United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Mid Term Exam, Trimester: Spring 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 expected output of the following code [4]

```
public class SomeClass {
  SomeClass(){
     System.out.println("I am the base constructor");
  }
  SomeClass(int a){
     this();
     System.out.println("I have an extra value: "+ a);
     this.someMethod(a);
  }
  SomeClass(int a, double b){
     this(a);
     System.out.println("I have more values: " + b);
  }
  public void someMethod(){
     System.out.println("I have no param");
  }
  public void someMethod(int c){
     System.out.println("I borrow "+c+" form a constructor");
  }
}
```

```
public class Main {
  public static void main(String[] args) {
     SomeClass s = new SomeClass(5,6.3);
     s.someMethod();
  }
}
```

2. Consider the following three classes and the output of the program. Then answer (a),(b) and (c).

```
package M1;

public class Human {
   private int id;
   String intelligence;
   protected boolean bravery;
}

public class Monster {
   String name;
   double weight;
```

```
  public void increaseWeight(double weight) {
      System.out.println("New weight of monster="+(this.weight+weight));
   }

   boolean eat(Human h){

 //check intelligence and return a boolean value
   }

   String scare(boolean bravery){
     //check bravery and return a string
   }
}
```

```
package M1.M2;

public class Test {
   public static void main(String[] args) {

      Human h1=new Human(1,"low",true);
      Human h2=new Human(2,"high",false);

      Monster m1= new Monster ("CookieMonster", 100);

      if (m1.eat(h1)==true) {
         System.out.println("Monster has eaten human "+h1.id);
      }else{
         System.out.println("Human escaped");
      }
      System.out.println(m1.scare(h2.bravery));
   }
}
```

Expected Outcome:

To avoid getting scared or eaten be brave or intelligent.
To avoid getting scared or eaten be brave or intelligent.
New weight of monster=110.0
Monster has eaten human 1
Human is too brave to scare.

(a) Correct the given code (don't modify the methods yet) by editing or adding any lines. You cannot remove any line from the code. Also write necessary getter methods (Assume the variables are read-only). 2]

(b) Observe the output given and write necessary constructors and blocks accordingly. [3]

(c) Implement the eat() and scare() methods. The **eat()** method checks the intelligence of a human- if it's "high", returns true, otherwise it calls the increaseWeight() method and then returns false. The **scare()**

method checks a human's bravery- if it's false, it prints a line: "Human scared." otherwise it will print: "Human is too brave to scare." [3]

3. Given the following information write the necessary code to implement AdvancedCalculator class. [6]

```
public class Calculator {
   public int a;
   public int b;
   Calculator(int firstNumber, int secondNumber){
      this.a = firstNumber;
      this.b = secondNumber;
   }
   public int sum(){
      return a+b;
   }
   public int subtract(){
      return a-b;
   }

}
```

```
public class AdvancedCalculator // is a Child of Calculator class

{



}
```

```
public class main {
   public static void main(String[] args) {
      AdvancedCalculator c = new AdvancedCalculator(1,2, new int[]{3, 4, 5});
      System.out.println(c.subtract());
      System.out.println(c.sum());
   }
}
```

Expected output

```
Subtraction result: -1
Summation result: 15
```

Following criteria must be fulfilled:

- AdvancedCalculator class has a constructor that uses this keyword to set the value of an attribute called numbers, which is an array of int type.

- Inside AdvancedCalculator class, override the sum() method from its parent. Then use the super keyword to utilize sum() method of its parent class Calculator to find out the sum of first two numbers. Then, add additional lines of code to add the elements of numbers[] with the sum and return the total summation.

Hints:

- 1st line of Expected outcome is the subtraction of 1st two numbers 1 and 2.
- 2nd line of Expected outcome is the summation of 1st two numbers 1 and 2 along with all the elements of **numbers[]** which are 3,4 and 5.

4. Answer Following Questions: [3]

a. Write down the output of the following Code:

```
public class Parent {
    public static int count = 0;
    public static void printDetails(){
        count++;
        System.out.println("I am in Parent Class: " + count);
    }
}
public class Child extends Parent{
    public static void printDetails()
    {
        count++;
        System.out.println("I am in a Child Class: " + count);
    }
}
public class Main {
    public static void main(String[] args) {
        Child x = new Child();
        x.printDetails();
        x.printDetails();
        Parent y = new Parent();
        y.printDetails();
        Child.printDetails();
        Parent.printDetails();
        x.printDetails();
    }
}
```

5. Find out the errors in the following code and explain the reasons of Errors in those particular lines. [3]

```
class Point{
int x, y;
final int f = 10;
final Point p = new Point(1, 2);
public Point(int x, int y){
        this.x = x;
        this.y = y;
}
}
class Check{
        public static void main(String args[]){
                Point point = new Point(5, 5);
                point.f = 5;
                point.p.x = 10;
                point.p = new Point(1, 5);
```

```
        }
}
```

6. Answer the following questions based on the attached code snippet:

```
public class Cake {
    protected String name;
    protected double rate;

    public Cake(String n, double r) {
        name = n;
        rate = r;
    }

    public double calcPrice(){
        System.out.println("Print the calculated price.");
    }
   public printDetails(){
        System.out.println("Prints the detail.");
    }
}

class OrderCake extends Cake{
    double weight;

    public OrderCake(String n, double r, double w){
        super(n, r);
        weight = w;
    }
    //override calcPrice & printDetails
}

class ReadymadeCake extends Cake{
    int quantity;

    public ReadymadeCake(String n, double r, int q){
        super(n, r);
        quantity = q;
    }
    //override calcPrice & printDetails
}

class Main{
    public static void main(String[] args) {
        Cake cake[];

        // Complete the code

        for (int i = 0; i < cake.length; i++) {
            cake[i].printDetails();
        }
```

```
    }
}
```

a. calcPrice() method in the Cake class is used to calculate the total price of a cake. You need to override this method in each of the derived classes: OrderCake and ReadymadeCake. Price is calculated as per the following rules: [4]

- OrderCake: *rate * weight*
- ReadymadeCake: *rate * quantity*

Also, override another method printDetails() in each class. This method will print the information about a particular cake according to the following format:

Name: <name>
Rate: <rate>
Weight/Quantity: <value>
Total Price: <price>

Now, complete the *OrderCake* and *ReadymadeCake* class by overriding calcPrice() and printDetails() methods.

b. Create an array of three cake objects. First two objects are of OrderCake type and later one object is of ReadymadeCake type in the main() method. Assign some values to the class attributes while creating those objects. If you successfully create the array, your output will look like as follows: [2]

Name: OrderCake
Rate: 150
Weight: 3
Total Price: 450
… …. … …. ….. …

Name: ReadymadeCake
Rate: 200
Quantity: 2
Total Price: 400
…. …… …… …… …… ..

Question 01:

I am the base constructor
I have an extra value: 5
I borrow 5 form a constructor
I have more values: 6.3
I have no param

Question 02:

a)

```
package M1;
public class Human {
	private int id;
	private String intelligence; // Make intelligence private
	protected boolean bravery;

	public Human(int id, String intelligence, boolean bravery) { // Add constructor
		this.id = id;
		this.intelligence = intelligence;
		this.bravery = bravery;
	}
	public int getId() {
		return id;
	}
	public String getIntelligence() {
		return intelligence;
	}
	public boolean isBrave() {
		return bravery;
	}
}
// Monster class remains unchanged
package M1.M2;
public class Test {
	public static void main(String[] args) {
		// Code remains unchanged
	}
}
```

b)

```
package M1;
// Human class with constructor already added in (a)
public class Monster {
	private String name; // Make name private
	private double weight; // Make weight private

	public Monster(String name, double weight) {
		// Add print statement
		System.out.println("To avoid getting scared or eaten be brave or intelligent.");
		this.name = name;
		this.weight = weight;
	}
	// Other methods remain unchanged
}
// Test class remains unchanged
```

c)

```
public class Monster {
    // ……other code
    boolean eat(Human h) {
        if (h.getIntelligence().equals("high")) {
            return true;
        } else {
            increaseWeight(10);
            return false;
        }
    }
    String scare(boolean bravery) {
        if (!bravery) {
            System.out.println("Human scared.");
        } else {
            System.out.println("Human is too brave to scare.");
        }
    }
}
```

Question 03:

```
public class AdvancedCalculator extends Calculator {
    private int[] numbers;

    public AdvancedCalculator(int firstNumber, int secondNumber, int[] numbers) {
        super(firstNumber, secondNumber);
        this.numbers = numbers;
    }

    @Override
    public int sum() {
        int sum = super.sum();   // Use the sum() method of the parent class to find the sum of first two numbers
        for (int number : numbers) {
            sum += number;         // Add each element of the array to the sum
        }
        return sum;
    }
}
```

Question 04:

```
I am in a Child Class: 1
I am in a Child Class: 2
I am in Parent Class: 3
I am in a Child Class: 4
I am in Parent Class: 5
I am in a Child Class: 6
```

Question 5:

```
class Point {
    int x, y;
    int f = 10;
    Point p = null; // Initialize p to null

    public Point(int x, int y) {
```

```
            this.x = x;
            this.y = y;
        }
    }

    class Check {
        public static void main(String args[]) {
            Point point = new Point(5, 5);
            point.f = 5;
            point.p = new Point(1, 2); // Initialize p to a new Point object
            point.p.x = 10;
            point.p = new Point(1, 5);
        }
    }
```

Question 6:

a)

```
class OrderCake extends Cake {
    double weight;

    public OrderCake(String n, double r, double w) {
        super(n, r);
        weight = w;
    }

    @Override
    public double calcPrice() {
        return rate * weight; // Calculate price for OrderCake
    }

    @Override
    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Rate: " + rate);
        System.out.println("Weight: " + weight);
        System.out.println("Total Price: " + calcPrice());
    }
}

class ReadymadeCake extends Cake {
    int quantity;

    public ReadymadeCake(String n, double r, int q) {
        super(n, r);
        quantity = q;
```

```
    }

    @Override
    public double calcPrice() {
        return rate * quantity; // Calculate price for ReadymadeCake
    }

    @Override
    public void printDetails() {
        System.out.println("Name: " + name);
        System.out.println("Rate: " + rate);
        System.out.println("Quantity: " + quantity);
        System.out.println("Total Price: " + calcPrice());
    }
}
```

b)

```
class Main {
    public static void main(String[] args) {
        Cake cake[] = new Cake[3]; // Create an array of Cake objects

        // Create the first two objects of OrderCake type
        cake[0] = new OrderCake("OrderCake", 150.0, 3.0);
        cake[1] = new OrderCake("OrderCake", 150.0, 4.0);

        // Create the third object of ReadymadeCake type
        cake[2] = new ReadymadeCake("ReadymadeCake", 200.0, 2.0);

        // Print the details of each cake object in the array
        for (int i = 0; i < cake.length; i++) {
            cake[i].printDetails();
        }
    }
}
```