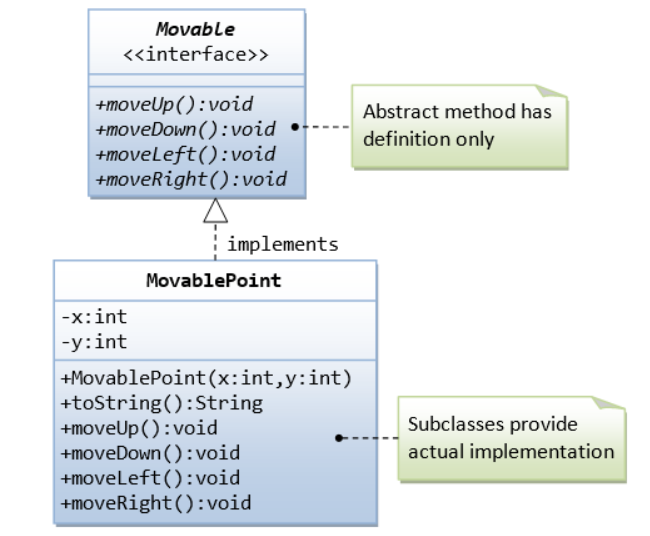
Assignment (BCAC391)

**(Interface, abstract method)**



**testMovableInterface.java**

We can also upcast subclass instances to the Movable interface, via polymorphism, similar to an abstract class.

public class testMovableInterface {

public static void main(String[] args) {

MovablePoint p1 = new MovablePoint(1, 2);

System.out.println(p1);//(1,2)

p1.moveDown();

System.out.println(p1);//(1,3)

p1.moveRight();

System.out.println(p1);//(2,3)

// Test Polymorphism

Movable p2 = new MovablePoint(3, 4); // upcast

p2.moveUp();

System.out.println(p2);//(3,3)

MovablePoint p3 = (MovablePoint)p2; // downcast

System.out.println(p3);//(3,3)

}

}

**Expected Output:**

(1,2)

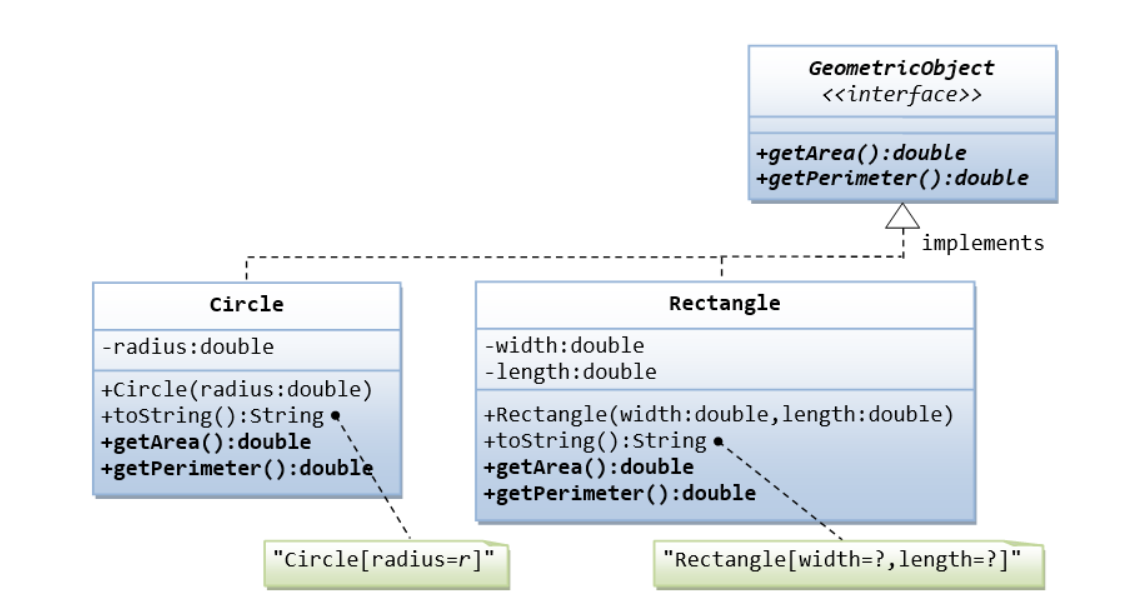
(1,3)

(2,3)

(3,3)

(3,3)

1. Write an **interface** called **GeometricObject**, which contains 2 abstract methods: getArea() and getPerimeter(), as shown in the class diagram. Also write an implementation class called Circle. Mark all the overridden methods with annotation @Override.



Below is a test driver **(testGeometricObjec.java)** to test the above class:

public class testGeometricObjects {

public static void main(String[] args) {

Circle c1=new Circle(10.0);

System.out.println(c1);

Circle c2=new Circle(20.0);

System.out.println("Area:"+c2.getArea());

System.out.println("Perimeter:"+c2.getPerimeter());

Rectangle r1=new Rectangle(10.0, 5.0);

System.out.println(r1);

Rectangle r2=new Rectangle(5.0,2.0);

System.out.println("Area"+r2.getArea());

System.out.println("Perimeter"+r2.getPerimeter());

}

}

**Expected Output is:**

Circle[radius=10.0]

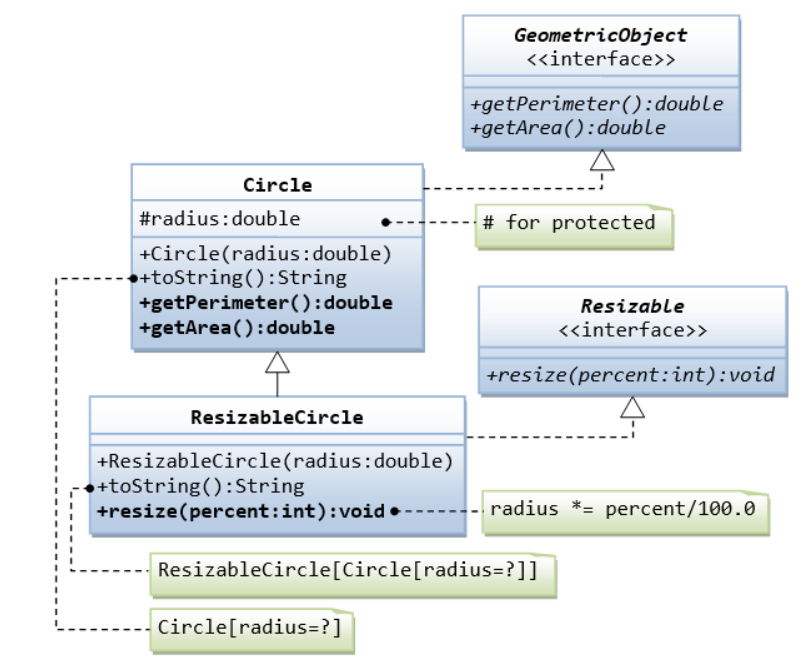
Area:1256.6370614359173

Perimeter:125.66370614359172

Rectangle[length=10.0, width=5.0]

Area10.0

Perimeter14.0



Below is the test drive (**testResizableGeometricObject.java**) code to implement above class:

public class testResizableGeometricObject {

public static void main(String[] args) {

// TODO code application logic here

Circle c1=new Circle(10.0);

System.out.println(c1);

System.out.println("Area"+c1.getArea());

System.out.println("Perimeter"+c1.getPerimeter());

ResizableCircle r1=new ResizableCircle(200.0);

System.out.println(r1);

r1.resize(25);

System.out.println(r1);

}

}

**Expected Output:**

Circle[radius=10.0]

Area314.1592653589793

Perimeter62.83185307179586

ResizableCircle[Circle[radius=200.0]]

ResizableCircle[Circle[radius=50.0]]