United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Mid Term Exam. Trimester: Summer 2023

Course Code: CSE-1115, Course Title: Object Oriented Programming

Total Marks: 30, Duration: 1 hours 45 minutes

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules

1. Write the output of the following program. **[5]**

```
public class Vehicle {
    private String brand;
    private String model;

    static {
        System.out.println("Initializing Vehicle
class...");
    }

    {
        System.out.println("Initializing an instance of
the Vehicle class...");
    }

    public Vehicle() {
        System.out.println("Creating a default
vehicle.");
        brand = "Unknown";
    }

    public Vehicle(String brand, String model) {
        System.out.println("Creating a customized
vehicle of brand: " + brand + " and model: " +
model);
        this.brand = brand;
        this.model = model;
    }

   public void honk() {
        System.out.println("The vehicle emits a
honking sound.");
    }
   public void honk(String sound) {
        System.out.println("The vehicle emits a
custom honking sound: " + sound);
    }
    static{
        System.out.println("Making sure of
initialization...");
    }

    public void info(){
        System.out.println("model="+model+"
brand="+brand);
    }
    public static void main(String[] args) {
        Vehicle defaultVehicle = new Vehicle();
        defaultVehicle.honk();
        defaultVehicle.info();

        Vehicle truck = new Vehicle("Ford", "F-
150");
        truck.honk("Loud horn sound");
        truck.info();
    }
}
```

2. Consider the following class named ElectronicDevice representing a generic electronic device. **[5]**

```
public class ElectronicDevice {
   String brand;
   double price;

   public ElectronicDevice(String brand, double price) {
      this.brand = brand;
      this.price = price;
   }

   public void displayInfo() {
      System.out.println("Brand: " + brand);
      System.out.println("Price: $" + price);
   }
}
```

Now write the necessary codes to fulfill the requirements as follows:

1. Create a class named Smartphone - a child class of ElectronicDevice.
2. The Smartphone class has three additional attributes: model (String), operatingSystem (String), and IMEI (String). It must not be possible to set the value of IMEI outside of the class.
3. Create a constructor in Smartphone that takes brand, price, model, operatingSystem, and IMEI as arguments and sets the values of the attributes. This constructor invokes the constructor of ElectronicDevice.
4. There is a method named *displayInfo* in SmartPhone that shows the brand, price, model, and operatingSystem. The method invokes the *displayInfo* of ElectronicDevice.
5. There must be an option to fetch the IMEI outside of the class.

3. Suppose that you are assigned to compute the volumes of different geometrical objects, i.e., cylinder, sphere and cone. Thus, you are required to do the tasks systemically as follows: **[1+1+1+1+1+3 = 8]**

.

- Write a Java Class called **Myobject** that has a private member variable**: *r*** which represents radius of the shape. Add the following function: **findVolume()** that returns -1.0 (since no geometrical object is given).
- Write a child class **Sphere** from **Myobject**. Include the function findVolume() that computes the volume *v* of a sphere as follows:

$$v = \frac{4}{3}\pi r^3$$

- Write another child class **Cylinder** from **Myobject**. Include a private variable ***h*** and the function findVolume() that computes the volume *v* of a cylinder as follows:

$$v = \pi r^2 h$$

- Write a child class **Cone** from **Cylinder**. Include the function findVolume() that computes the volume *v* of a cone as follows:

$$v = \frac{1}{3}\pi r^2 h$$

- Add only the necessary Getter methods for each variable in the above classes. Make the necessary parameterized constructors to set the values of the variables.
- Now test your program from main by computing the volumes of different geometrical objects provided in Table 1.

Table 1: List of different geometrical objects and their radii and heights (if applicable)

Myobject	r	h
Sphere1	2.5	
Cone1	1.9	8.9
Cylinder1	1.5	6.5
Cone1	2.7	5.7
Sphere1	3.5	

Hint: Make 5 objects of Myobjects and use each child class reference to each of these objects according to Table 1 using the concept of heterogeneous collection. Next, sum up the volumes which can be obtained by calling the function **findVolume()** through each object**.**

4. Consider the following two classes and the output of the program. Read the comments carefully, **correct errors** in the code of the following **StaticContext** class, and **rewrite** the code for the **StaticContext** class. You can edit, add, or remove any lines excluding the commented ones. You can also write necessary constructors and blocks in the StaticContext class if required. **[6]**

```
package rollbar;

 //You can’t remove or modify this FinalContext class.
 public class FinalContext {

    public final void calculate(){
        System.out.println("calculate method is called");
    }
}
```

```
package rollbar;

public class StaticContext {
   final static int value; //You can't modify or remove this line of
code
   private double mark;
   private int count;

   @override
   public void calculate(){System.out.println("calculate method is
called");}

   private int getCount() {
       return ++count;
   }

   private static double getMark() {
       return mark;
   }
  // You can’t modify the following main method.
   public static void main(String... args) {
        count++;
        System.out.println("count= "+getCount());
        System.out.println("value = "+value);
        FinalContext sv = new StaticContext();
        System.out.println("mark= "+((StaticContext)sv).getMark());
        sv.calculate();
  }
}
```

Expected Outcome:

```
count= 2
value = 8
mark= 90.0
calculate method is called
```

5.1 **[1 x 3 = 3]**

a. Can a class be abstract and final simultaneously? Why or why not?

b. Abstract classes can be created without a single instance variable of method declared inside it. Can you think of any reason why you may want to create such an abstract class?

c. Why does a class with an abstract method need to be declared abstract? (Just answering "Compiler will give error" will not get you any marks)

5.2 **[1 x 3 = 3]**

a. Consider the following code:

```
abstract class Animal {}

class Baby extends Animal {

public double age;

}

class Cat extends Animal {

 public void sleep(int time) {

System.out.println("Sleeping");

  }

}
```

```
public class Main {

public void speak(Animal target) {

//write your code here

  }

}
```

Now, complete the speak method such that it prints (You cannot change any other part of the program other the speak method)

i. "WAAHHH" if the variable target is instance of the class Baby,

ii. "Meow" if the variable target is instance of the class Cat,

iii. "Grrrrr" if the variable target is instance of any other subclass of Animal

Question 01:

```
Initializing Vehicle class...
Making sure of initialization...
Initializing an instance of the Vehicle class...
Creating a default vehicle.
The vehicle emits a honking sound.
model=null brand=Unknown Initializing an instance of the Vehicle class…
Creating a customized vehicle of brand: Ford and model: F-150
The vehicle emits a custom honking sound: Loud horn sound
model=F-150 brand=Ford
```

Question 02:

```
public class Smartphone extends ElectronicDevice {
        String model;
        String operatingSystem;
        private String IMEI;
        public Smartphone(String brand, double price, String model, String
        operatingSystem, String IMEI) {
                super(brand, price);
                this.model = model;
                this.operatingSystem = operatingSystem;
                this.IMEI = IMEI;
        }

        public void displayInfo() {
                super.displayInfo();
                System.out.println("Model: " + model); System.out.println("Operating
                System: " + operatingSystem);
                }

        public String getIMEI() {
                return IMEI;
        }
}
```

Question 03:

```
public class Myobject {
        private double r;
        public double findVolume(){
                return -1.0;
                }
        public Myobject(double x){
                r = x;
        }
        public double getr(){
                return r;
        }
}


public class Sphere extends Myobject{
        public double findVolume(){
                double r1 = getr();
                double x = 3.0/4.0*Math.PI * Math.pow(r1, 3);
                return x;
                }
                public Sphere(double x){
                        super(x);
                }
}



public class Cylinder extends Myobject{
        private double h;
        public double findVolume(){
                double r1 = getr();
                double x = Math.PI * r1*r1*h;
                return x;
        }
        public Cylinder(double x, double y){
                super(x);
                h = y;
        }
        public double geth(){
                return h;
        }
}
```

```
public class Cone extends Cylinder{
        public double findVolume(){
                double r1 = getr();
                double h1 = geth();
                double x = Math.PI * r1*r1*h1/3.0;
                return x;
        }
        public Cone(double x, double y){
                super(x, y);
        }
}


public class JavaApplication5 {
        public static void main(String[] args) {
                Myobject [] sp = new Myobject[5];
                //r = 2.5, 1.9, 1.5, 2.7, 3.5;
                //h = 8.9, 6.5, 5.7
                sp[0] = new Sphere(2.5);
                sp[1] = new Cone(1.9, 8.9);
                sp[2] = new Cylinder(1.5, 6.5);
                sp[3] = new Cone(2.7, 5.7);
                sp[4] = new Sphere(3.5);
                double v = 0.0;
                double t ;

                for(int i = 0; i < 5; i++){
                        t = sp[i].findVolume();
                        System.out.println("Individual volume = " + t);
                        v += t;
                }
                System.out.println("Total volume: " + v);
        }
}
```

Question 04:

```
package rollbar;
public class FinalContext {
        public final void calculate(){
                System.out.println("calculate method is called");
        }
}


package rollbar;
public class StaticContext extends FinalContext {
        private final static int value;
        private double mark;
        private static int count;
        static {
                value = 8;
                count = 0;
        }
        StaticContext(){
                mark = 90;
        }
        private static int getCount() {
                return ++count;
        }
        private double getMark() {
                return mark;
        }

        // calculate method is removed

        public static void main(String... args) {
                count++;
                System.out.println("count= "+getCount());
                System.out.println("value = "+value);
                FinalContext sv = new StaticContext();
                System.out.println("mark= "+((StaticContext)sv).getMark());
                sv.calculate();
        }
}
```

Question 5.1:

a. No, because by making an abstract class final, you are also restricting it from being inherited by other classes. So, there is no way to be able to even create a subclass object which may be assigned to a variable of that abstract class.

b. We may create an abstract class without any instance variables or methods to create a class hierarchy, where we do not want to add any features to the top superclass

c. Since a method declared as abstract does not have any body, if an object of the class with an abstract method is created, it won’t be able to execute that abstract method and would give error. To prevent making the abstract method get called, we declare the entire class as abstract so that an object of that class cannot be initialized

Question 5.2:

```
public void speak(Animal target) {


        if (target instanceof Baby) {
                System.out.println("WAAHHH");
        }
        else if (target instanceof Cat) {
                System.out.println("Meow");
        }
        else if (target instanceof Animal) {
                System.out.println("Grrrr");
        }
}
```