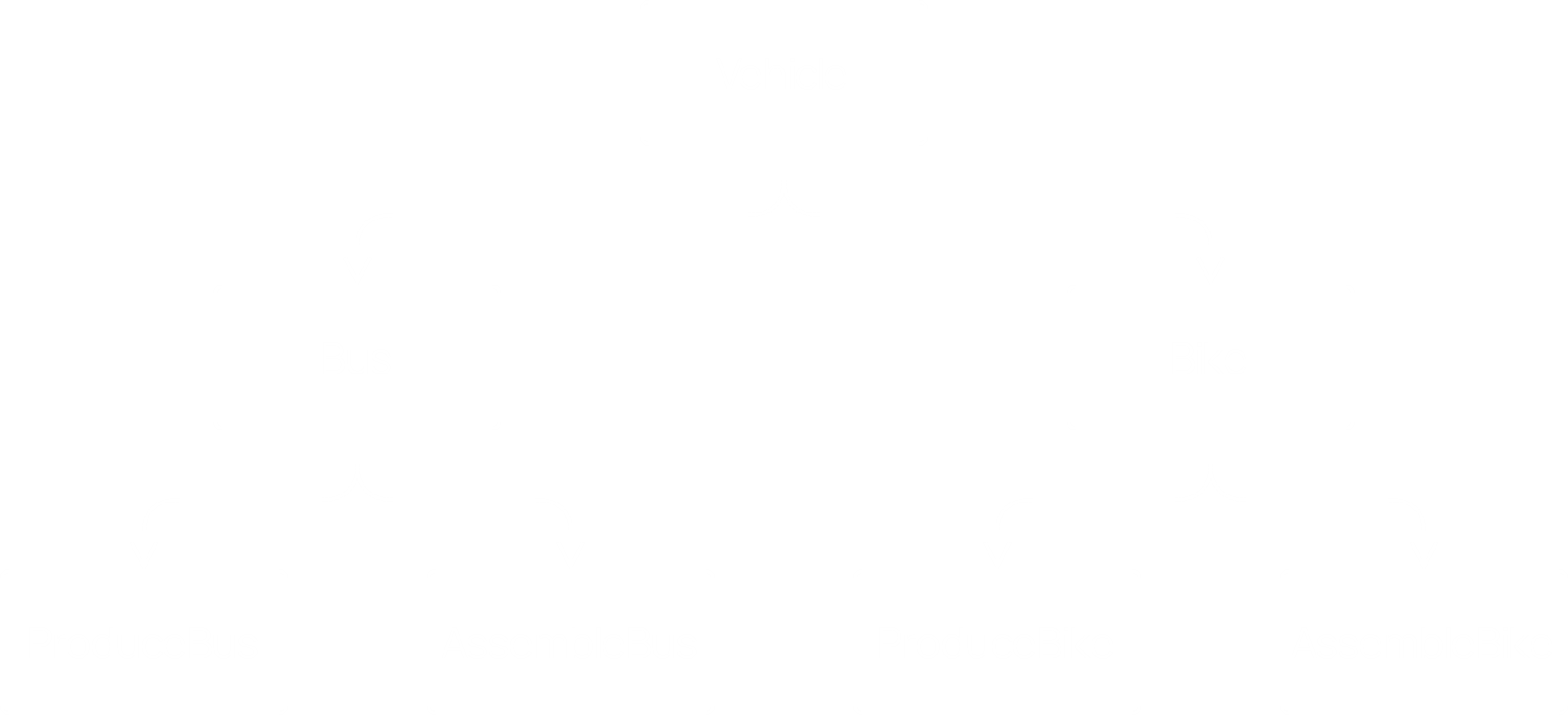
Bridge Pattern

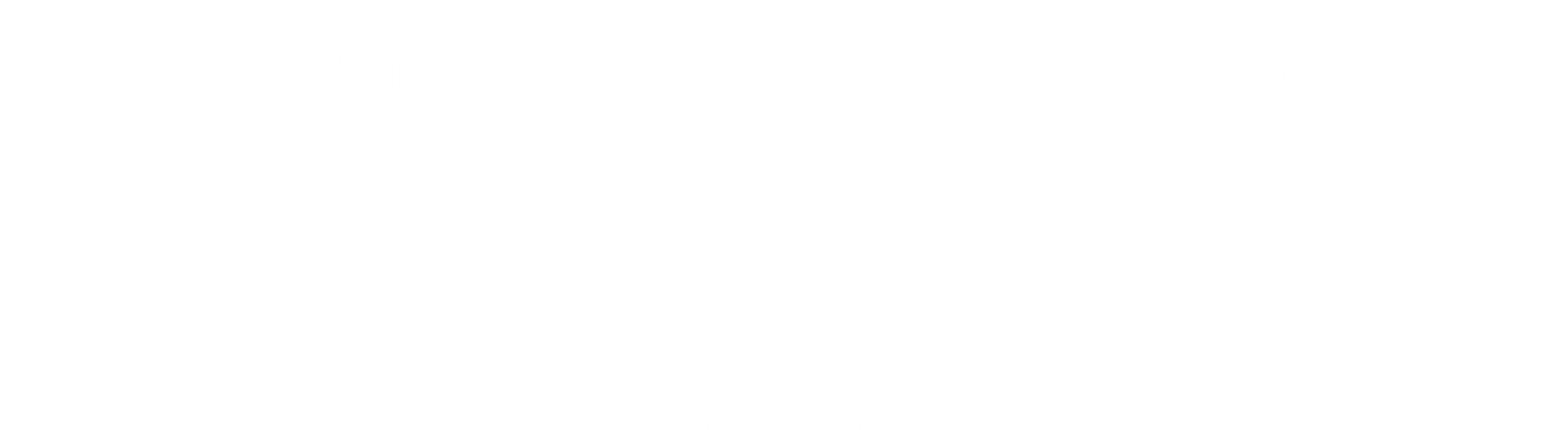
The **bridge pattern** is a **structural pattern** that lets us split a large class or a set of closely related classes into two hierarchies: abstraction and implementation. The two hierarchies can then be developed independently. The pattern involves an interface which acts as a **bridge**. The bridge makes the functionality of the concrete classes independent from interface implementer classes.

The following diagrams show a few examples to make the setup clear:

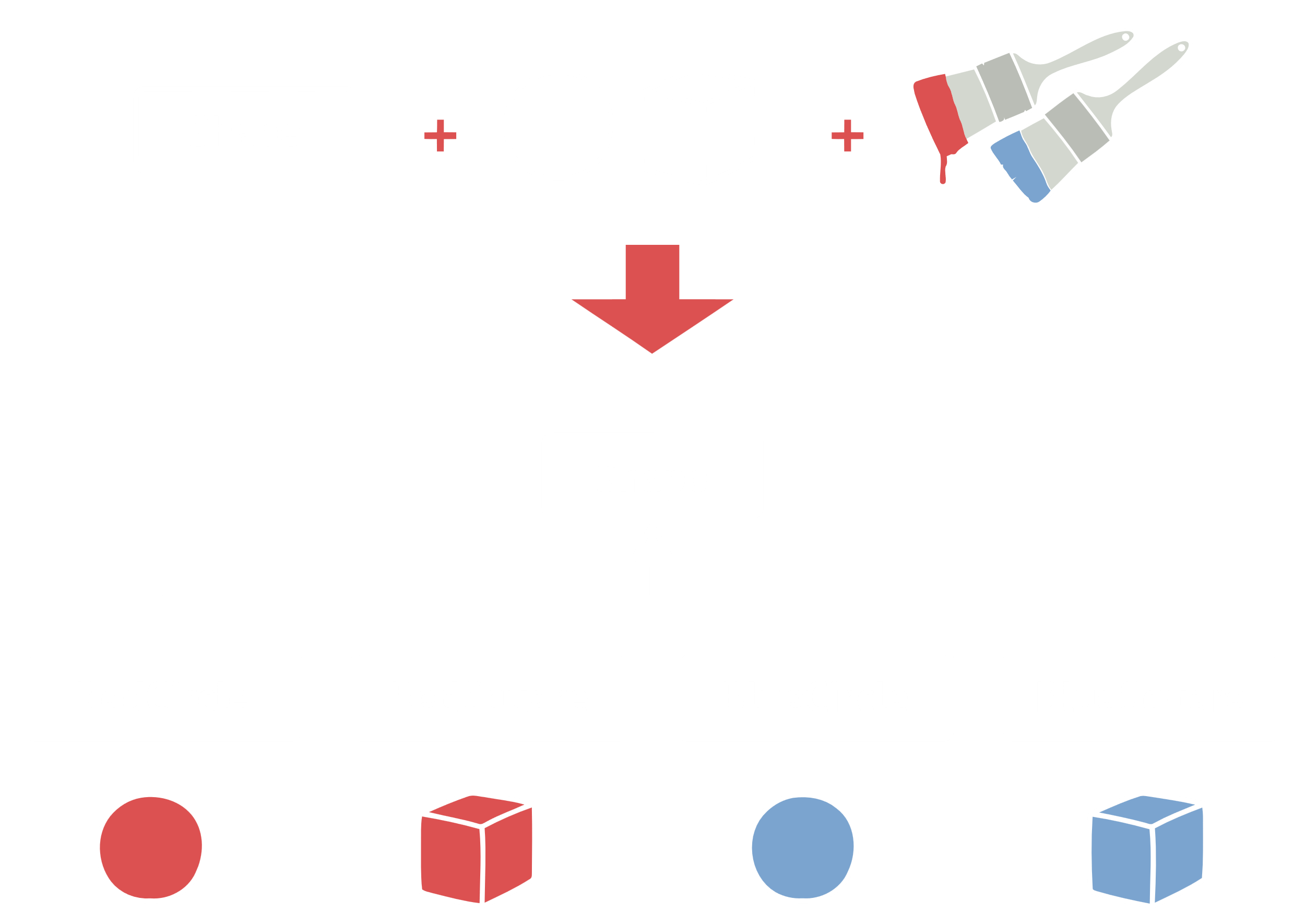
Without Bridge:



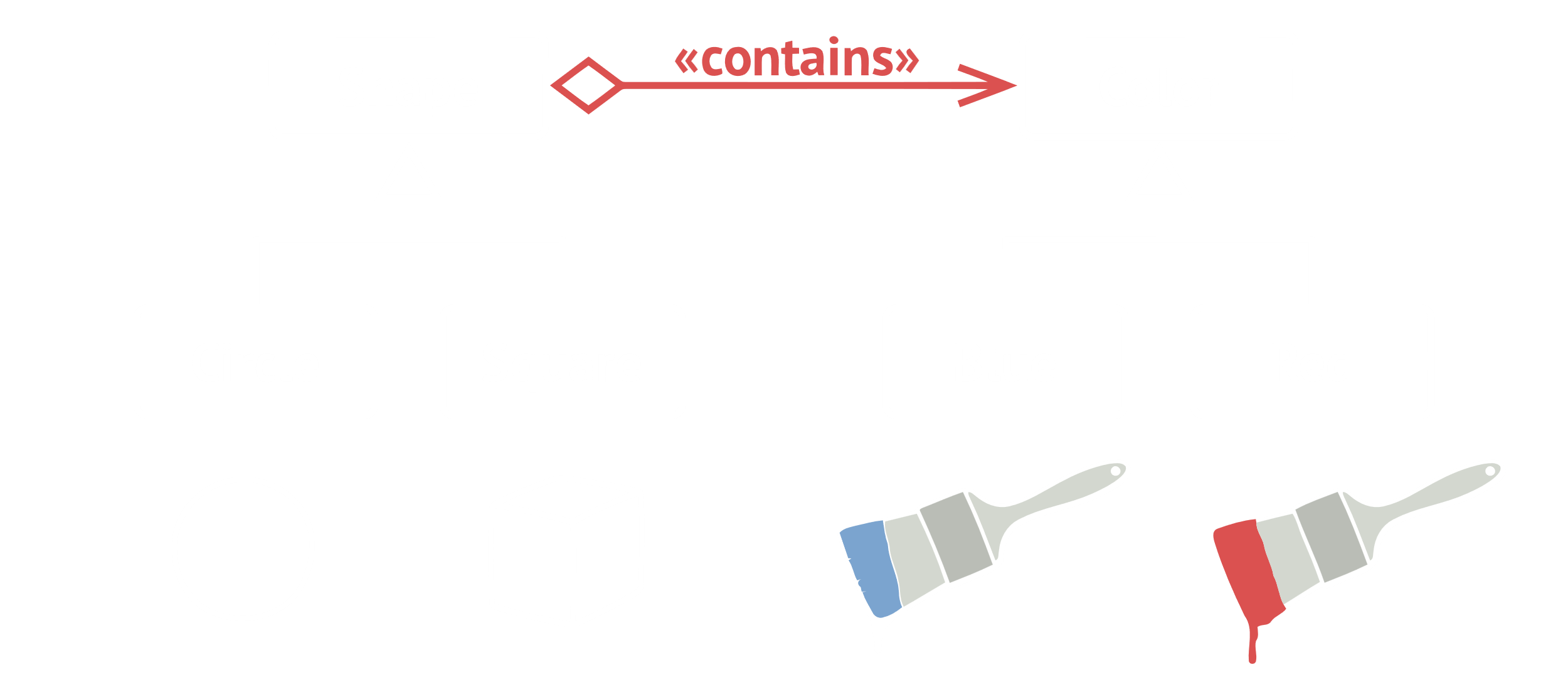
With Bridge:



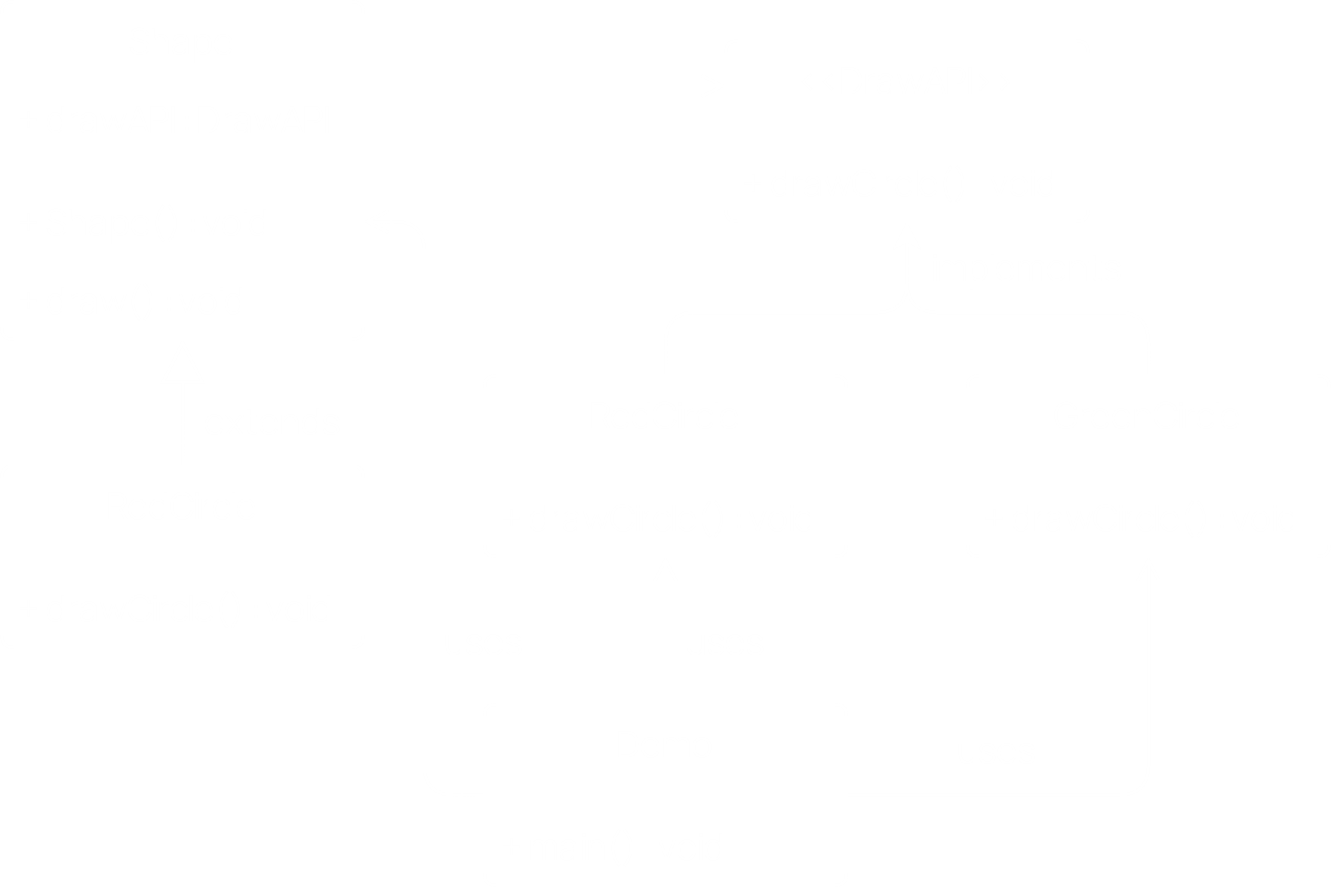
Without Bridge:



With Bridge:



UML Diagram:



Implementation:

abstract class Vehicle {  
 protected *Workshop* workShop1;  
 protected *Workshop* workShop2;  
  
 protected Vehicle(*Workshop* workShop1, *Workshop* workShop2) {  
 this.workShop1 = workShop1;  
 this.workShop2 = workShop2;  
 }  
  
 abstract public void manufacture();  
}  
  
class Car extends Vehicle {  
 public Car(*Workshop* workShop1, *Workshop* workShop2) {  
 super(workShop1, workShop2);  
 }  
  
 @Override  
 public void manufacture() {  
 System.*out*.print("Car ");  
 workShop1.work();  
 workShop2.work();  
 }  
}  
  
class Bike extends Vehicle {  
 public Bike(*Workshop* workShop1, *Workshop* workShop2) {  
 super(workShop1, workShop2);  
 }  
  
 @Override  
 public void manufacture() {  
 System.*out*.print("Bike ");  
 workShop1.work();  
 workShop2.work();  
 }  
}  
  
interface *Workshop* {  
 abstract public void work();  
}

class Produce implements *Workshop* {  
 @Override  
 public void work() {  
 System.*out*.print("Produced");  
 }  
}  
  
class Assemble implements *Workshop* {  
 @Override  
 public void work() {  
 System.*out*.print(" And");  
 System.*out*.println(" Assembled.");  
 }  
}  
  
class Demo {  
 public static void main(String[] args) {  
 Vehicle vehicle1 = new Car(new Produce(), new Assemble());  
 vehicle1.manufacture();  
 Vehicle vehicle2 = new Bike(new Produce(), new Assemble());  
 vehicle2.manufacture();  
 }  
}

JAVA