The University of Nottingham Ningbo China

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, AUTUMN SEMESTER 2020-2021

DEVELOPING MAINTAINABLE SOFTWARE

Time allowed: **Sixty (60) Minutes (One Hour)**

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer All FOUR questions

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn examination paper over until instructed to do so

INFORMATION FOR INVIGILATORS:
Collect both the exam papers and the answer booklets at the end of the exam.

SECTION A: Object-Oriented Concepts and Maintenance Principles

Question 1: Developing software maintenance is realised through the concepts of object-oriented programming (OOP).

a. Illustrate **FIVE** OOP concepts in Java with explanations.

[5 marks]

b. Explain **TWO** differences between polymorphism, method overloading, and method overriding?

[6 marks]

Polymorphism	Method Overloading	Method Overriding
1.		
2.		

Question 2: There are six relationships in class diagrams where classes are interrelated to each other in specific ways. Provide Java sample code and discuss **TWO** differences between "Aggregation" and "Composition".

[6 marks]

Aggregation	Composition
1.	
2.	

Question 3: Explain the differences between continuous integration (CI), continuous delivery (CD) and continuous deployment.

[6 marks]

End of Section A: Total 23 marks

SECTION B: Design Principles

Question 4: Consider the following scenario and answer the questions below.

Jane would like to set up several thermal sensors in her room to monitor the room temperature. At the beginning, she bought several plain sensors, which measure the change of the temperature and act by displaying the latest measured value on the sensor's screen. Later, Jane's friend helped her set up several additional sensors, which exhibit some degree of intelligence: the sensor acts not only by displaying the latest temperature, but also by behaving as alarms. Out of these sensors with alarming functionalities, some of them beep if the temperature is higher than 40 °C, while others not only beep if the temperature is higher than 40 °C, but also send Jane an alerting SMS if the temperature is higher than 60° C.

Jane would like to implement a software based on the abovementioned scenario. It is expected to apply both the **Adapter** and **Observer** design patterns.

a) Draw the class diagram for this software. Show all the necessary information including class names, relationships, fields, and methods. For fields and methods, showing the essential ones that reflect the Adapter and Observer patterns would be sufficient. A class named **Tester** containing a `main()` method should be made available to manipulate the temperature change of the room. **[11 Marks]**

b) According to the class diagram above, implement the Java code. Note: (1) You do not need to implement the `Tester` class. (2) The *display*, *beep* and *sendSMS* functionalities do not need to be implemented, use output message "`displaying (or beeping, sending SMS)`" instead. (3) Stick to Java syntax as much as possible. Minor syntactical errors can be tolerated, but not majority of pseudo code.

[11 Marks]

Hint for a) and b):

Certain class names that would probably useful for this question are **`Room`**, **`PlainSensor`**, and **`IntelligentSensor`**. Their constructors would probably be:

```
1) +Room(temperature : double)

2) +PlainSensor(room : Room)

3) +IntelligentSensor(room : Room, type: String)
```

c) Based on your implementation, what should be the output, if the `main()` method of the **Tester** class is defined as follows? **[5 Marks]**

```
public class Tester {
    public static void main(String[] args) {
        Room room = new Room(10);
        new PlainSensor(room);
        new IntelligentSensor(room,"SMSAlarm");
        new IntelligentSensor(room,"BeepAlarm");
        room.setTemperature(70);
    }
}
```

End of Section B: Total 27 marks