King Saud University

College of Computer and Information Sciences

Information Technology Department

CSC 113: Computer Programming

Second Semester 1445

Hotel Management System

Phase 2

Prepared by

Student name	ID	Work division
Leader: Arwa Almutairi	444201055	Writing the code Code Review and Testing Editing and proofreading the report Updating UML Diagram
Ruba Alshammari	444200470	Writing the code Code Review and Testing Editing and proofreading the report
Gadah Alqahtani	444200371	Writing the code Code Review and Testing Debugging the code

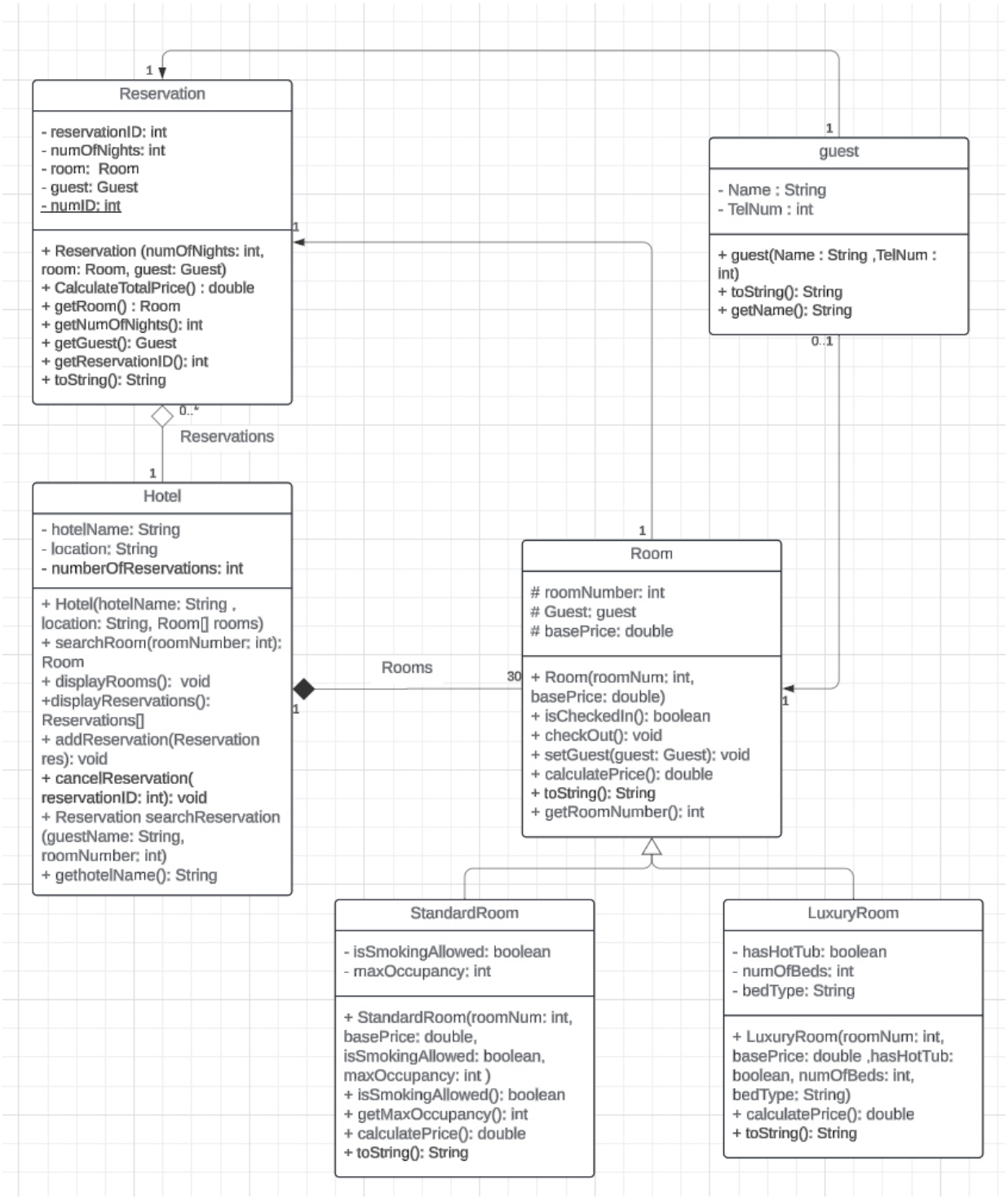
Supervised By: Amani Alahmadi

Introduction:

Our Hotel Management System is an essential Java program for hotel staff, playing a vital role in enhancing efficiency and ensuring smooth management of guest accommodations. Its user-friendly design and comprehensive features enable staff members to efficiently handle various hotel services, such as processing reservations, and retrieving reservation details using unique IDs. By optimizing these essential tasks, the system empowers staff to deliver excellent service and improve guest experiences.

Our Hotel Management System Java program has been enhanced with graphical user interface (GUI) components to improve user interaction and experience. This upgrade includes the addition of two frames to facilitate user input and output. The main modifications involve implementing functionalities for adding reservations, loading and saving reservation data from/to files, displaying reservation information, and searching for a specific reservation. Additionally, exceptions are handled to ensure durability and error-free operation.

UML diagram:


Reservation
- reservationID: int
- numOfNights: int
- room: Room
- guest: Guest
- numID: int
+ Reservation (numOfNights: int, room: Room, guest: Guest)
+ CalculateTotalPrice() : double
+ getRoom() : Room
+ getNumOfNights(): int
+ getGuest(): Guest
+ getReservationID(): int
+ toString(): String
guest
- Name : String
- TelNum : int
+ guest(Name : String ,TelNum : int)
+ toString(): String
+ getName(): String
Reservations
Hotel
- hotelName: String
- location: String
- numberOfReservations: int
+ Hotel(hotelName: String , location: String, Room[] rooms)
+ searchRoom(roomNumber: int): Room
+ displayRooms(): void
+displayReservations(): Reservations[]
+ addReservation(Reservation res): void
+ cancelReservation(reservationID: int): void
+ Reservation searchReservation (guestName: String, roomNumber: int)
+ gethotelName(): String
Rooms
Room
roomNumber: int
Guest: guest
basePrice: double
+ Room(roomNum: int, basePrice: double)
+ isCheckedIn(): boolean
+ checkOut(): void
+ setGuest(guest: Guest): void
+ calculatePrice(): double
+ toString(): String
+ getRoomNumber(): int
StandardRoom
- isSmokingAllowed: boolean
- maxOccupancy: int
+ StandardRoom(roomNum: int, basePrice: double, isSmokingAllowed: boolean, maxOccupancy: int)
+ isSmokingAllowed(): boolean
+ getMaxOccupancy(): int
+ calculatePrice(): double
+ toString(): String
LuxuryRoom
- hasHotTub: boolean
- numOfBeds: int
- bedType: String
+ LuxuryRoom(roomNum: int, basePrice: double ,hasHotTub: boolean, numOfBeds: int, bedType: String)
+ calculatePrice(): double
+ toString(): String

Implementation of exceptions:

In Our Hotel Management System program, exceptions are handled at various points to address potential errors and provide appropriate feedback to the user:

1.Input Validation (unchecked exception):

When staff members input data such as phone number, number of nights, and room number, the program expects valid numeric inputs. If the user enters invalid data (e.g., non-numeric characters), a NumberFormatException is thrown.

To handle this, each input field is wrapped in a try-catch block where the program attempts to parse the input as an integer. If parsing fails, a message dialog informs the user of the error, and the input field is reset to a default value of zero. This ensures that the program can continue to execute without crashing due to invalid input.

2. File Handling (checked exception):

When the program attempts to load or save a file, various file-related exceptions may occur, such as FileNotFoundException and IOException.

To handle these exceptions, the file operations are enclosed within try-catch blocks. For example, when loading a file, if the specified file does not exist, a FileNotFoundException is caught, and a message dialog informs the user that the file does not exist. Similarly, when saving a file, if an IOException occurs during the file writing process, the exception is caught, and an error message is displayed to the user.

This approach ensures that the program gracefully handles file-related errors and provides feedback to the user, enhancing the overall user experience and robustness of the system.

3. Invalid reservation ID Exception (checked exception):

In the system, there is a user defined exception called invalidReservationIDException. This exception is thrown when an invalid reservation ID is encountered, such as a zero or a non-positive integer value.

To handle this custom exception, the program includes a try-catch block specifically for catching instances of invalidReservationIDException. When a reservation ID input is processed, if it is found to be invalid, the exception is caught, and a message dialog informs the user of the error, prompting them to enter a valid reservation ID.

By implementing comprehensive exception handling mechanisms across the program, users receive informative feedback in the event of errors, enhancing the program's reliability and facilitating a smoother user experience.

GUI Description

In the first frame, labeled "Add Reservation," staff can input guest information such as name, phone number, number of nights, and room number. Exception handling is implemented to validate staff inputs for phone number, number of nights, and room number. Error messages are displayed to guide users in case of invalid inputs.
File management functionalities are incorporated to enable users to load reservation data from a file and save reservation data to a file. Users can specify the file name for loading, and appropriate messages are displayed to confirm successful file operations. Exception handling is implemented to handle file-related errors, such as file not found or errors during file reading/writing.

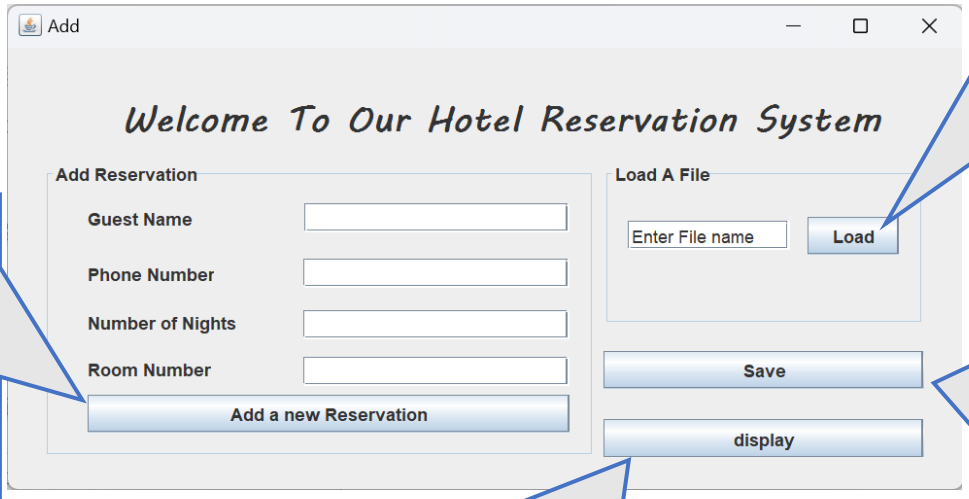
File management functionalities are incorporated to enable staff members to load reservation data from a file and save reservation data to a file. staff can specify the file name for loading or saving, and appropriate messages are displayed to confirm successful file operations. Exception handling is implemented to handle file-related errors, such as file not found or errors during file reading/writing.

The second frame, labeled "Reservation Information," allows staff to display all reservations and search for specific reservations by reservation ID. It features a text area for displaying reservation information and a text field for entering the reservation ID to search for. When displaying all reservations, exception handling ensures that only valid reservation data is displayed, preventing any potential errors.

When Clicked:

loads the reservations from the named file into the system.

A message dialog shows success of loading appears.



When Clicked:

The reservation is added to the system.

A message dialog shows success of add operation appears.

All fields are cleared after adding a Reservation.

When Clicked:

Takes the user to the next frame.

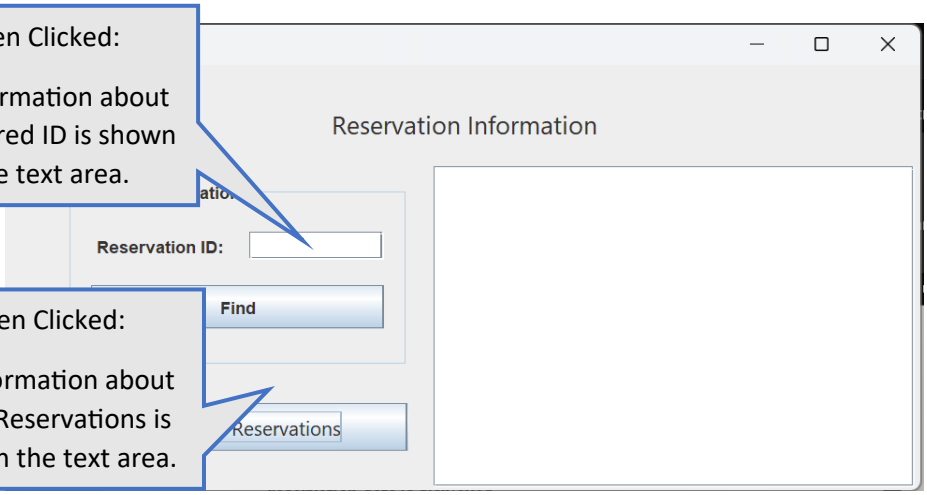
When Clicked:

Saves the reservations to a file.

A message dialog shows success of saving appears.

When Clicked:

The information about the entered ID is shown in the text area.



When Clicked:

The information about All the Reservations is shown in the text area.