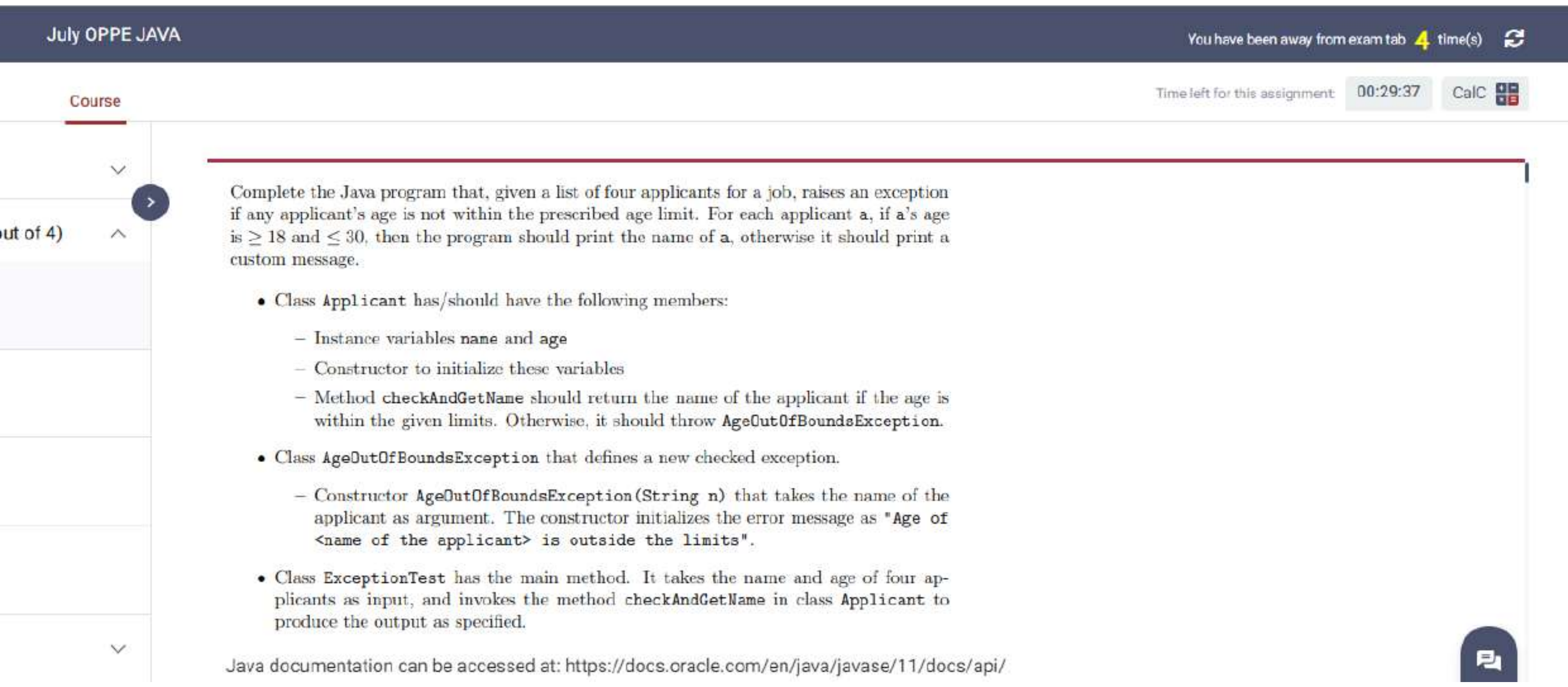
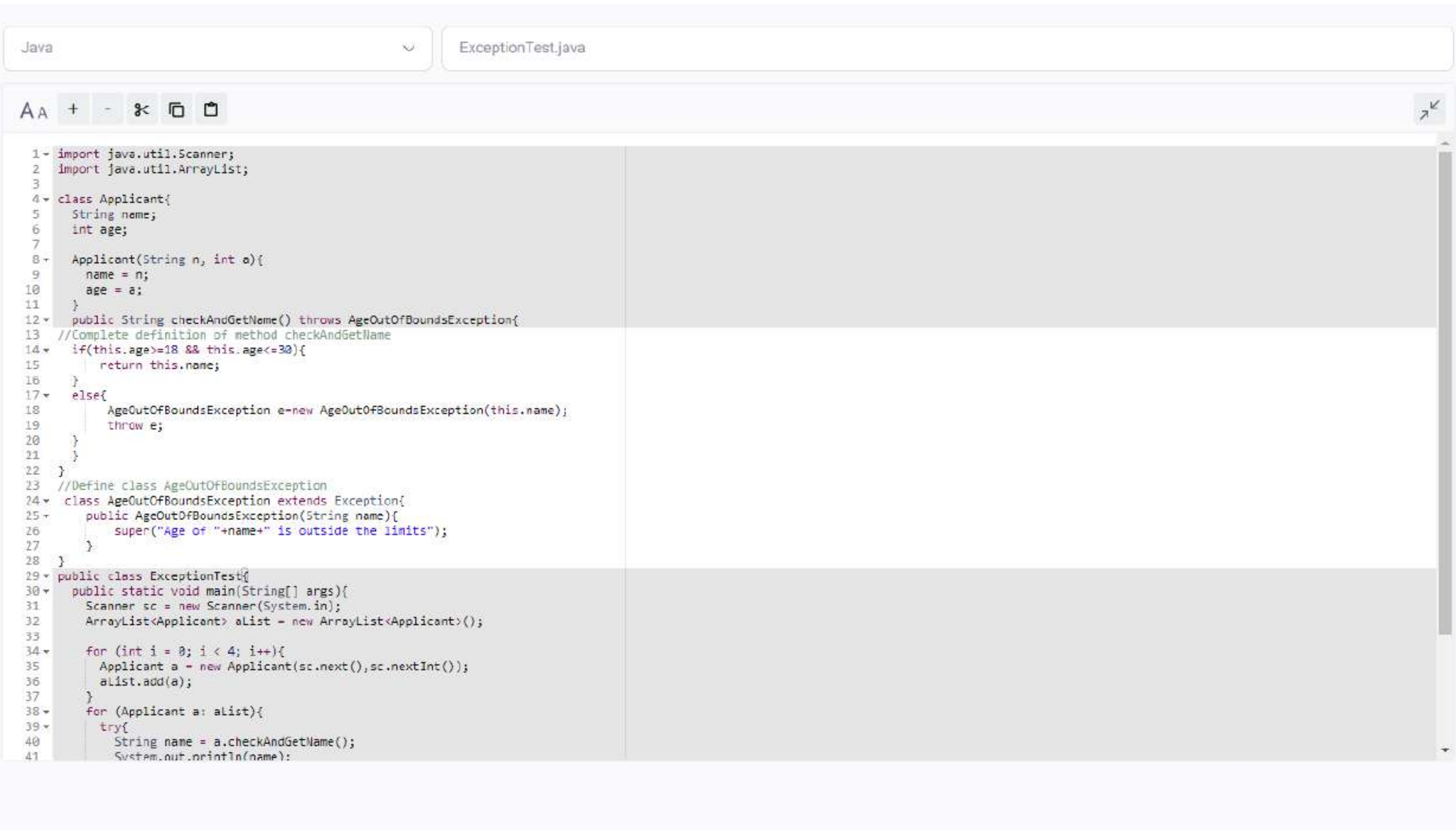

July OPPE JAVA
You have been away from exam tab 4 time(s)
Course
Time left for this assignment 00:29:37
CalC
ut of 4)

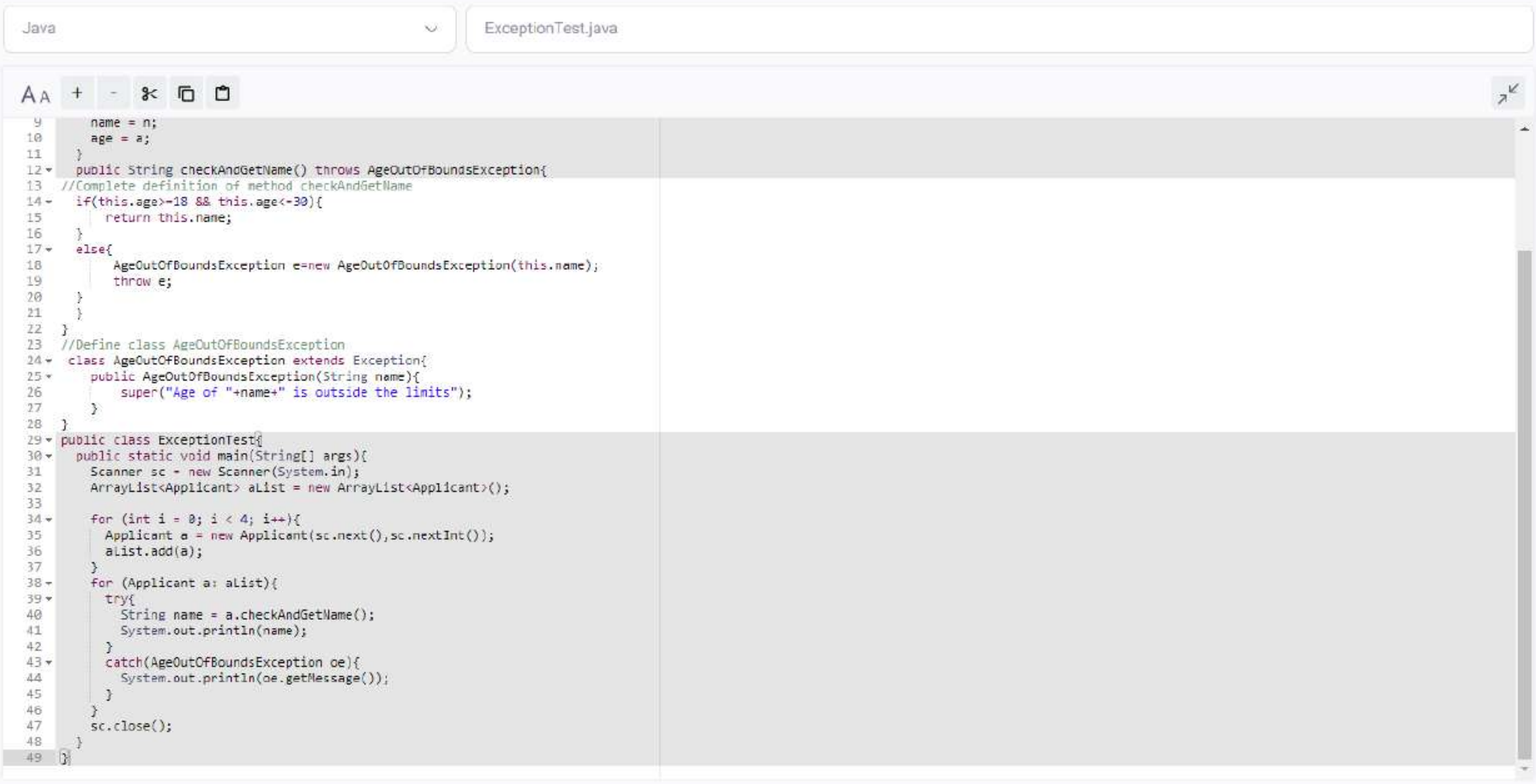
Complete the Java program that, given a list of four applicants for a job, raises an exception if any applicant's age is not within the prescribed age limit. For each applicant `a`, if `a`'s age is ≥ 18 and ≤ 30, then the program should print the name of `a`, otherwise it should print a custom message.

- Class `Applicant` has/should have the following members:
  - Instance variables `name` and `age`
  - Constructor to initialize these variables
  - Method `checkAndGetName` should return the name of the applicant if the age is within the given limits. Otherwise, it should throw `AgeOutOfBoundsException`.
- Class `AgeOutOfBoundsException` that defines a new checked exception.
  - Constructor `AgeOutOfBoundsException(String n)` that takes the name of the applicant as argument. The constructor initializes the error message as `"Age of <name of the applicant> is outside the limits"`.
- Class `ExceptionTest` has the main method. It takes the name and age of four applicants as input, and invokes the method `checkAndGetName` in class `Applicant` to produce the output as specified.

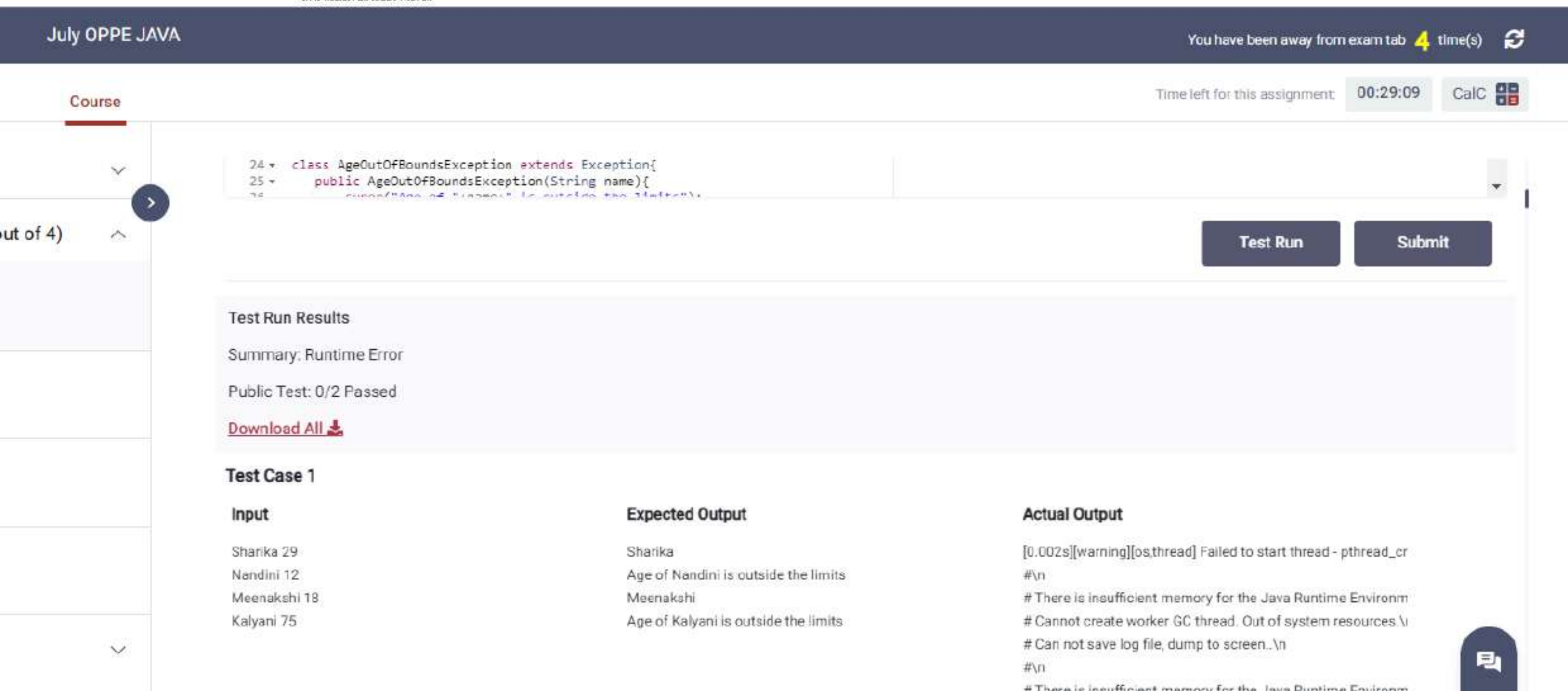
Java documentation can be accessed at: https://docs.oracle.com/en/java/javase/11/docs/api/



```java
import java.util.Scanner;
import java.util.ArrayList;

class Applicant{
  String name;
  int age;

  Applicant(String n, int a){
    name = n;
    age = a;
  }
  public String checkAndGetName() throws AgeOutOfBoundsException{
//Complete definition of method checkAndGetName
  if(this.age>=18 && this.age<=30){
      return this.name;
  }
  else{
       AgeOutOfBoundsException e=new AgeOutOfBoundsException(this.name);
       throw e;
  }
  }
}
//Define class AgeOutOfBoundsException
 class AgeOutOfBoundsException extends Exception{
    public AgeOutOfBoundsException(String name){
        super("Age of "+name+" is outside the limits");
    }
}
public class ExceptionTest{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    ArrayList<Applicant> aList = new ArrayList<Applicant>();

    for (int i = 0; i < 4; i++){
      Applicant a = new Applicant(sc.next(),sc.nextInt());
      aList.add(a);
    }
    for (Applicant a: aList){
      try{
        String name = a.checkAndGetName();
        System.out.println(name);
```



```java
    name = n;
    age = a;
  }
  public String checkAndGetName() throws AgeOutOfBoundsException{
//Complete definition of method checkAndGetName
  if(this.age>=18 && this.age<=30){
      return this.name;
  }
  else{
       AgeOutOfBoundsException e=new AgeOutOfBoundsException(this.name);
       throw e;
  }
  }
}
//Define class AgeOutOfBoundsException
 class AgeOutOfBoundsException extends Exception{
    public AgeOutOfBoundsException(String name){
        super("Age of "+name+" is outside the limits");
    }
}
public class ExceptionTest{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    ArrayList<Applicant> aList = new ArrayList<Applicant>();

    for (int i = 0; i < 4; i++){
      Applicant a = new Applicant(sc.next(),sc.nextInt());
      aList.add(a);
    }
    for (Applicant a: aList){
      try{
        String name = a.checkAndGetName();
        System.out.println(name);
      }
      catch(AgeOutOfBoundsException oe){
        System.out.println(oe.getMessage());
      }
    }
    sc.close();
  }
}
```



July OPPE JAVA

You have been away from exam tab 4 time(s)

Course

Time left for this assignment 00:29:09 CalC

out of 4)

```
class AgeOutOfBoundsException extends Exception{
   public AgeOutOfBoundsException(String name){
```

Test Run Submit

Test Run Results

Summary: Runtime Error

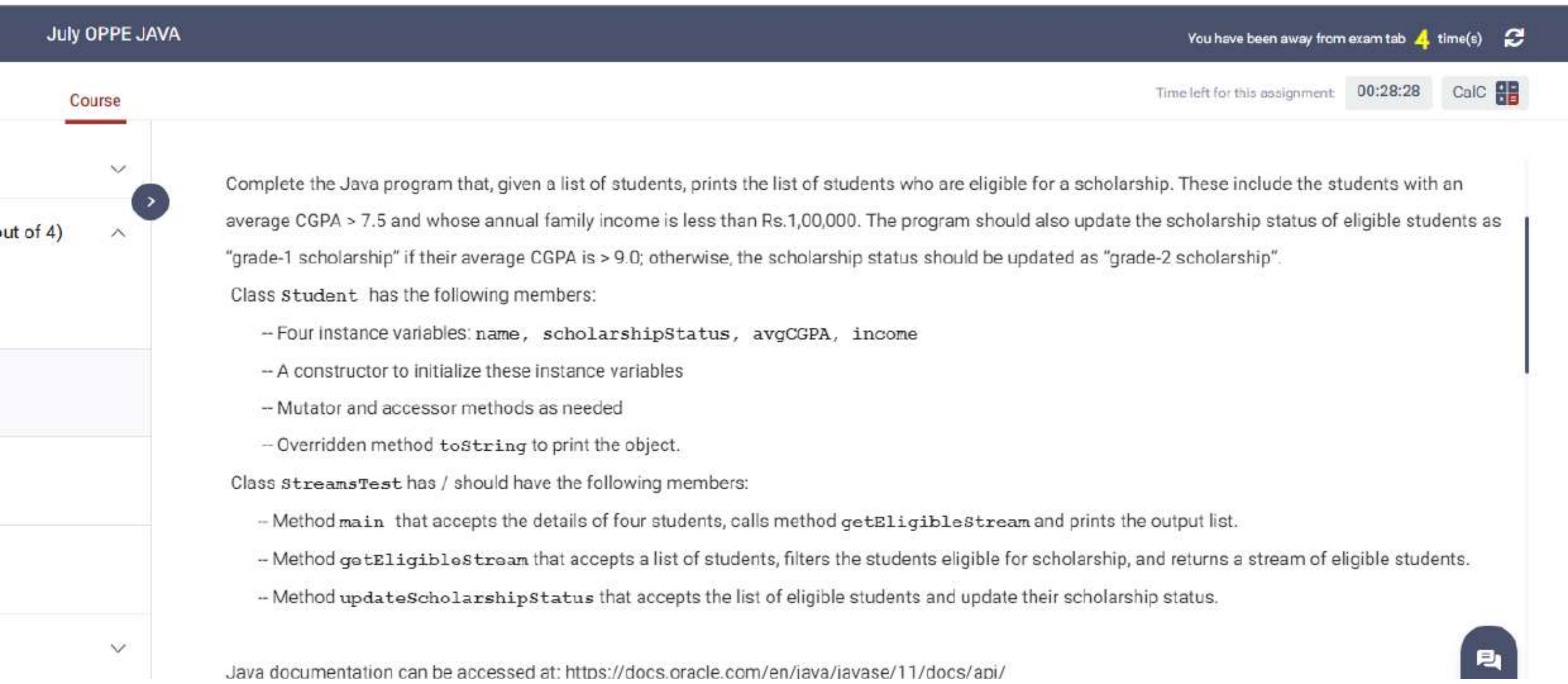
Public Test: 0/2 Passed

Download All

Test Case 1

Input	Expected Output	Actual Output
Sharika 29 Nandini 12 Meenakshi 18 Kalyani 75	Sharika Age of Nandini is outside the limits Meenakshi Age of Kalyani is outside the limits	

```
[0.002s][warning][os,thread] Failed to start thread - pthread_cr
#\n
# There is insufficient memory for the Java Runtime Environm
# Cannot create worker GC thread. Out of system resources.\
# Can not save log file, dump to screen..\n
#\n
```


July OPPE JAVA
You have been away from exam tab 4 time(s)
Course
Time left for this assignment 00:28:28
CalC
ut of 4)

Complete the Java program that, given a list of students, prints the list of students who are eligible for a scholarship. These include the students with an average CGPA > 7.5 and whose annual family income is less than Rs.1,00,000. The program should also update the scholarship status of eligible students as "grade-1 scholarship" if their average CGPA is > 9.0; otherwise, the scholarship status should be updated as "grade-2 scholarship".

Class `Student` has the following members:

-- Four instance variables: `name, scholarshipStatus, avgCGPA, income`

-- A constructor to initialize these instance variables

-- Mutator and accessor methods as needed

-- Overridden method `toString` to print the object.

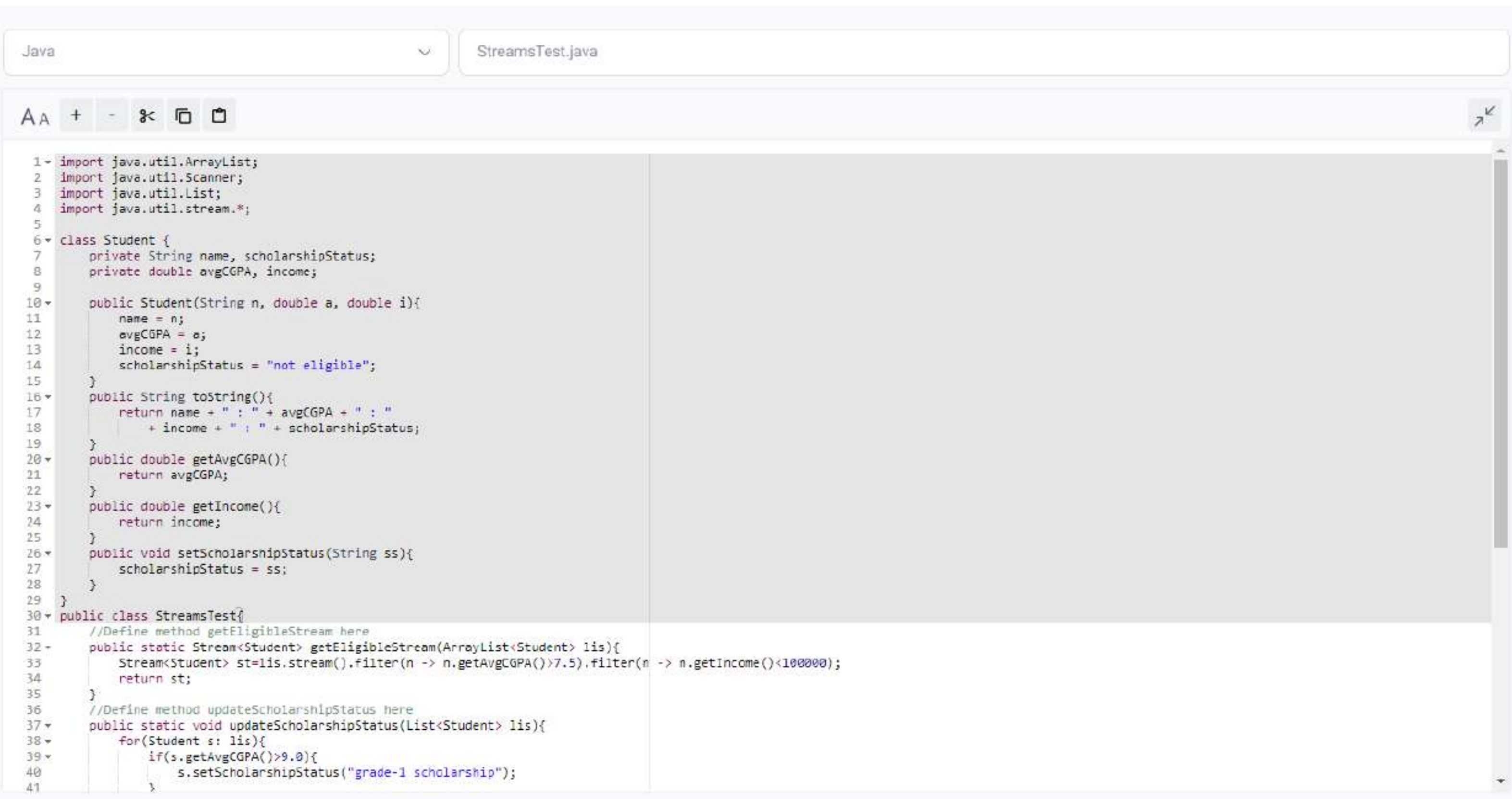
Class `StreamsTest` has / should have the following members:

-- Method `main` that accepts the details of four students, calls method `getEligibleStream` and prints the output list.

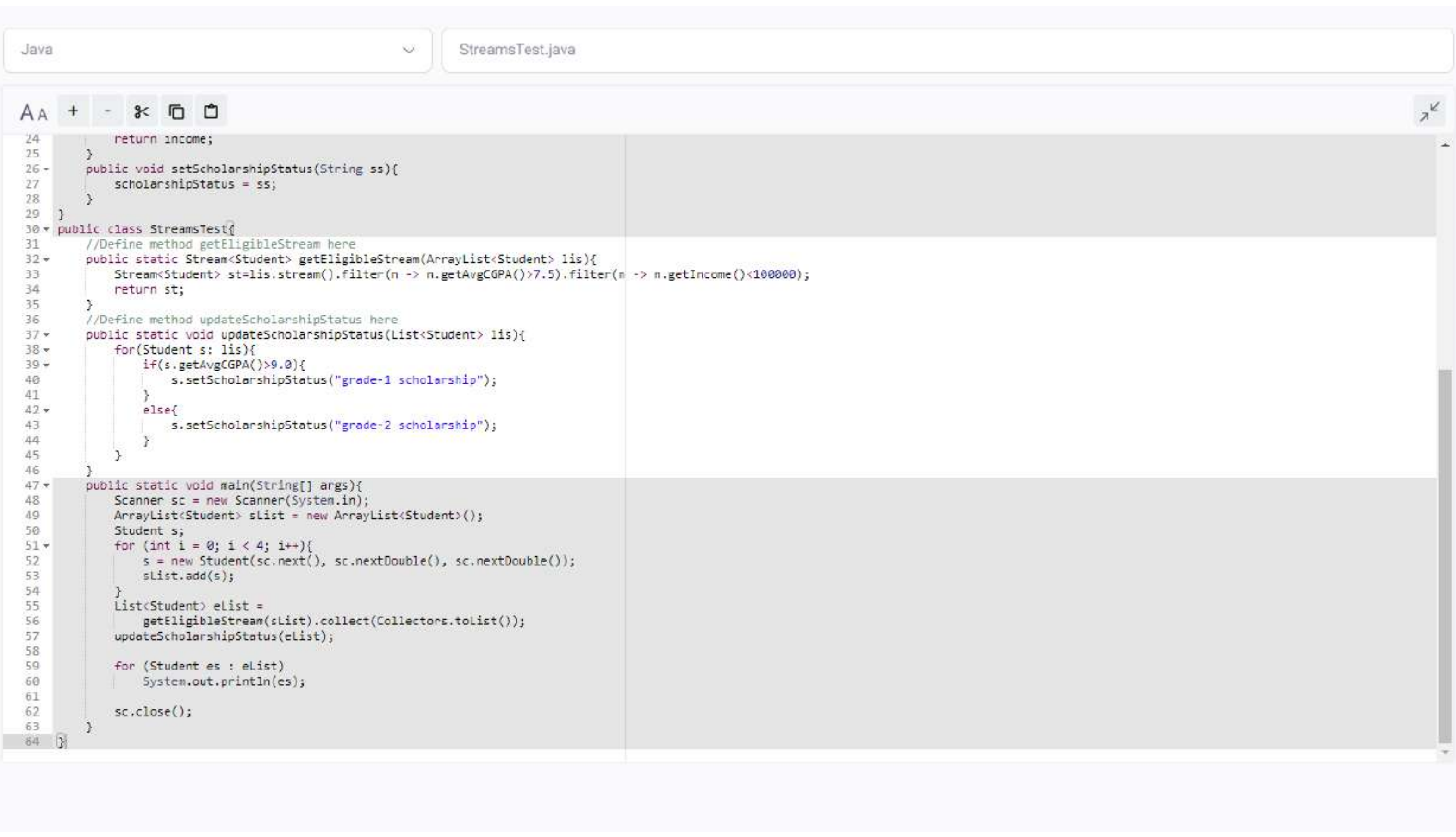
-- Method `getEligibleStream` that accepts a list of students, filters the students eligible for scholarship, and returns a stream of eligible students.

-- Method `updateScholarshipStatus` that accepts the list of eligible students and update their scholarship status.

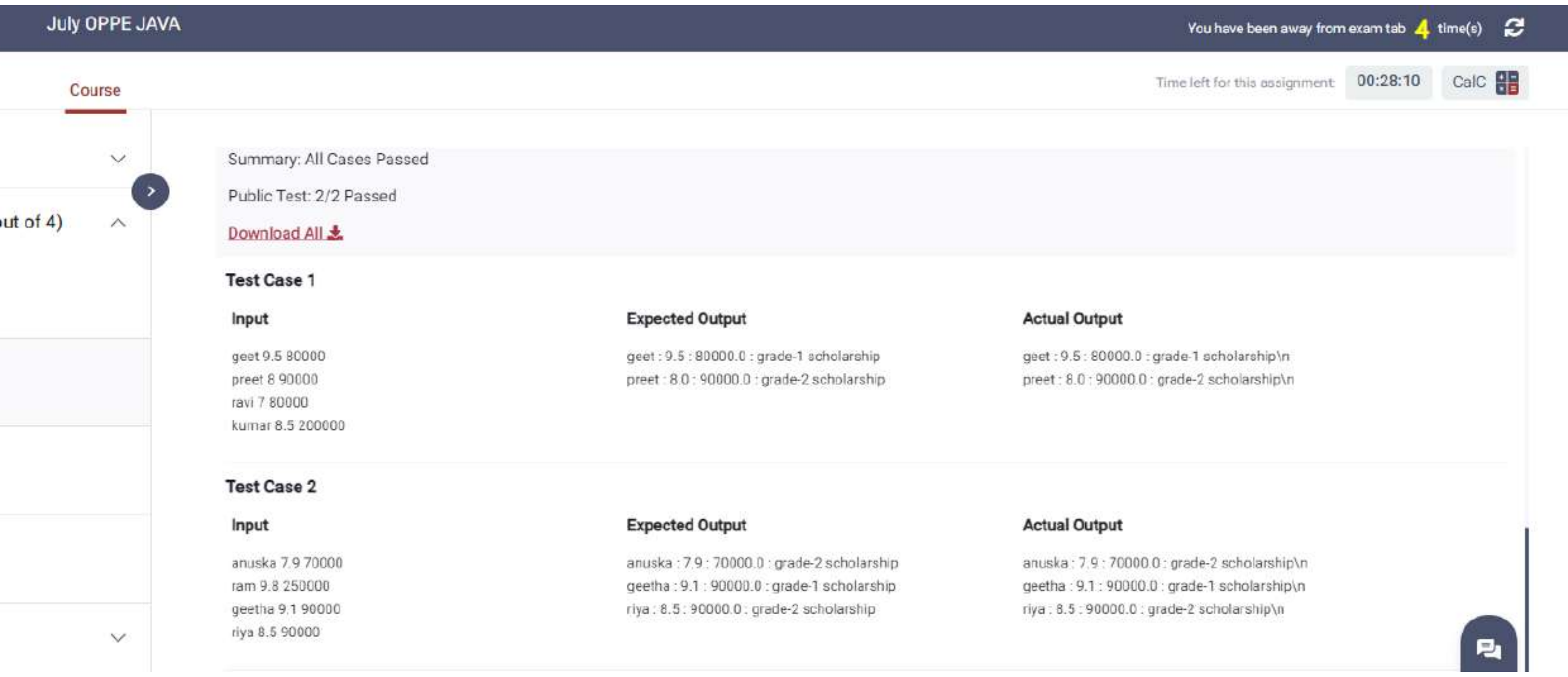
Java documentation can be accessed at: https://docs.oracle.com/en/java/javase/11/docs/api/



```java
import java.util.ArrayList;
import java.util.Scanner;
import java.util.List;
import java.util.stream.*;

class Student {
    private String name, scholarshipStatus;
    private double avgCGPA, income;

    public Student(String n, double a, double i){
        name = n;
        avgCGPA = a;
        income = i;
        scholarshipStatus = "not eligible";
    }
    public String toString(){
        return name + " : " + avgCGPA + " : "
            + income + " : " + scholarshipStatus;
    }
    public double getAvgCGPA(){
        return avgCGPA;
    }
    public double getIncome(){
        return income;
    }
    public void setScholarshipStatus(String ss){
        scholarshipStatus = ss;
    }
}
public class StreamsTest{
    //Define method getEligibleStream here
    public static Stream<Student> getEligibleStream(ArrayList<Student> lis){
        Stream<Student> st=lis.stream().filter(n -> n.getAvgCGPA()>7.5).filter(n -> n.getIncome()<100000);
        return st;
    }
    //Define method updateScholarshipStatus here
    public static void updateScholarshipStatus(List<Student> lis){
        for(Student s: lis){
            if(s.getAvgCGPA()>9.0){
                s.setScholarshipStatus("grade-1 scholarship");
            }
```



```java
        return income;
    }
    public void setScholarshipStatus(String ss){
        scholarshipStatus = ss;
    }
}
public class StreamsTest{
    //Define method getEligibleStream here
    public static Stream<Student> getEligibleStream(ArrayList<Student> lis){
        Stream<Student> st=lis.stream().filter(n -> n.getAvgCGPA()>7.5).filter(n -> n.getIncome()<100000);
        return st;
    }
    //Define method updateScholarshipStatus here
    public static void updateScholarshipStatus(List<Student> lis){
        for(Student s: lis){
            if(s.getAvgCGPA()>9.0){
                s.setScholarshipStatus("grade-1 scholarship");
            }
            else{
                s.setScholarshipStatus("grade-2 scholarship");
            }
        }
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        ArrayList<Student> sList = new ArrayList<Student>();
        Student s;
        for (int i = 0; i < 4; i++){
            s = new Student(sc.next(), sc.nextDouble(), sc.nextDouble());
            sList.add(s);
        }
        List<Student> eList =
            getEligibleStream(sList).collect(Collectors.toList());
        updateScholarshipStatus(eList);

        for (Student es : eList)
            System.out.println(es);

        sc.close();
    }
}
```


July OPPE JAVA
You have been away from exam tab 4 time(s)
Course
Time left for this assignment 00:28:10
CalC
ut of 4)

Summary: All Cases Passed

Public Test: 2/2 Passed

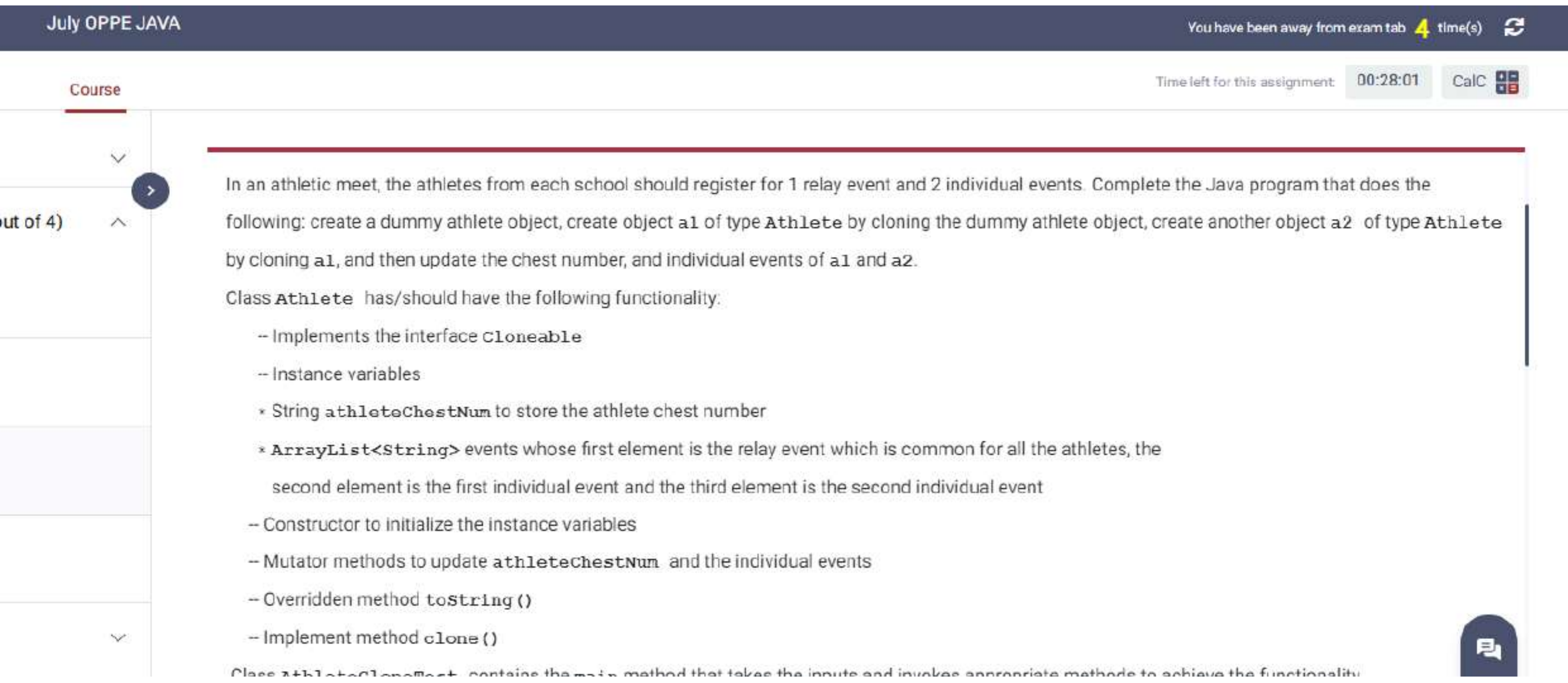
Download All

Test Case 1

Input	Expected Output	Actual Output
geet 9.5 80000 preet 8 90000 ravi 7 80000 kumar 8.5 200000	geet : 9.5 : 80000.0 : grade-1 scholarship preet : 8.0 : 90000.0 : grade-2 scholarship	geet : 9.5 : 80000.0 : grade-1 scholarship\n preet : 8.0 : 90000.0 : grade-2 scholarship\n

Test Case 2

Input	Expected Output	Actual Output
anuska 7.9 70000 ram 9.8 250000 geetha 9.1 90000 riya 8.5 90000	anuska : 7.9 : 70000.0 : grade-2 scholarship geetha : 9.1 : 90000.0 : grade-1 scholarship riya : 8.5 : 90000.0 : grade-2 scholarship	anuska : 7.9 : 70000.0 : grade-2 scholarship\n geetha : 9.1 : 90000.0 : grade-1 scholarship\n riya : 8.5 : 90000.0 : grade-2 scholarship\n


July OPPE JAVA
You have been away from exam tab 4 time(s)
Course
Time left for this assignment 00:28:01
CalC
ut of 4)

In an athletic meet, the athletes from each school should register for 1 relay event and 2 individual events. Complete the Java program that does the following: create a dummy athlete object, create object `a1` of type `Athlete` by cloning the dummy athlete object, create another object `a2` of type `Athlete` by cloning `a1`, and then update the chest number, and individual events of `a1` and `a2`.

Class `Athlete` has/should have the following functionality:

-- Implements the interface `Cloneable`

-- Instance variables

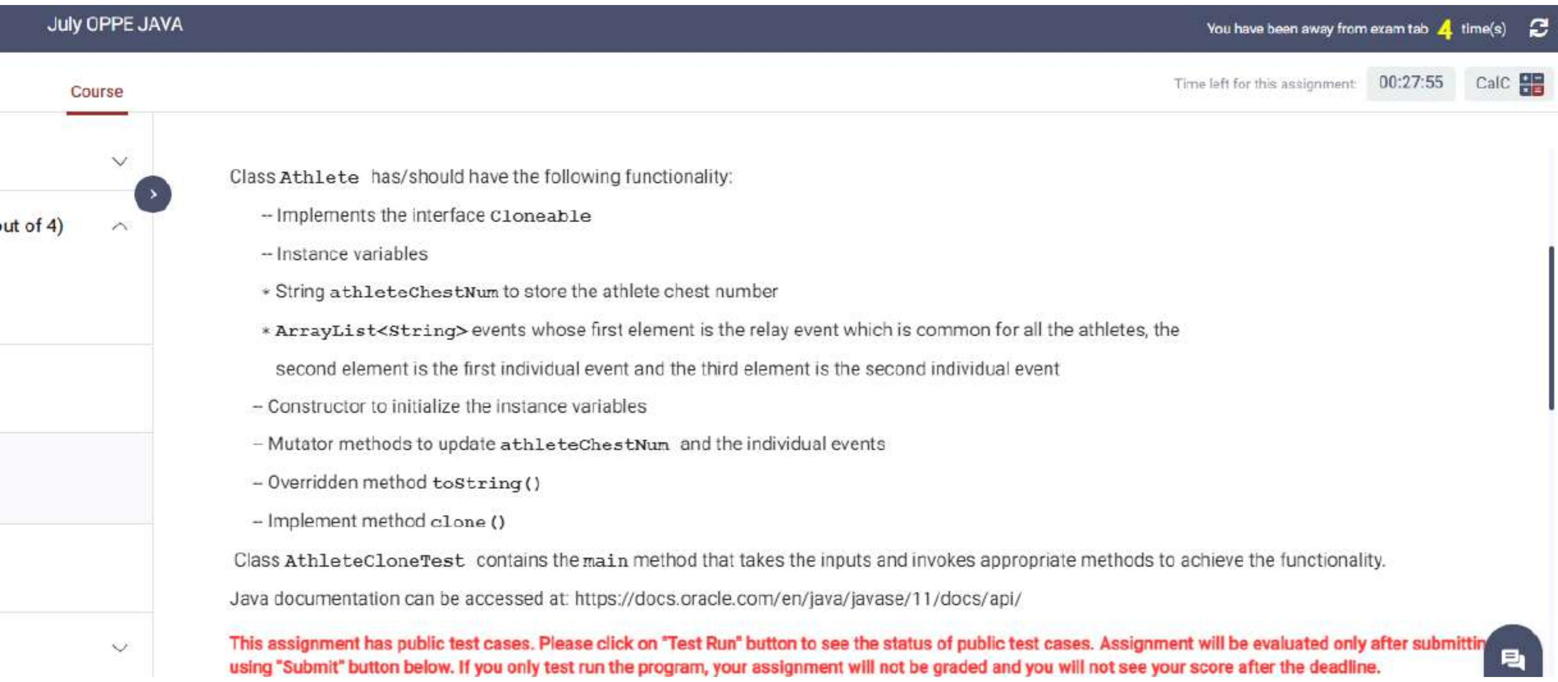
* String `athleteChestNum` to store the athlete chest number
* `ArrayList<String>` events whose first element is the relay event which is common for all the athletes, the second element is the first individual event and the third element is the second individual event

-- Constructor to initialize the instance variables

-- Mutator methods to update `athleteChestNum` and the individual events

-- Overridden method `toString()`

-- Implement method `clone()`


July OPPE JAVA
You have been away from exam tab 4 time(s)
Course
Time left for this assignment 00:27:55
CalC
ut of 4)

Class `Athlete` has/should have the following functionality:

-- Implements the interface `Cloneable`

-- Instance variables

* String `athleteChestNum` to store the athlete chest number

* `ArrayList<String>` events whose first element is the relay event which is common for all the athletes, the second element is the first individual event and the third element is the second individual event

-- Constructor to initialize the instance variables

-- Mutator methods to update `athleteChestNum` and the individual events

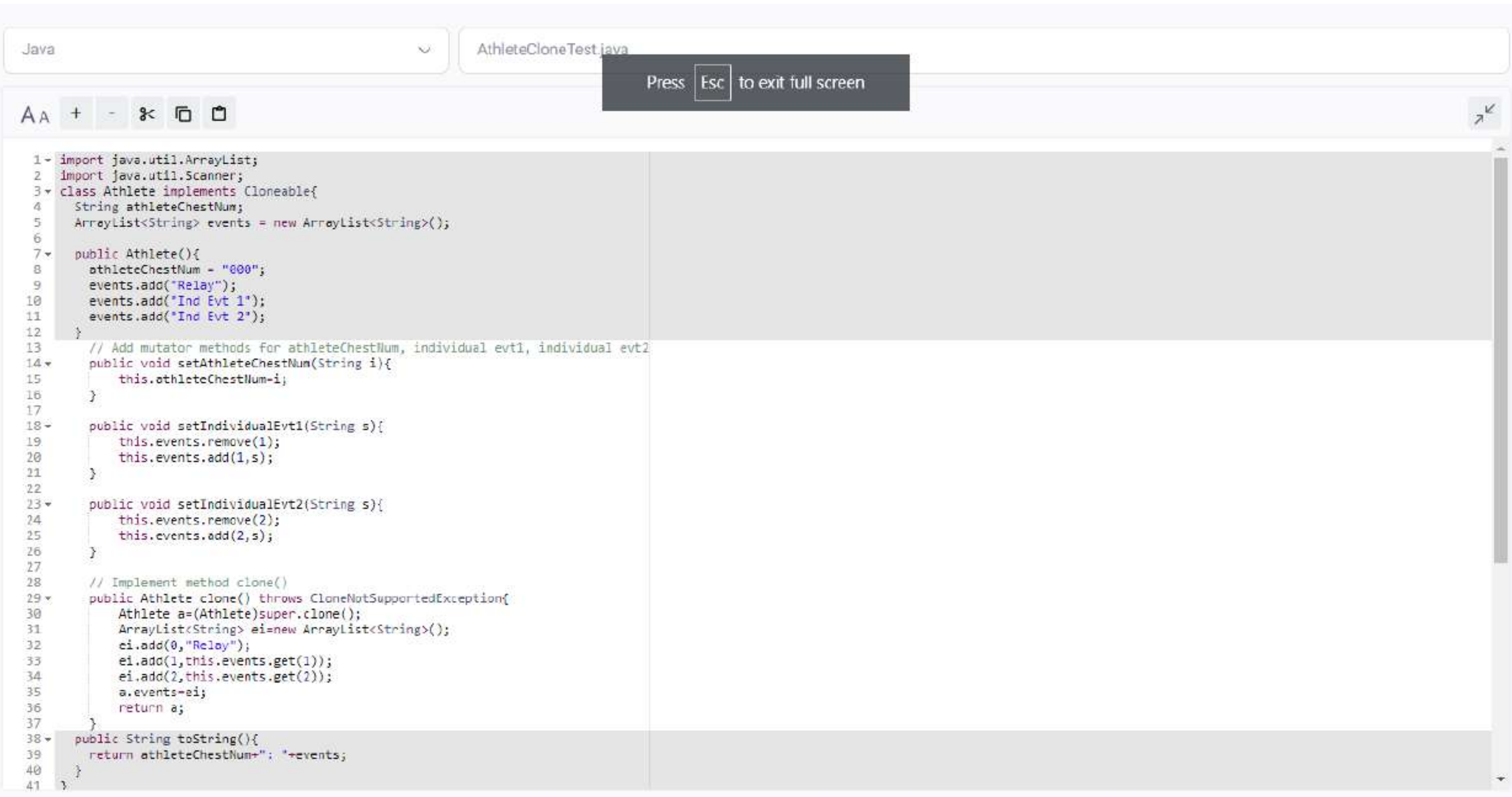
-- Overridden method `toString()`

-- Implement method `clone()`

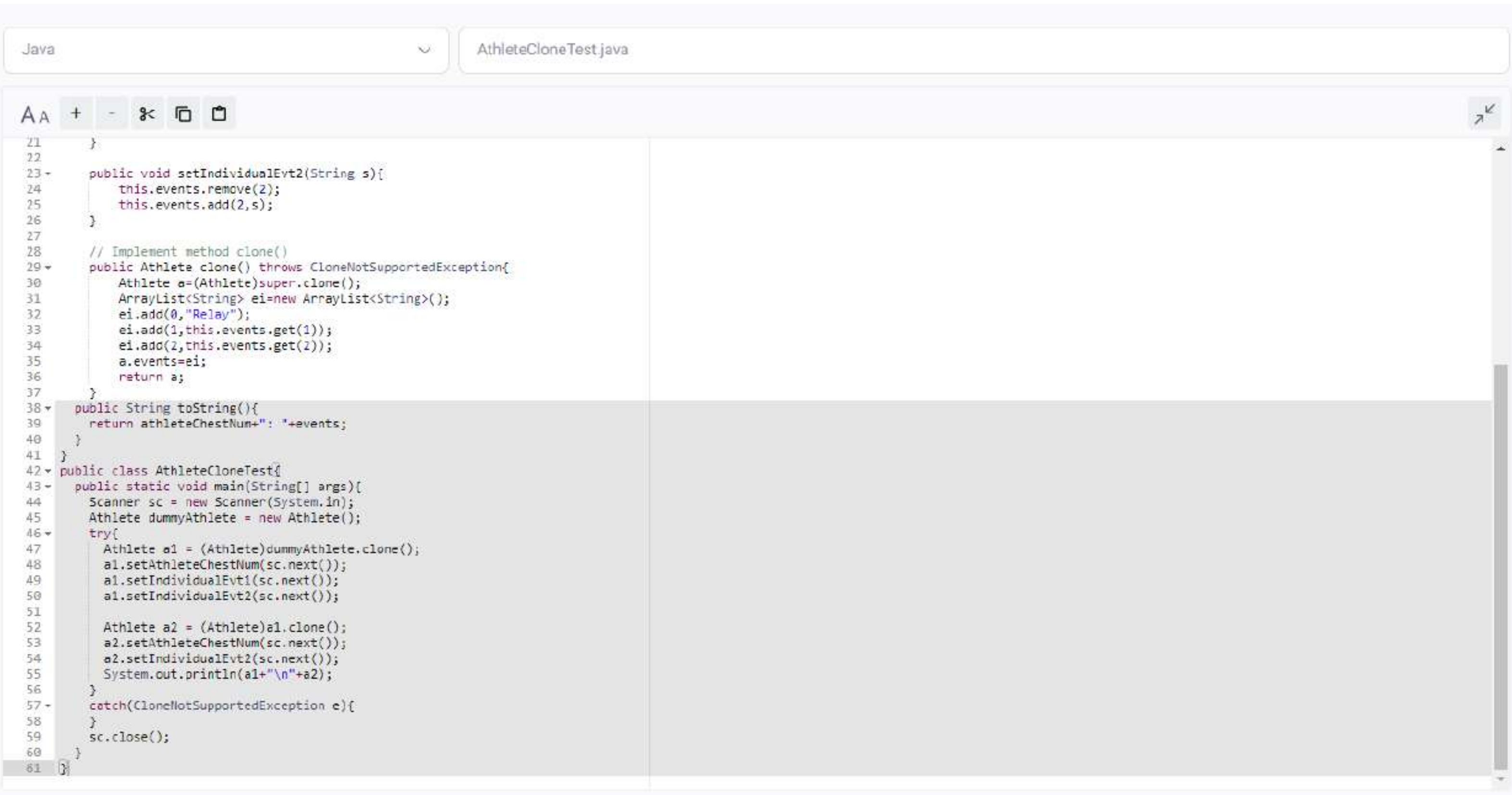
Class `AthleteCloneTest` contains the `main` method that takes the inputs and invokes appropriate methods to achieve the functionality.

Java documentation can be accessed at: https://docs.oracle.com/en/java/javase/11/docs/api/

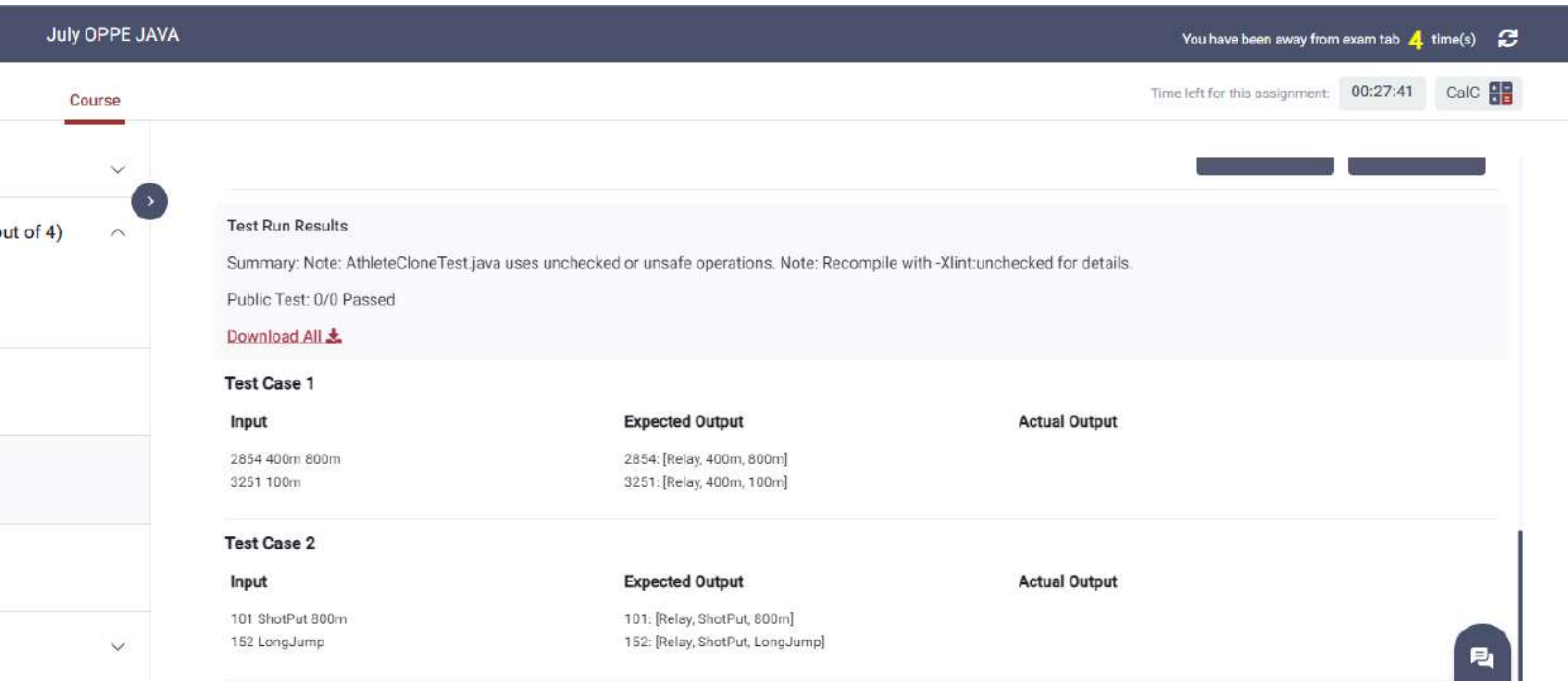
This assignment has public test cases. Please click on "Test Run" button to see the status of public test cases. Assignment will be evaluated only after submittin
using "Submit" button below. If you only test run the program, your assignment will not be graded and you will not see your score after the deadline.



```java
import java.util.ArrayList;
import java.util.Scanner;
class Athlete implements Cloneable{
  String athleteChestNum;
  ArrayList<String> events = new ArrayList<String>();

  public Athlete(){
    athleteChestNum = "000";
    events.add("Relay");
    events.add("Ind Evt 1");
    events.add("Ind Evt 2");
  }
    // Add mutator methods for athleteChestNum, individual evt1, individual evt2
    public void setAthleteChestNum(String i){
        this.athleteChestNum=i;
    }

    public void setIndividualEvt1(String s){
        this.events.remove(1);
        this.events.add(1,s);
    }

    public void setIndividualEvt2(String s){
        this.events.remove(2);
        this.events.add(2,s);
    }

    // Implement method clone()
    public Athlete clone() throws CloneNotSupportedException{
        Athlete a=(Athlete)super.clone();
        ArrayList<String> ei=new ArrayList<String>();
        ei.add(0,"Relay");
        ei.add(1,this.events.get(1));
        ei.add(2,this.events.get(2));
        a.events=ei;
        return a;
    }
  public String toString(){
    return athleteChestNum+": "+events;
  }
}
```



```java
    }

    public void setIndividualEvt2(String s){
        this.events.remove(2);
        this.events.add(2,s);
    }

    // Implement method clone()
    public Athlete clone() throws CloneNotSupportedException{
        Athlete a=(Athlete)super.clone();
        ArrayList<String> ei=new ArrayList<String>();
        ei.add(0,"Relay");
        ei.add(1,this.events.get(1));
        ei.add(2,this.events.get(2));
        a.events=ei;
        return a;
    }
  public String toString(){
    return athleteChestNum+": "+events;
  }
}
public class AthleteCloneTest{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    Athlete dummyAthlete = new Athlete();
    try{
      Athlete a1 = (Athlete)dummyAthlete.clone();
      a1.setAthleteChestNum(sc.next());
      a1.setIndividualEvt1(sc.next());
      a1.setIndividualEvt2(sc.next());

      Athlete a2 = (Athlete)a1.clone();
      a2.setAthleteChestNum(sc.next());
      a2.setIndividualEvt2(sc.next());
      System.out.println(a1+"\n"+a2);
    }
    catch(CloneNotSupportedException e){
    }
    sc.close();
  }
}
```


July OPPE JAVA

You have been away from exam tab 4 time(s)

Course

Time left for this assignment: 00:27:41 CalC

ut of 4)

Test Run Results

Summary: Note: AthleteCloneTest.java uses unchecked or unsafe operations. Note: Recompile with -Xlint:unchecked for details.

Public Test: 0/0 Passed

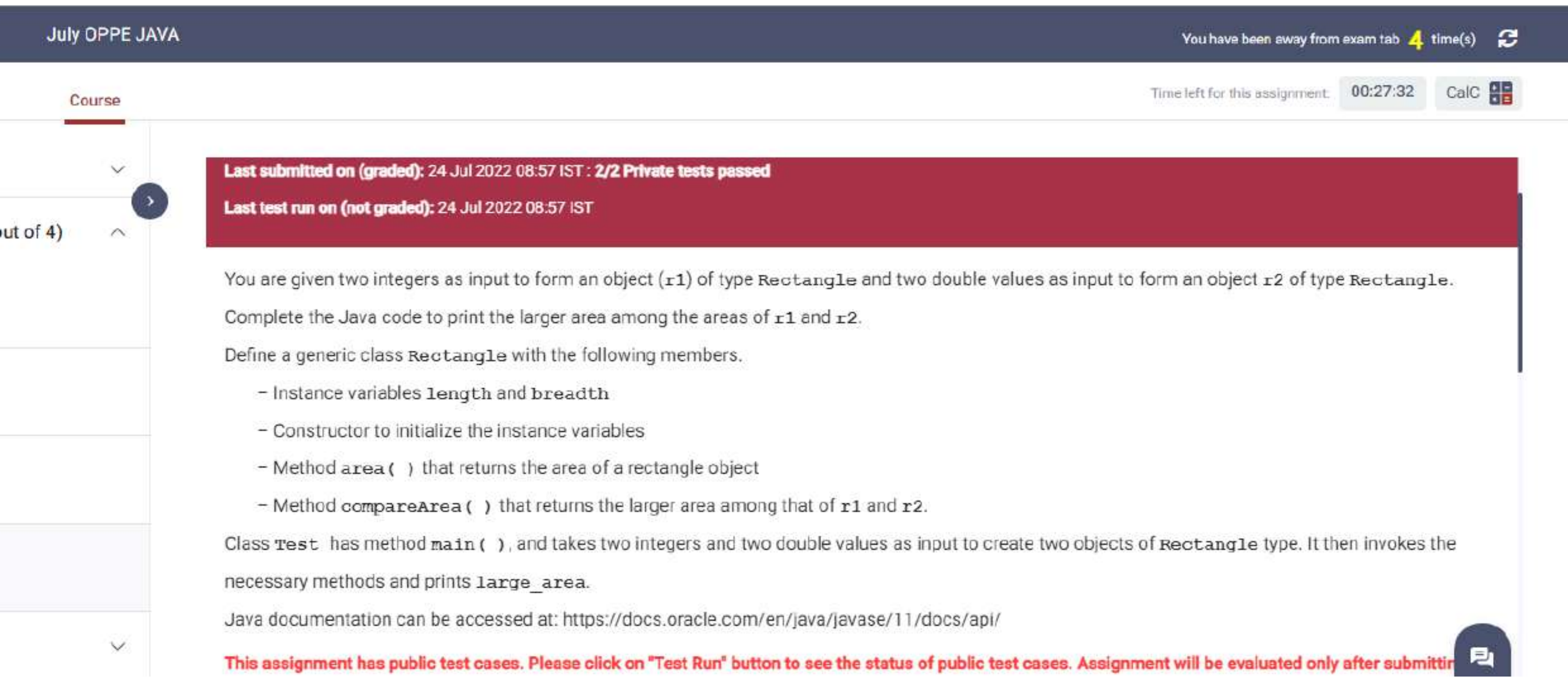
Download All

Test Case 1

Input	Expected Output	Actual Output
2854 400m 800m 3251 100m	2854: [Relay, 400m, 800m] 3251: [Relay, 400m, 100m]	

Test Case 2

Input	Expected Output	Actual Output
101 ShotPut 800m 152 LongJump	101: [Relay, ShotPut, 800m] 152: [Relay, ShotPut, LongJump]	


July OPPE JAVA

You have been away from exam tab 4 time(s)

Course

Time left for this assignment: 00:27:32 CalC

ut of 4)

Last submitted on (graded): 24 Jul 2022 08:57 IST : **2/2 Private tests passed**

Last test run on (not graded): 24 Jul 2022 08:57 IST

You are given two integers as input to form an object (`r1`) of type `Rectangle` and two double values as input to form an object `r2` of type `Rectangle`.

Complete the Java code to print the larger area among the areas of `r1` and `r2`.

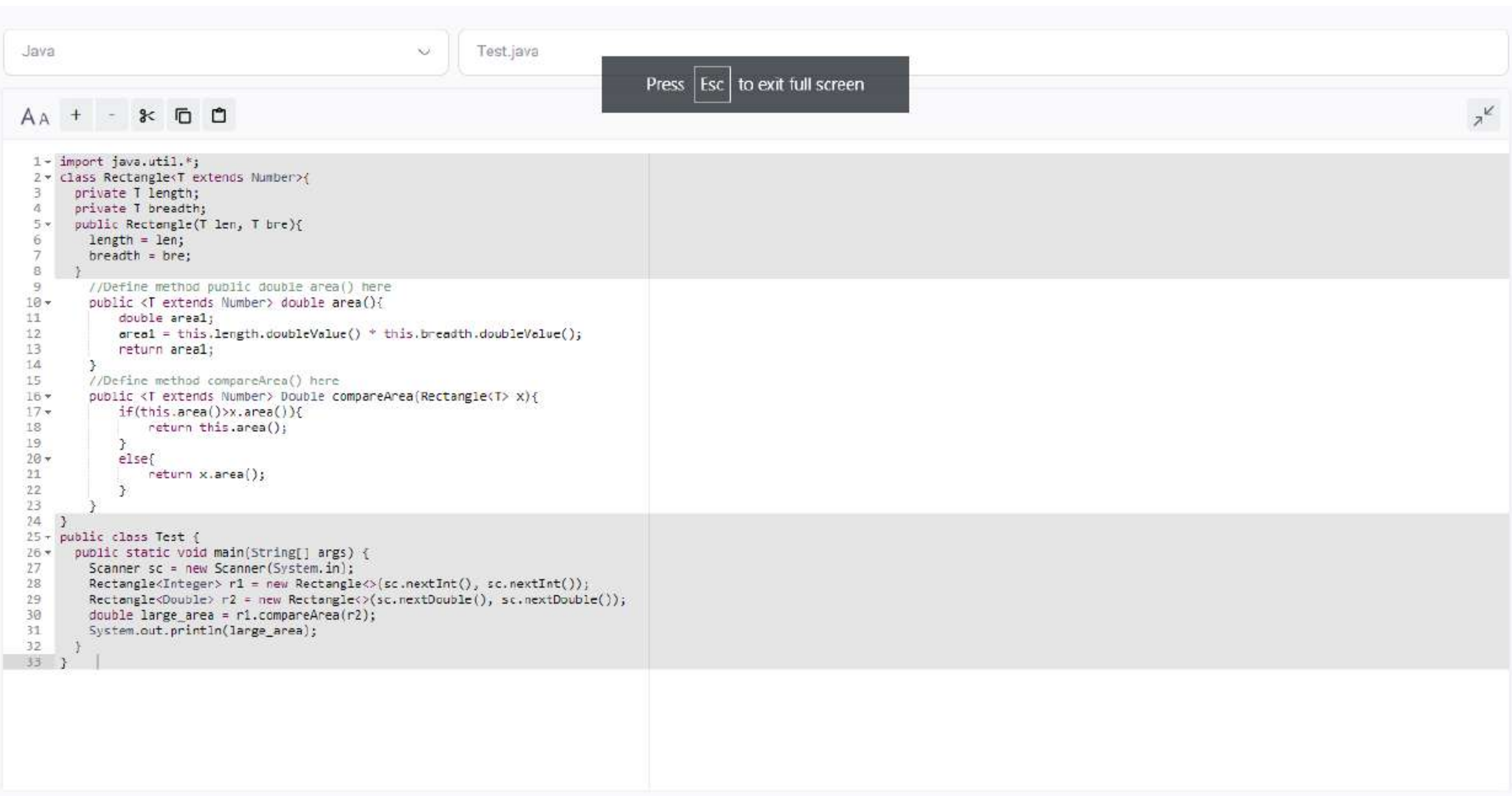
Define a generic class `Rectangle` with the following members.

- Instance variables `length` and `breadth`
- Constructor to initialize the instance variables
- Method `area( )` that returns the area of a rectangle object
- Method `compareArea( )` that returns the larger area among that of `r1` and `r2`.

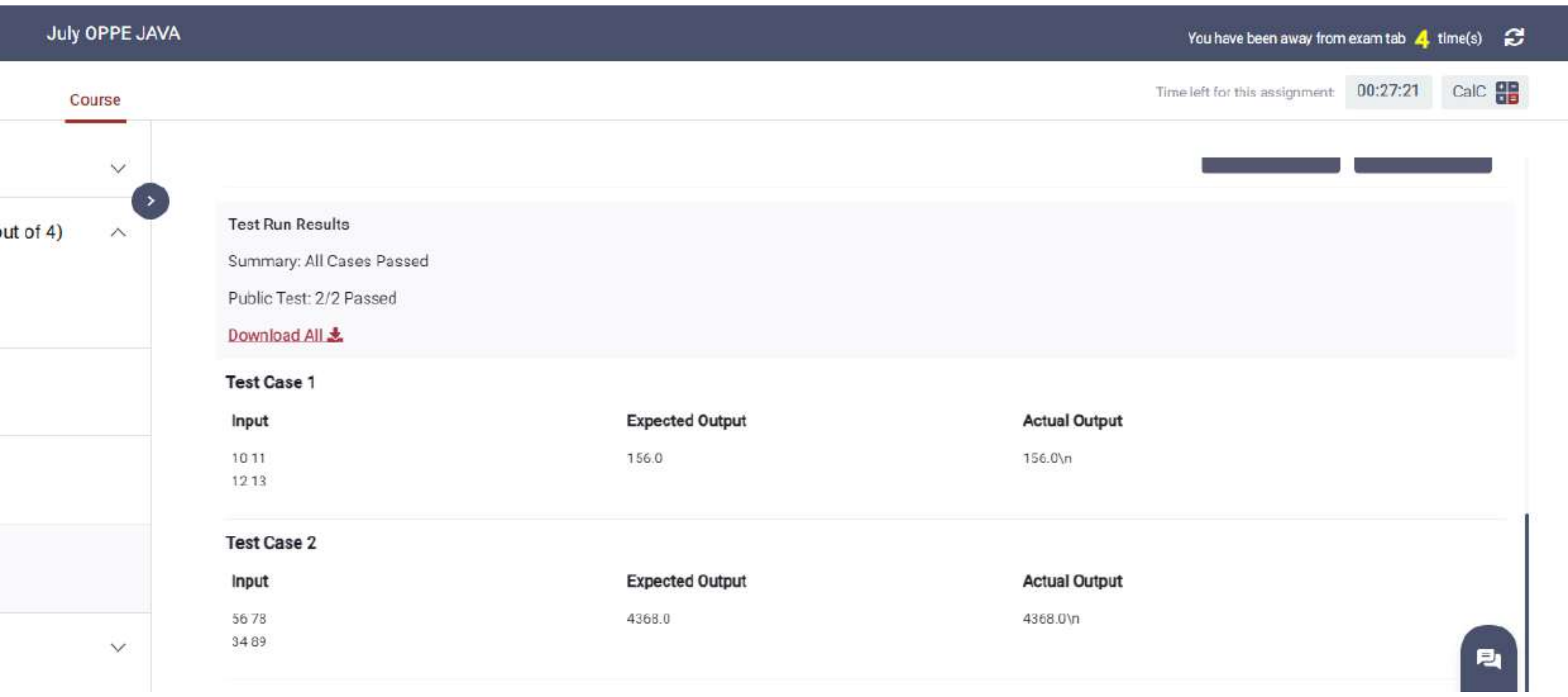
Class `Test` has method `main( )`, and takes two integers and two double values as input to create two objects of `Rectangle` type. It then invokes the necessary methods and prints `large_area`.

Java documentation can be accessed at: https://docs.oracle.com/en/java/javase/11/docs/api/

This assignment has public test cases. Please click on "Test Run" button to see the status of public test cases. Assignment will be evaluated only after submittin



```java
import java.util.*;
class Rectangle<T extends Number>{
  private T length;
  private T breadth;
  public Rectangle(T len, T bre){
    length = len;
    breadth = bre;
  }
    //Define method public double area() here
    public <T extends Number> double area(){
        double area1;
        area1 = this.length.doubleValue() * this.breadth.doubleValue();
        return area1;
    }
    //Define method compareArea() here
    public <T extends Number> Double compareArea(Rectangle<T> x){
        if(this.area()>x.area()){
            return this.area();
        }
        else{
            return x.area();
        }
    }
}
public class Test {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Rectangle<Integer> r1 = new Rectangle<>(sc.nextInt(), sc.nextInt());
    Rectangle<Double> r2 = new Rectangle<>(sc.nextDouble(), sc.nextDouble());
    double large_area = r1.compareArea(r2);
    System.out.println(large_area);
  }
}
```


July OPPE JAVA
You have been away from exam tab 4 time(s)
Course
Time left for this assignment 00:27:21
CalC
ut of 4)

Test Run Results

Summary: All Cases Passed

Public Test: 2/2 Passed

Download All

Test Case 1

Input	Expected Output	Actual Output
10 11 12 13	156.0	156.0\n

Test Case 2

Input	Expected Output	Actual Output
56 78 34 89	4368.0	4368.0\n