First Name (Print): _____ **Last Name (Print):** _____

Student Number: _____

Irving K. Barber Faculty of Science
COSC 121 Midterm I Exam (Practice Only)

Instructor: Dr. Bowen Hui
~~Wednesday, February 11, 2025~~
~~Time: 2:00 pm - 3:20 pm~~

Instructions:

This is a closed-book exam. No calculators are allowed. Turn off your cell phone and put it away. Have your student ID card out on your desk.

This exam has 6 pages (excluding this cover page).

Questions	**Your Score**	**/ Possible Marks**
Question 1		/ 7
Question 2		/ 6
Question 3		/ 5
Question 4		/ 5.5
Question 5		/ 7
Question 6		/ 2
Total		/ 32.5

Question 1.

Consider the following `TransitPass` and `StudentPass` classes:

```
public class TransitPass {
    protected double baseFare;
    public TransitPass( double baseFare ) { this.baseFare = baseFare; }
    public double calcFare( int trips ) {
        return trips * baseFare;
    }
    public String toString() { return "Transit Pass: " + baseFare; }
}

public class StudentPass extends TransitPass {
    private double discountRate;
    public StudentPass( double baseFare, double discount ) {
        super( baseFare );
        this.discountRate = baseFare - discount;
    }

    public double calcFare( int trips ) {
        return trips * discountRate;
    }
    public String toString() { return "Student Pass: " + discountRate; }
}
```

Now, define a subclass called `SeniorPass` that inherits from `TransitPass`. In particular, the subclass should:

- Define a constructor that makes use of `super()` in setting up the attributes
- Override the `calcFare()` method to include the class name (i.e., "Senior Pass") and the recipient in the return string

Lastly, define a test class with a `main()` method that:

- Creates a `TransitPass` reference variable called `pass`
- Initialize `pass` as a `StudentPass` object with a discount of 0.50 and call that object's `calcFare()` method on two trips
- Change `pass` to refer to a `SeniorPass` object whose discount will be the full fare (seniors ride for free) and call that object's `calcFare()` method on two trips

Grading scheme:

- [**1 pt**] `SeniorPass` class declaration
- [**1 pt**] `SeniorPass` constructor
- [**1 pt**] `calcFare()` method in `SeniorPass` class
- [**1 pt**] `pass` type
- [**1 pt**] `pass` object instantiations
- [**1 pt**] Calls the appropriate `calcFare()` methods
- [**1 pt**] Overall syntax and correctness

Question 2.

You are designing a subscription management system for a company. Different subscription types (e.g., trial, monthly, and yearly) share common data and behavior, but compute their fees differently.

Write an abstract class called `Subscription` that satisfies the following requirements:

- Subscriptions have in common:
  - a `protected` string variable called `type`
  - a `protected` double variable called `basePrice`
  - a `protected` boolean variable called `multipleUsersAllowed`
- The class must include the following `public` methods:
  - a constructor that initializes both attributes
  - a concrete method called `getMultipleUsersAllowed` that simply returns the value of `multipleUsersAllowed`
  - an abstract method called `calcMonthlyCost` that does not take any input, computes the monthly fee for the customer based on the type of subscription, the base price, and whether multiple users are allowed, and returns the cost as a double variable

Grading scheme:

- **[1 pt]** Correct attributes
- **[1 pt]** Correct constructor definition
- **[1 pt]** Correct method declaration for `getMultipleUsersAllowed`
- **[1 pt]** Correct method definition for `getMultipleUsersAllowed`
- **[1 pt]** Correct method declaration for `calcMonthlyCost`
- **[1 pt]** Overall syntax and correctness

Question 3.

Suppose your friend is developing a media player program and wrote a `Podcast` class and a `LiveStream` class. You suggested adding a `Seekable` interface as a way to improve the design of the code so that both classes must define common methods. Define an interface called `Seekable` that is compatible with the following definitions of `Podcast` and `LiveStream` classes and includes both "seek" methods.

```
class Podcast implements Seekable {
    private int time = 0;
    public void seekForward(int seconds) {
        time += seconds;
    }
    public void seekBackward(int seconds) {
        time -= seconds;
    }
}

class LiveStream implements Seekable {
    private int time = 0;
    public void seekForward(int seconds) {
        time += seconds;
    }
    public void seekBackward(int seconds) { }  // cannot rewind
}
```

Grading scheme:

- **[1 pt]** Correct interface definition
- **[2 pts]** Method declarations (0.5 pt each)
- **[1 pt]** Correct method visibility modifiers
- **[1 pt]** Overall syntax and correctness

Question 4.

What is the output of the following program? If there is an error, just write "error".

```
public class ExceptionTrace {
    public static void main(String[] args) {
        System.out.println("A");
        try {
            process(5);
            System.out.println("B");
        }
        catch (NullPointerException e) {
            System.out.println("C");
        }
        catch (Exception e) {
            System.out.println("D");
        }
        finally {
            System.out.println("E");
        }
        System.out.println("F");
    }

    public static void process(int n) throws Exception {
        System.out.println("G");
        try {
            calculate(n);
            System.out.println("H");
        }
        catch (ArithmeticException e) {
            System.out.println("I");
            throw e;
        }
        finally {
            System.out.println("J");
        }
        System.out.println("K");
    }

    public static void calculate(int x) {
        System.out.println("L");
        int y = x / 0;               // ArithmeticException
        System.out.println("M");
    }
}
```

Grading scheme:

- **[0.5 pt]** Each letter that should be printed and its placement order
- **[0.5 pt]** Each letter that should be omitted

Question 5.

See the cloneable exercise (posted at: https://cmps-people.ok.ubc.ca/bowenhui/121/lectures/9-oopExercises.pdf). Be comfortable making deep copies of objects.

Question 6.

What is the data saved to the `user_data.csv` file after running the program `CsvUserInput` below? The answer depends on user input – come up with feasible user inputs that would make this program work as expected, then indicate what is saved into the output file based on your user inputs.

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

public class CsvUserInput {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String filename = "user_data.csv";

        try (PrintWriter writer = new PrintWriter(filename)) {
            // Write CSV header
            writer.println("Name,Age,Country");

            // Loop 3 times to get user input
            for (int i = 1; i <= 3; i++) {
                System.out.print("Enter name #" + i + ": ");
                String name = scanner.nextLine();

                System.out.print("Enter age #" + i + ": ");
                String age = scanner.nextLine();

                System.out.print("Enter country #" + i + ": ");
                String country = scanner.nextLine();

                // Write CSV line
                writer.println(name + "," + age + "," + country);
            }

            System.out.println("Data saved to " + filename);
        } catch (FileNotFoundException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

Grading scheme:

- **[1 pt]** Correct answer for each line