United International University (UIU)

Dept. of Computer Science & Engineering (CSE)
Final Exam, Trimester: Summer 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

1a **Take a look at the definitions of the two interfaces and the abstract class given below:** **1+4 =5**

Student (abstract class): **Attributes:** • firstName (String) • lastName (String) • studentID (String) • age (int) **Methods:** • **void register()** – prints which courses the student is registered to. **Yet to be defined.** • **void display() -** defined to print all attributes of the student class.	**TA (interface):** **Attributes:** String ROLE = "teaching assistant" **Methods:** **void assistProfessor()-** should print which professor the TA is assisting this semester. **Yet to be defined.**	**RA (interface):** **Attributes:** String role = "research assistant" **Methods:** **void conductResearch()-** should print research work the RA is currently working on. **Yet to be defined.**

Now, a class is defined to represent students who are both a TA and a RA. **This class implements the two interfaces and extends the abstract class.** It also has a few attributes and methods of its own as defined below:

Class UG_SuperStudent:

Attributes:

- string Professor - holds the name of the professor the student is working as a TA of.
- string research - holds the title of the research the student is currently working on.
- int salary - holds the total salary the student gets as a TA and RA.

Methods:

void display() - overrides the display() method to print all information for a UG_ SuperStudent.

Now, **write a code snippet in java to implement ONLY the class UG_SuperStudent** by completing the code snippet given below.

```
class UG_SuperStudent //inherit other 3 classes appropriately
{
   //write necessary attributes & override all necessary abstract functions so that it's a concrete class
   //For overriding abstract functions, you may just print a line as you see fit.
   //No need to write constructors or other non-abstract methods.
}
```

1b **1x5 =5**

```
public class TestException {
public static void main(String[] args) {
   Scanner scan = new Scanner(System.in);
   int flag = 0;
   while(flag = = 0)
   {
      System.out.println("Enter 2 integers.");
      try{
         int a = scan.nextInt();
         int b = Integer.parseInt(scan.nextLine().trim());
         int c = a/b;
         System.out.println("Result: " + c);
         flag = 1;
      }
      catch(ArithmeticException e)
      {  System.out.println("A"); }
      catch(NumberFormatException e)
      {   System.out.println("C");}
      catch(Exception e)
      {
         System.out.println("B");
         if (scan.hasNextLine())
         scan.nextLine();
      }
      finally
      { System.out.println("D"); }
   }
   scan.close();
   System.out.println("E");
}}
```

Analyze the java code given above and decide the output for each of the test cases given below. Writing only the input-output table given below by filling the '?' marks in your script will be enough.

Sample Input	Output
Enter two integers: 2 e	?
Enter two integers: e 5	?
Enter two integers: f f	?
Enter two integers: 5 0	?
Enter two integers: 2 2	?

2a. You have a class called "Book" with attributes: title, author, and numberOfAvailableCopies, which indicates how many copies of the book are available in the store. When someone buys a book, the available copies decrease by 1. If the available copies reach zero, the "buyBook" method in the "Book" class should raise a custom exception called "BookOutOfStock" with the message: "[BookOutOfStockException] The value of numberOfAvailableCopies is 0."
Now, implement this functionality by adding code to the specified portion below.

```
import java.util.ArrayList;
import java.util.List;
class Book {
   String title; String author;
   int numberOfAvailableCopies;

   public Book(String title, String author, int numberOfAvailableCopies) {
      this.title = title; this.author = author;
      this.numberOfAvailableCopies = numberOfAvailableCopies;
   }

   void buyBook(){// Complete the method}
}
```

Then, consider the following class named Library. Library class contains a list of instances of class Book and the title of each book is unique. The method named buyBook takes a title as an argument and if the title is found in the list, then calls the buyBook method of Book class. As per the instruction of this question, the buyBook method of the Book class might throw an exception. Hence, this exception needs to be handled in the buyBook method of the Library class and show the user a message that "The book you are willing to order is unavailable". **[2+2]**

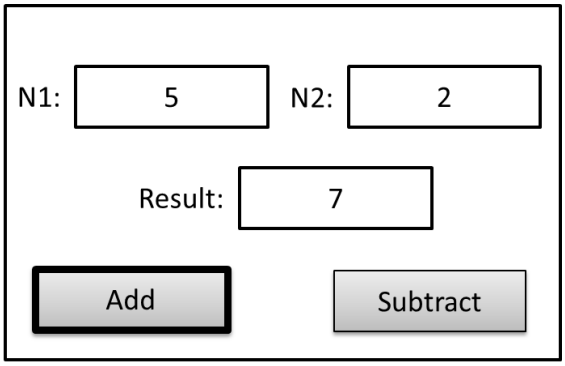
```
class Library {
   List< Book > books;
   Library(){
      books = new ArrayList<>();
   }
   void addBook(Book book){
      for(int i=0; i<books.size(); i++){
         if(books.get(i).title.equals(book.title)){return;}
      }
      books.add(book);
   }

   void buyBook(String bookTitle){
      for(int i=0; i<books.size(); i++){
         Book b = books.get(i);
         if(b.title.equals(bookTitle)){
            // Write your code here
            b.buyBook();
            books.set(i, new Book(b.title, b.author, b.numberOfAvailableCopies-1));
            break;
         }
      }
   }
}
```

2b. You have to write a program that **reads two integers from the file input.txt, then adds the two numbers, and prints the sum in the file output.txt.** You may use other equivalent file input and output streams instead as well.

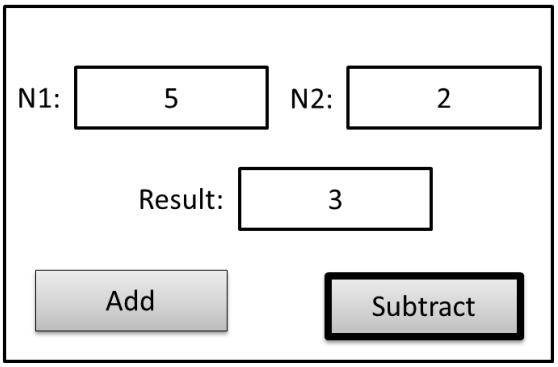
You must also use File objects to represent the two files input.txt and output.txt. The file input.txt contains two integers. The output.txt file is empty. They are all in the same folder as the program **FileTest.java**.

Remember to add necessary try-catch blocks, and to close any streams that you use. [6]

3.



When "Add" button is pressed



When "Subtract" button is pressed

Take a look at the sample GUI given above. It contains three text fields, two for taking input of two integers and one for displaying the result. It also has two buttons –

- "Add" - adds N1 and N2 and displays it on Result.
- "Subtract" - subtracts N2 from N1 and displays it on Result.

Now**, write ONLY the appropriate event handling function** that implements the functionality of these two buttons. Assume variable names for GUI elements and listeners as necessary. **[10]**

4. (a) Consider the class below:

```
public class Student {
private double gpa;
private String name;
public Student(double g, String n){
        gpa = g;
        name = n;
}
public double get_gpa(){return gpa;}
public String get_name(){return name;}
}
```

Now write the lines of code in the appropriate place of the class Myproject4 for the following operations:

[1+0.5+0.5+0.5+0.5+2]

- Add 4 students [Elias with gpa 3.5, Sourav with gpa 3.2, Barakat with gpa 4.5, Bidu with gpa 2.5] in ArrayList.
- Remove the student with index 1 of ArrayList
- Set a student Ali with gpa 3.7 at index 2.
- Add a student Hasil with 2.95 at index 1.
- Sort the array according to higher order of gpa.
- Compute the average gpa of the students in ArrayList.

```
import java.util.ArrayList;
import java.util.Comparator;
public class Myproject4 {
   static void display(ArrayList <Student> c){
      for(int i = 0; i < c.size(); i++){
        System.out.println("Name:"+c.get(i).get_name()+",gpa = "+c.get(i).get_gpa());
      }
       System.out.println();
   }

   public static void main(String[] args) {
      ArrayList <Student> c = new ArrayList <Student>();
      // Write your code here
      display(c);
      c.sort(Comparator.comparing(Student::get_name));
      // Modify the above line of code here
      display(c);
      double s;
      // Write your code here
      System.out.println("average gpa = " + s);
   }
}
```

4. (b) Suppose that we require evaluating the value of z which is given by

$$z = \frac{1}{5} \times \frac{1}{8} \times \frac{1}{11} \times \frac{1}{14} \ldots \frac{1}{n}.$$

We want to split the above problem in three parts and run each part in a different thread. That is, each part will run concurrently. When the threads stop, the results will be collected and multiplied together to form the final value of z.

Now write the lines of code in the appropriate place of the class Myproject5 to accomplish evaluating the value of z for $n = 38$. [1+2+2]

[Check next page for the code]

```
public class Mythread extends Thread{
   private int tid; // Thread ID
   private int startValue, endValue, inc;
   private double z;
   public Mythread(int id, int sv, int ev){

     // Write your code here
      inc = 3; // increment
      z = 1.0;
   }
   public void run(){
       // Write your code here to compute partial z
   }


   public double get_z(){return z;}
}

public class Myproject5 {
   public static void main(String[] args) {
      Thread t1 = new Mythread(1, 5, 14);
      Thread t2 = new Mythread(2, 17, 26);
      Thread t3 = new Mythread(3, 29, 38);
      t1.start();
      t2.start();
      t3.start();

      try {
         t1.join();
      } catch (InterruptedException ex) {
         ex.printStackTrace();
      }


     try {
         t2.join();
      } catch (InterruptedException ex) {
         ex.printStackTrace();
      }

      try {
         t3.join();
      } catch (InterruptedException ex) {
         ex.printStackTrace();
      }

      if(!t1.isAlive() && !t2.isAlive() && !t3.isAlive()){
         double r = ((Mythread)t1).get_z();
       // Write your code here to compute final z
      }
   }
}
```