CONFIDENTIAL

UNIVERSITI TEKNOLOGI MALAYSIA
FACULTY OF COMPUTING

TEST 2

SEMESTER II 2017/2018

SUBJECT CODE : SCSJ2154
SUBJECT NAME : OBJECT ORIENTED PROGRAMMING
YEAR/COURSE : 2 (SCSJ / SCSV / SCSB / SCSR)
TIME : 8.00 pm – 10.00 pm (2 Hours)
DATE : 25 April 2018 (Wednesday)
VENUE : MPK1-10 (Block N28)

INSTRUCTIONS TO THE STUDENTS:

- Read the problem and instructions carefully.
- References to any resources by any means except **OOP Lab Module** are strictly prohibited.
- You are given **TWO HOURS** to complete the test inclusive of the submission of your program.
- You must answer all the questions.
- You can download the java file for **Question 1** and input file for **Question 2** via UTM's e-learning system.
- Both of your programs must follow the input and output as shown in the examples.

SUBMISSION PROCEDURE:

- Only the source code (*i.e.* the file with the extension **.java**) is required for the submission.
- Submit the source code via the **UTM's e-learning system**.

This question booklet consists of 7 pages inclusive of the cover page.

QUESTION 1 – ERROR DEBUGGING **(40 Marks)**

You are given a Java source code (**`TestOrder.java`**) with errors (syntax errors and/ or logical errors, if any) and the sample of output in **Figure 1**.

```
1   import java.Scanner;
2
3   class Coffee
4   {
5         private String coffeeId;
6         private String flavour;
7         private double pricePerUnit;
8
9         public void Coffee(String id, String n, double p)     {
10              coffeeId = id;
11              flavour = n;
12              pricePerUnit = p; }
13
14        public String getId()
15        {     return coffeeId;  }
16
17        public String getFlavour()
18        {     return flavour;   }
19
20        public String getPrice()
21        {     return pricePerUnit;    }
22
23        public void display()
24        {             System.out.printf("%-15s%-15s%-
25  10.2f\n",coffeeId,flavour,pricePerUnit);  }
26  }
27
28
29  class DiscountedItem extends Coffee
30  {
31        private double discRate;
32
33        class DiscountedItem(double r){
34              super();
35              discRate = r;     }
36
37        public double getPrice()
38        {     return (getPrice()-(getPrice()*discRate));}
39  }
40
41
42  class Customer
    {
          private String customerName;

          public Customer(String n)
          {     customerName = n; }

          public String getName()
```

```
43        {     return customerName; }
44  }
45
46
47  class CoffeeOrder
48  {
49        private int orderId, unit;
50        private Customer customer;
51        private Coffee item;
52
53        public CoffeeOrder(int id, String c, String i,int u)
54        {     customer = c;
55              orderId = id;
56              item = i;
57              unit = u;
58        }
59
60        public double calcTotalPrice(Coffee item)
61        {           return getPrice()*unit;       }
62
63        public String toString()
64        {     return "\nOrder ID: " +orderId+"\nCustomer Name:
65  "+getName()+"\nFlavour: " +getFlavour()+ "\nQuantity: "
66  +unit+"\nPrice per Unit: "+getPrice()+"\nTotal Price: "
67  +calcTotalPrice();      }
68  }
69
70  public class TestOrder
71  {
72         public static void main(String[] args)
73         {
74              Coffee myCoffee = new Coffee("CC001", "Caramel",
75  10.00);
76              DiscountedItem yourCoffee = new Coffee("CC008", "Latte
77  ", 8.00,0.10);
78              Coffee hisCoffee= new Coffee("CC009", "Mocha", 9.00);
79
80              System.out.println("\n ITEM DESCRIPTION");
81              System.out.println("********************");
82              System.out.printf("%-15s%-15s%-25s\n", "Coffee ID",
83  "Flavour", "Price");
84              System.out.printf("%-15s%-15s%-25s\n", "*********",
85  "********", "******");
86
87              Vector<Coffee> menu = new Vector<Coffee>();
88              menu.add(myCoffee);
89              menu.add(yourCoffee);
90              menu.add(hisCoffee);
91
              for (int i = 0; i < Coffee.size; i++)
              {
                   menu.display();
              }


  System.out.println("************************************");
```

```
92          Customer [] c= new Customer[3];
93          c[0] = new Customer("Arif");
94          c[1] = new Customer("Hakim");
95          c[3] = new Customer("Nuha");
96
97        ArrayList<Object> order = ArrayList<Object>;
98
99            order.addElement(new CoffeeOrder(1, c[0],myCoffee,2));
100           order.addElement(new CoffeeOrder(2, c[1],yourCoffee,3));
101           order.addElement(new CoffeeOrder(3, c[2],hisCoffee,1));

              System.out.println("List of orders :\n");

              for (int i = 0; i < order.length(); i++)
              {

      System.out.println(((ArrayList)order(i)).toString());
              }     }}
```

You are required to debug the errors, compile and run the program. The program should produce the output as in Figure 1.

```
ITEM DESCRIPTION
********************
Coffee ID      Flavour        Price
*********      ********       ******
CC001          Caramel        10.00
CC008          Latte          8.00
CC009          Mocha          9.00
**********************************
List of orders :

Order ID: 1
Customer: Arif
Flavour: Caramel
Quantity: 2
Price per Unit: 10.0
Total Price: 20.0

Order ID: 2
Customer: Hakim
Flavour: Latte
Quantity: 3
Price per Unit: 7.2
Total Price: 21.6

Order ID: 3
Customer: Nuha
Flavour: Mocha
Quantity: 1
Price per Unit: 9.0
Total Price: 9.0
Press any key to continue . . .
```

Figure 1: Sample output

QUESTION 2 – PROBLEM SOLVING **(60 Marks)**

You are being appointed as the developer a new patient management system in a clinic to manage the patient records. The system would be based on the following UML.

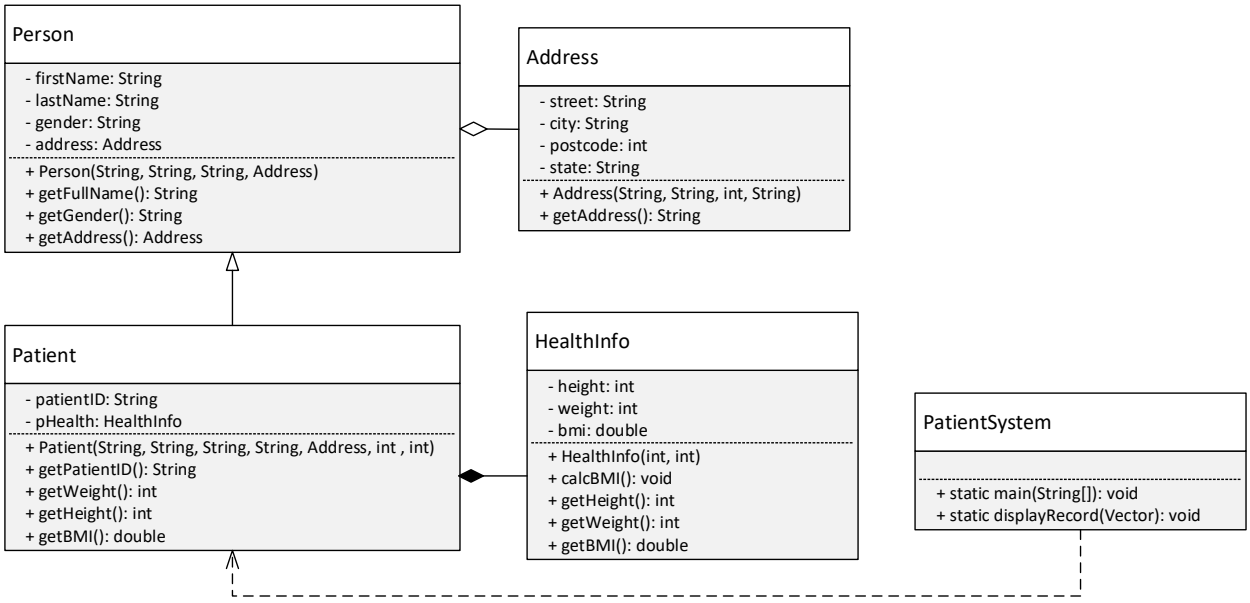

Figure 1: UML Diagram

You must write a complete JAVA program consists of ***Person.java, Address.java, Patient.java, HealthInfo.java*** and ***PatientSystem.java*** based on the instruction below:

a. Write a class named **Person** that consists of the information such as first name, last name, gender and the aggregation of the class of **Address.** The **Person** class also consists of a constructor with four arguments which are first name, last name, gender, and the Address object. It also contains the methods that access the full name, gender, and address.

b. Write a class named **Patient,** which inherited from **Person** class. The class consists of specific information such as patient ID as well as the composition of **HealthInfo** class. The **Patient** class also provides functions such as a constructor with arguments to initialize a patient with their patient ID, names, gender, address, height and weight. It also contains the accessor methods to get the information of patient ID, weight, height and BMI from the class of **HealthInfo**.

   ** Every time a patient is added, a message will be displayed such as following:

   "*Patient: <full name here> has been added*"

c. Write a class named **Address** which is aggregated into **Person** class. The class consists of information such as street name, city, postal code, and state. This class has a constructor that takes up four arguments for the address info as well as an accessor method to get the complete address.

d. Write a class named **HealthInfo** which contains the specific measurements for a patient (height, weight, and BMI). The class has a two-argument constructor, a function named `calcBMI` to calculate the Body Mass Index (BMI) for the patient based on their weight and height. The class also consists of the accessor methods for those attributes. **Note:** height is in centimeters (cm), and weight is in kilograms (kg). Formula to calculate BMI:

$$BMI = \frac{weight(kg)}{(height(m))^2}$$

e. You are given the incomplete file **PatientSystem.java** which is the driver program of the **Patient** class. You are required to do the following tasks:

   a. Create a dynamic list to hold the objects of **Patient**.
   b. Then initialize **THREE** patients and addresses based on **Table 1** below.
   c. Save those patient into the dynamic list.
   d. After initializing the three patients and saved into the list, the program will invoke the `displayRecord` function to display the list of patients.
   e. Write the code to remove the 2nd patient in the list and then invoke the `displayRecord` function again to display the list of patients.
   f. Complete the `displayRecord` function to display the list of patients are shown in **Figure 2.**

Table 1: Patient info

Patient ID	Name	Gender	Street	City	Postal code	State	Height	Weight
P0001	Akmal Adnan	Male	Jalan Pahlawan	Skudai	81300	Johor	180	82
P0002	Syafiq Yusof	Male	Jalan Flora	Skudai	81300	Johor	186	80
P0003	Mei Ling Koh	Female	Jalan Bakti	Skudai	81300	Johor	168	45

```
Patient: Akmal Adnan has been added
Patient: Syafiq Yusof has been added
Patient: Mei Ling Koh has been added

Patient Record Management System
================================
No   PatientID Name                 Gender    Address                                   Height    Weight    BMI
--   --------- ----                 ------    -------                                   ------    ------    ---
1    P0001     Akmal Adnan          Male      Jalan Pahlawan, 81300 Johor, Skudai       180       82        25.31
2    P0002     Syafiq Yusof         Male      Jalan Flora, 81300 Johor, Skudai          186       80        23.12
3    P0003     Mei Ling Koh         Female    Jalan Bakti, 81300 Johor, Skudai          168       45        15.94

------------------
Total Patients: 3

Patient Record Management System
================================
No   PatientID Name                 Gender    Address                                   Height    Weight    BMI
--   --------- ----                 ------    -------                                   ------    ------    ---
1    P0001     Akmal Adnan          Male      Jalan Pahlawan, 81300 Johor, Skudai       180       82        25.31
2    P0003     Mei Ling Koh         Female    Jalan Bakti, 81300 Johor, Skudai          168       45        15.94

------------------
Total Patients: 2

Press any key to continue . . .
```

Figure 2: Output of **PatientSystem.java**