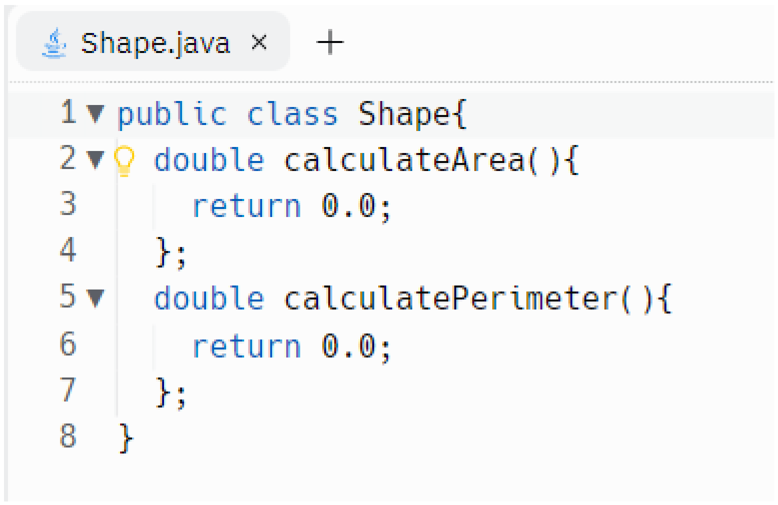
Note code smells are somewhat opinionated and is not necessarily an issue but rather an indication of a potential issue. However, if that potential is unlikely given the use case of the application then it is not worthy to spend time addressing.

Inappropriate Abstraction

When a concrete parent class is created but would never be instantiated by a user.

Example:
Shape is defined as a concrete class but judging by its implementation, the user would never need to create an instance of shape. Additionally, it provides no useful concrete methods that are used as-is so there is little reason for it to be an abstract class either, use an interface instead.



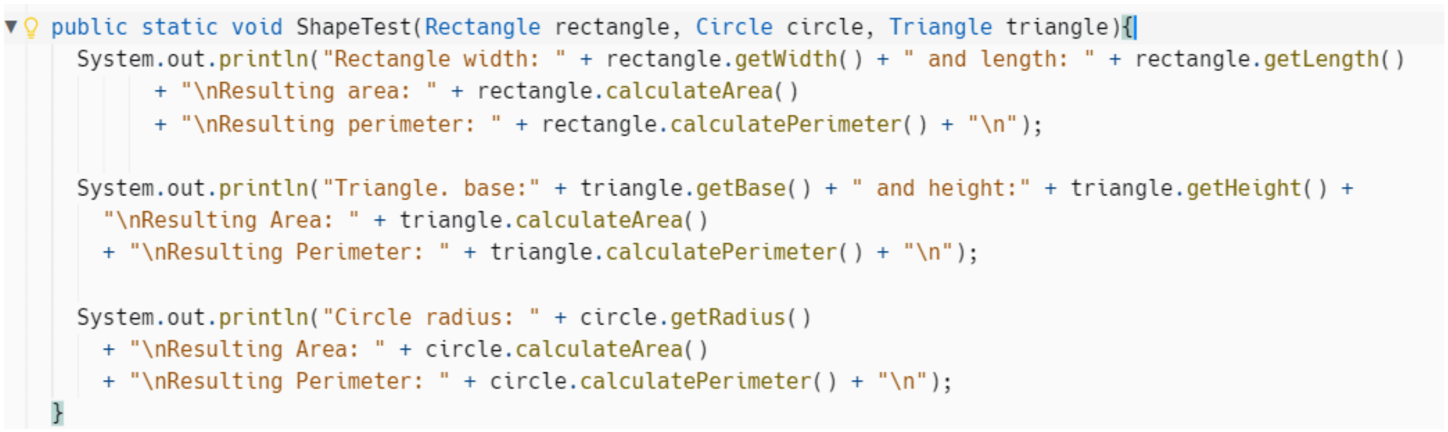
Shape.java

```java
public class Shape{
  double calculateArea(){
    return 0.0;
  };
  double calculatePerimeter(){
    return 0.0;
  };
}
```

Redundant Code (OCP Violation)

When code blocks are repetitive at a glance, often a loop with polymorphism may be introduced.
The intent of the user is to test the methods by printing dimensions, perimeter and area of the shapes. However, it is done in a redundant manner. To handle additional shapes a similar code block must be copied and pasted.

Before:



```java
public static void ShapeTest(Rectangle rectangle, Circle circle, Triangle triangle){
    System.out.println("Rectangle width: " + rectangle.getWidth() + " and length: " + rectangle.getLength()
          + "\nResulting area: " + rectangle.calculateArea()
          + "\nResulting perimeter: " + rectangle.calculatePerimeter() + "\n");

    System.out.println("Triangle. base:" + triangle.getBase() + " and height:" + triangle.getHeight() +
      "\nResulting Area: " + triangle.calculateArea()
      + "\nResulting Perimeter: " + triangle.calculatePerimeter() + "\n");

    System.out.println("Circle radius: " + circle.getRadius()
      + "\nResulting Area: " + circle.calculateArea()
      + "\nResulting Perimeter: " + circle.calculatePerimeter() + "\n");
}
```

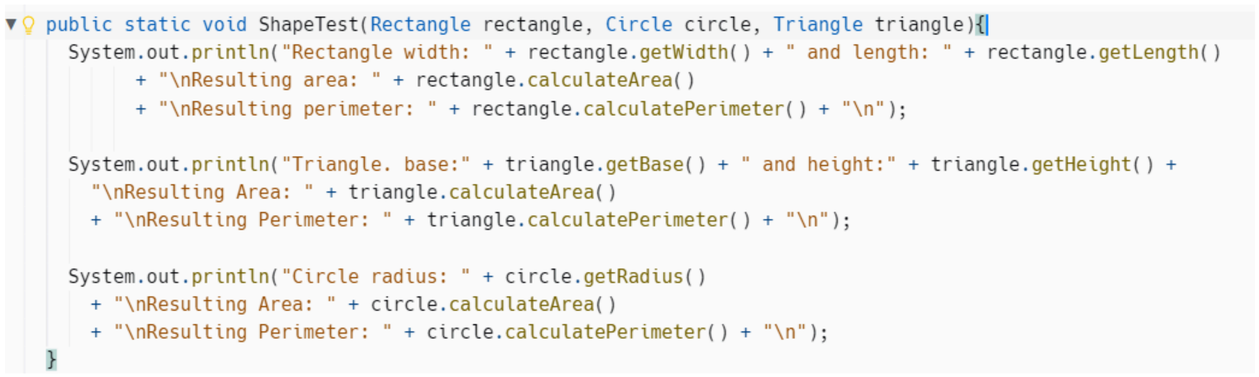
After:

```java
public static void ShapeTest(Shape[] shapes){
  for (Shape shape: shapes){
        System.out.println(shape.toString()
        + "\nResulting area: " + shape.calculateArea()
        + "\nResulting perimeter: " + shape.calculatePerimeter() + "\n");
  }
}
```

Unnecessary Specificity (Dependency Inversion Violation)

When a user is coupled to several low-level (child) classes but wishes to invoke some general behavior in them.

Example:



```java
public static void ShapeTest(Rectangle rectangle, Circle circle, Triangle triangle){
    System.out.println("Rectangle width: " + rectangle.getWidth() + " and length: " + rectangle.getLength()
            + "\nResulting area: " + rectangle.calculateArea()
            + "\nResulting perimeter: " + rectangle.calculatePerimeter() + "\n");

    System.out.println("Triangle. base:" + triangle.getBase() + " and height:" + triangle.getHeight() +
      "\nResulting Area: " + triangle.calculateArea()
      + "\nResulting Perimeter: " + triangle.calculatePerimeter() + "\n");

    System.out.println("Circle radius: " + circle.getRadius()
      + "\nResulting Area: " + circle.calculateArea()
      + "\nResulting Perimeter: " + circle.calculatePerimeter() + "\n");
}
```

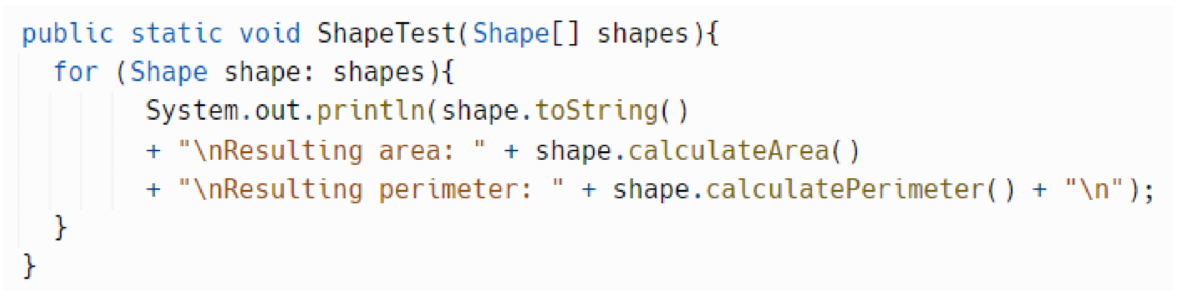
The ShapeTest method works with exactly 3 child shapes, rectangle, circle and triangle but does not invoke any behavior specific to the children.

For each child object the method prints 3 things:

1. Dimensions
2. Area
3. Perimeter

These are all behaviors that can be exhibited by any shape and thus should be invoked via the shape type.

ShapeTest doesn't care that the objects are rectangle\circle\triangle because it only invokes general Shape behavior.



```java
public static void ShapeTest(Shape[] shapes){
  for (Shape shape: shapes){
        System.out.println(shape.toString()
        + "\nResulting area: " + shape.calculateArea()
        + "\nResulting perimeter: " + shape.calculatePerimeter() + "\n");
  }
}
```

Deep Nesting (SRP Violation)

When a function is creating multiple nested scopes, there could be an opportunity to break it down into reusable functions.

"If you need more than 3 levels of indentation, you're screwed anyway, and should fix your program" -Linus Todrvalis