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 C

Duration: 3 Hours

October 2019

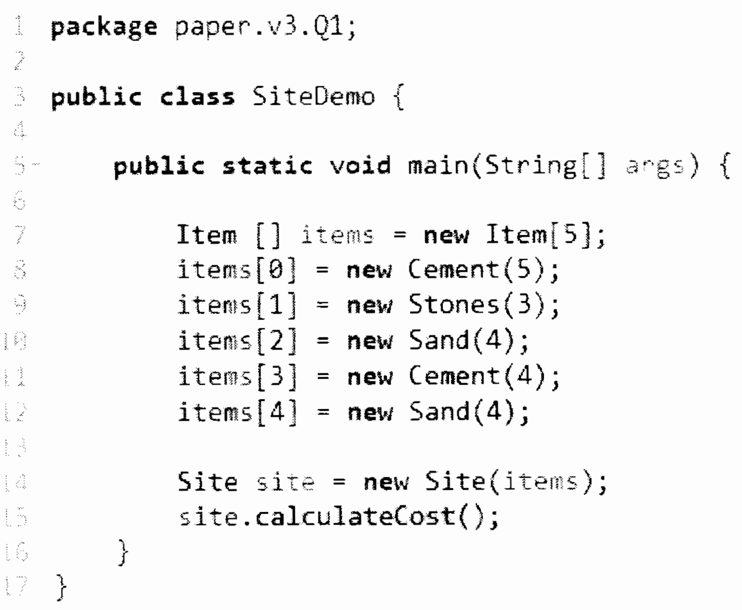
Instructions to Candidates:

- 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 items for the Site (Cement, Sand, Stones) in deferent quantities of cubic meters. Quantities are passed through the constructor.

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



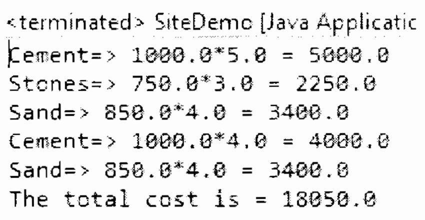
```java
package paper.v3.Q1;

public class SiteDemo {

    public static void main(String[] args) {

        Item [] items = new Item[5];
        items[0] = new Cement(5);
        items[1] = new Stones(3);
        items[2] = new Sand(4);
        items[3] = new Cement(4);
        items[4] = new Sand(4);

        Site site = new Site(items);
        site.calculateCost();
    }
}
```



```
<terminated> SiteDemo [Java Applicatic
Cement=> 1000.0*5.0 = 5000.0
Stones=> 750.0*3.0 = 2250.0
Sand=> 850.0*4.0 = 3400.0
Cement=> 1000.0*4.0 = 4000.0
Sand=> 850.0*4.0 = 3400.0
The total cost is = 18050.0
```

i). First implement the **IConstruction** 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 **Cement, Sand** and **Stones** and override all abstract methods **displayItem( ), getCost()** and **getQuality( )** .
(3 x 3 = 09 marks)

iv). Add the property qtyInCubes and overload the constructor of each class (**Cement, Sand** and **Stones)** and use the data type double.
(2 x 3 = 06 marks)

v). Similarly create a class called **Site** and implement the **IConstruction** 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) Site 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 **Paper01C**

Question 2 (25 marks)

This question is based on the **exception handling** in a sensor station monitor the temperature continuously and it basically check the temperature in three states. If temperature < 0.0. known as "freeze state", and if (temperature > 0.0 && temperature < 40.0) known as "Normal state" and if it increases beyond the 40.0 considered as "Danger state".

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

(02 marks)

b) Create another three constants as private final String to store (**FREEZE_STATE, NORMAL_STATE**, and **DANGER_STATE**) and those should have values respectively ("Freeze state", "Normal state", and "Dangerous for Health")

(03 marks)

c) Overload the constructor with temperature, TemperatureException (temperature) **if(temperature <= 0.0)** the message should be **"Freeze state"** and **if(temperature > 0.0 && temperature < 40.0)** the message is **"Normal state"** and display the message in console **"Program terminate"** else message should be **"Dangerous for Health".**

(05 marks)

d) Create the class **SensorStation** and implement the main() method and the monitorTemperature () method

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

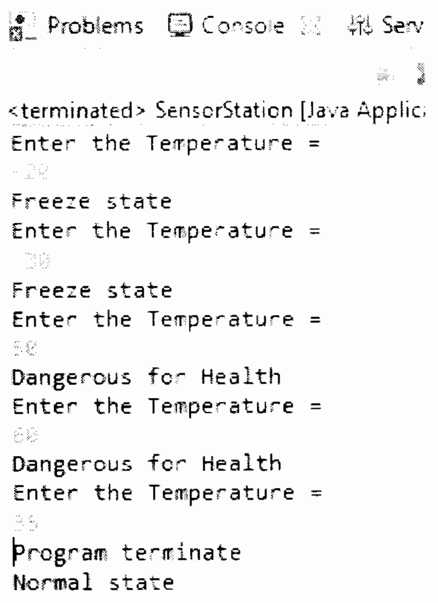
   (01 mark)

   ii) Check the **temperature < 0.0 or temperature > 40** or any other condition it should throw **TemperatureException** with passing the temperature as the parameter

   (04 marks)

iii) In main() method you should invoke the monitorTemperature() and if it throws the exception it should **continuously read the key board inputs** until you enter the temperature in **"Normal state"** (value in between 0 to 40). Once you enter the temperature within the range (0 to 40) you should **terminate the running process**. Refer the below screenshot as well when formulating the answer

(10 marks)



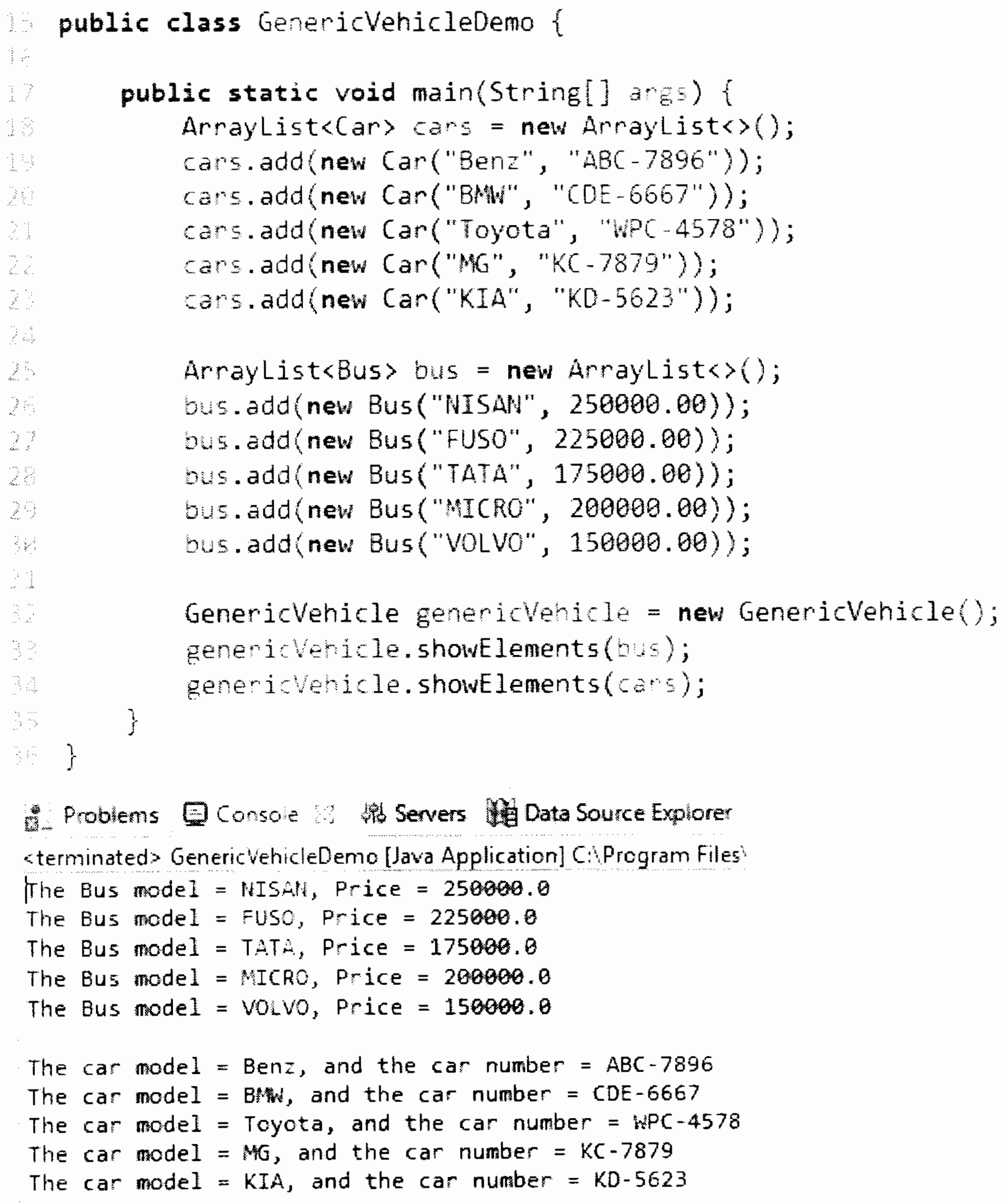
```
<terminated> SensorStation [Java Applic
Enter the Temperature =
-20
Freeze state
Enter the Temperature =
-30
Freeze state
Enter the Temperature =
50
Dangerous for Health
Enter the Temperature =
60
Dangerous for Health
Enter the Temperature =
35
Program terminate
Normal state
```

Save the project as **Paper02C**

Question 3 (20 marks)

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

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



```java
public class GenericVehicleDemo {

    public static void main(String[] args) {
        ArrayList<Car> cars = new ArrayList<>();
        cars.add(new Car("Benz", "ABC-7896"));
        cars.add(new Car("BMW", "CDE-6667"));
        cars.add(new Car("Toyota", "WPC-4578"));
        cars.add(new Car("MG", "KC-7879"));
        cars.add(new Car("KIA", "KD-5623"));

        ArrayList<Bus> bus = new ArrayList<>();
        bus.add(new Bus("NISAN", 250000.00));
        bus.add(new Bus("FUSO", 225000.00));
        bus.add(new Bus("TATA", 175000.00));
        bus.add(new Bus("MICRO", 200000.00));
        bus.add(new Bus("VOLVO", 150000.00));

        GenericVehicle genericVehicle = new GenericVehicle();
        genericVehicle.showElements(bus);
        genericVehicle.showElements(cars);
    }
}
```

```
<terminated> GenericVehicleDemo [Java Application] C:\Program Files\
The Bus model = NISAN, Price = 250000.0
The Bus model = FUSO, Price = 225000.0
The Bus model = TATA, Price = 175000.0
The Bus model = MICRO, Price = 200000.0
The Bus model = VOLVO, Price = 150000.0

The car model = Benz, and the car number = ABC-7896
The car model = BMW, and the car number = CDE-6667
The car model = Toyota, and the car number = WPC-4578
The car model = MG, and the car number = KC-7879
The car model = KIA, and the car number = KD-5623
```

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

(02 marks)

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

(02 marks)

iii). Implement the **IVehicle** interface in the **Bus** class and override the method **showVehicleDetails ()** to print the model and the price.

(02 marks)

iv). Create a class called **Car** and implement the two properties called **model** (String) and **number** (String) and the values should be assigned through the **overloaded constructor**.

(02 marks)

v). Implement the **IVehicle** interface in the **Car** class and override the method **showVehicleDetails ()** to print the **model** and the **number**.

(02 marks)

vi). Now create the generic class called **GenericVehicle** and implement the method **showElements** should support passing **generic array list** (either Cars array list or Busses array list). The **showElements ()** method should have an iteration and within the iteration, the each element should call the **showEmployeeDetails**() method to print the Car and Bus 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

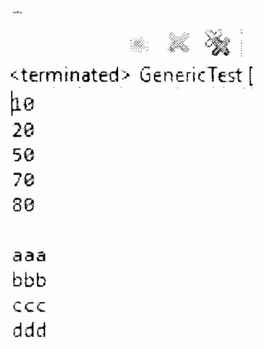
(05 marks)

```java
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 **Paper03C**

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 implement the Strategy Design Pattern based on the university degree programs (**PhDPrograms, MScPrograms, and BScPrograms**).

i). Design two interfaces **IFestival** and **IPrograms**. Each interface you should declare methods (in **IFestival** interface declare the method **void performEvent()** and **double getBudget()** and in **IPrograms** interface declare methods **void offerPrograms(), double getCost()**)

(02 marks)

ii). Then design 3 classes **RoboFest**, **GameFest**, and **CodeFest** and those classes should implement the **IFestival** interface and override all methods with in the class.

(06 marks)

iii). Similarly design another 3 classes **PhDPrograms**, **MScPrograms**, and **BScPrograms** and those classes should implement the **IPrograms** interface and override necessary methods as well.

(06 marks)

iv). Design an **Abstract** class **Students** and draw aggregation relationship for two interfaces **(IFestival,** and **IPrograms),** you should set those two behaviors with using two set methods **setFestival ( )** and **setPrograms( )**. (Those "set" methods are used to dynamically add festivals and degree programs features to Students)

(06 marks)

b) Now for the above two student types you can add different events such as **RoboFest**, **CodeFest**, and **GameFest** and based on the event budget should be different and **assume you can't add more than one event for Student**.

i). Then design another two methods called **offerPrograms (),** and **conductEvents ()** methods respectively through the declared interfaces of the Student class

(02 marks)

ii). Apart from that within the **Students** class you should have two **abstract** methods **displayStudents() and displayCost()**

(02 marks)

iii). Now **extends** the **Students** class in the **UndergraduateStudents, PostGraduateStudents** classes and override abstract methods

(02 marks)

iv). Please refer the **output of the test class** when you draw the **class Diagram**.

```java
public class TestStratergy {

    public static void main(String [] args){

        Students poStudents = new PostGraduateStudents();
        poStudents.setFestival(new CodeFest());
        poStudents.setPrograms(new MScPrograms());
        poStudents.displayStudents();

        System.out.println();

        Students unStudents = new UndergraduateStudents();
        unStudents.setFestival(new RoboFest());
        unStudents.setPrograms(new BScPrograms());
        unStudents.displayStudents();
    }
}
```

```
Offer MSc Programs
Perform CodeFest Event for 300000.0
Cost for the postgraduate program is = 500000.0
Display Post gratuate students

Offer BSc degree programs
Perform Robo Fest Event for 600000.0
Cost for the undergraduate program is = 120000.0
Display under gratuate students
```

Design your UML diagram in below space and you should sbmit 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