BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

K. K. BIRLA GOA CAMPUS
FIRST SEMESTER 2024-2025

CS F213 (Object Oriented Programming) – Lab 9 Assignment, Date: 16/10/24

Problem Title: Advanced Rectangle Analysis with Abstract Classes and Interfaces

---

Objective:

Enhance your Java program using abstract classes and the `Comparable` interface to manage rectangles.

Problem Description:

1. **Abstract Class `Figure`:**
   - Define an abstract class `Figure` that includes abstract methods for calculating the area and perimeter.
   - This class serves as a base for different geometric figures.
2. **Class `Rectangle`:**
   - Extend the abstract class `Figure`.
   - Implement the `Comparable<Rectangle>` interface.
   - Include properties for `width` and `height`.
   - Implement methods to calculate the area and perimeter.
   - Override the `compareTo` method to compare rectangles based on their area.
   - Implement methods to determine if one rectangle can fit inside another, considering rotations of 0 or 90 degrees.
3. **Class `RectangleManager`:**
   - Include a static method `public static void analyzeRectangles(List<Rectangle> rectangles)` that performs the following analyses:
     a. **First Output:** Determines and prints the **maximum number of rectangles** (from the list, excluding Rectangle 0) that can **fit inside** Rectangle 0. Rectangles can be rotated by 0 or 90 degrees to fit.
     b. **Second Output:** Determines and prints the **maximum number of rectangles** (from the list, excluding Rectangle 0) that Rectangle 0 can **fit into**. Rectangles can be rotated by 0 or 90 degrees.
     c. **Third Output:** Identifies and prints the rectangle from the list (excluding Rectangle 0) that has the **closest aspect ratio** to Rectangle 0. (Assume that only one rectangle will have the closest aspect ratio.) When determining the closest aspect ratio:
     - **Definition:** The aspect ratio of a rectangle is the ratio of its width to its height.

        - **Calculation:** For each rectangle, calculate both the original and rotated aspect ratios:
          Aspect Ratios of a Rectangle = { Width / Height, Height / Width }
        - **Comparison:** Compare these aspect ratios to both the original and rotated aspect ratios of Rectangle 0.
        - **Closest Match:** Find the rectangle (excluding Rectangle 0) with the smallest absolute difference in aspect ratio compared to Rectangle 0, considering rotations of 0 or 90 degrees.
    - d. **Fourth Output:** Determines and prints the **maximum number of rectangles** (from the list, **including Rectangle 0**) that can be **nested inside each other in sequence**, considering rotations. Each rectangle in the sequence must fit inside the next one.

Input:

- A list of rectangles with specified dimensions.
- Rectangle 0 is the primary rectangle for analysis.

Output:

- **First Output:** Maximum number of rectangles that can fit inside Rectangle 0.
- **Second Output:** Maximum number of rectangles that Rectangle 0 can fit into.
- **Third Output:** Index of Rectangle with the closest aspect ratio to Rectangle 0, considering rotations. (Assume that only one rectangle will have the closest aspect ratio.)
- **Fourth Output:** Maximum number of rectangles that can be nested inside each other in sequence.

Constraints:

- All rectangle dimensions are positive integers.
- Rotations are limited to 0 or 90 degrees.
- Use abstract classes and interfaces as specified.
- Ensure your solution is efficient and well-structured.

Example 1:

Input: The input given below corresponds to 5 rectangles. The first line contains the number of rectangles in the array. The remaining lines contain width and height of each rectangle

```
5
10 20
5 5
15 5
20 10
5 25
```

Output: The output corresponds to the 4 integer values for the above problem instance.

```
3
1
3
4
```

Explanation:
There are 3 rectangles that can fit inside the rectangle at index position 0.
There is only 1 rectangle that the Rectangle 0 can fit into.
Rectangle 3 has the closest aspect ratio to Rectangle 0.
Rectangle 0 can contain Rectangle 3, which, in turn, can contain Rectangle 2, which, in turn, can contain Rectangle 1. So the answer is 4.

Instructions:

Follow the steps given below to complete the OOP MidSem Long problem.

Step 1: Read Lab Question

Read and understand the Lab problem given above.

Step 2: Edit the Solution java file

Edit the solution java files to solve the given problem.

For Lab 9, the main java file is named: `L9_Q1_Main.java`. Note that `L9` refers to Lab 9 and `Q1` refers to Question 1.

Step 3: See the input and the expected output

Use the following command to see the input and the expected output for test case `T1`. Note that `L9` refers to Lab 9, `Q1` refers to Question 1 and `T1` refers to test case 1.

```
:~$ ./RunTestCase L9 Q1 YourBITSId T1
```

Use your own (13 character) BITS Id in place of `YourBITSId` in the above command. Type your BITS Id in upper case (capital letters). Ensure that you enter your BITS Id in 202XA7PSXXXXG format.

Run the command from within the folder containing the executables and the java files.

Step 4: Modify the solution java file

Modify the solution Java file and repeat Step 3 until all the test cases are passed. The test cases are numbered `T1`, `T2`, etc.

Step 5: Passing evaluative test cases

There are evaluative test cases whose expected output is masked. The evaluative test cases are numbered `ET1`, `ET2`, etc. The lab marks will be based on the evaluative test cases. Use the following command to check whether your solution passes the evaluative test cases.

```
:~$ ./RunTestCase L9 Q1 YourBITSId ET1
```

Ensure that your solution passes all the evaluative test cases ET1, ET2, etc. The expected output of the evaluative test cases are *hidden*; therefore, only the hash value is shown.

Step 6: Create submission zip file

Use the following command to create the submission zip file.

```
:~$ ./CreateSubmission L9 YourBITSId
```

The first command line argument must correspond to the lab number and the second command line argument must be your 13 character BITS id.

The above command will create a zip file. The command will also list the evaluative test cases that your solution program has passed.

Step 7: Upload submission zip file

Upload the submission zip file created in Step 6. Do not change the name of the zip file or modify any file inside the zip file. You will not be awarded marks if the zip file is tampered with in any manner.