00269

Student ID:

Machine No:

Sri Lanka Institute of Information Technology

B.Sc. Honours Degree in Information Technology

Specialized in Information Technology

Final Examination (Computer Based)
Year 2, Semester 1 (2023)
Paper Version F

IT2030 – Object Oriented Programming

Duration: 3 Hours

November 2023

Instructions to Candidates:

- This paper contains four questions. Answer All Questions.
- Marks for each question are given on the paper. Total Marks: 100.
- Create a folder on the Desktop with your student registration number and store all your program files inside that.
- Create a separate Project for each question. The name of the project is provided in the question.
- This paper contains 06 pages including the Cover Page.

Instructions to Candidates when submitting:

- Save all your work.
- Delete all unnecessary files from each project folder (There should be 4 folders for 4 projects named as Question01, Question02, Question03 and Question04 inside your ID folder, and each folder should contain the source code (.java files)) only.
- Zip the Student ID folder (Zipped folder also should be named with Student ID number).
- Upload the zipped folder to the correct link.

Question 1 (30 marks)

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

a) Write a Java program that allows the user to manage a list of books in a library using a PriorityQueue. The program should perform the following tasks.

i) Create an empty PriorityQueue of book titles (Strings) to store the book names. Prompt the user to enter the titles of books one by one. Continue accepting book titles as keyboard inputs until the user enters "done." Add each entered book title to the created PriorityQueue. (04 marks)

ii) Once the user has finished entering book titles, ask the user to enter the title of the book they want to remove. If the book title is found in the PriorityQueue, remove it. Display a message indicating whether the book was successfully removed or not, in the case the title was not found. (06 marks)

iii) Find and display the total number of books in the PriorityQueue. (02 marks)

iv) Print all the remaining book titles in the PriorityQueue. (03 marks)

Save the project as **Question01A**

b) Create a Java program that manages the inventory of products using a HashMap. The program should allow user to add products, update their quantities, remove products, and view the current inventory.

i) Create a HashMap to manage the inventory, where the keys are unique product codes (strings) and the values are objects representing each product. Each product object should include the product name (string), the quantity in stock (integer), and the price per unit (a double) (06 marks)

ii) Implement a method to add a product to the inventory. The method should take the product code, product name, quantity, and price per unit as parameters. If a product with the same code already exists, update its details. (04 marks)

iii) Create a method to remove a product from the inventory based on the product code. (02 marks)

c) In the main method, create a sample inventory with 5 products, and display the product details. Then remove 2 products and display the remaining products again. (03 marks)

Save the project as **Question01B**

Question 2 (30 marks)

This question is based on the **Design Patterns** implementation.

a) You are going to implement the **Abstract Factory Pattern** based on the scenario of **Vehicle** factory that turns out Cars or Busses for the specified model. Refer the Screenshot of the Console Output and the **VehicleDemo** class. You should implement the rest of the other classes.

i). Implement **VehicleProducer** class that returns either **CarFactory** or **BusFactory** outputs through **getVehicle()** operation. Refer the screenshot in Demo class. (02 marks)

ii). Design **CarFactory** and **BusFactory** classes according to the **Singleton Pattern** and let **getModel(String vehicle)** select the factories.

(06 marks)

iii). Design 03 classes for Car factory called (**BMW, Benz, and RollsRoys**) according to the **singleton pattern**. Each car should have a **displayVehicle()** method you have to override.

(03 marks x 03 = 09 marks)

iv). Create Design 03 classes for Bus factory called (**Volvo, Fuso, and TATA**) according to the **singleton pattern**. Each bus should have a **displayVehicle()** method you must override.

(03 marks x 03 = 09 marks)

b) Now design three interfaces to **IBus**, **ICar**, and **VehicleFactory** to override common behaviors.

i). Now design **IBus** and **ICar** interfaces and declare the **displayVehicle()** operation in each.

(01 mark x 2 = 02 marks)

ii). Design **VehicleFactory** as an Interface and declare the **getModel** operation to select vehicle option either Bus or Car

(02 marks)

Hint: Refer to the main program and package hierarchy given below

Save the project as **Question02**

Main Program:

```
3  public class VehicleSelectionDemo {
4
5      private static final String CAR = "Car";
6      private static final String BUS = "Bus";
7
8      public static void main(String[] args) {
9
10         ((ICar) VehicleProducer.getVehicle(CAR).getModel("RollsRoys")).displayVehicle();
11         ((ICar) VehicleProducer.getVehicle(CAR).getModel("Benz")).displayVehicle();
12         ((ICar) VehicleProducer.getVehicle(CAR).getModel("BMW")).displayVehicle();
13
14         ((IBus) VehicleProducer.getVehicle(BUS).getModel("Volvo")).displayVehicle();
15         ((IBus) VehicleProducer.getVehicle(BUS).getModel("Fuso")).displayVehicle();
16         ((IBus) VehicleProducer.getVehicle(BUS).getModel("TATA")).displayVehicle();
17     }
18 }
```

Console | Problems | Javadoc | Declaration

<terminated> VehicleSelectionDemo [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Sep 29, 2023, 1:26:40 PM)

```
Factory turns out RollsRoys Car
Factory turns out Benz Car
Factory turns out BMW Car
Factory turns out Volvo Bus
Factory turns out Fuso Bus
Factory turns out TATA Bus
```

Package Hierarchy:

- oop.exam.q1
  - Benz.java
  - BMW.java
  - BusFactory.java
  - CarFactory.java
  - Fuso.java
  - IBus.java
  - ICar.java
  - RollsRoys.java
  - TATA.java
  - VehicleFactory.java
  - VehicleProducer.java
  - VehicleSelectionDemo.java
  - Volvo.java

Question 3 (20 marks)

This question is based on the **Threads implementation**.

You are tasked with simulating a resource allocation system for a multi-user environment. The system manages a set of resources, and users can request to allocate and deallocate these resources from the Resource Manager.

a) Create a class named **ResourceManager** that manages a pool of resources. (02 marks)

   i). Implement the requestResource() method where a user can request a resource. If a resource is available, it is allocated to the user. If not, the user waits until a resource becomes available. (04 marks)

   ii). Implement the releaseResource() method where users can request to release the resource they have acquired. (04 marks)

b) Create a class called **User** that simulates a user. Each user should simulate users requesting to acquire and release resources from a Resource Manager with a sleeping time of 1000 milliseconds in between acquiring and releasing. Each user needs to have a unique userId. (06 marks)

c) The main class should initialize the **ResourceManager** object with 3 resources and 5 user threads, and the program should display messages indicating when users request and release resources. Refer to the partial code of the Main class and the output below. (4 marks)

```
public class Main {
    public static void main(String[] args) {
        int TResources = 3;
        int TUsers = 5;
        ResourceManager rm = new ResourceManager(TResources);

        for (int i = 1; i <= TUsers; i++) {
            UserThread userThread = new UserThread(i, rm);
            userThread.start();
        }    }     }
```

Save the project as **Question03**

Sample Output:

```
User 1 acquires Resource 0
User 5 acquires Resource 1
User 4 acquires Resource 2
User 3 is waiting for a resource.
User 2 is waiting for a resource.
User 4 releases Resource 0
User 2 acquires Resource 0
User 3 is waiting for a resource.
User 1 releases Resource 0
```

Question 4 (20 marks)

This question is based on the **OOP concepts**.

Create an abstract class called "**Person**" with attributes as ID (int) and a name (String). This class should also include a default constructor and an overloaded constructor to initialize these attributes. Additionally, add an abstract method called "DisplayDetails()". Implement another abstract class called "**Course**" class with attributes such as code (String) and a course name (String). It should include a default constructor and an overloaded constructor to initialize these attributes. Add an abstract method called "DisplayCourseDetails()".

The concrete "**Student**" class should be inherited from the "**Person**" class. It should have an additional attribute as enrolledCourses which can hold a list to hold the enrolled courses. This class should include a default constructor and an overloaded constructor to initialize the attributes. Additionally, implement the following methods:

- EnrollInCourse(Course c): This method should allow students to enroll in a course by adding the course to their list of enrolled courses.
- DisplayDetails(): This method should override the abstract method from the "Person" class to display the student's details, including their ID, name, and the list of enrolled courses.

The concrete "**Teacher**" class should also be inherited from the "**Person**" class and have an attribute called coursesTaught which can hold a list of courses they teach. This class should include a default constructor and an overloaded constructor to initialize the attributes. Additionally, implement the following methods:

- TeachCourse(Course c): This method should allow teachers to add a course to their list of courses they teach.
- DisplayDetails(): This method should override the abstract method from the "Person" class to display the teacher's details, including their ID, name, and the list of courses they teach.

Implement a class called "**InstitutionManagementApp**" with the main method. In the main method, create 2 instances each of "Student" and "Teacher" classes and initialize their attributes using the respective constructors. Allow 3 students to enroll in courses and teachers to add courses they teach. Display the details of students, teachers, and courses using the provided methods, taking advantage of the abstract class structure for common attributes and methods.

Save the project as Question04