



United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Final Exam, Trimester: Fall 2023

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

Total Marks: 40, Duration: 2 Hours

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

Q1: (a) Consider the interface *Transaction*

[4+4]

```
public interface Transaction {
    public void give();
    public void receive();
}
```

The *Payment* class implements interface *Transaction*. Code for the *Payment* class is provided as follows:

```
public abstract class Payment implements Transaction{
    double amount;
    String currency = "BDT";
    @Override
    public void give(){
        //Write code here
        System.out.println(currency+" "+amount+" is paid");
    }
    @Override
    public void receive() {
        System.out.println("[Nothing to receive]");
    }
    public abstract double getCharge();
    public double getTax(){
        //Write code here
    }
}
```

Create three classes *CashPay*, *CreditCardPay* and *CheckPay* that are subclasses of *Payment*. Complete the code for *Payment* class. Write all the necessary codes according to the following description:

- I. There is no charge added for cash payment.
- II. 2.5% charge is added to the amount for credit card payment.
- III. 10 taka flat charge is added for payment with check.
- IV. 15.6% VAT is added for all types of payment
- V. *give()* method prints the amount to be given including charge and VAT.

(b) Consider the following code.

```
public interface Shape {
    double getArea();
}
class Rectangle implements Shape {
    private double length, width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    @Override
    public double getArea() {
        return length * width;
    }
}
class Square extends Rectangle{
    //write your code here
}
```

```

    }
class ShapeTest{
    //write code for draw method here
    public static void main(String[] args) {
        Shape r = new Rectangle(5,6);
        Shape s = new Square(3);
        draw(r);
        draw(s);
    }
}

```

- i. Complete the code for the Square class.
- ii. Implement the method *draw* in ShapeTest class so the following output is produced.

```

drawing over 30.0 area
drawing over 9.0 area

```

Q2: (a) Suppose that you are given a text file named “input.txt” with random texts. Now you have to check how many consonants are in the file and write it to another text file named “output.txt”. For example, if the input file contains the following text - “Don’t be upset”, you should write “7” in the output file. Both the files should be in the “src” directory. Now, write a program in Java to perform this task. [6+6]

(b)

- i. What happens if an exception is not caught or handled in Java?
- ii. Write the output of the following code.

```

import java.util.*;
import java.lang.*;
import java.io.*;

public class Main{
    public static void main (String[] args) throws java.lang.Exception{
        int arr[]= new int[4];
        int x = 10, y = 1;
        try{
            try{
                arr[4] = x / (y - 1);
                System.out.println("a");
            }
            catch(ArithmeticException e){
                System.out.println("b");
                arr[4] = x / (y - 1);
            }
            catch(IndexOutOfBoundsException e){
                System.out.println("c");
                arr[4] = x / (y - 1);
            }
            finally{
                System.out.println("d");
            }
        }
        catch(Exception e){
            System.out.println("e");
        }
        finally{
            System.out.println("f");
        }
    }
}

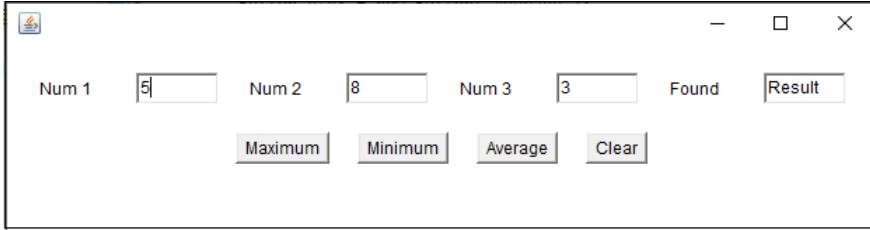
```

Q3: (a) Suppose that you are required to write a Java program for implementing a GUI using any layout. GUI initially contains a single button named *ADD*. *ADD* button dynamically adds a new button to the frame when it is clicked. Each new button should have a unique label too (such as b1, b2, b3 ...). Note that the buttons or labels should not overlap. Now do the following task to demonstrate your skill:

[4+4]

Write only the codes to create the user interface and an event-handling method for *ADD* button.

(b) Assume that the following window is already created.



The names for text field variable are *n1*, *n2*, *n3*, and *result*. For buttons, the names are *max*, *min*, *avg*, and *clr*.

Now write the proper event handling method to do the following task,

- I. After clicking the maximum button, the *result* field should show the maximum of *n1*, *n2*, and *n3*.
- II. The minimum button will set the minimum value of *n1*, *n2*, and *n3* in the *result* field.
- III. Average button will find the average of all these inputs.
- IV. Clear button will clear all the text fields.

Q4: Consider the following Address class:

[6]

```
public class Address {
    String building_number, area, city;
    int zip_code;

    Address(String building_number, String area, String city, int zip_code){
        this.building_number = building_number;
        this.area = area;
        this.city = city;
        this.zip_code = zip_code;
    }

    int getZip_code(){
        return zip_code;
    }
}
```

Based on the above class, complete the following tasks:

```
class Test{
    public static void main(String[] args){
        //Task 1: Create an empty ArrayList of Address type

        /*
        Task 2: Add the following objects in the ArrayList
        "19/A","Dhanmondi","Dhaka",1209
        "2/A","Tejgaon","Dhaka",1215
        "65","Nirala","Khulna",9100
        */

        /*
        Task 3: Add the below object at index 1 of the ArrayList
        "215","Aamtola","Barishal",8200
        */

        /*
        Task 4: Set the object at index 2 to
```

```

"36", "Gulshan", "Dhaka", 1212
*/

/*
Sort the arraylist in, ascending order of zip codes using a
comparator for comparing objects of Address class [You can also
define the Comparator as a separate class if you want]
*/

/*
Task 5: Delete the object at index 2
*/
}
}

```

Q5: Asfaq, a first-year programming student, learned how to find the highest number from an array using the Java program. Excited with his newfound knowledge, he started to wonder if he could concurrently do the same for multiple arrays. One of his seniors told him to use Thread which facilitates to run multiple tasks concurrently. Based on the idea, in order to help Asfaq, your task is to

- write a Java program that can find the highest numbers from 4 integer arrays concurrently and next, compute the maximum of these highest numbers and print the maximum number. Consider the arrays as {3, 1, -5, 10}, {-2, 6, 7, 8, 0}, {12, -6, 4, 2, 1}, {10, 5, -9, 18, 7}.

[6]