United International University
Department of Computer Science and Engineering

CSE 1115: Object Oriented Programming Final: Summer 2025

Total Marks: 40 Time: 2 hours

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

Answer all six questions. The numbers on the right of the questions denote their marks.

1. Examine the following code and answer parts **(a)** and **(b)** to produce the **expected output** shown below. **Note:** Do not **modify** any line of the given code.

Main.java

```
interface Writable {
    String write();
}
interface Refillable {
    void refill(String color);
}
abstract class AbstractFountainPen { // Do not modify this line
}
class FountainPen extends AbstractFountainPen implements
   WritableAndRefillable {
    FountainPen(String s) {
        super(s);
    }
}
public class Main {
    public static void main(String[] args) {
        Refillable matador = new FountainPen("Matador");
        Writable pentonic = new FountainPen("Pentonic");
        matador.refill("Blue");
        System.out.println(pentonic.write());
    }
}
```

Expected Output

```
Refilling Matador with color Blue
Writing with Pentonic
```

(a) Write the interface **WritableAndRefillable** which has the functionality for both writing and refilling. (2)

(b) Write necessary codes in the **AbstractFountainPen** and **FountainPen** classes only.**(You can choose any of the classes or both to write your code.)** (4)

2. Write the **output** of the following program: (8)

Exception.java

```
public class Question_2_1 {
    static void checkA() {
        System.out.println("Inside checkA");
        int[] arr = new int[4];
        try {
            System.out.println("Inside try of checkA");
            arr[5] = 7;
            throw new NumberFormatException("Thrown from try of checkA");
        } catch (NumberFormatException e) {
            System.out.println("Exception caught inside catch of checkA: "
                + e.getMessage());
        } finally {
            System.out.println("Inside finally of checkA");
        }
    }

    static void checkB() {
        System.out.println("Inside checkB");
        try {
            System.out.println("Inside try of checkB");
            checkA();
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception caught inside catch1 of checkB");
            throw new ArithmeticException("Thrown from catch1 of checkB");
        } catch (Exception e) {
            System.out.println("Exception caught inside catch2 of checkB");
            throw new IllegalArgumentException("Thrown from catch2 of
                checkB");
        }
    }
}
```

```
public static void main(String []args){
    System.out.println("Main function start");
    try {
        System.out.println("Inside outer try of main");
        try {
            checkB();
            System.out.println("Inside inner try of main");
            throw new RuntimeException("Thrown from try of main");
        } catch (ArithmeticException e) {
            System.out.println("Exception caught inside inner catch of main
                : " + e.getMessage());
        } finally {
            System.out.println("Inside inner finally of main");
        }
    }
    catch(RuntimeException e){
        System.out.println("Exception caught inside outer catch of main: "
            + e.getMessage());
    }
    System.out.println("Main function end");
}
```

3. Suppose you are creating **a guessing game** where player 1 writes a sentence and player 2 tries to guess the number of words in the sentence, with the following setup: (8)

- Player 1 types a sentence in a **text field** named **T1** and clicks a **button** named **B1** to submit his sentence.
- Player 2 types a number in a **text field** named **T2** and clicks a **button** named **B2** to submit his guess for the number of words in the sentence given by player 1.
- A label named **L1** displays the text **"Player 2 wins"** if player 2 correctly guesses the number of words of player 1's sentence, and displays **"Player 1 wins"** if player 2 is wrong. The text is displayed when button **B2** is clicked.

Write the code for the **event-handling** part of the game above. You can **assume** that the components have already been created and that a sentence has words separated by spaces only. **You must use only one method to handle both button clicks.**
A visual demonstration is given below:

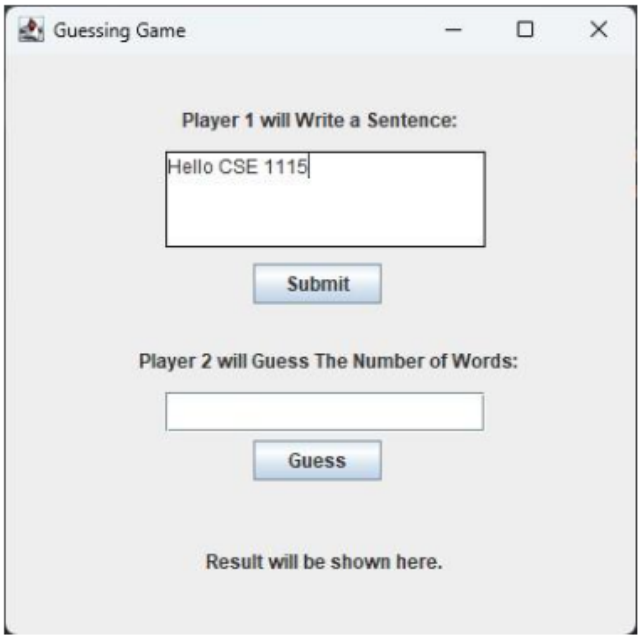

Figure 1: Player 1's turn

Figure 2: Player 2's turn

4. Observe the following code:

EligibilityList.java

```
class Applicant implements Comparable<Applicant>
{
    String name;
    double cgpa;
    int probation;
    Applicant(String name, double cgpa, int probation) {
        this.name = name;
        this.cgpa = cgpa;
        this.probation = probation;
    }
}
```

```
public class EligibilityList {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        ArrayList<Applicant> all = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            String name = sc.next();
            double cgpa = sc.nextDouble();
            int probation = sc.nextInt();
            Applicant applicant = new Applicant(name, cgpa, probation);
            all.add(applicant);
        }
        sc.close();
    }
}
```

(a) Write necessary codes to sort the **Applicants** based on their **cgpa** in **descending** order. (2)

(b) Build a new list of Applicants named **eligible** from the existing list **all**, with the applicants who has no probation (probation = 0). (2)

(c) Display the information of **Top 3 (or fewer)** applicants from the **eligible** list in the format **"name - cgpa"**. (2)

5. You have an input file named **input.txt** with multiple lines. Each line contains words separated by spaces. Your task is to find the **number of words in the entire file**. **The words will contain only alphabetic characters (A–Z, a–z)**. Write the result to an **output.txt** file in the following format: (6)

input.txt	**output.txt**
Object Oriented Java is Fun File Handling is cooler	Number of words: 9

6. Write a Java program that takes an integer **n** as input and calculates **n!(n factorial)** using **2 threads**. (6)

- The first thread should multiply the numbers from **1 to n/2**.
- The second thread should handle the remaining numbers.
- The main thread should then combine the partial results to produce the final factorial.
- Hint: Factorial of a number **n** means multiplying all numbers from **1 to n**. For example, **3! = 1*2*3.**