Project Log of Inf2C-SE Coursework 3

Keqi Li(s1408733) JinHong Lu (s1569197)

November 2015

$23^{rd} November$

Aim

Get a basic understanding of the coursework and finish the setup and reading of the instructions and codes.

Achievements

Set Up of Version Control: We have created a Github Repository at `https://github.com/KeqiLi/BikeScheme`
Original files are all pushed to the repository for version control. The address above is the master repository of this project.
Create Class Implementations: User and Bike. (Details on the design of the classes will be explained in the Project.pdf) Both classes and some changes are merged into the repository. The log file of the project repository will be provided in the Appendix.

Reflection

Today, we have difficulty in understanding of the whole project code. we might have to spend more time on reading it. Therefore, we have decided to book a time slot with tutor to discuss and ask for our queries.

$25^{th} November$

Aim

Discuss with Prof. Paul Jackson and resolve some confusing questions about the demonstration implementation.

Achievements

Code: We have read through the demo tests provided and grasped a basic understanding of how the tests should work. We then created initialised tests in the SystemTest class for target use cases.
After the discussion with Prof. Paul Jackson, it is realised that the demo use cases and config can be changed. So we configures the design noticing more devices, e.g. bikesensor, bikelock.
Code style To follow convention, we modified the eclipse setting for java and replaced the "tabs-only" setting to "space-only" setting and made each indentation 4 space.

Reflection

After meeting with Prof. Paul Jackson, we have resolved most of our questions, there are still areas that we do not understand. We have booked another time slot to discuss about the project with tutor.

$26^{th} November$

Aim

We plan to figure out the project today and start implementing.

Achievements

Code: We have created Bike, User classes and fully completed Use Cases AddDStation, AddBike and HireBike with our own implementation. Moreover, we have planned the overview of the system, which help us implement the other use cases easily in the next few days.

Created Event Tests to test the cases that: If in one event, any parameter is not identical to the respective parameter of another event, they are different events.

Reflection

We have fully understood about the project. We are planning to start implementing Use Cases.

$27^{th} November$

Aim

We continue to implement the target task for the rest use cases.

Achievements

Code: We have created Trip class, which contains parameters of startStation, endStation, start and end time in day and minutes. Nevertheless, we have completed the rest use cases for the target task, RegisterUser, Return bike, ViewUserActivity and ViewOccupancy. After that, we have planned how to do the supplementary task for the additional use cases and how to implement the interface devices, BankServer, FaultLight and FaultButton.
Reflection: We have followed the plan through. Currently we are facing problem that implementing supplementry tasks with implementing new devices and observers.

$28^{th} November$

Aim

We start to implement the additional devices for interfaces.

Achievements

Code: After spending time to figure out how to implement the additional devices, BankServer, FaultLight and FaultButton, we have completed the three interface devices. BankServer and FaultLight are implemented as Output Device, whereas FaultButton is implemented as Input Device.

$29^{th} November$

Aim

With the implementation of the device classes yesterday, we start to implement the additional use cases for supplementary task.

Achievements

Code: We have completed the supplementary task for the use cases, RemoveBike, ReportFault, FindFreePoints, viewStats. Furthermore, we have created some Observers for these use cases.

$30^{th} November$

Aim

Finish up the supplementary tasks and write report design.

Achievements

Code: Finished implementing the bank server class and Charge User Use Case.

Appendix - Notes on Discussions with Instructors

Notes on Discussion with Prof. Paul Jackson on 25th Nov

We held a discussion meeting at 14:20 for 20 minutes and cleared some understanding:

1. It is allowed to change the demonstration tests/configurations provided. However the configuration is some standard design, we might want to add more features in it instead of replacing it.

2. Use Cases might need to work together instead of running by itself. For example: addDStation() creates two some docking stations. addBike() adds some bikes, which calls AddDStation() to make an environment. registerUser() registers a new user, which also needs to set up environment of docking station... After all, when testing viewUserActivity(), it will have to call the previous tests, e.g. addDStation(), registerUser(), hireBike(), returnBike().

3. When testing use cases, for one input there can be more than one output expected, as well as a list of output events; it should follow our design.

4. On the course webpage and piazza, there are specifications of how the output should look like. It is required to follow the standard for more objective marking.

5. Paul suggested us to read and fully understand the Event class and the web pages. There will be useful information we need or I asked.

6. Component-level tests are time-consuming and less mark-awarding, so it is suggested to finish all the other first before even thinking about it. It is the same for the supplementary use cases.

7. Tests of Event class follow some JUnit Tests guidelines to make sure the class works fine all the time.(A bug was found by someone.)

8. In the project log, it is recommended to record all the test failures or modification.

With tutor Dib on Thursday 26th Nov

After the discussion with Tutor Dib, we have understood that how to avoid NullPointer Exception, which is caused by that we did not call the Observer for the Use Case.
Furthermore, he suggested us some ways to implement the use cases. We also realised that we should first target the minimum level of the expectation then move on to the higher level of the expectation.