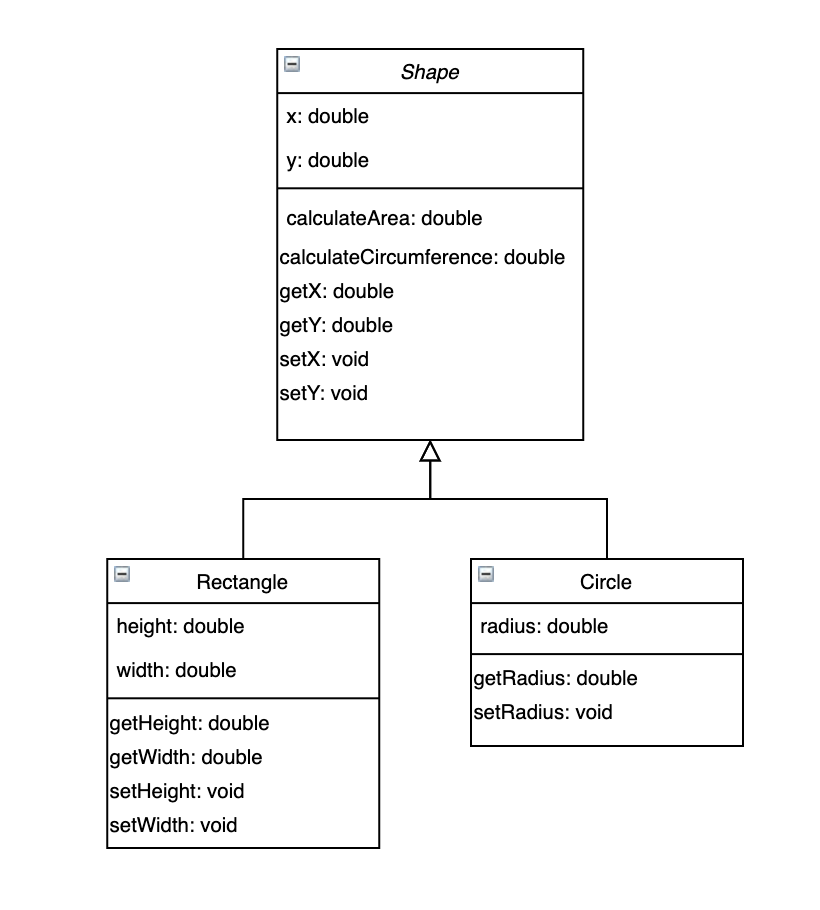
**PSG COLLEGE OF TECHNOLOGY**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**23MX26 - Java Programming Laboratory**

**Hands-on Worksheet 8.2 Inheritance with Array List**

2. You will be completing the class PolymorphismPractice. Shape, Circle, and Rectangle have already been written for you. Provided below is a UML diagram that gives a visual representation of this lab.



Step 1

In the main method, initialize an ArrayList that holds objects of type Shape.

Step 2

|  |
| --- |
| ArrayList and inheritance An ArrayList is a powerful tool that allows you to store a large number of objects without having to preallocate memory. This tool becomes enhanced with the inclusion of inheritance. Although an abstract class cannot be instantiated normally, it can be used as the argument to an ArrayList constructor. When used in this style, the ArrayList can now store any type of object who inherits from the abstract class used in the declaration. Let’s say we have an abstract class called Shape, we can declare an ArrayList as such:  **ArrayList<Shape>** list **=** **new** **ArrayList<Shape>();**  This list now has the ability to store any object that inherits from the Shape class. The same methodology can be applied to interfaces; any class which implements a specific interface can be stored in an Array or ArrayList of that type of interface. Where this really comes in to play is with polymorphism. |

Let’s now fill the ArrayList with 5 Circles and 5 Rectangles (for a total of 10 objects). A for loop is definitely helpful in this step, and also consider using Math.random().

Testing

To test this step, you can use the ArrayList [**size()**](https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#size--) method to make sure it has a size of 10, or try printing out all 10 items in the ArrayList.

Step 3

Let’s complete the getInfo method. Start by editing this method header for getInfo so that takes in an ArrayList of Shape objects as a parameter. Now we will be iterating over the list and adding information to a StringBuilder. For this method use a **[foreach](https://www.javatpoint.com/for-each-loop)** loop to iterate through the list. For each shape in the array, add the shape information to your StringBuilder *HINT: use the toString method for that Object*.

Enhanced For Loops

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | import java.util.ArrayList**;**  **public** **class** **test** **{**  **public** **static** **void** **main(String[]** args**)** **{**  **ArrayList<Shape>** list **=** **new** **ArrayList<Shape>();**  list**.**add**(new** **Circle(**12**,** 2**,** 2**));**  list**.**add**(new** **Rectangle(**13**,** 21**,** 2**,** 2**));**  list**.**add**(new** **Triangle(**12**,** 10**,** 2**,** 2**));**  list**.**add**(new** **Circle(**5**,** 5**,** 5**));**  list**.**add**(new** **Rectangle(**5**,** 5**,** 2**,** 2**));**    *//foreach loop*  **for** **(Shape** s **:** list**)** **{**  **System.**out**.**println**(**s**.**getClass**()** **+** ": " **+** s**.**calculateArea**());**  **}**  **}**  **}** |

Let’s also add the lines “Has area: *the area of the shape*”, as well as “Has circumference: *the circumference of the shape*”. Please make sure that the area and circumference found are rounded to two decimal places. Separate every object with a new line character. Finally, return the String of all the information found.

**An example of how this should look when printed:**

Circle with radius: 70.84, Centered at X: 69.27 Y: 89.29

Has area: 15764.34

Has circumference: 445.08

Rectangle with height: 9.34 and width: 70.01, Centered at X: 9.69 Y: 84.71

Has area: 654.07

Has circumference: 158.70

Testing

Call and print the getInfo method inside your main method.

Step 4

Let’s now complete the findCircles method. Start by editing this method header for findCircles so that it takes in an ArrayList of Shape objects as a parameter. Now, using another for each loop, and the **[instanceof](https://www.javatpoint.com/downcasting-with-instanceof-operator)** keyword add the information of **ONLY** Circles to a StringBuilder (make sure every Circle Object’s info is separated by a new line). *HINT use the toString method*

Finally return a string holding all of the different Circle’s information.

**An example of how this should look when printed:**

Circle with radius: 70.84, Centered at X: 69.27 Y: 89.29

Circle with radius: 23.29, Centered at X: 1.62 Y: 26.34

Testing

Call and print the findCircles method inside your main method.