



Advanced Software Engineering Assignment

Delivery date 14/04/2020

In this assignment, we intend to become more familiar with the concepts presented in the software testing chapter. For this purpose,

Consider a project that was implemented with the aim of booking trains. This project is a small library.

used to build a simple train reservation system. It is provided under Java which can be

The library provides sections for defining trains, cities, trips, and tickets. These sections should ensure that

in the final system, trips are compatible with each other (e.g., they do not overlap or conflict) and that

management features (e.g., a way to delay a trip) are provided. The Java code for this project is in the folder

attached to the assignment file, but a summary of how to use and implement it is as follows:

It is below.

- To compile and run, type the following command:

```
mvn package
```

- In order to use the codes of the different sections, consider the following information:

• To create an instance of the system, you must create an object of type `TicketReservationSystem`,

which requires an instance of the `TicketReservationSystemImpl` class, which also requires

It has a time zone selection. For this purpose, consider the following codes:

```
// Choosing the time zone of the system, to parse times/dates
// Here we use the time zone of the system
ZoneId timeZone = ZoneId.systemDefault();

// Creating the system
TicketReservationSystem system = new TicketReservationSystemImpl(timeZone);
```

After creating the system, you need to enter the basic information about cities, trains, etc., as shown in the code below.

Create:

```
// Creating and adding cities in the system
City rennesCity = new CityImpl("Rennes");
City nantesCity = new CityImpl("Nantes");
system.addCity(rennesCity);
system.addCity(nantesCity);

// Creating and adding trains in the system
Train smallTrain = new TrainImpl("Small Train", 2);
Train biggerTrain = new TrainImpl("Bigger Train", 10);
system.addTrain(smallTrain);
system.addTrain(biggerTrain);
```

Now it's time to create a trip. To do this, you need to enter the arrival and departure dates and other information as in the code below.

Define travel information:

```
Instant departure1 = TimeManagement.createInstant("2022-05-12 12:00", timeZone);
Instant arrival1 = TimeManagement.createInstant("2022-05-12 13:00", timeZone);
Instant departure2 = TimeManagement.createInstant("2022-05-12 15:00", timeZone);
Instant arrival2 = TimeManagement.createInstant("2022-05-12 16:00", timeZone);
try {
    Trip trip1 = system.createTrip(rennesCity, nantesCity, smallTrain, departure1, arrival1);
    Trip trip2 = system.createTrip(rennesCity, nantesCity, biggerTrain, departure2, arrival2);
} catch (TripException e) {
    // Do something if a constraint is unfulfilled
}
```

To book a ticket, follow the code below:

```
try {
    Ticket ticket1 = trip1.bookTicket("Alice");
    Ticket ticket2 = trip1.bookTicket("Bob");
} catch (ReservationException e) {
    // Do something if reservation was not possible
}
```

To cancel a ticket, follow these instructions:

```
trip1.cancel(ticket1);
```

• To change your ticket, keep the following code in mind:

```
List<Trip> alternatives = system.findPossibleExchanges(ticket2);
try {
    ticket2.exchangeTicket(alternatives.get(0));
} catch (ReservationException e) {
    // Do something if the exchange is not possible
}
```

You can also use the following code to find information about available trips:

```
// Search all trips from a city, on a given day
List<Trip> results1 = system.findAvailableTrips(rennesCity, LocalDate.of(2022, 5, 12));

// Search only trip between two cities, on a given day
List<Trip> results2 = system.findAvailableTrips(rennesCity, nantesCity, LocalDate.of(2022, 5, 12));
```

• To cancel a trip, use the following command:

```
system.cancelTrip(trip1);
```

Finally, keep the following code in mind to delay your trip:

```
// Delay departure
system.delayTripDeparture(trip2, Duration.ofHours(1));

// Delay arrival
system.delayTripArrival(trip2, Duration.ofMinutes(30));
```

• Now, according to this explanation, you should,

In the first part of this assignment, from the perspective of black box testing and input parameter segmentation, [•] **analysis**

And calculate the total number of unit tests required?

In the second part of the assignment, there are 5 scenarios and test cases (Case Test) to check the accuracy [•] of the

Design the functionality of the project codes implemented for the purpose of train reservations .

In the third part of this assignment, 5 scenarios based on the Gherkin structure for system acceptance testing

Design the final document in a way that covers the main needs expressed in the assignment text.

Give.

Now, according to the codes in the attached project and the explanations given in the previous sections, you should:

In the fourth part of this assignment, using JUnit 5, a test class containing the following fixtures will be created.

the scenarios stated in the second part of the assignment. Also, 10 Method Tests based on

Implement .

In the fifth part of this assignment, using Cucumber, 3 Step definitions are based on

Implement the scenarios outlined in the third section of the assignment .

In the sixth part of this assignment, you must use Mockito and create 2 Method Tests.

Implement new using mocking .

Please note:

Each section of this assignment carries 0.5 points and must be completed **individually** .

sections, a text report including descriptions and the total number of tests. As an output of the first to third

Prepare and submit the requirements for the first part and the scenarios for the second and third parts.

Create a short video of the tests for sections four to six and **share the link** .

at the beginning of **your submitted PDF file** of your assignment response.

Output:

Submit the final report as a PDF file to the LMS system.

Good luck.

Poet