NATIONAL UNIVERSITY OF SINGAPORE

CS2103/T – SOFTWARE ENGINEERING

(Semester 1: AY2017/18)

Part 2 (mock)

Time Allowed : 1 Hour

INSTRUCTIONS TO STUDENTS

- Please write your Student Number only. Do not write your name.
- This assessment paper contains **FOUR** questions and comprises **FIVE** printed pages.
- You are required to answer **ALL** questions.
- This is an **OPEN BOOK** assessment.
- You may **use pencils** to write answers.

STUDENT NO: _____

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/4	
Q2	/4	
Q3	/4	
Q4	/4	
Total	/16	

Q1 [4 marks] Illustrate the structure of the following *problem domain* (related to a role playing game) using a suitable UML diagram.

> A player can build structures, fight with other players, and trade (i.e. buy/sell) assets with other players. Some assets are used for building, some are used for fighting, and some are used as currency for trading with other players. Two types of assets that can be used as currency are gold bars and salt bags. All assets have an associated value. In addition, salt bags have expiry dates. A trade between two players involves a buyer giving up one or more assets to receive one or more assets from the seller. Buyer is the player who initiates the trade.

Q2 [4 marks] Illustrate the interactions caused by calling the `Car#start()` given below using a suitable UML diagram.

```
class Car{
    Engine engine;
    boolean isReady;

    void start(){
        if(isReady){
            engine.start();
        } else {
            engine.warm();
            showDelay();
        }

    }

    private void showDelay() {
        //...
    }
}
```

```
class Engine{
    Valve valve;

    void start(){
        valve.open();
    }

    public void warm() {
        //...
    }
}

class Valve{
    void open(){
        //...
    }
}
```

Q3 [2+2 = 4 marks] Design an efficient and effective test cases for the method given below. Show the intermediate steps to your test case design. Give at least 7 but no more than 10 test cases.

```
/**
 * Returns the class size of the specified module.
 * @param moduleCode should be in the range 1000..6999
 * @throws Exception if the moduleCode is not in range or the user is
 * not logged in or if the user is not an instructor
 */
int getClassSize(int moduleCode)throws Exception{
    //...

}
```

Q4 [4 marks] Suggest at least 5 code quality improvements to the code below. One example given.

```
/**
 * Add student to the list of students enrolled in the module.
 * @param studentNumber should be a valid student number
 * @param gitHubId cannot be null
 * @throws InvalidUserException if the given gitHubId is not a valid GitHub user
 * @throws IncorrectDataException if the student number is not a valid student number
 */
void addStudentToModule(int studentNumber, String gitHubId)
                        throws InvalidUserException, IncorrectDataException {

    assert isExistingStudent(studentNumber) : "student is already in the module";
    assert gitHubId != null;

    log(Logger.WARN, "adding student to the module");

    if(studentNumber > classSize)
          throw new IncorrectDataException("invalid student number");

    if(!new GitHubConnect().isValidUser(gitHubId))
           throw new InvalidUserException();

    Student addStudent = new Student(studentNumber, gitHubId);

    classList.add(addStudent); //adds the student
    ui.updateStudentCount(classList.size());

}
```

Remove redundant comment