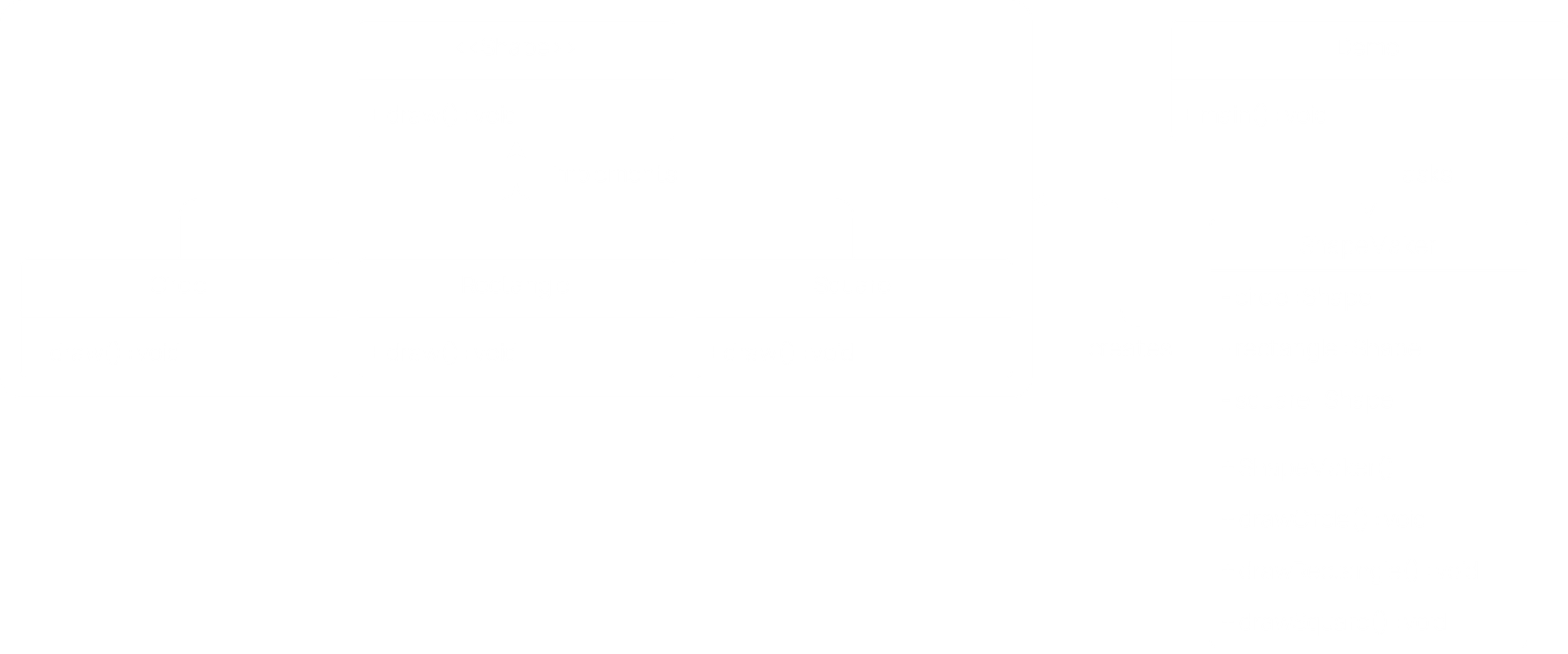
Façade Pattern

The **façade pattern** hides the complexities of the system and provides the client with an interface they can use to access the system. It is a **structural pattern** that creates a single class with simplified methods the client can use which delegates calls to methods of existing system classes.

public interface *Shape* {  
 void draw();  
}  
  
public class Rectangle implements *Shape* {  
 @Override  
 public void draw() {  
 System.*out*.println("Rectangle::draw()");  
 }  
}  
  
public class Square implements *Shape* {  
 @Override  
 public void draw() {  
 System.*out*.println("Square::draw()");  
 }  
}  
  
public class Circle implements *Shape* {  
 @Override  
 public void draw() {  
 System.*out*.println("Circle::draw()");  
 }  
}

public class ShapeMaker {  
 private *Shape* circle;  
 private *Shape* rectangle;  
 private *Shape* square;  
  
 public ShapeMaker() {  
 circle = new Circle();  
 rectangle = new Rectangle();  
 square = new Square();  
 }  
  
 public void drawCircle(){  
 circle.draw();  
 }  
 public void drawRectangle(){  
 rectangle.draw();  
 }  
 public void drawSquare(){  
 square.draw();  
 }  
}  
  
public class Demo {  
 public static void main(String[] args) {  
 ShapeMaker shapeMaker = new ShapeMaker();  
  
 shapeMaker.drawCircle();  
 shapeMaker.drawRectangle();  
 shapeMaker.drawSquare();  
 }  
}

JAVA



The **façade pattern** is quite similar to the **abstract factory** pattern. However, the abstract factory pattern deals specifically with object creation (making it a creational pattern) whereas the façade pattern can be used for anything, not just creation (thus making it a structural pattern).