizes the Employee Class and its inheritors (Reception, Management) to add and remove managers and receptionists to the application. This is done by creating objects of Reception and or Management by using their overwritten toString method to write to the files. The General manager has a special method for creating passwords for its' employees called generateSecurePassword. This method creates a randomly generated password using all the characters and digits available on the keyboard. This password is then assigned to the newly created object inside the addManager method which then gets used to write to the Employee file.

Manager

The Manager Class is one of the Classes that uses GUI to create an interface for the hypothetical manager. Inside this class we have several methods that help the manager manage the rooms by adding, removing, and updating the rooms. The Manager GUI provides the manager with a simple and understandable interface that helps the Manager navigate through the application. The Manager has the ability to add, remove and update rooms as they please. By using the updateRoom method the manager can update the already existing rooms in the hotel by changing their status and price and even their type. Adding rooms is a simple operation for the manager, by clicking on the addRoom button, the manager is provided with combo boxes and text fields to add rooms. Removing rooms is another simple operation that the manager can easily do by choosing and selecting a room from the provided table and clicking the remove button.

Receptionist

The Receptionist is the most important and theoretically the most used and sophisticated part of the application. The purpose of the receptionist class is to check in and check out customers using the tables provided to the user. By using the checkIn and checkOut methods created in the receptionist class the user can easily check in customers and checkout customers at any given time. The user interface for the Receptionist is very user friendly and easy to use. The user can see the available rooms inside a table to see which rooms are occupied. They can use this information to make their next decision on what to do in the application. The Receptionist has the ability to check in customers by inputting their information into the text fields provided to them. After the check-in operation is done using the checkIn method the customers info and the room that's been occupied will be added to another table in a different Panel which the receptionist can access to easily check out the residents out of the hotel.

LoginPage

In loginPage class, login operations are handled. Login can be done as general manager, manager, and receptionist. By entering a correct username and password, and choosing the correct role, the user will be directed to the page of the specific role. Username and password are saved in the “employee.txt” file.

Main

The main class was created purely for running the application and testing. The main class starts by creating a login object by reference. The constructor for the login object created by reference is executed on EDT (Event Dispatch Thread). The rest of the program runs from there by the user.

UML DIAGRAMS

metin, çizgi, yazı tipi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu