

|  |
| --- |
| Principle of Software Design & Development  Homework 2 |
|  |
| Fall 2020 - 2021  Eskişehir Technical University  By:  Şevval Türkyılmaz 49927389416 &  Yağmur DOĞAN 10963330588 &  Pelin Sena VAROL 64360013520 |



# Let’s revise the first report of our project and then complete the missing sections and explain changes. Here is our answers from Homework I.

# 1. Our imagined software development scenario.

# We’ve selected a subject which was not in the suggested topics list. Which is “Hotel Reservation System”.

# 2. Our requirements list was like that:

# The user enters the number of guests and the date.

# The application offers the rooms according to the user’s wishes.

# The user can sort the room options according to their choices and then selects.

# After the room is selected, necessary information is taken from the user, and the user makes the payment.

# The user can cancel the reservation up to one day before the check-in date and also takes the money back. Last minute changes are not allowed.

# Whether the reservation process successful or not, a message is displayed on the app’s screen without exiting the application.

# After the whole booking process, the user receives an e-mail and also can communicate with the hotel at any time via this e-mail address.

# 3. Our use case diagram according to Homework I.

# 4. Our 3 use cases from the report of Homework I.

# Use Case 1: Main Reservation

# The user opens the application of the hotel.

# The user enters the desired check-in and check-out dates.

# The user enters the number of the people who will stay.

# Rooms that meet the requirements are listed.

# The user displays room options, room features (like view type) and payment amount.

# The user can sort the options and selects the room they wanted.

# VI.1. If the desired room is not available, the user returns to the previous screen.

# VI.2. If the user wants to continue, s/he selects another room and continue to process if new selected room is available.

# VI.3. If the user wants to quit the application, a message displays and the app closes.

# Necessary personal information are obtained from the user.

# The user makes the payment for booking.

# Application confirms the reservation.

# A message appears on the screen.

# The user exits the application.

# Use Case 2: Cancel Reservation

# The user opens the main reservation section again.

# The user can cancel the reservation before at least one day from the check-in date

# The amount that the user paid returns to the user’s bank account.

# Use Case 3: Report

# The user receieves an e-mail which includes the reservation information.

# The user can make suggestions or ask questions via this

# e-mail address.

# 5. Our UML design and the given details about our project was like that according to Homework I Report.

# 

# KingSuit, Suit, Deluxe and Classic classes are the child of RoomType thus there is an inheritance relationship between them.

# In PersonalInformation class while we are storing the information we used encapsulation.

# Room uses RoomType’s attributes for listRooms and sortRooms methods. Also it uses RoomType’s roomPrice attribute for calculateTotalPrice method.

# Room uses Customer’s check-in check-out dates attributes for calculateTotalPrice method.

# Room uses Customer’s guestNumber attribute for listRooms method.

# Room uses Sort’s methods for sortRooms method.

# Payment actualize due to the calculateTotalPrice method from Room class. With Payment’s pay method, the program directs the customer to the 3D secure payment page.

# With Customer’s attributes, App class checks if the room is available or not (checkRoomAvailability).

# If the customer wants to cancel reservation, the App class runs cancelReservation method.

# If everything goes well, confirmReservation method runs.

# With App’s displayMessage method, the app displays a message before the app closes.

# App’s getReservationInfo method stores the information of the reservation. After the confirmation, app directs details to the Report class. With these details, Report’s sendConfirmationMail method runs and sends the reservation information to the user.

# HOMEWORK II FINAL REPORT

# What did we change in our project?

# Firstly, we removed the *‘roomType’* class from our project and we added all of its attributes to the ‘Room’ class. Because it didn’t do anything other than having just this attributes. In this way, our design has been better. And our subclassses KingSuit, Suit, Deluxe and Classic, now extends directly ‘Room’ instead of ‘RoomType.’

# Secondly, we’ve added *open()* ve *close()* methods to the ‘App’ class. These methods didn’t exist in our previous UML diagram. These methods allow us to to print a message on the screen while opening and closing our application.

# Also, we removed the *‘displayMessage()’* method that was previously in the App class. Because now our close() method does work that this method does before.

# And we replaced the *checkRoomAvailability()* method in the App class with the *checkReservationAvailability()* method. Now, this method takes our customers id as a parameter and returns a boolean value according to there is a reservation with the given id or not. This method provides control over the reservation process, not the room anymore.

# With this change, the alternative path of the 6th step of our first use case has removed. Now, we are checking the availability of the ‘reservation’. In this way, we added our alternative path to *Use Case 2.* So, when we want to cancel the reservation, we first check whether there is a reservation with the given id in this method. If there is no reservation with the given id, user should enter the id again. If the given id is correct, the cancellation process will complete successfully.

# Another new method, which does not exist in the last design, is *getRoomInfo().* This method that we added to ‘App’ class, allows us to print the properties of the room the user has selected. In the ‘Report’ class, we have a method called *sendConfirmationMail()*, to inform the customer. In this method, we are using *getRoomInfo()* method from ‘App’ class.

# And the last thing about our App class is, we had a method with the name confirmReservation() in our last design. Now we’ve also added an *addReservation()* method to the app class. While this method adds a new reservation, the other shows that the process are confirmed.

# We’ve added the *id* value as a field to the ‘Customer’ class. Also we’ve added its getter and setter methods. Also we declared public int *generateRandomId()* to get a random id. (field id: int 🡪 private final int id;)

# We did the sorting methods by price and by starRates in the ‘Sort’ class. But we removed the *sortByView ()* method from our project. We decided that the other three methods are more logical and sufficient for our project.

# We removed the *sortRooms()* method in the ‘Room’ class because we are listing the rooms first in our code(*listRooms())* and then print them with our sort methods.(*sortByPrice methods* or *sortByStarRates*). We already have methods that do the work this method will do. So, we don’t need this anymore.

# Which object-oriented principles did we use?

# There is a relationship between the Customer class and PersonalInformation class. ‘Customer’ extends ‘PersonalInformation’ and takes all of the information of the customer except check-in, check-out date and guestNumber. [There is Inheritance-Generalization here] Also we are using encapsulation here while we are storing the information.

# Also there is inheritance between our room types and the ‘Room’ class.

# We also provided encapsulation using private fields in Customer, Room and PersonalInformation classes.

# And we have a lot of associations and links between our classes.

# Our new Use Case Diagram: (simply containing main goals of our use cases)

# 

# Our new UML Diagram: (with new changes and parameters of methods)

# 

# (The arrow goes to sort from customer is hard to draw and this is bad for the appearance of UML diagram. Because of that, the arrow goes under of the Room class.)

# What is the summary of our code and project?

# Firstly, the application takes necessary information of the customer. (We gave these informations with Customer.txt file.)

# Then, application lists all of the available rooms in the hotel.

# Then, applications sorts the rooms according to price and star rates. For price, the app sorts both ascending and descending way and for star rates sorts in descending form.

# Then, the user can select the room s/he wanted.

# Then, the application prints the information of the selected room. User can see the price, the view type and the other properties again.

# Then, the application continues with the payment process. User should enter the price of the selected room in this section. The application checks if the given price is correct or not. If the price is correct, the applications directs user to 3DSecure.

# Then, at the end, the applications gives two choices to the user. They can log out from the application or they can cancel the reservation if they want.

# If they want to cancel the reservation, they should enter their id first.

# Then, application checks the reservation with the given id. If there is a reservation with the given id, the application cancels that reservation. Then, the application closes.

# If they choose ‘exit’, a message displays first and the application closes.

NOTE: “The software should not expect any input from the console during running the jar file.” There was such a sentence in the Homework 2 pdf, however, since our system is based on reservation, our every transaction is usually done at user request. We send an e-mail to teacher Gökhan Göksel and told the situation. Then, he said “Main metodunda default olarak veremiyor musun inputu kullanıcıdan almış gibi? Olmuyorsa kalsın öyle kullanıcı girsin inputu.”. That's why we pulled the data of user over a txt file and rearranged all other methods except the ‘room selection process’ so that they do not receive any input from the user.[Hocam çok fazla input girişi almamak adına kendimiz bir müşterinin bilgilerini bir text dosyasına yazıp o kişi için rezervasyon işlemi gerçekleşecek şekilde devamını getirdik. Sadece oda seçimi ve ödeme/isteğe göre iptal etme ve uygulamadan çıkma kısımlarında seçimi size bıraktık. Direkt kendimiz sabit bir şekilde girmek istemedik.]