Lab Exercise 6
CS 2334

October 1, 2015

Introduction

In this lab, you will experiment with using inheritance in Java through the use of abstract classes and interfaces. You will implement a set of classes that represent various shapes. In addition, your implementation will facilitate the comparison of shape objects, even when they are different shapes.

Learning Objectives

By the end of this laboratory exercise, you should be able to:

1. Create and extend abstract classes and methods
2. Use interfaces to define standard behavior across multiple classes
3. Appropriately select an abstract class or interface based on the requirements of the solution

Proper Academic Conduct

This lab is to be done individually. Do not look at or discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net.

Preparation

1. Import the existing lab6 implementation into your eclipse workspace.

   (a) Download the lab6 implementation:
   `http://www.cs.ou.edu/~fagg/classes/cs2334/labs/lab6/lab6-initial.zip`

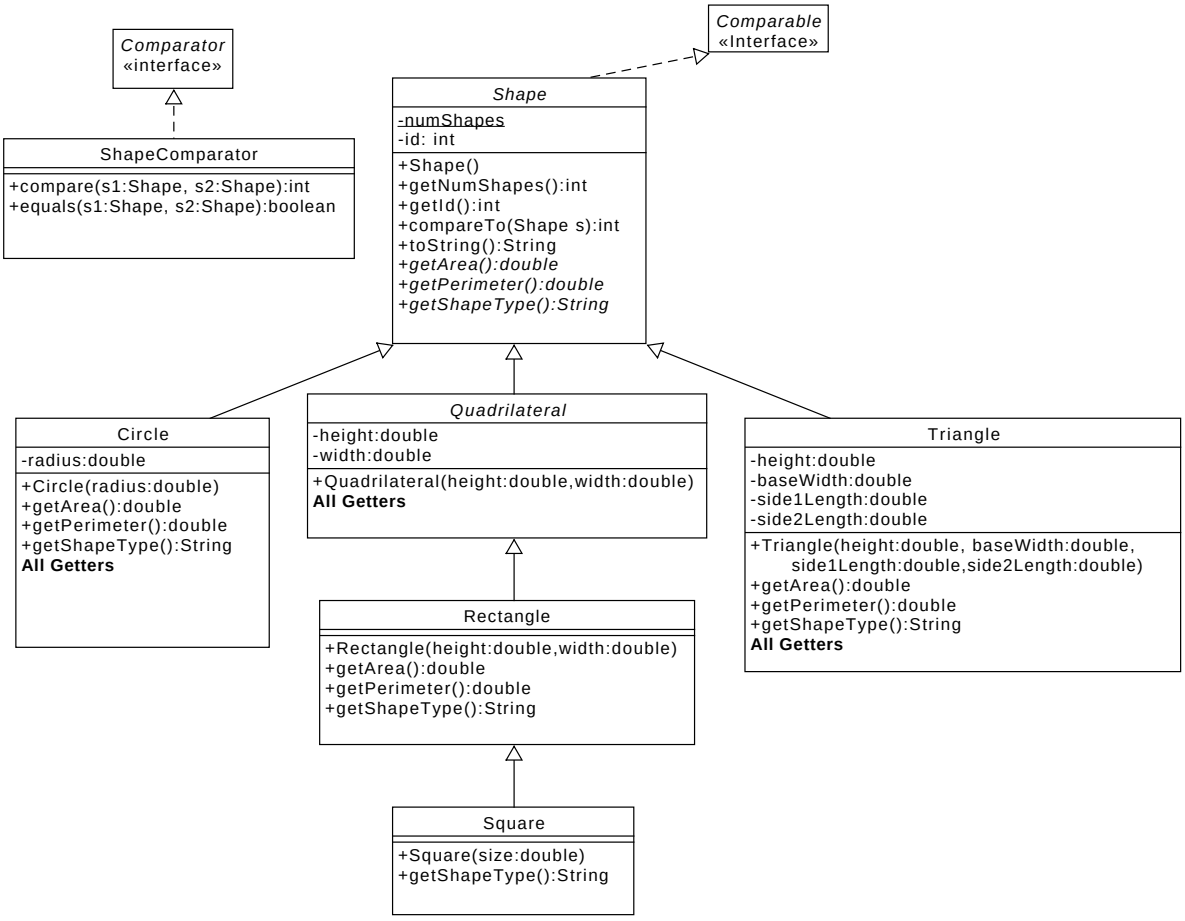
   (b) In Eclipse, select *File/Import*

   (c) Select *General/Existing projects into workspace*. Click *Next*

   (d) Select *Select archive file*. Browse to the lab6-initial.zip file. Click *Finish*

Representing Different Shapes

Below is the UML representation of a set of classes that represent various shapes. Your task will be to implement this set of classes and an associated set of JUnit test procedures.



The classes in italics represent abstract classes or interfaces. The concrete child classes must implement all methods from the abstract parent classes In this lab, **Shapes** and **Quadrilateral** are the abstract classes.

The line from **Shape** to **Comparable** indicates that **Shape** must implement the **Comparable** interface. Similarly, the **ShapeComparator** class must implement

the **Comparator** interface. **ShapeComparator** compares shapes based on their perimeter.

All instances of a **Shape** are given a unique int *id*. These are to be assigned by the **Shape** constructor. The instance of a shape is assigned an *id* of 0 (zero); the next is assigned 1.

In the **Triangle** class, the lengths of the three sides are *baseWidth*, *side1Length* and *side2Length*.

Lab 6: Specific Instructions

Start from the class files that are provided in lab6-initial.zip.

1. Create a new Java class (or modify a provided one) for each object class described in the UML diagram
    - Be sure that the class name is exactly as shown
    - You must use the default package, meaning that the package field must be left blank
2. Implement the attributes and methods for each class
    - Use the same spelling for instance variables and method names as shown in the UML
    - Do not add functionality to the classes beyond what has been specified
    - Don't forget to document as you go!
3. Create *Lab6Test* class and JUnit to thoroughly test all of your code
    - You need to convince yourself that everything is working properly
    - Make sure that you cover all the classes and methods while creating your test. Keep in mind that we have our own tests that we will use for grading.

Final Steps

1. Generate Javadoc using Eclipse.
    - Select *Project/Generate Javadoc...*

- Make sure that your project is selected, as are the classes: Driver, Circle, Quadrilateral, Rectangle, Triangle, Square, Shape, ShapeComparator, and Lab6Test (check for these individually!)
- Select *Private* visibility
- Use the default destination folder
- Click *Finish*

2. Open the *lab6/doc/index.html* file using your favorite web browser or Eclipse (double clicking in the package explorer will open the web page). Check to make sure that that all of your classes are listed and that all of your documented methods have the necessary documentation.

3. If you complete the above instructions during lab, you may have your implementation checked by one of the TAs.

Submission Instructions

- All required components (source code and compiled documentation) are due at 11:59pm on Friday, October 2nd.
- Prepare your submission file:
  1. Select the project in the *Package Explorer* window.
  2. Right-click. Select *Export*
  3. Select *General/Archive File*
  4. Expand your project and verify that both the *src* and *doc* folders are selected, as well as all of their contents
  5. Enter the archive file name: *lab6.zip* (note that you may want to browse to a different destination folder)
  6. Select *Save in zip format*
  7. Click *Finish*
- Submit your zip file to the lab6 folder on D2L.

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 35 points

Program formatting: 10 points

(10) The program is properly formatted (including indentation, curly brace and semicolon locations).

(5) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 15 points

(15) The program is using proper data types and method calls.

(10) There is one error in data type or method call selection.

(5) There are two errors in data type or method call selection.

(0) There are multiple errors in data type and method call selection.

Required Methods: 10 points

(10) All of the required methods are implemented.

(7) A component of one required method is missing.

(4) A method is not implemented or components are missing from two methods.

(0) Two or more required methods are not implemented or components from three or more methods are missing.

Proper Execution: 30 points

Output: 15 points

(15) The program passes all tests.

(10) The program fails one test.

(5) The program fails two tests.

(0) The program fails three or more tests.

Execution: 15 points

(15) The program executes with no errors.

(8) The program executes, but there is one minor error.

(0) The program does not execute.

Documentation and Submission: 35 points

Project Documentation: 5 points

(5) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 10 points

(10) Every method contains all of the required documentation elements ahead of the method prototype.

(7) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements.

(0) The method documentation is missing more than two of the required documentation elements.

Inline Documentation: 10 points

(10) Every method contains appropriate inline documentation.

(7) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 10 points

(10) The correct zip file name is used and has the correct contents.

(5) The correct zip file name is used, but one required component is missing.

(0) An incorrect zip file name is used or more than one required component is missing.