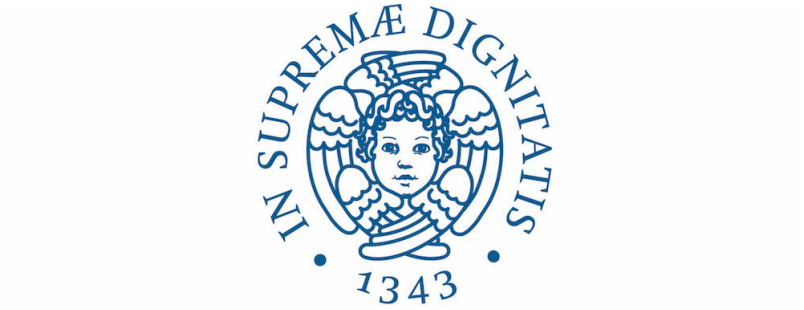

UNIVERSITÀ DI PISA

MSc in Computer Engineering
Information Systems
A.A. 2019/2020

Workgroup Task 0

Designing and implementing a simple JAVA application connecting to a relational database

Task Report

Chiara Bonsignori
Marco Del Gamba
Alessio Ercolani
Federico Garzelli

Contents

1 Specification

The application manages room reservations for an hotel chain with different branches.

1.1 Functionalities

There are two actors who interact with the application: Receptionist and Costumer.

A Receptionist can

- register a Customer
- verify the presence of a free room with a given capacity in a given hotel for a given period of time
- insert a new reservation
- update a reservation
- delete an upcoming reservation

A Customer can

- see his or her upcoming reservations

1.2 Data Flow

The front-end of the application provides a command line interface, handles user's input and prints result messages to the screen. Receptionist and Customer interact with the application with a command line interface by inserting the digit corresponding to the command they want to execute. First, they have to login, then they type commands to perform different operations.

Input data are passed to the logic modules of the application which recognize commands, perform different operations on the database according to their type and handle eventual errors.

The back-end components handle the interaction with the hotel database in order to check credentials and manage reservations and pass results back to the logic module.

1.3 Classes Responsibilities

Terminal Class *Terminal* manages the interaction with the user. It receives user's input, checks its syntax and invokes commands. Moreover, it shows the user result messages.

User, Receptionist, Customer Class *User* executes the commands of the application. A *User* can be a *Receptionist* or a *Customer*. As described in section 1.1, each of them implements its own set of operations while common functionalities, such as the login, are implemented by *User* class.

Reservation Class *Reservation* represents a model for the reservation of a room in a hotel during a certain period of time. *Reservation* objects are instantiated by *Terminal* class or by *DatabaseManager* respectively with user's input or database information and then are passed to *User*.

Room Class *Room* represents a model for a room in a hotel. *Room* objects are instantiated by *Terminal* class or by *DatabaseManager* respectively with user's input or database information and then are passed to *User*.

DatabaseManager Class *DatabaseManager* manages the interaction with the database. It opens and closes connections and performs queries to provide results to the other classes.

2 Use Cases

Figure 1: Use Cases Diagram

3 ER Diagram and Tables

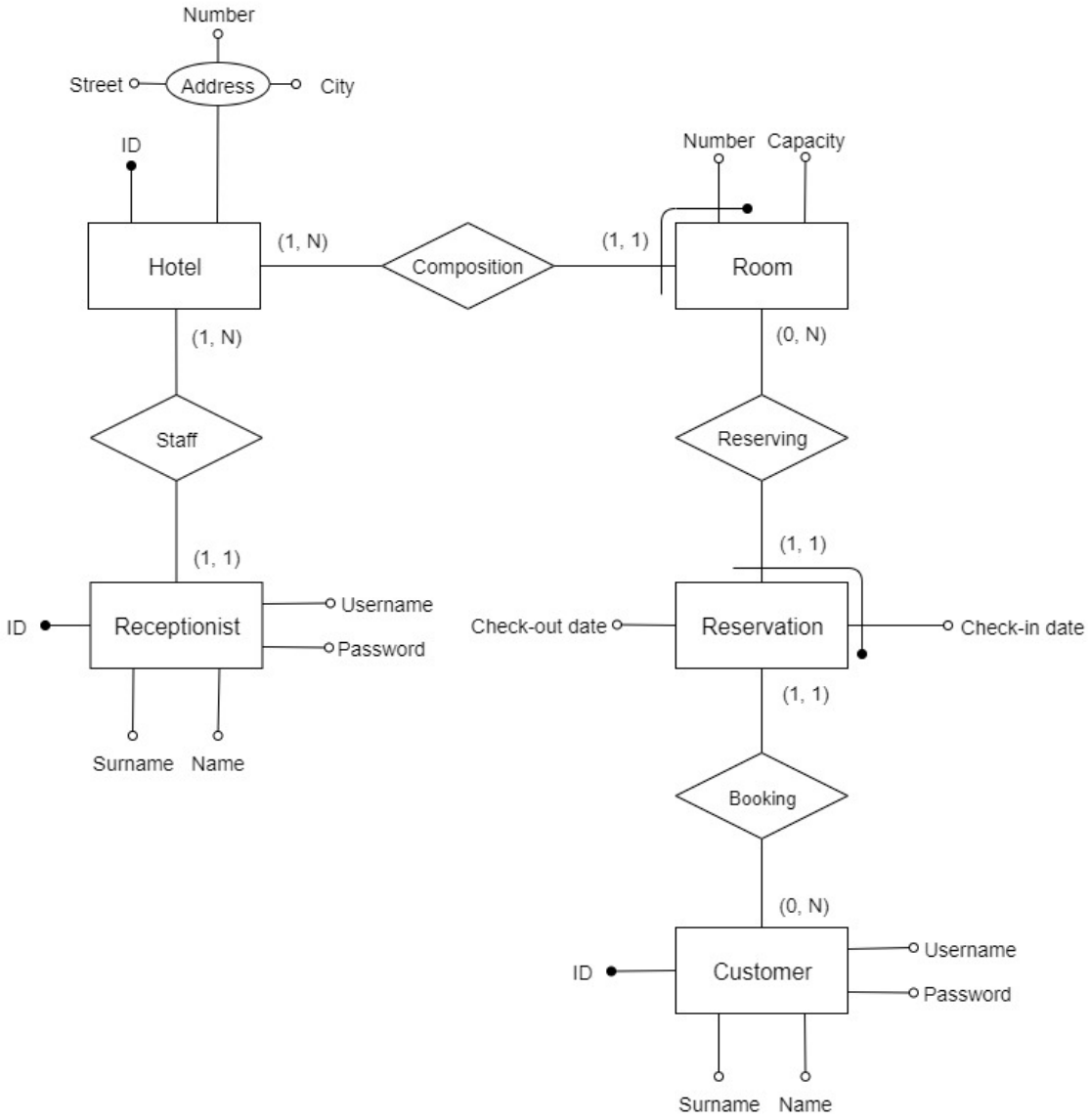

Figure 2: ER Diagram

Table List				
Customer	**Receptionist**	**Room**	**Reservation**	**Hotel Table**
Id	Id	Hotel	Hotel	Id
Name	Username	Number	Room	Street
Surname	Password	Capacity	Check-In	Number
Username	Name		Check-Out	City
Password	Surname		Customer	
	Hotel			

Table 1: List of tables

4 Example of Usage

4.1 Receptionist

```
Who are you? Type C if you are a custumer or R if you are a receptionist: R
Please, insert your username: z1
Please, insert your password: a
Hi z1!
You can choose one of the following command:
1. Insert reservation
2. Update reservation
3. Delete reservation
4. Add customer
5. Help
6. Exit
Select a command: 1
Insert details of the new reservation
Hotel: 5
Username: a
Insert check-in date (in format YYYY-MM-DD): 2020-12-02
Insert check-out date (in format YYYY-MM-DD): 2020-12-04
Number of guests: 4
0) Room 4 in Hotel 5: capacity 4
1) Room 5 in Hotel 5: capacity 4
2) Room 7 in Hotel 5: capacity 4
Select room: 0
New insertion completed.
Select a command: 2
Insert hotel, room and check-in data of the reservation to delete
Hotel: 5
Room: 4
Check-in date (in format YYYY-MM-DD): 2020-12-02
Insert information of the new reservation
New check-in date (in format YYYY-MM-DD): 2020-12-03
New check-out date (in format YYYY-MM-DD): 2020-12-04
New number of guests: 2
0) Room 1 in Hotel 5: capacity 2
1) Room 2 in Hotel 5: capacity 2
2) Room 9 in Hotel 5: capacity 2
Select room: 0
Update completed.
Select a command: 3
Insert hotel, room and check-in data of the reservation to delete
Hotel: 5
Room: 1
Check-in date (in format YYYY-MM-DD): 2020-12-03
Delete completed.
Select a command: 4
Insert name, surname, username and passoword of the new customer
Name: bg
Surname: bg
Username: ccc
Password: ccc
Added customer.
Select a command: 5
You can choose one of the following command:
1. Insert reservation
2. Update reservation
3. Delete reservation
4. Add customer
5. Help
6. Exit
Select a command: 6
Bye z1!
```

Figure 3: Example of usage by a Receptionist

4.2 Customer

```
Who are you? Type C if you are a custumer or R if you are a receptionist: C
Please, insert your username: a
Please, insert your password: a
Hi a!
You can choose one of the following command:
1. Show reservations
2. Help
3. Exit
Select a command: 1
Reservation:
        Customer a
        Room 5 in Hotel 1: capacity 2
Select a command: 2
You can choose one of the following command:
1. Show reservations
2. Help
3. Exit
Select a command: 3
Bye a!
```

Figure 4: Example of usage by a Customer

5 Test and Validation

The system is tested by using JUnit. The following functionalities are tested:

- the insertion, the update and the deletion of a new reservation
- the absence of a certain customer in the database, the insertion of the customer and the presence in the database

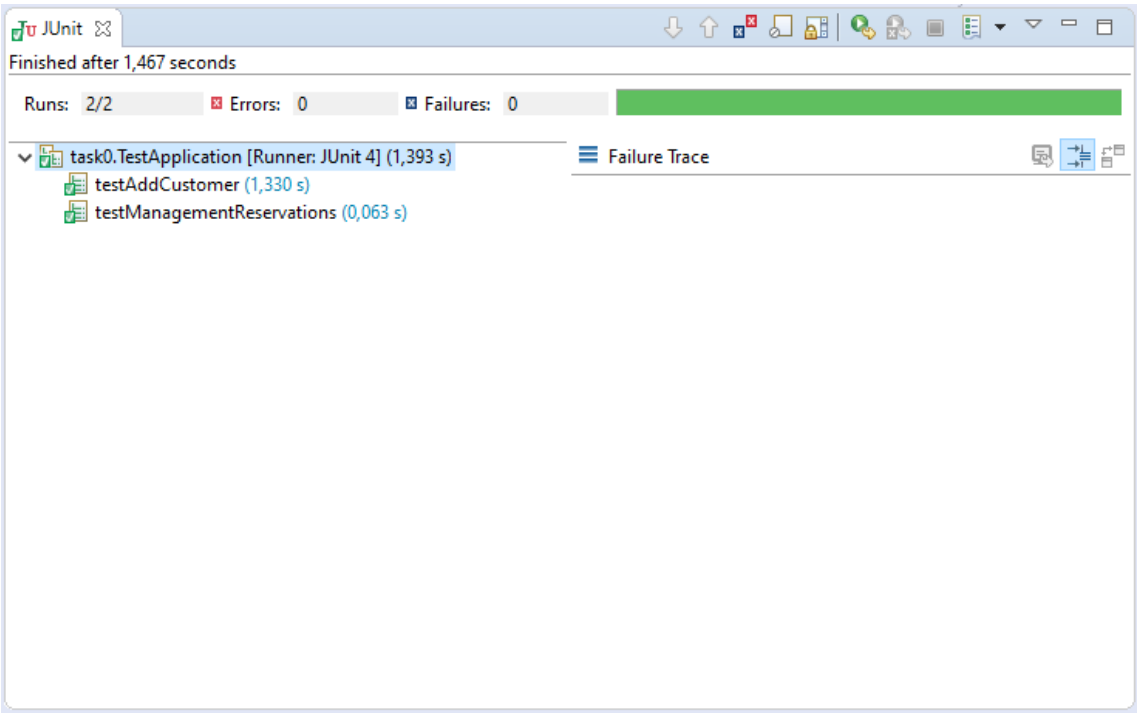

Figure 5: JUnit test