Sri Lanka Institute of Information Technology

B.Sc. Honours Degree in Information Technology

Specialized in Information Systems Engineering

Final Examination

Year 2, Semester I (2019)

IE2021 – Object Oriented Programming

Paper D

Duration: 3 Hours

October 2019

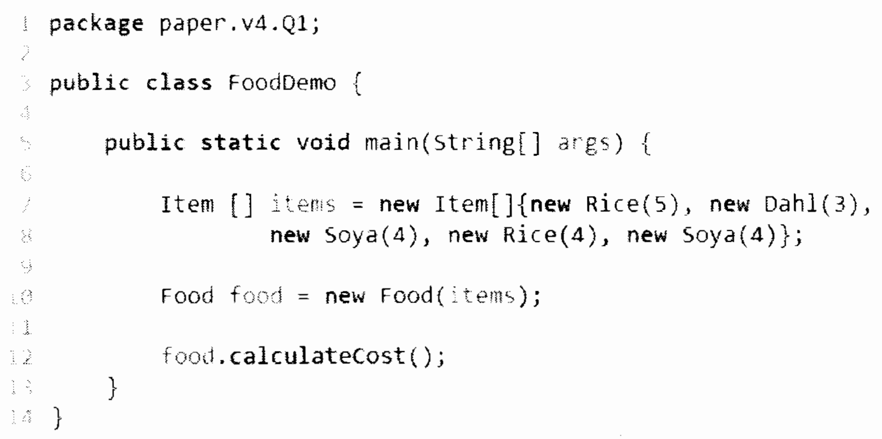
Instructions to Candidates:

- This paper contains **Four** questions. **Answer All** Questions.
- Fill **Student Details** in the last page
- Marks for each question are given in the paper
- Total Marks is 100.
- Create a separate Project for each question. The name of the project is provided. Save each Java program using the class name given.
- Store all your program files in the Desktop Folder provided
- This paper contains 10 pages with the Cover Page.

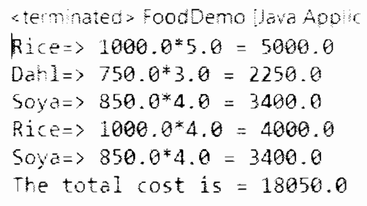
Question 1 **(30 marks)**

This question is based on the **Object-Oriented Programming (OOP) concepts**. You are going to add deferent Food items (Rice, Dahl, Soya) in deferent quantities of kilograms. Quantities are passed through the constructor.

a) You can refer the output is given in **FoodDemo** class and adjust your code accordingly



```java
package paper.v4.Q1;

public class FoodDemo {

    public static void main(String[] args) {

        Item [] items = new Item[]{new Rice(5), new Dahl(3),
                new Soya(4), new Rice(4), new Soya(4)};

        Food food = new Food(items);

        food.calculateCost();
    }
}
```



```
<terminated> FoodDemo [Java Applic
Rice=> 1000.0*5.0 = 5000.0
Dahl=> 750.0*3.0 = 2250.0
Soya=> 850.0*4.0 = 3400.0
Rice=> 1000.0*4.0 = 4000.0
Soya=> 850.0*4.0 = 3400.0
The total cost is = 18050.0
```

i). First implement the **IAccountable** interface and declare **void calculateCost( )** method.
(02 marks)

ii). Then implement and **Item** abstract class and declare the methods with return types **displayItem( ):String, getCost( ): double, getQuantity( ):double**
(04 marks)

iii). Create concrete classes called **Rice, Soya** and **Dahl** and override all abstract methods **displayItem( ), getCost()** and **getQuality( )** .
(3 x 3 = 09 marks)

iv). Add the property qtyInKilos and overload the constructor of each class (**Rice, Soya** and **Dahl)** and use the data type double.
(2 x 3 = 06 marks)

v). Similarly create a class called **Food** and implement the **IAccountable** interface with in the class and **override the calculateCost( )** method. Then overload the constructor to pass the array of Items (E.g. :- Item [ ]).
(03 marks)

b) Food class should calculate the cost in each item and it should print the sub total of each order line and at the end it should print the total cost.

i). When calculate the line total it should multiply each item's cost with the given quantity and use iteration to print item name, cost, quantity and the line total as per the console output.

(06 marks)

Save the project as **Paper01D**

Question 2 (25 marks)

This question is based on the **exception handling** in a Doppler meter monitor the speed continuously and it basically check the speed in three states. If speed ≥ 100. known as "High speed", and if (speed < 100 && speed >= 50.0) known as "Normal speed" and if it is less than the 50.0 considered as "Too Slow".

a) Create the custom exception class **HighSpeedException** and with properties **message, and speed** and implement getter methods for them

(02 marks)

b) Create another three constants as private final String to store different states of speed (**HIGH_SPEED**, **NORMAL_SPEED**, and **TOO_SLOW**) and those should have values respectively ("High Speed". "Normal Speed". and "Too slow")

(03 marks)

c) Overload the constructor with speed, HighSpeedException (speed) **if(speed ≥ 100)** the message should be **"High speed"** and **if(speed < 100 && speed >= 50.0)** the message is **"Normal speed"** and display the message in console **"Program terminate"** else message should be **"Too slow".**

(05 marks)

d) Create the class **DopplerMeter** and implement the main() method and the monitorSpeed () method

i) Create a static method called monitorSpeed () and that should read the speed as the keyboard input.

(01 mark)

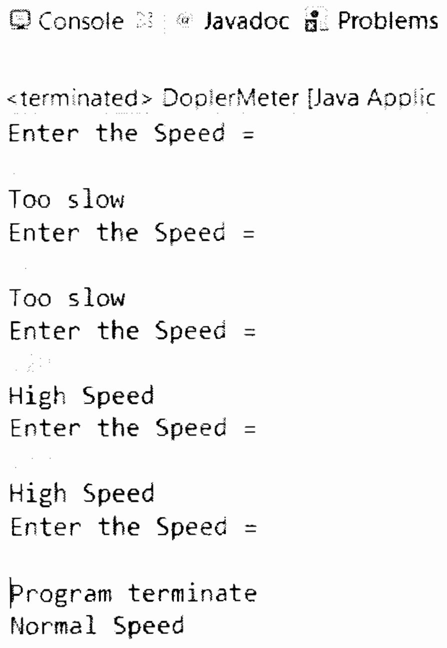
ii) Check the **speed >= 100 or speed < 50.0** or any other condition it should throw **HighSpeedException** with passing the speed as the parameter

(04 marks)

iii) In main() method you should invoke the monitorSpeed () and if it throws the exception it should **continuously read the key board inputs** until you enter the speed in **"Normal speed"** (value in between 50 to 100). Once you enter the speed within the range (50 to

100) you should **terminate the running process**. Refer the below screenshot as well when formulating the answer

(10 marks)



Console Javadoc Problems

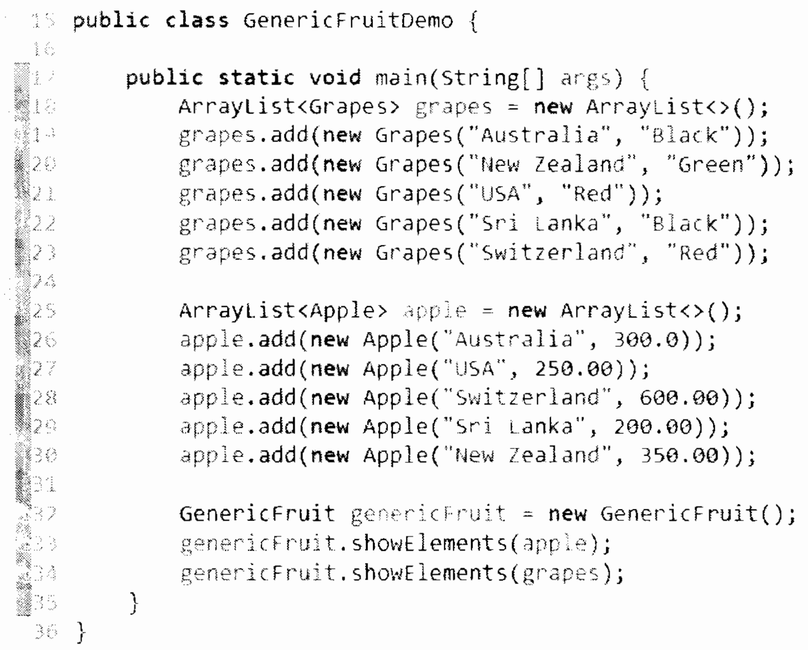
```
<terminated> DoplerMeter [Java Applic
Enter the Speed =

Too slow
Enter the Speed =

Too slow
Enter the Speed =

High Speed
Enter the Speed =

High Speed
Enter the Speed =

Program terminate
Normal Speed
```

Save the project as **Paper02D**

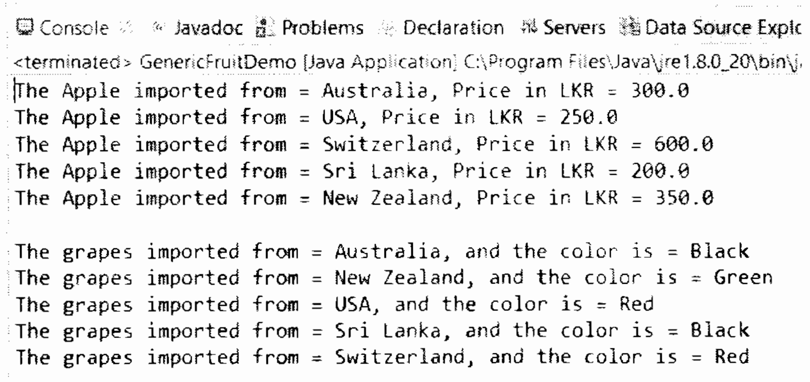
Question 3 (20 marks)

This question is based on the **Collection Framework and Generics**.

a) You should implement an array list of Apples and Grapes and use one Generic class called **GenericFruit** to display elements in both array lists. Please refer the **GenericFruitDemo** Test class and its execution output to fine-tune your results.



```java
public class GenericFruitDemo {

    public static void main(String[] args) {
        ArrayList<Grapes> grapes = new ArrayList<>();
        grapes.add(new Grapes("Australia", "Black"));
        grapes.add(new Grapes("New Zealand", "Green"));
        grapes.add(new Grapes("USA", "Red"));
        grapes.add(new Grapes("Sri Lanka", "Black"));
        grapes.add(new Grapes("Switzerland", "Red"));

        ArrayList<Apple> apple = new ArrayList<>();
        apple.add(new Apple("Australia", 300.0));
        apple.add(new Apple("USA", 250.00));
        apple.add(new Apple("Switzerland", 600.00));
        apple.add(new Apple("Sri Lanka", 200.00));
        apple.add(new Apple("New Zealand", 350.00));

        GenericFruit genericFruit = new GenericFruit();
        genericFruit.showElements(apple);
        genericFruit.showElements(grapes);
    }
}
```



Console Javadoc Problems Declaration Servers Data Source Explc

```
<terminated> GenericFruitDemo [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\j
The Apple imported from = Australia, Price in LKR = 300.0
The Apple imported from = USA, Price in LKR = 250.0
The Apple imported from = Switzerland, Price in LKR = 600.0
The Apple imported from = Sri Lanka, Price in LKR = 200.0
The Apple imported from = New Zealand, Price in LKR = 350.0

The grapes imported from = Australia, and the color is = Black
The grapes imported from = New Zealand, and the color is = Green
The grapes imported from = USA, and the color is = Red
The grapes imported from = Sri Lanka, and the color is = Black
The grapes imported from = Switzerland, and the color is = Red
```

i). Implement an interface **IFruit** and declare the method **displayFruitDetails ()** should return the output in **String** type.

(02 marks)

ii). Create a class called **Apple** and implement the two properties called **country** (String) and **price** (double) and values should be assigned through the **overloaded constructor**.

(02 marks)

iii). Implement the **IFruit** interface in the **Apple** class and override the method **displayFruitDetails ()** to print the country and the price.

(02 marks)

iv). Create a class called **Grapes** and implement the two properties called **country** (String) and **color** (String) and the values should be assigned through the **overloaded constructor**.

(02 marks)

v). Implement the **IFruit** interface in the **Grapes** class and override the method **displayFruitDetails ()** to print the **country** and the **color**.

(02 marks)

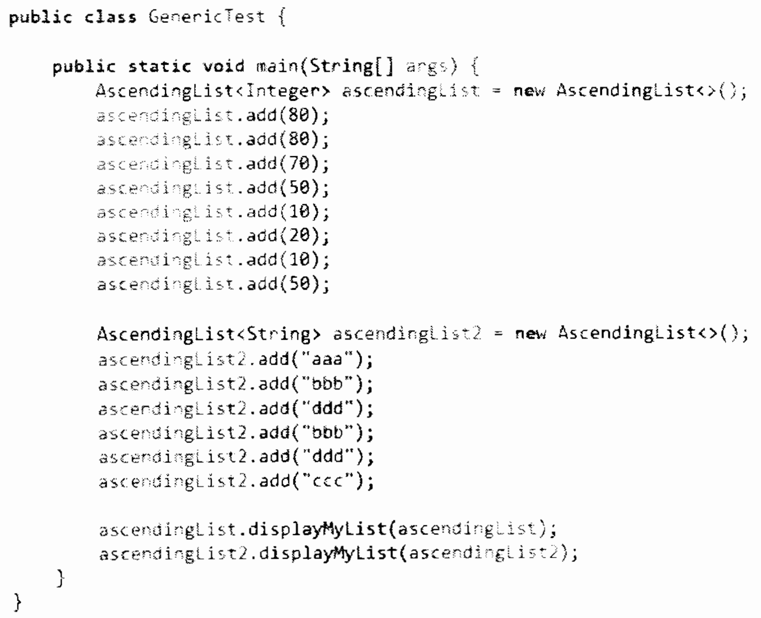
vi). Now create the generic class called **GenericFruit** and implement the method **showElements** should support passing **generic array list** (either Grapes array list or Apples array list). The **showElements ()** method should have an iteration and within the iteration, the each element should call the **displayFruitDetails** () method to print the Grapes and Apple details as per the given output.

(05 marks)

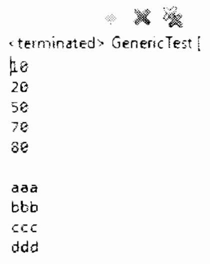
b) You should create a class called **AscendingList** and that should store list of elements. Elements should be stored according to the **Ascending order** and it should **remove all duplicate elements** as well.

i). Implement the method called **displayMyList()** it should print elements according to the ascending order. Refer the **GenericTest** class and the console output to adjust your results accordingly

(05 marks)



```
public class GenericTest {

    public static void main(String[] args) {
        AscendingList<Integer> ascendingList = new AscendingList<>();
        ascendingList.add(80);
        ascendingList.add(80);
        ascendingList.add(70);
        ascendingList.add(50);
        ascendingList.add(10);
        ascendingList.add(20);
        ascendingList.add(10);
        ascendingList.add(50);

        AscendingList<String> ascendingList2 = new AscendingList<>();
        ascendingList2.add("aaa");
        ascendingList2.add("bbb");
        ascendingList2.add("ddd");
        ascendingList2.add("bbb");
        ascendingList2.add("ddd");
        ascendingList2.add("ccc");

        ascendingList.displayMyList(ascendingList);
        ascendingList2.displayMyList(ascendingList2);
    }
}
```



```
<terminated> GenericTest [
10
20
50
70
80

aaa
bbb
ccc
ddd
```

Save the project as **Paper03D**

Question 4 (25 marks)

This question is based on the **Design Patterns** implementation. Read the following scenario and draw the detailed design class diagram of the following design pattern. [**You should use the correct UML notation to get the full marks**]

a) You are going to Design the Strategy Design Pattern based on the scenario for meal preparation of a Restaurant. You can prepare three meals for the day (**Breakfast, Dinner, and Lunch**) with time duration of **(60 minutes, 45 minutes, and 30 minutes**).

i). Design two interfaces **IPrepareQuickly** and **IPrepareDeliciously**. Each interface you should declare methods (in **IPrepareQuickly** interface declare the method **void deliveryTime()** and in **IPrepareDeliciously** interface declare methods **void addFlavour()** and **double getCost()**)

(02 marks)

ii). Then design 3 classes **ChickenFlavour**, **FishFlavour**, and **EggFlavour** and those classes should implement the **IPrepareDeliciously** interface and override all methods with in the class.

(06 marks)

iii). Similarly design another 3 classes **OneHour**, **ThirtyMinutes**, and **FourtyFiveMinutes** and those classes should implement the **IPrepareQuickly** interface and override the method as well.

(03 marks)

iv). Design and draw an **Abstract** class called **Meal** and draw aggregation relationship for two interfaces (**IPrepareQuickly**, and **IPrepareDeliciously**), you should set those two behaviors with using two set methods **setFlavour()** and **setDuration()**. (Those "set" methods are used to dynamically add prepare **quickly** and **prepare deliciously** features to meal)

(06 marks)

b) Now for the above three meals you can add different flavors such as **chicken, fish**, and **egg** and **assume you can't add more than one flavor per meal**.

i). Then design another two methods called **mealWithFlavour()**, and **mealInDuration()** and you should call relevant **addFlavour()** and **deliveryTime()** method respectively through the declared interfaces of the Meal class

(02 marks)

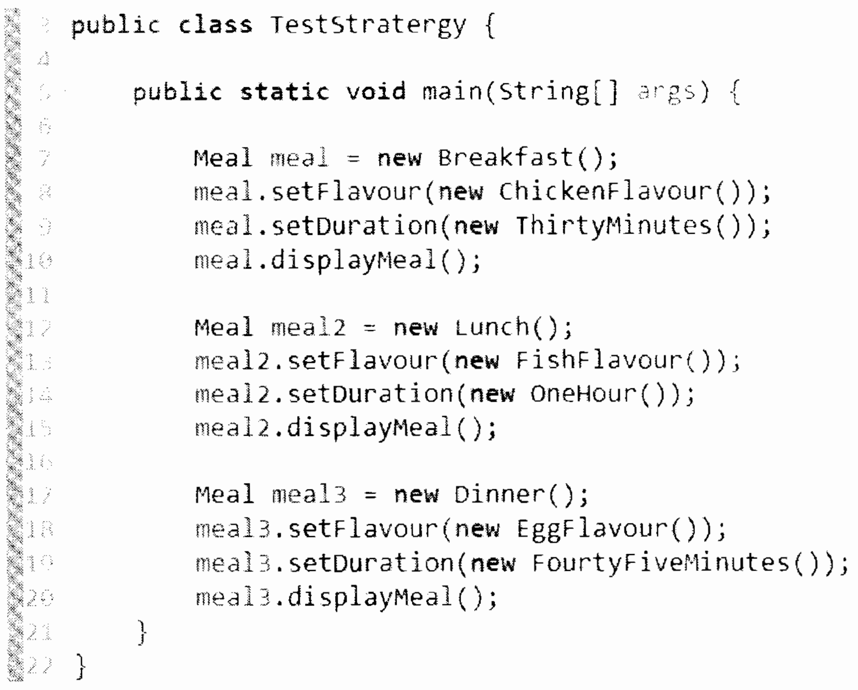
ii). Apart from that with in the **Meal** class you should add two **abstract** methods **displayMeal() and displayCost()**

(01 mark)

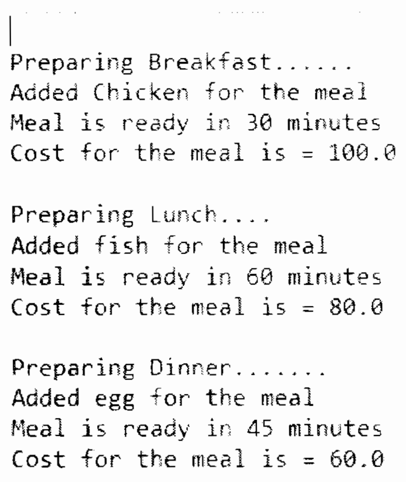
iii). Now **extends** the **Meal** class in the **Breakfast, Lunch and Dinner** classes and draw your diagram.

(05 marks)

iv). Please refer the **output of the test class** when you design your diagram.



```java
public class TestStratergy {

    public static void main(String[] args) {

        Meal meal = new Breakfast();
        meal.setFlavour(new ChickenFlavour());
        meal.setDuration(new ThirtyMinutes());
        meal.displayMeal();

        Meal meal2 = new Lunch();
        meal2.setFlavour(new FishFlavour());
        meal2.setDuration(new OneHour());
        meal2.displayMeal();

        Meal meal3 = new Dinner();
        meal3.setFlavour(new EggFlavour());
        meal3.setDuration(new FourtyFiveMinutes());
        meal3.displayMeal();
    }
}
```



```
Preparing Breakfast......
Added Chicken for the meal
Meal is ready in 30 minutes
Cost for the meal is = 100.0

Preparing Lunch....
Added fish for the meal
Meal is ready in 60 minutes
Cost for the meal is = 80.0

Preparing Dinner.......
Added egg for the meal
Meal is ready in 45 minutes
Cost for the meal is = 60.0
```

Design your UML diagram in below space and you should submit the paper at the end of the exam.

COMPULSORY TO FILL BEFORE STARTING THE EXAM

Student ID :

Student Name:-

Machine No :-

Machine IP Address :-

Location :-

Question Number	Marks
Q1	
Q2	
Q3	
Q4	
TOTAL	

Evaluated Lecturer :- ....

Signature:- ....

End of The Examination Paper