# Inf2C Software Engineering 2019-20
# Coursework 3

HONGYU YU S1719616
HAO ZHOU S1862323

# Self-Assessment

1. Implementation of extension submodule (10%)

We believe we could get 10% in here because we successfully implement all the extension submodule and the unit test. And we pass the unit tests completely.

2. Peer review of another group's submodule (10%)

We think in this peer review we could get 8%. Actually, during one hour, my groupmate and me try our best to write down all the advantage and improvement for the odd group (group 33). Also due to the report, we also do some improvement on our code and design.

3. Tests (35%)

We strongly believe we could get 35 % on this because we write down all the necessary class and interface. Also, we keep communicating with each other frequently so that we have deep understanding on our own code and also could understand the code given by the partner. Furthermore, we successfully meet all the conditions and demands like
1."cover all key use cases and check they are carrying out the necessary steps."
2."have some variety of test data"
3."use MockDeliveryService"
All of this demand we finish it wonderfully.

4. • Unit tests for Location and DateRange(5%)

We have the Unit tests about the Location and DateRange, which are all pass the tests.so we could get full mark 5%.

5. Systems test including implemented extension to pricing / valuation (5)

We have the system test including implement the extension to pricing and valuation. So, we should get full mark 5%

6. Mock and test pricing/valuation behaviour given other extension (challenging) (5%)

It is a very challenging task that we don't have enough time to finish it, so we ignore it, but in our code. We have successfully designed a system which is available to use for renting a bike, so I think we should get a litter mark about this, maybe (3%).?

7. . Code (45%)

We believe that we could get this mark (43%) is because our code actually meets all the demand.
Firstly, we write the correct interface with pricing and valuation policies like doubleDeclingBalanceDepreciation class and linearDepreciation class. Secondly, we also have the pricing class and the valuation class in our code. Moreover, all the function in our code could be successfully implemented in our class, and the implement could pass the system test. In the design, we try our best the keep low coupling and high cohesion. Also, in each class

we try to reduce the unrelative attributes in order to reduce coupling. And in each class could have its own useful function to satisfy the condition that high-cohesion. In readability, we have some comment inside the code and we keep the name of data is easy to read and understand. Furhthermore, we have supplied the Javadoc on Location and DateRange class.

## Revision to design

I want to describe how we consider about the whole system. At first, bikeProvider need to upload all the bike information into the system which would store in the database. Then in the database we get all the bike information and the available date of the bike. Furthermore, the customer would submit the request of the bike and submit it into the system. In our QuoteProcessor, it would process and return the collection of the quote which is satisfy the conditions customer want. Then customer would pick one which is favor and then put this quote back to QuoteProcessor. Then our QuoteProcessor would update the bike information and notify the rental system to change the state of bike. The rental system would notify the provider that bike is rent, the customer that the payment is success, and the driver is time to delivery the bike. This is the whole design.
Also, we have some accurate example about this. For example, in our calculateQuote class,

```
public Collection<Quote> caculateQuote(RentalNeed rental_need,
Collection<BikeProvider> providerSet, double a, int choosen_type,
LocalDate ld)
```

The rental need is the need provided by the customer. The ProviderSet is the collection of bikeProvider, which could be used for developing the quote that meet the customer needs. The int chosen type which is use for determine the Depreciation rate, 1 means LinearDepreciation 2.meansDoubleDecliningBalanceDepreciation and the other means getOriginalReplacementValue() multiply the deposit rate. So, In here, we consider three different situation and write it down one by one. Many examples like this would be shown in our code.

In summary, we try our best to finish this work, and spending all day and night, at least over 50h.We hope we could get an excellent score.