

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 1 REPORT

**EBRU KARDAŞ
141044049**

Course Assistant: Fatma Nur Esirci

1 INTRODUCTION

1.1 Problem Definition

A *simple* hotel management system is designed and implemented with Java 8.0 on IntelliJ IDEA. The management system design is simple. Receptionist user services are booking & canceling room, checking-in and checking-out. Hotel guest user services are booking room and cancel reservation.

1.2 System Requirements

When Main program runs, inputs are willing from user via terminal in order to:

- is user receptionist (R/r) or guest(G/g)?
- Does the user want to give a spesific hotel file (like different room system .csv file)? For this choice, type Y/y for yes, N/n for no. If y is typed, filename is wanted.
- User fullname is wanted to login if receptionist or to make process if guest.

-If user is receptionist, receptionist's password is wanted. If name and password is correct, receptionist services are printed on the terminal.

For these services, each service has a spesific number to take the request:

- If number 1 typed, service book room is given to book room. So it is, guest name, room number and person number that will stay in the room are wanted.
- If number 2 typed, service cancel book is given to cancel from room. So it is, guest name and room number are wanted.
- If number 3 typed, service check-in is given to check-in. So it is, guest name, room number and person number that will stay in the room are wanted.
- If number 4 is typed, service check-out is given to check-out. So it is, guest name and room number are wanted.
- If number 5 is typed, system exits.

After each process, hotel rooms are shown if it is empty or not. If a room is not empty, room information (Empty/Booked/Reserved, guest name, person number in the room) is shown to receptionist.

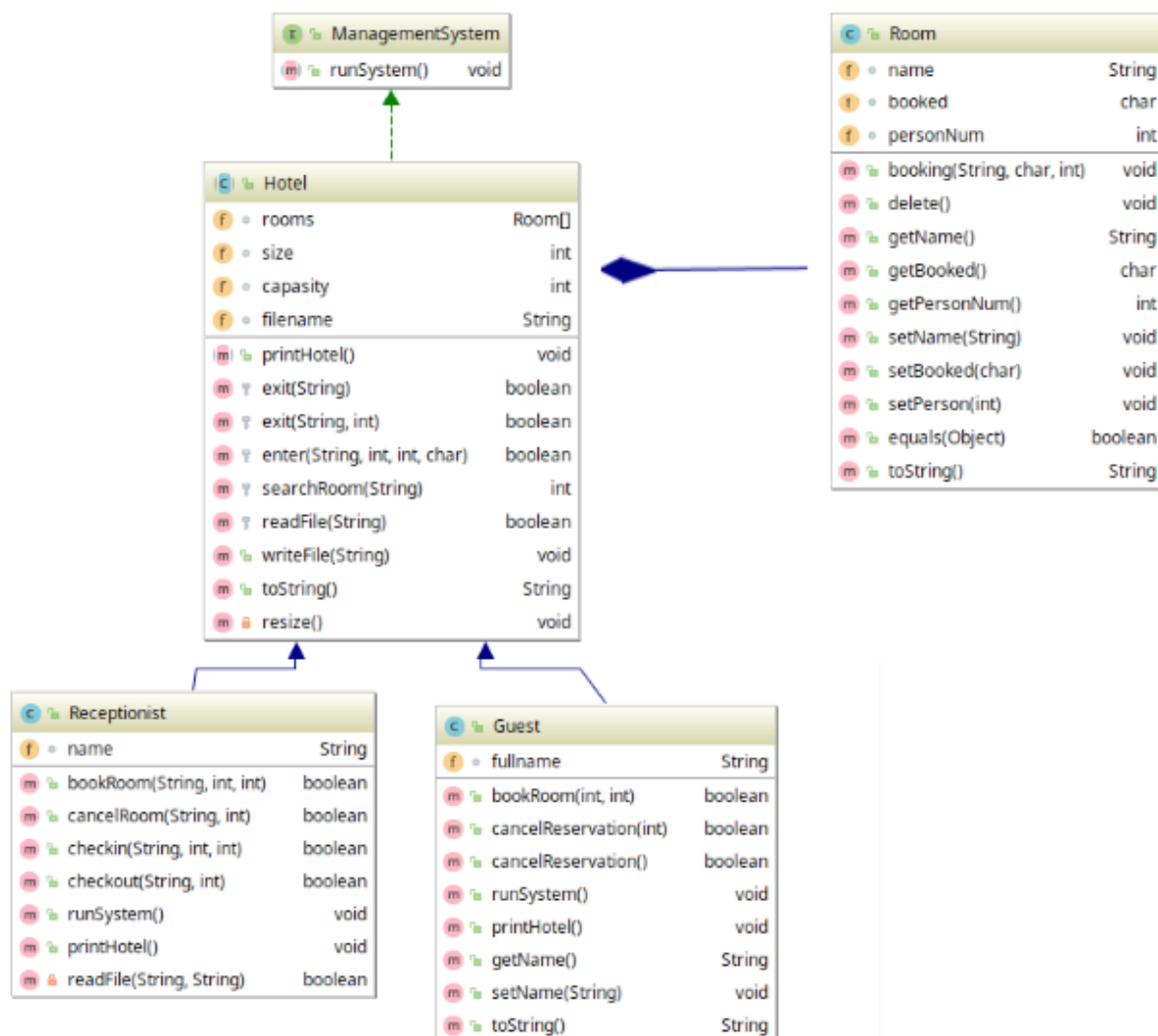
-If user is guest, each service has a specific number to take the request:

- If number 1 typed, service book room is given to book room. So it is, room number and person number that will stay in the room are wanted.
- If number 2 typed, service cancel book is given to cancel from room. So it is, guest name and room number(optional) are wanted. If room number didn't entered, then there is one room for that guest, the room is searched for that guest and canceled.
- If number 3 typed, system exits.

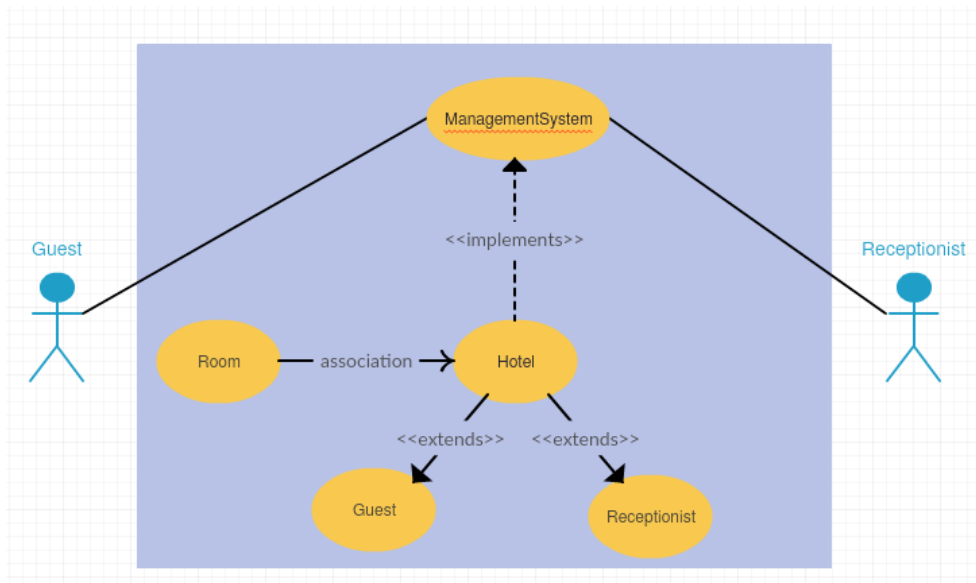
After each process, hotel rooms are shown if it is empty or not.

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams



Guest and Receptionist (user-person) are interact with system on Management System. Inputs are taken via terminal. **But for test**, inputs are taken from .txt file. File format shown in test cases.

2.3 Other Diagrams (optional)

No need other diagrams

2.4 Problem Solution Approach

ManagementSystem is a class that contains only one method which is a simple kind of gui from terminal. Usage is in the system requirements.

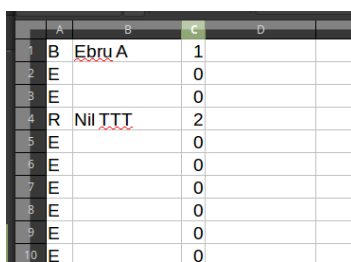
Room class takes information about the person who books the room, the situation if the room is booked and how many person that will stay in the room.

Hotel class takes dynamic Room array to serve services, like exit from room, entering to room, file read & write, to Guests and Receptionists. Room data is received from .csv file.

* One guest can book more than one room.

** If more than one room are booked, when canceling or checking-out from the room, room number must be given. If it is not given, first found room by the guest name will be canceled/checked-out.

*** Receptionist must be given in the "recep.csv" file. Information order is (receptionist name,password).



	A	B	C	D
1	B	Ebru A	1	
2	E		0	
3	E		0	
4	R	Nil TTT	2	
5	E		0	
6	E		0	
7	E		0	
8	E		0	
9	E		0	
10	E		0	

In .csv file(UTF8), each line is room number and there are some information about hotel in order to:

Column A: The room is empty, booked or reserved, represented by character.

Column B: Guest name.

Column C: person number that will stay in room.

Guest class is for hotel guests to serves

- cancelReservation that takes room number or nothing if only one room is booked by that guest, and

- bookRoom that takes room number and person number

attributes. This attributes are provided from Hotel class which is super class. All requirements are given in system requirements.

Receptionist class is for receptionists to serves

- bookRoom that takes guest name, room number and person number;

- cancelRoom that takes guest name and room number;

- checkin that takes guest name, room number and person number, and

- checkout that takes guest name and room number

attributes. This attributes are provided from Hotel class which is super class. If it is chosen, hotel rooms can be displayed whether it is booked (with guest name) or empty.

3 RESULT

3.1 Test Cases

Main Tests:

Input file is .txt format. In the .txt file, wanted inputs are separated by '-' character.

“inputRecep.txt” --> outputRecep.txt

r-y-rooms3.csv-Ebru Kar-123-1-Eylul Kalay-5-2-2-Nil TTT-4-5

INFORMATION MUST BE GIVEN

r: chosen receptionist
y: hotel .csv file will be given answer
rooms3.csv: hotel .csv source file
Ebru Kar: receptionist name
123: receptionist password

SERVICES (OPTIONAL)

1: Book room service

Eylul Kalay: Guest name that wants to book room

5: Room number

2: Person number

2: Cancel reservation service

Nil TTT: Guest name that wants to cancel reservation

4: Room number

3: Check-in service

Eylul Kalay: Guest name that wants to book room

7: Room number

1: Person number

4: Check-out service

Nil TTT: Guest name that wants to cancel reservation

4: Room number

5: Exit system

“inputGuest.txt” --> “outputGuest.txt”

G-n-DENEME-1-5-2-2-5-2-4-3

G: guest chosen

n: an input .csv hotel file is default (“rooms.csv”)

DENEME: Guest name

SERVICES

1: Book room service

5: Room number

2: Person number

2: Cancel reservation service

5: Room number

2: Cancel reservation service

4: Room number

3: Exit system

Unit Tests:

GuestTest:

bookRoom:

- For empty room from given .csv hotel file (TRUE)
- For *non-empty room* from given .csv hotel file (FALSE)
- For *invalid room number* from given .csv hotel file (FALSE)
- For empty room from default “rooms.csv” hotel file (TRUE)
- For *non-empty room* from default “rooms.csv” hotel file (FALSE)
- For empty room, *invalid person number* from default “rooms.csv” hotel file

(FALSE)

cancelReservation(with room number parameter):

- For *non-exist guest name* from given .csv hotel file (FALSE)
- For exist guest name but *her own oom number* from given .csv hotel file (FALSE)
- For exist guest name but *invalid room number* from given .csv hotel file (FALSE)
- For exist guest name & valid room number from given .csv hotel file (TRUE)
- For *invalid room number* from default “rooms.csv” hotel file (FALSE)
- For valid room number from default “rooms.csv” hotel file (TRUE)

cancelReservation(no parameter):

- For *non-exist guest name* from default “rooms.csv” hotel file (FALSE)

- For exist guest name from default "rooms.csv" hotel file (TRUE)

ReceptionistTest:

bookRoom

- For non-exist guest name from default "rooms.csv" hotel file (TRUE)
- For exist guest name but *non-empty room* from default "rooms.csv" hotel file

(FALSE)

- For exist guest name but *invalid person number* from default "rooms.csv" hotel file

(FALSE)

- For exist guest name but *invalid room number* from default "rooms.csv" hotel file

(FALSE)

- For non-exist guest name from given .csv hotel file (TRUE)
- For exist guest name but *non-empty room* from given .csv hotel file (FALSE)
- For exist guest name but *invalid person number* from given .csv hotel file (FALSE)
- For exist guest name but *invalid room number* from given .csv hotel file (FALSE)
- For invalid input .csv file (EXCEPTION CATCHED)
- For wrong password (EXCEPTION CATCHED)
- For wrong reception name (EXCEPTION CATCHED)

cancelRoom

- For exist guest name but *not her room number* from default "rooms.csv" hotel file

(FALSE)

- For *non-exist guest name* from default "rooms.csv" hotel file (FALSE)
- For *non-exist guest name* and *invalid room number* from default "rooms.csv" hotel

file (FALSE)

- For exist guest name from default "rooms.csv" hotel file (TRUE)
- For *non-exist guest name* from given .csv hotel file (FALSE)
- For exist guest name but *invalid room number* from given .csv hotel file (FALSE)
- For exist guest name from given .csv hotel file (TRUE)
- For invalid input .csv file (EXCEPTION CATCHED)

checkin

- For non-exist guest name from default "rooms.csv" hotel file (TRUE)
- For exist guest name but *non-empty room* from default "rooms.csv" hotel file

(FALSE)

- For exist guest name but *invalid person number* from default "rooms.csv" hotel file

(FALSE)

- For exist guest name but *invalid room number* from default “rooms.csv” hotel file (FALSE)
- For non-exist guest name from given .csv hotel file (TRUE)
- For exist guest name but *non-empty room* from given .csv hotel file (FALSE)
- For exist guest name but *invalid person number* from given .csv hotel file (FALSE)
- For exist guest name but *invalid room number* from given .csv hotel file (FALSE)

checkout

- For exist guest name but *not her room number* from default “rooms.csv” hotel file (FALSE)
- For *non-exist guest name* from default “rooms.csv” hotel file (FALSE)
- For *non-exist guest name* and *invalid room number* from default “rooms.csv” hotel file (FALSE)
- For exist guest name from default “rooms.csv” hotel file (TRUE)
- For *non-exist guest name* from given .csv hotel file (FALSE)
- For exist guest name but *invalid room number* from given .csv hotel file (FALSE)
- For exist guest name from given .csv hotel file (TRUE)

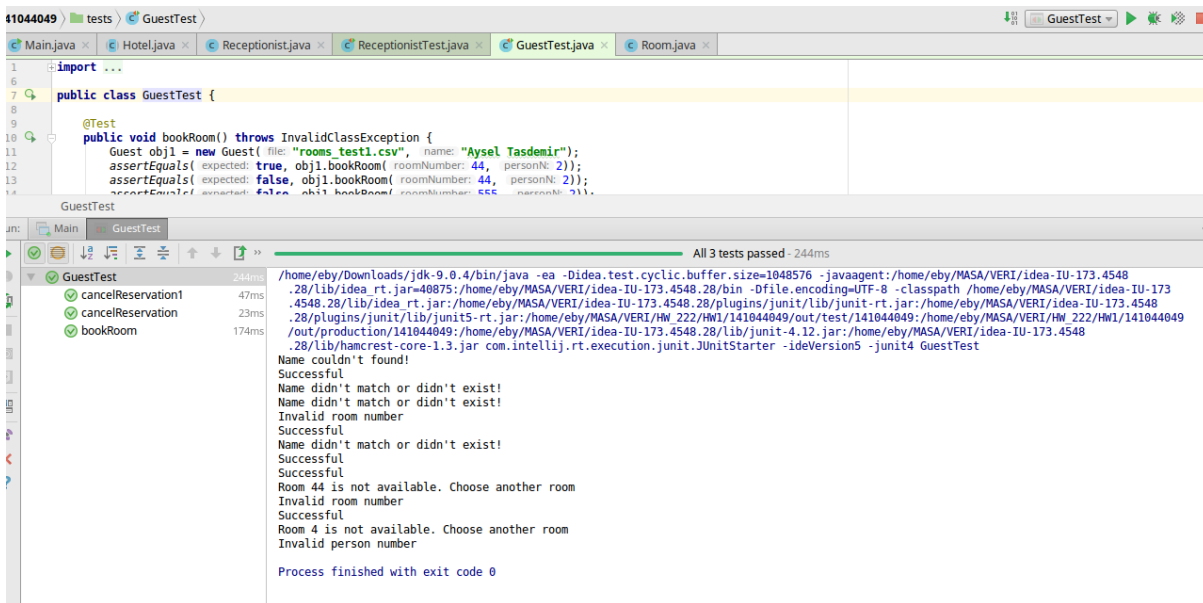
3.1 Running Results

Main test results are in output folder. Because of long outputs, all outputs are in .txt files.

“inputGuest.txt” --> “outputGuest.txt”

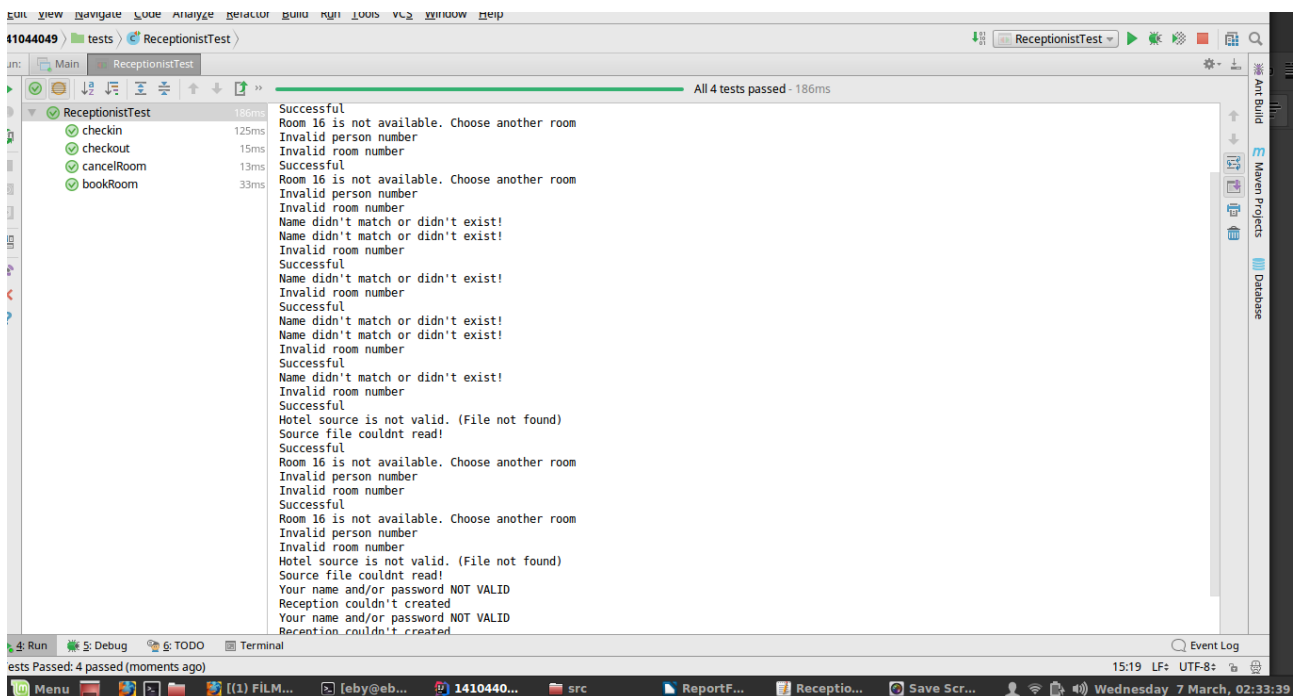
“inputRecep.txt” --> “outputRecep.txt”

Guest Unit Tests:



The screenshot shows the IntelliJ IDEA IDE with the `GuestTest.java` file open. The code defines a `GuestTest` class with a `@Test` method `bookRoom()` that throws `InvalidClassException`. The test method creates a `Guest` object and calls `bookRoom()` with room number 44 and person number 2. The IDE's output window shows the test results for `GuestTest`, indicating that all 3 tests passed in 244ms. The output also shows the JVM command line and the test runner's output, which includes messages like "Name couldn't found!", "Successful", "Name didn't match or didn't exist!", "Invalid room number", "Room 44 is not available. Choose another room", "Invalid room number", "Room 4 is not available. Choose another room", "Invalid person number", and "Process finished with exit code 0".

Receptionist Unit Test:



The screenshot shows the IntelliJ IDEA IDE with the `ReceptionistTest.java` file open. The code defines a `ReceptionistTest` class with several `@Test` methods: `checkin`, `checkout`, `cancelRoom`, and `bookRoom`. The IDE's output window shows the test results for `ReceptionistTest`, indicating that all 4 tests passed in 186ms. The output also shows the JVM command line and the test runner's output, which includes messages like "Successful", "Room 16 is not available. Choose another room", "Invalid person number", "Invalid room number", "Name didn't match or didn't exist!", "Invalid room number", "Hotel source is not valid. (File not found)", "Source file couldnt read!", "Room 16 is not available. Choose another room", "Invalid person number", "Invalid room number", "Room 16 is not available. Choose another room", "Invalid person number", "Invalid room number", "Hotel source is not valid. (File not found)", "Source file couldnt read!", "Your name and/or password NOT VALID", "Reception couldn't created", and "Your name and/or password NOT VALID".