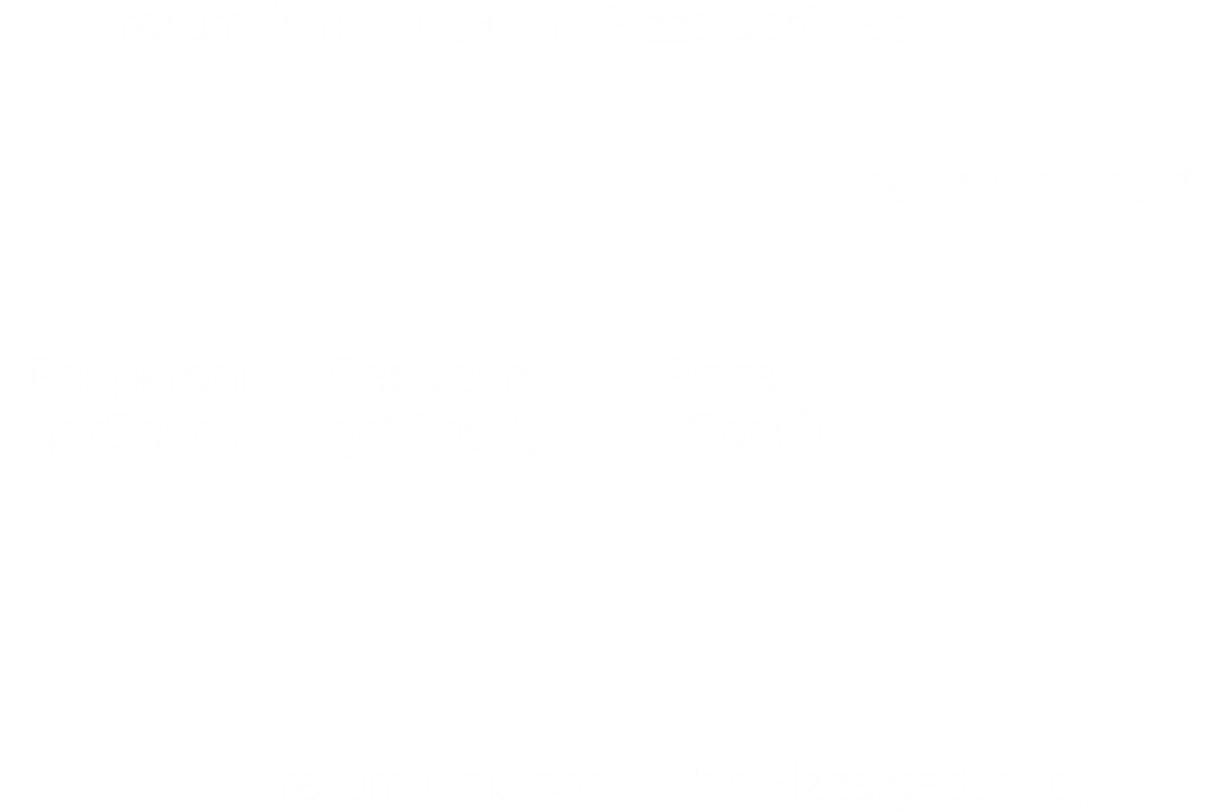
Decorator Pattern

The **Decorator Pattern** allows us to add new functionality to an existing object without changing its fundamental structure. This is a **structural pattern**, since the pattern adds wrappers to existing classes. A decorator wraps the original class and provides additional functionality while keeping the original class’s method signatures intact.



As opposed to the **Strategy Pattern**, which creates an object using various parts, the Decorator Pattern allows us to add new functionality to the object at runtime. The original object is not modified.

Suppose we have classes that draw various types of shapes.

public interface *Shape* {  
 void draw();  
}  
  
public class Rectangle implements *Shape* {  
 @Override  
 public void draw() {  
 // code  
 }  
}

public class Circle implements *Shape* {  
 @Override  
 public void draw() {  
 // code  
 }  
}

JAVA

We now want to add some color to the shapes. The basic drawing of the shape should not be changed. Some color should just be added to the basic drawing. Thus, we can use a decorator.

public abstract class ShapeDecorator implements *Shape* {  
 protected *Shape* decoratedShape;  
 public ShapeDecorator(*Shape* decoratedShape) {  
 this.decoratedShape = decoratedShape;  
 }  
  
 @Override  
 public void draw() {  
 // the original class's behaviour is maintained  
 decoratedShape.draw();  
 }  
}  
  
public class RedShapeDecorator extends ShapeDecorator {  
 public RedShapeDecorator(*Shape* decoratedShape) {  
 super(decoratedShape);  
 }  
 private void fillWithRed(*Shape* decoratedShape) {  
 // code  
 }  
  
 @Override  
 public void draw() {  
 decoratedShape.draw(); // original behaviour  
 fillWithRed(decoratedShape); // additional behaviour  
 }  
}

JAVA

This allows us to add decorators to the original shapes and still use the resulting objects as if they were the original shapes.

public class DecoratorPatternDemo {  
 public static void main(String[] args) {  
 *Shape* circle = new Circle();  
 *Shape* redCircle = new RedShapeDecorator(circle);  
 circle.draw(); // plain circle  
 redCircle.draw(); // circle, but red  
 }  
}

JAVA