Informatics 2C – Introduction to Software Engineering Coursework 2

Team Member: Teh Min Suan (s1817967)
Michitatsu Sato (s1807428)

Introduction

This system is a platform for consumer to book bikes online. It includes features such as online payment, bike delivery, online chat and many more which will be introduced later on.

UML Class Model

See figure 1 on page 3.

High-level description

- When a bike is being registered, system will generate an integer as bikeId to identify bike.
- Location is a class that contains street address and postcode. There might be two different places with same street name but different postcode. This class is to prevent confusion in those cases.
- Methods in Pricing, Deposite and Recording are all protected so unauthorized user cannot use them.
- BikeProvider is an instance of bike provider actor which holds all the information of bike provider including all its partner.
- BikeProvider class implements Pricing and Deposit so every BikeProvider instances can call their methods to have custom pricing and deposit.

- Payment information of Customer is recorded within the Customer class which cannot be accessed by all other classes.
- Customers get quotes by entering requirement which will then be converted to Condition class. The result is then passed to getQuotes() as argument. It returns a collection of bikes matching the attributes in Condition. Information that is not provided will be ignored during the search. The collection of bikes will then be printed as quotes to customer within the method.
- Booking quotes can be done by calling bookQuote() method in Customer class. The method receives bikeId of a Bike and returns a Boolean telling if it has succeeded or failed. Then, system notifies bike provider and print the order summary to customer by calling printSummary() and notifyProvider().
- If customer returns bike to bike provider, the bike provider has to record the return by calling recordCollected(). It will then send a message to Customer. Every partners of a bike provider can record bike returned provided that they enter the name of bike provider and bikeId as argument.
- When bike is being delivered, driver can update the status of bike using updateStatus() method. The driver can then record the bike as delivered after delivering. The method called will send a message to both Customer and BikeProvider as a proof of bike being delivered.

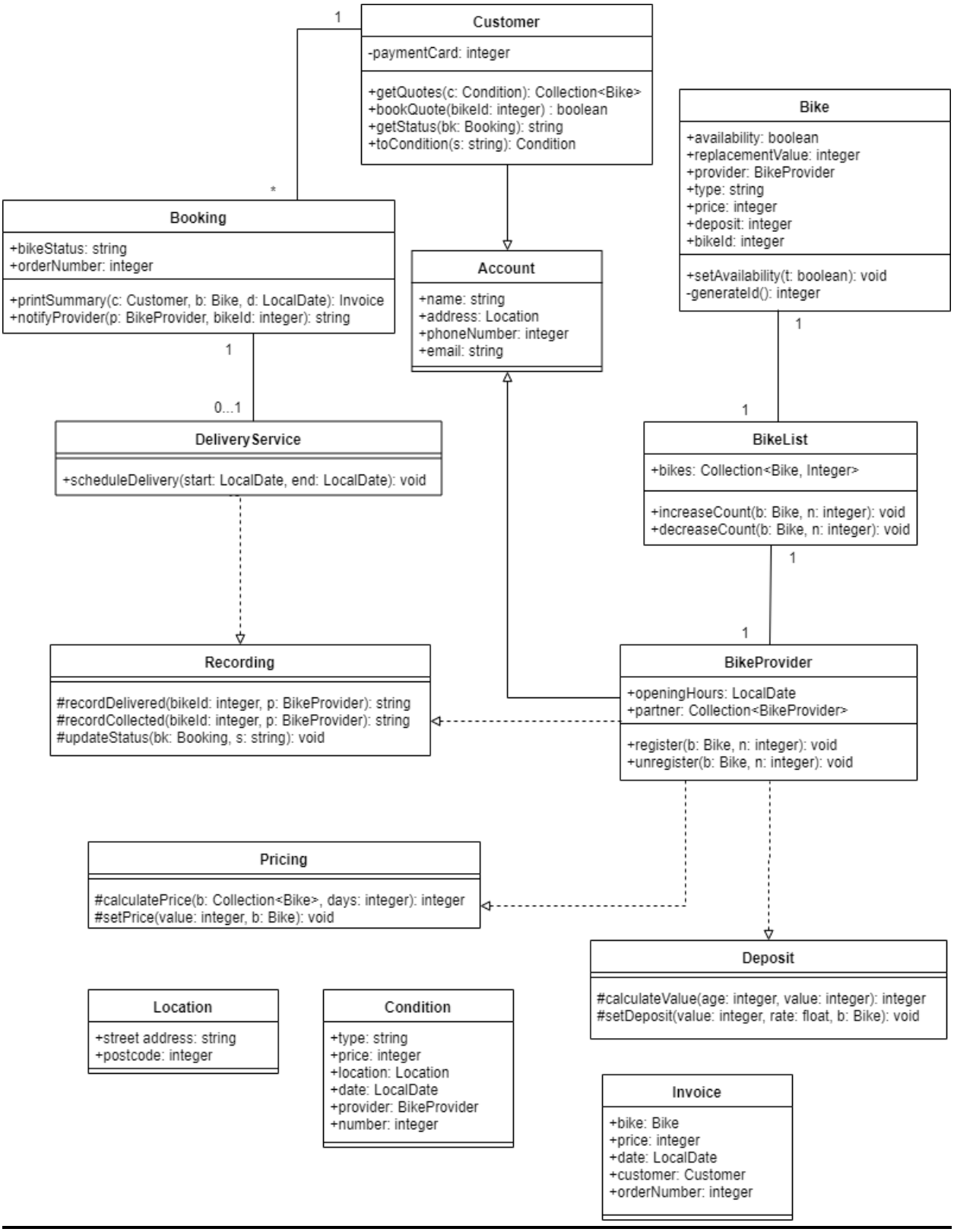

Figure 1: UML class diagram

UML Sequence Diagram

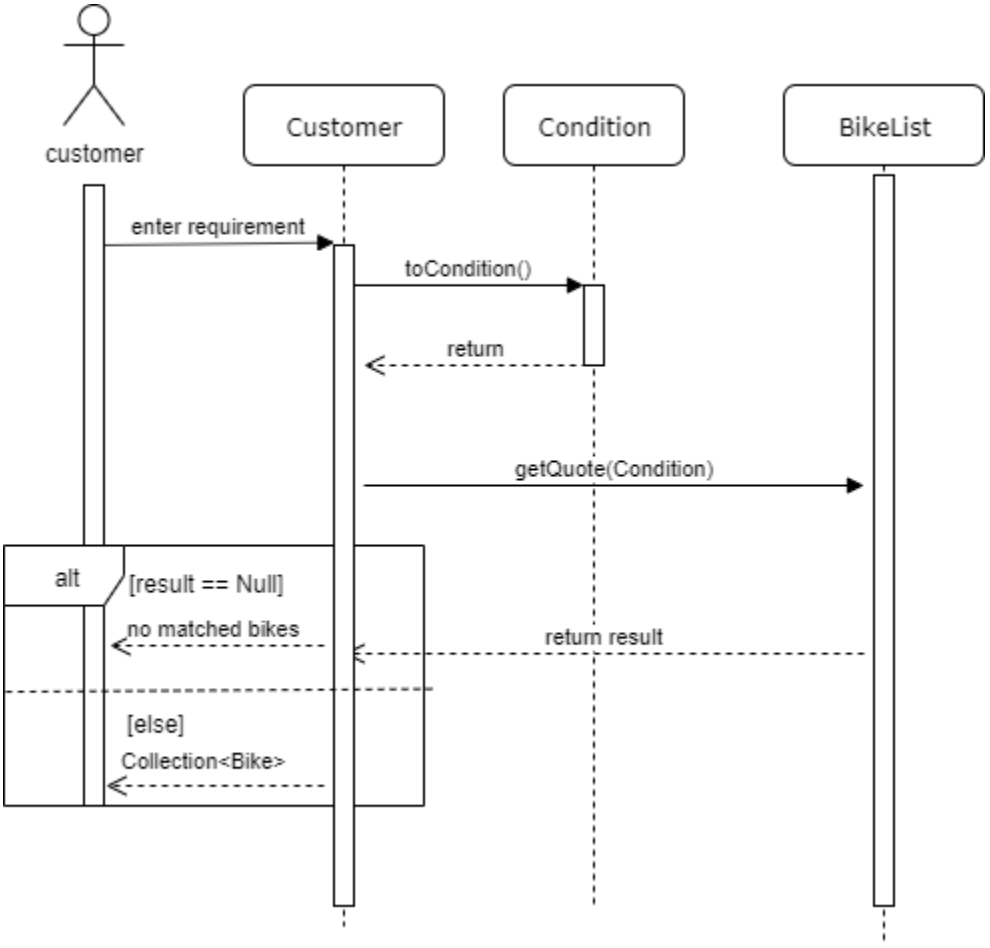

Figure 2: UML sequence diagram

UML Communication Diagram

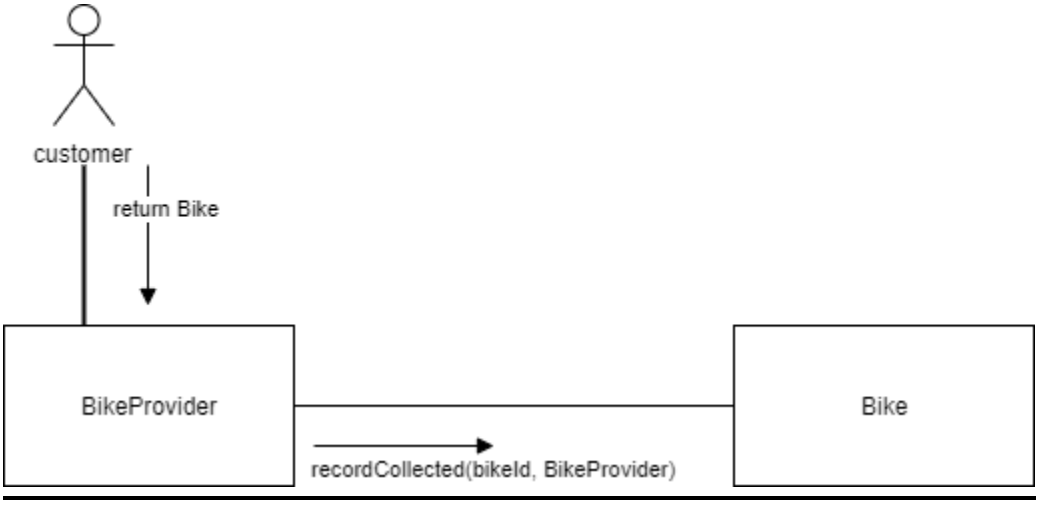

Figure 3: Return bikes to original provider

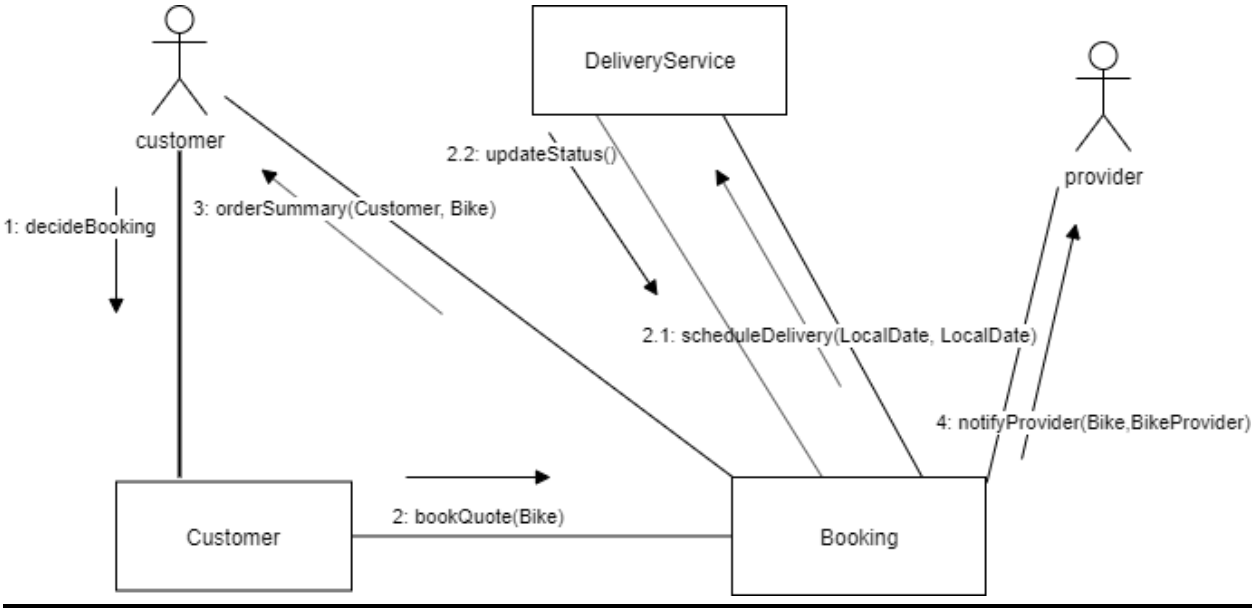

Figure 4: Book quote

Conformance to requirements

1. Bike originally has name as one of its attributes but it is hard to recognize the bike without any identification number. Thus, an attribute, bikeId, will be generated in every bike register.
2. Deposit was originally being calculated only when making a booking but that doesn't give any flexibility to bike provider to change the rate. Thus, an interface to calculate deposit is created and the information will be held by Bike class.
3. To be able to track the status of bike being delivered, there must be an attribute holding this information. A new attribute, bikeStatus is created for this purpose.
4. Initially, order summary was considered as a string that has all the information to be printed out. But it is hard to have a formatted string that holds all the booking information. Therefore, a new class, Invoice, is created to replace it.
5. Address was a string type but string can be easily modified. To ensure safety, a new class, Location, is created to save all the address information.

Self-assessment

Q 2.2.1 Static model 25%

- Make correct use of UML class diagram notation
  - Notation used should be all correct 5%
- Split the system design into appropriate classes
  - Several classes are created with important functionality 5%
- Include necessary attributes and methods for use cases
  - Each use cases should have suitable methods and attributes to be functional 5%
- Represent associations between classes
  - It should be done correctly 5%
- Follow good software engineering practices
  - Our design should be quite simple with multiple modules each having their own functionality 5%

Q 2.2.2 High-level description 15%

- Describe/clarify key components of design
  - Every part of our design is explained 10%
- Discuss design choices/resolution of ambiguities
  - We explained our choices of design as well as making sure ambiguities are solved 5%

Q 2.3.1 UML sequence diagram 20%

- Correctly use of UML sequence diagram notation

    - Notation used should be correct 5%
- Cover class interactions involved in use case
    - They should all be mentioned 10%
- Represent optional, alternative and iterative behavior where appropriate
    - We represented an alternative to getQuotes() when having no match in bike search 5%

Q 2.3.2 UML communication diagram 15%

- Communication diagram for return bikes to original use case
    - We should have correctly represented it in our UML communication diagram 8%
- Communication diagram for book quote use case
    - Same as above 7%

Q 2.4 Conformance to requirements 5%

- Ensure conformance to requirements and discuss issues
    - We brought up some issues we encountered and mentioned how we solved them 5%

Q 2.5.3 Design extensions 10%

- Specify interfaces for pricing policies and deposit/valuation polices
    - We did create 2 interfaces which are implemented by BikeProvider 3%
- Integrate interfaces into class diagram

- It is shown in our class diagram 7%

Q 2.6 Self-assessment

- Attempt a reflective self-assessment linked to the assessment criteria
  - We reviewed all of our works according to the assessment criteria at least once 5%
- Justification of good software engineering practice
  - Simple design. Only create class if it is necessary to do so. Every classes are created for its own functionality or, in another words, no overlapping functionality between classes. Design choices are explained clearly to ensure readers are able to understand the reason behind it.