United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Final Exam :: Fall 2021

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

Total Marks: **40** Time: **2hr**

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

Answer all the four questions from [Question 1 to 4]. There are two questions named Question 5. You need to answer one of these two.

Question 1 [3 + 2 + 3]

A. Consider the following two **interfaces** "P1" and "P2". Complete the following tasks now:

i. Write an **interface** named "P3" **inheriting both** "P1" and "P2"

ii. Now write a **concrete** class named "Concrete" **inheriting** from "P3" and **overriding** the required methods with proper signature i.e. proper access specifier, return type and parameters

N.B. *Just write only the solution codes*

`interface P1{`
`    double h2(int x);`
`}`
	`interface P2{`
	`    String k1();`
	`    void k2();`
	`}`

B. Answer the following questions for the following "Fall" class:

i. Suggest modifications for variable "**x**" so that Line Number 6 can be executed without any error

ii. Suggest modifications for variable "**y**" so that value of it cannot be changed after Line Number 3

```
1 public class Fall {
2     int x;
3     int y = 213;
4
5     public static void main(String[] args) {
6         x = 50; // This should work after modification of x
7     }
8 }
```

C. Create and assign an object of Person **using anonymous inner class** for both of Line 7 and 8 to produce output "*Hello, I'm an Engineer*" from line number 10 and "*Hello, I'm a Doctor*" from Line number 11.

```
 1 interface Person {
 2   void introduce();
 3 }
 4
 5 public class AnnonEx {
 6   public static void main(String[] args) {
 7     Person engineer; // Write your codes
 8     Person doctor; // Write your codes
 9
10     engineer.introduce(); // should print "Hello, I'm an Engineer"
11     doctor.introduce(); // should print "Hello, I'm a Doctor"
12   }
13 }
```

Question 2 [3 + 5]

A. Rewrite the given code such that it handles exceptions and the message “Exception handled successfully” should be displayed whether an exception occurs or not for both divide() and display_namelength() methods. **You can only change inside the main function**.

```
public class ExceptionTest {
    public static void main(String[] args) {
        calculator obj = new calculator();
        obj.divide();
        obj.display_namelength();
    }
}

class calculator {
    String name = null;
    int num1;
    int num2;

    public calculator() {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the first number");
        this.num1 = input.nextInt();
        System.out.println("Enter the second number");
        this.num2 = input.nextInt();
    }

    public void divide() {
        System.out.println(num1 / num2);
    }

    public void display_namelength() {
        System.out.println(name.length()); // name object can be null
    }
}
```

B. Complete the following code:

```
class MainClass {
    public static void main(String[] args) {
        try {
            int acc[] = {100, 101, 102, 103, 104, 105};
            double balance[] = {2000, 1500, 900, 1560, 1765.50};
            System.out.println("Account No\t" + "Balance\t");
            for (int i = 0; i < 5; i++) {
                System.out.println(acc[i] + "\t\t" + balance[i] + "\t");
                if (balance[i] < 1000) {
                    // Task 1: Throw a new user-defined exception class named
"MinimumBalanaceException" here in this block.
                }
            }
        }
        // Task 2: Write the catch block here. The catch block will catch the
exception generated by the custom user-defined class named
"MinimumBalanaceException."
    }
}

// Task 3: Write your MinimumBalanaceException class here. It will print the
message "Balance is low now."
```

Question 3 [8]

You are given a file, **input.txt**, that contains some even number of lines. Each line contains two values: a number and a string. For this task, you need to write a Java code to calculate the **sum** of the numbers in **two consecutive lines** of **input.txt** and write the sum in another file, **output.txt**. Check the following example for details:

input.txt	output.txt
123 100 50 50 45 55 …	Line 1: 223 Line 2: 100 Line 3: 100

Here, the first output is 223 which is equal to 123 + 100. The second output is 100 which is equal to 50 + 50 and so on. The three dots at the end of input.txt means that there are many more lines in the file.

Question 4 [8]

Complete the following code to create a GUI that has the same appearance as the given GUI(s). Some GUI objects are already created in the code. You need to create the rest of the objects and complete the code. After that, add event handling functionality to the code so that when the button is clicked, the inputs in the textfields are swapped (Check **Fig 1** & **Fig 2**). You need to perform following tasks to complete the GUI:

- Create the "Swap" button object
- Set the frame object's layout properly and call necessary methods
- Add components to the frame
- Make the frame visible
- Perform button click handling

[*You can also use corresponding AWT classes (Frame etc.) instead of Swing classes (JFrame) if you want*]

```java
public class GUITest {
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setSize(275, 120);
        JTextField tf1 = new JTextField(10);
        JTextField tf2 = new JTextField(10);

        // Add your code 1: Complete the GUI

        // Add your code 2: Complete button
        //                  click handling
    }
}
```

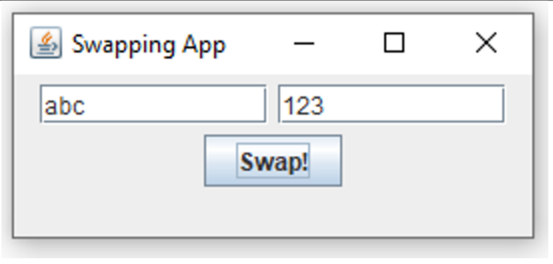

Fig 1: Before clicking Swap button

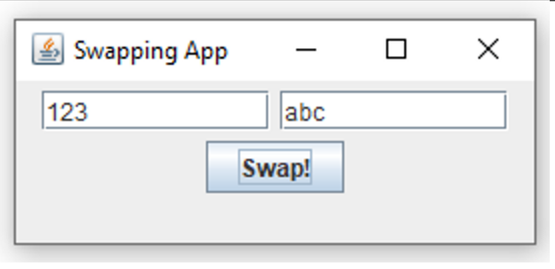

Fig 2: After clicking Swap button

Question 5 [4 + 4]

Write codes to complete the following the two tasks for the provided codes:

A. Find whether any student object with name “Peter” exists in the list using the **contains method of the “list” object**. Add required methods into the “Student” class to use the contains method

B. Now sort the “list” in **descending order of CGPA** using **Comparator**

```java
class Student {
    String name;
    int id;
    double cgpa;

    public Student(String name,
                   int id,
                   double cgpa) {
        this.name = name;
        this.id = id;
        this.cgpa = cgpa;
    }

    public Student(String name) {
        this.name = name;
        this.id = -1;
        this.cgpa = -1;
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        ArrayList<Student> list = new ArrayList();

        list.add(new Student("Wizard", 3, 3.88));
        list.add(new Student("Peter", 2, 3.5));
        list.add(new Student("Wanda", 1, 3.88));
        list.add(new Student("Thanos", 41, 3.9));
        list.add(new Student("Yelena", 7, 3.75));
        list.add(new Student("Thor", 15, 3.89));
    }
}
```

Or

Question 5 [4 + 4]

Write codes to perform the following two tasks:

A. Create a thread class named “NegativeFilter” **extending the “Thread”** class. Inside the run method it should **continuously** take an integer as input from console and output to console if it’s non-negative.
Write the public Main class and inside the main method, create an object of “NegativeFilter” and **start the thread**.

B. Create a thread class named “EvenFilter” **implementing the “Runnable”** interface. Inside the run method it should **continuously** take an integer as input from console and output to console if it’s an odd number.
Write the public Main class and inside the main method, create a thread object that takes an “EvenFilter” object and **start the thread**.