INF2C SOFTWARE ENGINEERING 2019-20 COURSEWORK 3

Capturing requirements for a federalized bike rental system

- HUACHENG SONG s1826390
- XUDONG ZHANG s1817972

Report

1. Revisions to design:

- Design document class diagram matches implemented system
    1. Obviously, our new document of class diagram shows a higher-level structure of our software system, as we match the newest implemented system.　Comparing to the coursework2, we add some new and important key classes to let our system more completely, for example, we decide to add Confirmation class, which could let system generate a receipt after customer booking a quote. This receipt would include many useful elements of quote for customers, like Order Number, order Summary, Deposit, Total Price, collection Mode and return Mode..., which could help users to check their quote quickly and correctly. In our class diagram, we have drawn all necessary classes in our software system and high light them.
- Discuss revisions made to design during implementation stage
    1. In our rental system, we believe that the key classes are Customer, Bike and Provider, because remained class will work and interact with this three-core class. In order to realize three main use cases (Getting Quote, Booking Quote, Return Bikes), we design three most complex methods in Customer class. First one which is call getQuote, this method generates a Collection of Quote, helping customers to get every suitable quote, which given by all providers. Second one which is called bookQuote, this method implements booking quote when customer has agreed to book provided bikes. It only requires user input the bike id showed on quote, it will do a quick check, then change the status of the booked bike, generate a confirmation for user to show all details of the booking bike, so that customer could check and clam down. This is a step that could double check the booked bike for customer. The last one method calls returnBike, when customer call this function, system will change bike Status into Tracking and get delivery service for returning used bike.
    2. In order to realize above three main scenarios, we finish many required class and methods, for example, we implement overlaps in DateRange class to check whether bike is available in required duration, and isNearto in Location class to check whether provider could provide their bike to customers hire location.
    3. We use Enumerate () method in java to show the Collection mode (Store Collection, Delivery) and Return mode (Original Provider, Partner Provider). This could help our system easy to choose corresponding options.

2. Self–assessment:

- Attempt a reflective self-assessment linked to the assessment criteria

    1. Extension submodules 10%/10%

    Implementation of extension submodule 10%

    Should implement extension submodule

    Should include unit tests for extension submodule

    Peer review of other group’　s submodule (up to 10% bonus marks)

2. Tests 27%/35%
System tests covering key use cases 20%/20%
Should have comments documenting how tests check the use cases are correctly implemented
Should cover all key use cases and check they are carrying out the necessary steps
Should have some variety of test data
We need to put more data to test.
Should use MockDeliveryService
We did not use MockDeliveryService to a large extent.
Unit tests for Location and DateRange 2%/5%
Systems test including implemented extension to pricing/valuation 5%/5%
Mock and test pricing/valuation behavior given other extension (challenging) 0%/5%
3. Code 45%/45%
Integration with pricing and valuation policies 10%/10%
System should correctly interface with pricing and valuation policies
System should correctly implement default pricing/valuation behavior
Functionality and correctness 25%/25%
Code should attempt to implement the full functionality of each use case ◊
Implementation should be correct, as evidenced by system tests
Quality of design and implementation 5%/5%
Your implementation should follow a good design and be of high quality
Should include some assertions where appropriate
Readability 5%/5%
4. Report 8%/10%
Code should be readable and follow coding standards
Should supply Javadoc comments for Location and DateRange classes
Revisions to design 3%/5%
Design document class diagram matches implemented system
Discuss revisions made to design during implementation stage
Self–assessment 5%/5%
Attempt a reflective self-assessment linked to the assessment criteria

Summary:

- In our rental system, most of system function use private variables to work, this could avoid any chaos when different functions call the same kind of variables. As well as, each function in our system have clear target, so system would not waste any times to deal with unnecessary step. Two interface pricing policies and deposit policies are designed in this system, this could increase the availability of the system and make renting more rational. Our system contains more specific use case (get Quote, book Quote, update the bike information …), so that this system could coverall requirements for renting bikes. In coursework2, we realized that the requirement would change from cw1 to cw2, so we made some important change to cover those requirements, like the class of controller and interface of pricing-deposit polices……Furthermore, we use teamwork to solve this coursework, so we could finished task faster and simpler. We will share our opinions when we are struggle, we could understand each other’ s part job very quickly, so we are confidence that other software engineer could understand our design.In conclusion, we trust we really did a good job