COMSATS UNIVERSITY ISLAMABAD (CUI)

DEPARTMENT OF COMPUTER SCIENCE
TERMINAL EXAMINATION SPRING - 2024

Course: CSC241-Object Oriented Programming
Maximum Marks: 50
Instructor: Mohsin Ahmed

Dated: 14-JUN-2024
Time Allowed: 180 Minutes
Class: BCS/BSE/BDS/BAI/BCT

Instructions:

- *All questions are self-explanatory and require no further explanations during exam time.*
- *Make sure that you have signed the attendance sheet before leaving the examination room.*

[CLO – 1: Demonstrate Fundamental Principles and Concepts of Object-Oriented Programming (OOP) programming.]

Question-1: Do as directed **[Marks 2+2+4+4]**

A. Examine the following code and report if there are any errors.

```
public class Test {
    public int a;
    public int b;
    public static int c;
    Test()
    { a=5; b=6; c=8;    }
    public String toString()
    {   return a+ " "+ b+ " "+ c;    }
   public static boolean  method1 (Test obj)
   {
       if(this.c == obj.c)
           return true;
       else
       return false;
    }
}
```

B. Which of the following fits the definition of immutable class?

i.
`class Person{`
`Age a; String name;`
`String getName(){return name;};`
`Age getAge(){return a};`
`}`
`class Age{ int y, m,d; }`
	ii.
	`class Person{`
	`Age a; String name;`
	`String getName(){return name;};`
	`void setAge(Age a){ this.a = a;}`
	`}`
	`class Age{ private int y, m, d; }`

iii.
`class Person{`
`private Age a; String name;`
`String getName(){return name;};`
`Age getAge(){return a};`
`}`
`class Age{ private int y,m,d; }`
	iv.
	`class Person{`
	`private Age a; String name;`
	`String getName(){return name;};`
	`}`
	`class Age{ int y,m,d; }`

C. **Consider the following class and provide method headers, specifying the appropriate access specifier, return type, and parameter(s).**

```
public class BookStore {
    String name;
    String[] books = new String[100];
    int [] price = new int[100];

    public BookStore(String name) {
        this.name = name;
    }
    _______ addABookinStore( ? )
    _______ isFullyOccupied (?)
    _______ validateData(?)
    _______ sellABook( ?)
    _______ findCostofTotalBooksInShop(?)
    _______ compareTwoBookStores( ?)
}
```

D. **Write output of executing the following Runner class.**

```
public class Distance{
   private  int foot;
   private int inches;
   public Distance(){
        this.foot = 3;
        this.inches = 11;
    }
   public Distance(int foot, int inches) {
        this.foot = foot;
        this.inches = inches;
    }
   public Distance(Distance obj) {
        this.foot = obj.foot;
        this.inches = obj.inches;
    }
    public void setFoot(int foot) {
        this.foot = foot;
    }
    public void setInches(int inches) {
        this.inches = inches;
    }
 public void addTwoDistances(Distance obj)
    {
        foot = foot + obj.foot;
        inches = inches + obj.inches;
    }
        public void display() {
        System.out.println(foot + "   " +
incnes);

    }
}
```

```
public class Runner {

  public static void main(String[] args){

   Distance [] d_array = new Distance[4];
   d_array[0] = new Distance(4,10);
   d_array[1] = new Distance();
   d_array[2] = new Distance(d_array[0]);
   d_array[3] = d_array[1];

   d_array[0].setFoot(7);
   d_array[1].setFoot(15);
   d_array[2].setInches(20);
   d_array[3].setInches(30);

   for(int i=0; i<d_array.length;i++) {
            d_array[i].display();
   }

System.out.println("===============");


d_array[0].addTwoDistances(d_array[1]);
d_array[0].display();
d_array[1].display();


    }

}
```

[CLO – 2: Apply the concepts of object-oriented programming principles along with interfaces and exception handling to solve a real-world problem.]

Question-2: **[Marks 8]**

Create a class diagram for a food delivery application with following specifications:

The application allows customers to browse various restaurants and place orders. Customers can log in, place orders, and leave reviews for restaurants. Restaurants manage their menus by adding, updating, or removing menu items. Each menu item has an associated cost. An order, placed by a customer, can consist of one or more menu items from a restaurant. Customers have the option to pay for their orders either online or with cash on delivery. Delivery personnel log in to accept delivery tasks, update the delivery status, and ensure timely delivery of orders to customers.
Constraint: All entities within the application must have the capability to update and display their profiles. The UML class diagram should adhere to all principles of the object-oriented paradigm.

Question-3: **[Marks 10]**

You are required to implement an inventory application for a medical store (Pharmacy) by using inheritance and polymorphism principles. Consider the following scenario.

There are two types of Items, Perishable, and Permanent. Every item has a name and an original price. Every Perishable item has an expiry date and quantity. The perishable item's price remains the same before the expiry date and becomes zero after it. Every Permanent item's price is depreciated daily by 0.02% since the date of entry into the pharmacy to the present date.
Additionally, Bulk items are permanent items and contain two extra attributes i.e. description and inPackage (yes/no). The price of a bulk item remains the same if it belongs to package category, if not, it depreciates similarly to the permanent items.
Create a class pharmacy which has a list of items. Create a constructor, a method to add items in the pharmacy, and another method that calculates the total cost of all items in the pharmacy in a polymorphic manner.
Note: You may assume a fully functional Date class is already available.

Question-4: **[Marks 5]**

As a developer you are working on creating a document formatting system that can format documents in different styles, such as PlainText and HTML. The system should be designed to allow for easy addition of new formatting styles in the future. To achieve this, you need to define a `DocumentFormatter` interface with a method `format(String text)`. Then, implement the `DocumentFormatter` interface in the following classes: `PlainTextFormatter` and `HtmlFormatter`. The `PlainTextFormatter` should return the text as plain text, prefixed with "PlainText: ". The `HtmlFormatter` should return the text wrapped in HTML tags (<html><body> and </body></html>). Additionally, write a class Document that has a constructor taking a `DocumentFormatter` and a String representing the content. The Document class should have a method `getFormattedContent()` that returns the formatted content using the provided `DocumentFormatter`.
To test the implementation, create instances of `PlainTextFormatter` and `HtmlFormatter`. Then, create a Document instance with each formatter and display the formatted content. The formatted content should be displayed appropriately for each formatter.

Question-5: **[Marks 5]**

You are required to create a generic class `OrderedPair` definition that should be flexible enough to handle any type of objects of the same type. This generic class will be useful in various situations e.g. representing key-value pairs, or any two related pieces of data. The generic class should contain two private fields, constructor to initialize these fields. In addition to this, you should implement getter and setter methods and lastly override the `toString()` method. Also show the working of this `OrderedPair` class in the Runner class.

[CLO – 3: Create a Program using Standard Libraries.]

Question-6: **[Marks 10]**

You are developing a car show room management system with the help of below mentioned classes. The information about the showroom is stored in a file named "ShowRoom.ser".

ShowRoom
- carsArray: Car[] - owner:String

Car
- model: String - color:String - registartionNumber: String - dateOfManufacture: Date

A. Write a method that can help in searching all cars which have been manufactured in the last five years.

B. Write a method that helps in deleting all cars that are of Black color.

Note: Consider that classes of ShowRoom, Car and Date are already available and have constructors, set, get and toString methods.

*****BEST OF LUCK*****