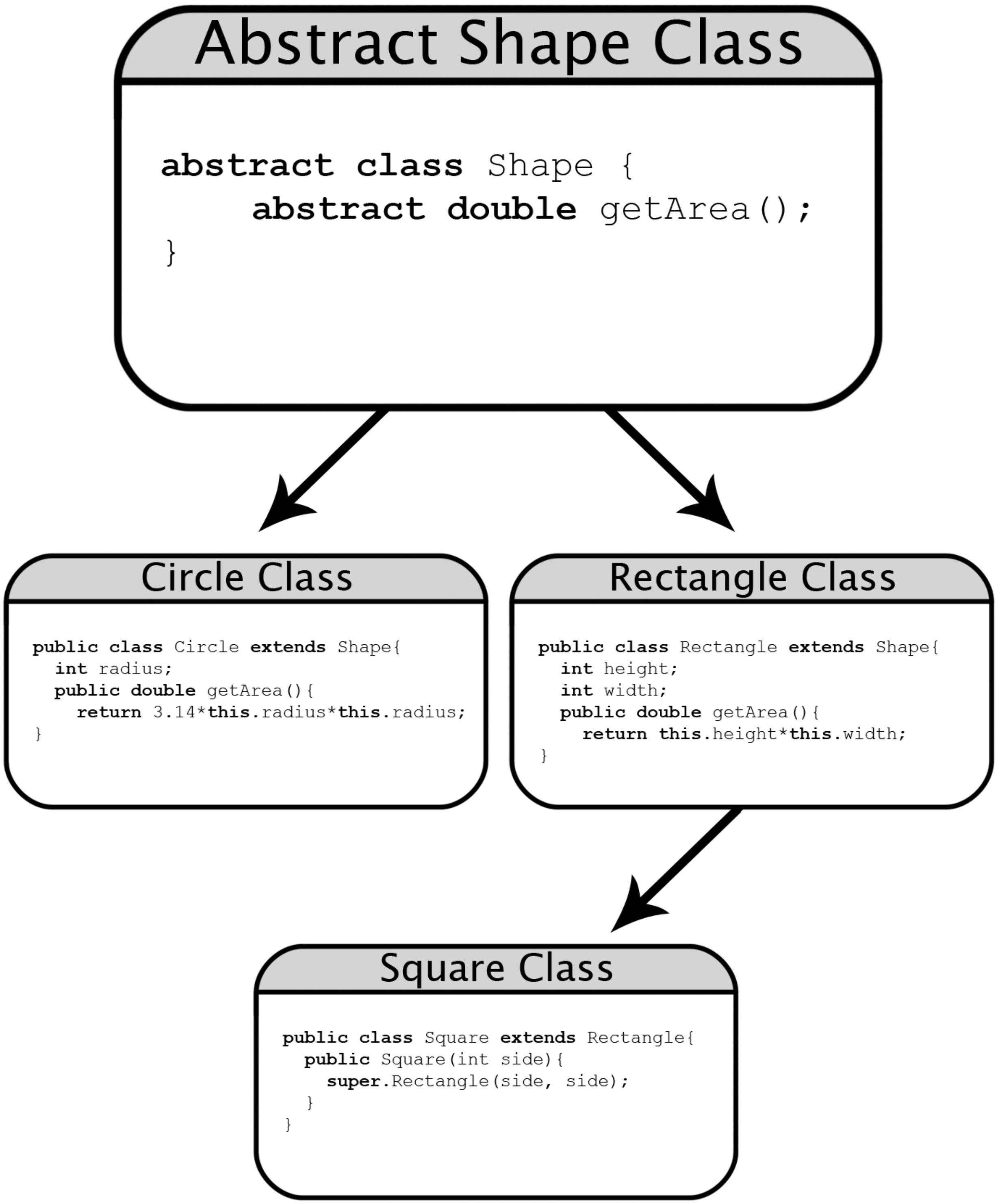
Polymorphism

Polymorphism is a method that a programmer can use to handle objects that, despite differing implementations and behavior, can be accessed through a common interface.

To implement polymorphism, a programmer would create a “base” abstract class that defines the methods that a class should have. The abstract base class cannot be instantiated itself and only extended. Then, classes can be created that “extend” the abstract class and provide actual implementations to the required methods.

For example, let’s say that a developer is creating a program that deals with several shapes. If the developer were clever, he would create an abstract “Shape” class. One method that the abstract Shape class could have would be a “getArea()” method, which would return the area of the shape. The developer would then subclass the abstract “Shape” class and provide specific implementations of these methods, including getArea(). For instance, the developer could choose to make a “Circle” and a “Rectangle” class. Both of these classes would have a getArea() method that would be implemented differently. The Circle.getArea method would look like Pi*r*r, while the Rectangle.getArea method would look like length*width. Now let’s say that the developer wants to create a “Square” class. The developer could create a subclass of the “Rectangle” class that accomplishes this. Providing that the developer still specifies a length and width (side = length = width), they could still use the Rectangle.getArea function.

Now the developer has a number of classes that subclass the abstract “Shape” class. Let’s say that our developer now needs to write a program that computes the total area of all the Shapes in a list. The developer can now simply iterate over the list and call the getArea() function on each shape without any regard for the shape itself.



```
abstract class Shape {
    abstract double getArea();
}
```

```
public class Circle extends Shape{
  int radius;
  public double getArea(){
    return 3.14*this.radius*this.radius;
}
```

```
public class Rectangle extends Shape{
  int height;
  int width;
  public double getArea(){
    return this.height*this.width;
}
```

```
public class Square extends Rectangle{
  public Square(int side){
    super.Rectangle(side, side);
  }
}
```