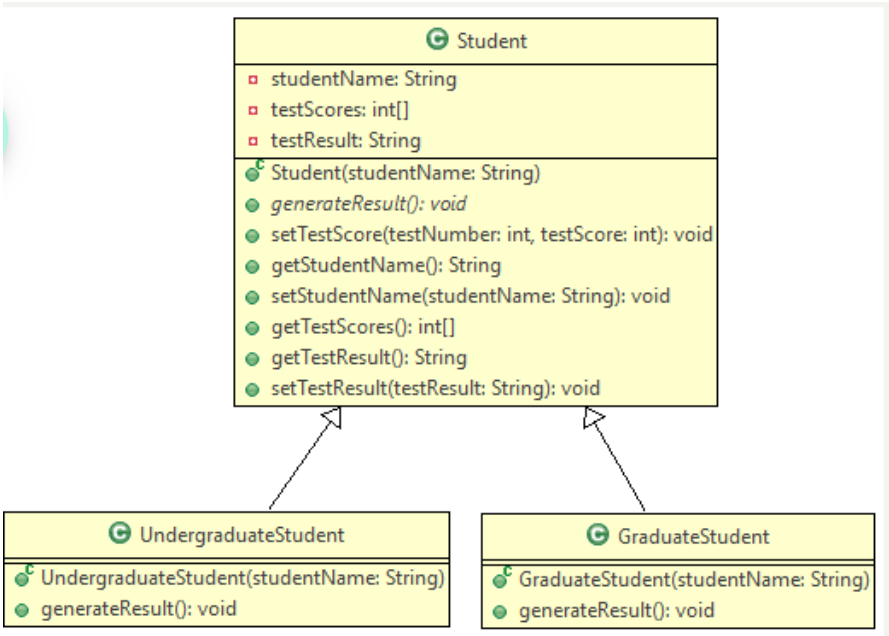
Lab 5

Topics :- Abstract class , Interface , Multithreading

Q.1

1. Anchor College offers both UnderGraduate and PostGraduate programs. The college stores the names of the students, their test scores and the final result for each student. Each student has to take 4 tests in total. You need to create an application for the college by implementing the classes based on the class diagram and description given below.



Method Description

Implement the getter and setter methods appropriately.

1. **Student(Class)**
2. **Student(String studentName)**

- Initialize the instance variable studentName with the value passed to the constructor and other instance variables to the default values.

1. **setTestScore(int testNumber, int testScore)**

- Set the value of the testScore in the appropriate position of testScores array based on the testNumber.

1. **UndergraduateStudent(Class)**
2. **UndergraduateStudent(String studentName)**

- Initialize the instance variable studentName with the value passed to the constructor and other instance variables to the default values.

1. **generateResult()**

- **Implement the abstract method of Student class by setting the value of testResult based on the below details.**

Average Score	Result
>=60	Pass
<60	Fail

Sample Input and Output For UndergraduateStudent

Input:-

Instance Variable	Values
name	**Jerry**
testScores	**{70,69,71,55}**

Output:-

Student Name : Jerry

Result : Pass

1. **PostGraduateStudent(Class)**
2. **PostgraduateStudent(String studentName)**

- Initialize the instance variable studentName with the value passed to the constructor and other instance variables to the default values.

1. **generateResult()**

- **Implement the abstract method of Student class by setting the value of testResult based on the below details.**

Average Score	Result
>=75	Pass
<75	Fail

Sample Input and Output For PostUndergraduateStudent

Input:-

Instance Variable	Values

name	Tom
testScores	{70,75,80,85}

Output:- Student Name : Tom

Result :

Pass

Ans.

Student.java

```
package lab_5;

abstract class Student {
private String StudentName;
private int[] testScores;
private String testResult;



public Student(String StudentName) {
	super();
	this.StudentName = StudentName;
}

public void setTestScore(int testNumber, int testScore){
	int length=testScores.length;
	testScores[length]=testNumber;

}



public String getStudentName() {
	return StudentName;
}

public void setStudentName(String studentName) {
	StudentName = studentName;
```

```
}



public int[] getTestScores() {
	return testScores;
}

public void setTestScores(int[] testScores) {
	this.testScores = testScores;
}

public String getTestResult() {
	return testResult;
}

public void setTestResult(String testResult) {
	this.testResult = testResult;
}

abstract void generateResult();

}
```

UnderGraduateStudent.java

```
package lab_5;

public class UnderGraduateStudnet extends Student{

	public UnderGraduateStudnet(String name) {
		super(name);
		// TODO Auto-generated constructor stub
	}
	@Override
	 void generateResult() {

		System.out.println("Student Name : "+ super.getStudentName());
		int totalScore=0;
		for(int marks:super.getTestScores()) {
```

```
                totalScore+=marks;
            }
//          System.out.println(totalScore);
            int getStatus=totalScore/super.getTestScores().length;
//          System.out.println(getStatus);
            if(getStatus >= 60) {
                super.setTestResult("PASS");
                System.out.println("Result : PASS");
            }
            else {
                super.setTestResult("FAIL");
                System.out.println("Result : FAIL");
            }

    }


}
```

GraduateStudent.java

```
package lab_5;

public class GraduateStudent extends Student {

    public GraduateStudent(String studnetName) {
        super(studnetName);
        // TODO Auto-generated constructor stub
    }

    @Override
    void generateResult() {
        System.out.println("Student Name : "+ super.getStudentName());
        int totalScore=0;
        for(int marks:super.getTestScores()) {
            totalScore+=marks;
        }
//      System.out.println(totalScore);
        int getStatus=totalScore/super.getTestScores().length;
//      System.out.println(getStatus);
```

```
            if(getStatus >= 75) {
                super.setTestResult("PASS");
                System.out.println("Result : PASS");
            }
            else {
                super.setTestResult("FAIL");
                System.out.println("Result : FAIL");
            }
        }


}
```

Main.java

```
package lab_5;

public class Main {
    public static void main(String[] args) {
        UnderGraduateStudnet obj1=new UnderGraduateStudnet("Jerry");
        int[] score={70,69,71,55};
        obj1.setTestScores(score);
        obj1.generateResult();

        GraduateStudent obj2=new GraduateStudent("Tom");
        int[] score2={70,75,80,85};
        obj2.setTestScores(score2);
        obj2.generateResult();

    }
}
```

/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Users/lakhman/Applications/IntelliJ IDEA Community Edition.app/Contents/lib/idea_rt.jar=49825:/Users/lakhman/Applications/IntelliJ IDEA Community Edition.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath

/Users/lakhman/Desktop/sem 4 coding/java/submission/out/production/submission lab5.lab_5.Main
Student Name : Lakhman
Result : PASS
Student Name : Dev
Result : PASS

Process finished with exit code 0

Q.2

Write a Java program as per the given description to demonstrate use of interface.

1. Define an interface **RelationInterface**.

Write three abstract methods: isGreater, isLess and isEqual.

All methods have a return type of boolean and take an argument of type Line with which the caller object will be compared.

1. Define the **Line** class implements the RelationInterface interface.

- It has 4 double variables for the x and y coordinates of the line.
- Define a **constructor** in **Line** class that initializes these 4 variables.
- Define a method **getLength()** that computes length of the line.

[double length = Math.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))].

- Implement the methods of interface in Line class

1. In class **CompareLines**.Java, create two objects of Line class, call the three methods to compare the lengths of the lines.

RelationInterface.java

Ans.

```
package lab_5;

public interface RelationInterface {
    boolean isGreater(Line otherLine);
    boolean isLess(Line otherLine);
    boolean isEqual(Line otherLine);
}
```

Line.java

```
package lab_5;



public class Line implements RelationInterface {
    private double x1, y1, x2, y2;

    public Line(double x1, double y1, double x2, double y2) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
    }

    public double getLength() {
        return Math.sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
    }

    @Override
    public boolean isGreater(Line otherLine) {
        return this.getLength() > otherLine.getLength();
    }

    @Override
    public boolean isLess(Line otherLine) {
        return this.getLength() < otherLine.getLength();
    }

    @Override
    public boolean isEqual(Line otherLine) {
        return this.getLength() == otherLine.getLength();
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -
javaagent:/Users/lakhman/Applications/IntelliJ IDEA Community
```

```
Edition.app/Contents/lib/idea_rt.jar=49836:/Users/lakhman/Applications/IntelliJ IDEA Community
Edition.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -
Dsun.stderr.encoding=UTF-8 -classpath /Users/lakhman/Desktop/sem 4
coding/java/submission/out/production/submission lab5.lab_2.CompareLines
Is line1 greater than line2? false
Is line1 less than line2? true
Is line1 equal to line2? false

Process finished with exit code 0
```

CompareLines.java

```
package lab_5;

public class CompareLines {
    public static void main(String[] args) {

        Line line1 = new Line(0, 0, 3, 4);
        Line line2 = new Line(0, 0, 5, 12);



        System.out.println("Is line1 greater than line2? " + line1.isGreater(line2));
        System.out.println("Is line1 less than line2? " + line1.isLess(line2));
        System.out.println("Is line1 equal to line2? " + line1.isEqual(line2));
    }
}
```

/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Users/lakhman/Applications/IntelliJ IDEA Community Edition.app/Contents/lib/idea_rt.jar=49836:/Users/lakhman/Applications/IntelliJ IDEA Community Edition.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /Users/lakhman/Desktop/sem 4 coding/java/submission/out/production/submission lab5.lab_2.CompareLines

Is line1 greater than line2? false
Is line1 less than line2? true
Is line1 equal to line2? false

Process finished with exit code 0

Q.3

In the producer–consumer problem, the producer and the consumer share a common, fixed-size buffer used as a queue. (Take buffer size as 1). The producer's job is to generate data, put it into the buffer. At the same time, the consumer is consuming the data (i.e. removing it from the buffer).The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer. Write a Java application consisting of all necessary classes to achieve this.

Ans.

Buffer.java

```
public class Buffer {
    private int data;
    private boolean empty;

    public Buffer() {
        empty = true;
    }

    public synchronized void produce(int newData) {
        while (!empty) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
        data = newData;
        empty = false;
        System.out.println("Produced: " + data);
        notify();
    }
```

```
    public synchronized int consume() {
        while (empty) {
            try {
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
        System.out.println("Consumed: " + data);
        empty = true;
        notify();
        return data;
    }
}
```

Producer.java

```
import java.util.Random;

public class Producer implements Runnable {
    private Buffer buffer;
    private Random random;

    public Producer(Buffer buffer) {
        this.buffer = buffer;
        this.random = new Random();
    }

    @Override
    public void run() {
        while (true) {
            int newData = random.nextInt(100); // Generate random data
            buffer.produce(newData); // Put data into the buffer
            try {
```

```
                Thread.sleep(random.nextInt(3000)); // Sleep for random time
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}
```

Consumer.java

```
import java.util.Random;

public class Consumer implements Runnable {
    private Buffer buffer;

    public Consumer(Buffer buffer) {
        this.buffer = buffer;
    }

    @Override
    public void run() {
        while (true) {
            buffer.consume(); // Consume data from the buffer
            try {
                Thread.sleep(new Random().nextInt(3000)); // Sleep for random time
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}
```

/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Users/lakhman/Applications/IntelliJ IDEA Community Edition.app/Contents/lib/idea_rt.jar=49942:/Users/lakhman/Applications/IntelliJ

IDEA Community Edition.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /Users/lakhman/Desktop/sem 4 coding/java/submission/out/production/submission lab5.que_3.Main
Produced: 58
Consumed: 58
Produced: 41
Consumed: 41
Produced: 12
Consumed: 12
Produced: 35
Consumed: 35
Produced: 0
Consumed: 0
Produced: 69
Consumed: 69
Produced: 82
Consumed: 82
Produced: 40
Consumed: 40
Produced: 27
Consumed: 27
Produced: 96
Consumed: 96
Produced: 67
Consumed: 67
Produced: 38
Consumed: 38
Produced: 4
Consumed: 4
Produced: 9
Consumed: 9
Produced: 48
Consumed: 48
Produced: 24
Consumed: 24
Produced: 72
Consumed: 72
Produced: 1
Consumed: 1
Produced: 19
Consumed: 19

Produced: 38
Consumed: 38
Produced: 34
Consumed: 34

Process finished with exit code 130 (interrupted by signal 2: SIGINT)

Q.4

Write a multithreaded Java application to produce a deadlock condition.

Ans.

```
public class DeadlockExample {
    private static final Object lock1 = new Object();
    private static final Object lock2 = new Object();

    public static void main(String[] args) {
        Thread thread1 = new Thread(() -> {
            synchronized (lock1) {
                System.out.println("Thread 1 acquired lock1");
                try {
                    Thread.sleep(100); // Introducing delay to make deadlock more apparent
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                synchronized (lock2) {
                    System.out.println("Thread 1 acquired lock2");
                }
            }
        });

        Thread thread2 = new Thread(() -> {
            synchronized (lock2) {
                System.out.println("Thread 2 acquired lock2");
                synchronized (lock1) {
                    System.out.println("Thread 2 acquired lock1");
                }
            }
        });
```

```
        thread1.start();
        thread2.start();
    }
```

/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Users/lakhman/Applications/IntelliJ IDEA Community Edition.app/Contents/lib/idea_rt.jar=49920:/Users/lakhman/Applications/IntelliJ IDEA Community Edition.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /Users/lakhman/Desktop/sem 4 coding/java/submission/out/production/submission lab5.DeadlockExample
Thread 1 acquired lock1
Thread 2 acquired lock2