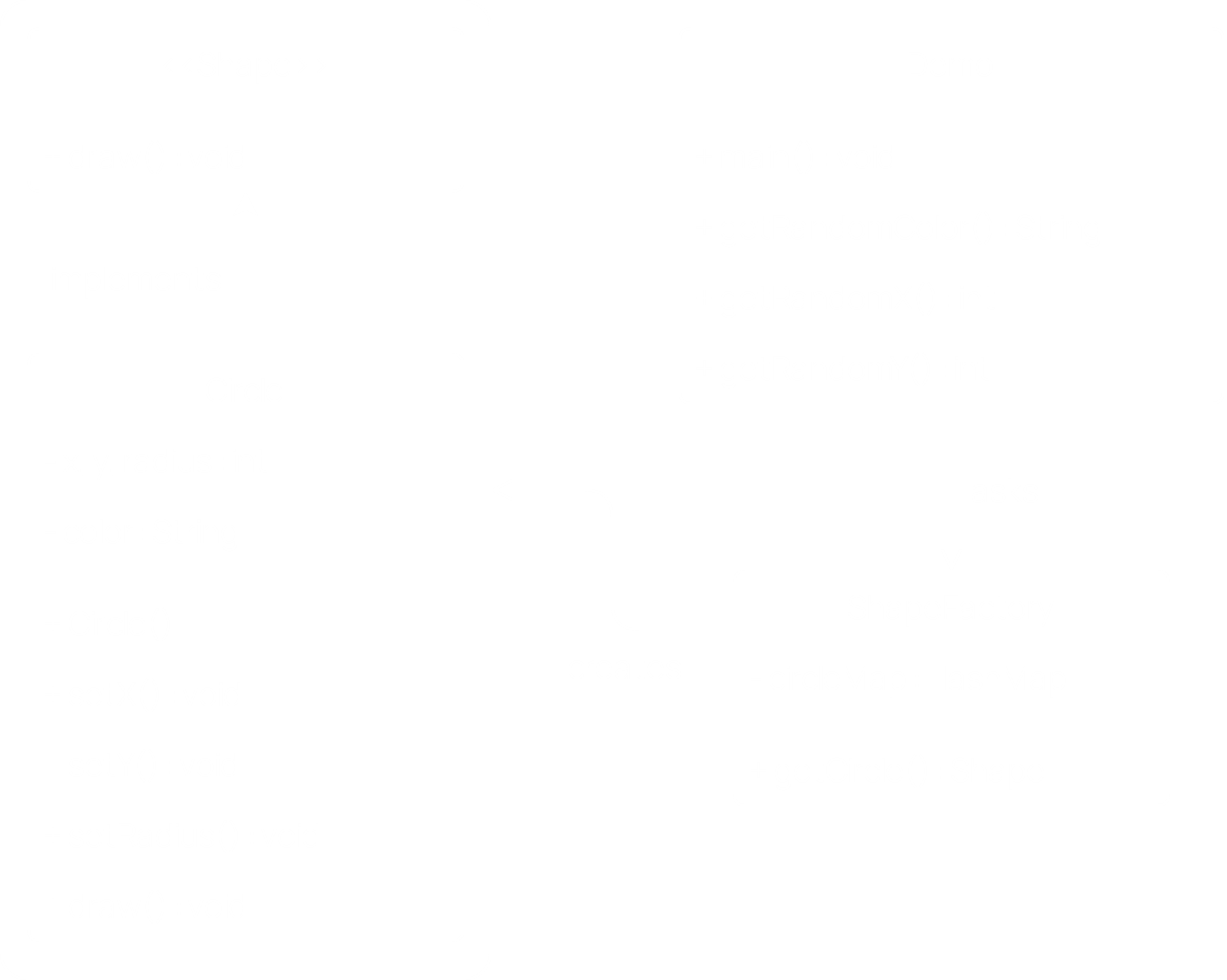
Flyweight Pattern

The **flyweight pattern** is used to reduce the number of objects created, thus decreasing the memory footprint and increasing performance. It is a **structural pattern** that stores created objects and tries to reuse them. New objects are only created if existing objects cannot be found.



public interface *Shape* {  
 void draw();  
}  
  
public class Circle implements *Shape* {  
 private String color;  
 public Circle(String color) {  
 this.color = color;  
 }  
   
 @Override  
 public void draw() {  
 System.*out*.println("Circle of color: " + color);  
 }  
}

public class ShapeFactory {  
  
 private static final HashMap *circleMap* = new HashMap();  
  
 public static *Shape* getCircle(String color) {  
 Circle circle = (Circle) *circleMap*.get(color);  
  
 if(circle == null) {  
 circle = new Circle(color);  
 *circleMap*.put(color, circle);  
 System.*out*.println("Creating circle of color : " + color);  
 }  
 return circle;  
 }  
}  
  
public class Demo {  
 private static final String *colors*[] = {"Red", "Green", "Blue"};  
   
 public static void main(String[] args) {  
  
 for(int i = 0; i < 20; ++i) {  
 String color = *getRandomColor*();  
 Circle circle = (Circle) ShapeFactory.*getCircle*(color);  
 circle.draw();  
 }  
 }  
   
 private static String getRandomColor() {  
 return *colors*[(int) (Math.*random*() \* *colors*.length)];  
 }  
}

JAVA