United International University
Department of Computer Science and Engineering

CSE 1115: Object Oriented Programming   Final: Fall 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 five(5) questions. The numbers on the right of the questions denote their marks.

1. (a) Find out the errors (if there are any) and correct them **without writing any code inside any class or interface.** (2)

Abstraction.java

```java
abstract class abs {
    abstract public void m1();
    public void m2(){}
}
interface I1{
    public void m3();
    public void m4();
}
interface I2{
    public void m5();
    public void m6();
}
```

```java
class Abstraction extends abs
   implements I1, I2{
    public void m1(){}
    public void m2(){}
    public void m3(){}
    public void m4(){}
    public void m5(){}
}
```

(b) Consider the following code (4)

ShapeTest.java

```java
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;
    }
}
```

```java
class Square extends Rectangle{
// Write your code here
}
public 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);
    }
}
```

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

Expected Output

```
drawing over 30.0 area
drawing over 9.0 area
```

2. Consider the following code:

Exception.java

```java
import java.util.Scanner;
// Task A: Write the code of InvalidBalanceException class here
public class TestException {
    static void checkBalance(int balance){
    // Write your code here
    System.out.println("Balance verified"); // do not change this line
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            System.out.println("Connecting...");
            try {
                String balance = sc.next();
                checkBalance(Integer.parseInt(balance));
                System.out.println("Transaction approved");
            }
            catch (NumberFormatException e) {
                System.out.println("Invalid amount entered");
            }
            finally {
                System.out.println("Processing request");
            }
            System.out.println("Please wait...");
        }
        catch (InvalidBalanceException e) {
            System.out.println("Transaction declined. " + e.getMessage());
        }
        finally {
            System.out.println("Thank you");
        }
        System.out.println("Good Goodbye");
    }
}
```

(a) Write the **InvalidBalanceException** class, which extends the Exception class and invokes the parent class constructor. (2)

(b) Complete the **checkBalance** method by throwing an **InvalidBalanceException** if the balance is less than 1000. The message should show **Insufficient balance**. (2)

(c) After completing the checkBalance method, find the output of the program for each of the following cases: (5)

i. $balance = 500$

ii. $balance = abc$

3. You are given a class Student that represents a student of a course section. (9)

Student.java

```
public class Student{
    String name;
    int id;
    double cgpa;
    Student(String name, int id, double cgpa){
        this.name = name;
        this.id = id;
        this.cgpa = cgpa;
    }
}
public class Sections {
    public static void main(String[] args){
        //Write code here
    }
}
```

Tasks:

(a) Create an **ArrayList** of student objects called **sectionA**(stores Student objects of Section A)

(b) Insert the following variables into the arraylist:

```
("A", 5, 3.61)
("D", 4, 3.82)
("K", 20, 3.57)
("M", 8, 3.70)
("L", 15, 3.45)
```

(c) Sort the arraylists in the **descending order of CGPA.**

(d) Display the **highest CGPA, lowest CGPA,** and **average CGPA** of the students in sectionA.

(e) Create another ArrayList named **Selected** and add to it all students from sectionA whose CGPA is **greater than or equal to the average CGPA.**

(f) Remove all students from the sectionA ArrayList.

(g) Print the details of all students in the Selected ArrayList using a **for-each loop.**

4. Due to a software error in a bank system, every transaction amount was overcharged by exactly **100 BDT**. All transaction records are stored in a text file named "transactions.txt" located inside the src folder. Each line in transactions.txt follows this format: (8)

```
SenderName sends ReceiverName: Amount
```

You are required to correct the transaction amounts and create a new file named `"transactions_updated.txt"`, where 100 BDT is subtracted from each transaction amount.

transactions.txt	**transactions_updated.txt**
Habib sends Labib : 1200	Habib sends Labib : 1100
Kamal sends Jamal : 3200	Kamal sends Jamal : 3100
Jalil sends Rumon : 4000	Jalil sends Rumon : 3900

Your program must:

- Read all transactions from transactions.txt
- Correct each transaction by subtracting 100 BDT from the amount
- Write the corrected transactions to transactions_updated.txt
- Work correctly for any number of transactions in the input file

5. Write a Java program to calculate the sum of the following series: (8)

$$S = 1^1 - 2^2 + 3^3 - \cdots - n^n$$

where n is an **even number**.

The program should use **two threads** as described below:

- **Thread 1:** Responsible for computing the terms where the base is an **odd number** (e.g., $1^1, 3^3, 5^5, \ldots$)
- **Thread 2:** Responsible for computing the terms where the base is an **even number** (e.g., $2^2, 4^4, 6^6, \ldots$)

Finally, the program should combine the results from both threads and display the **total sum** of the series.