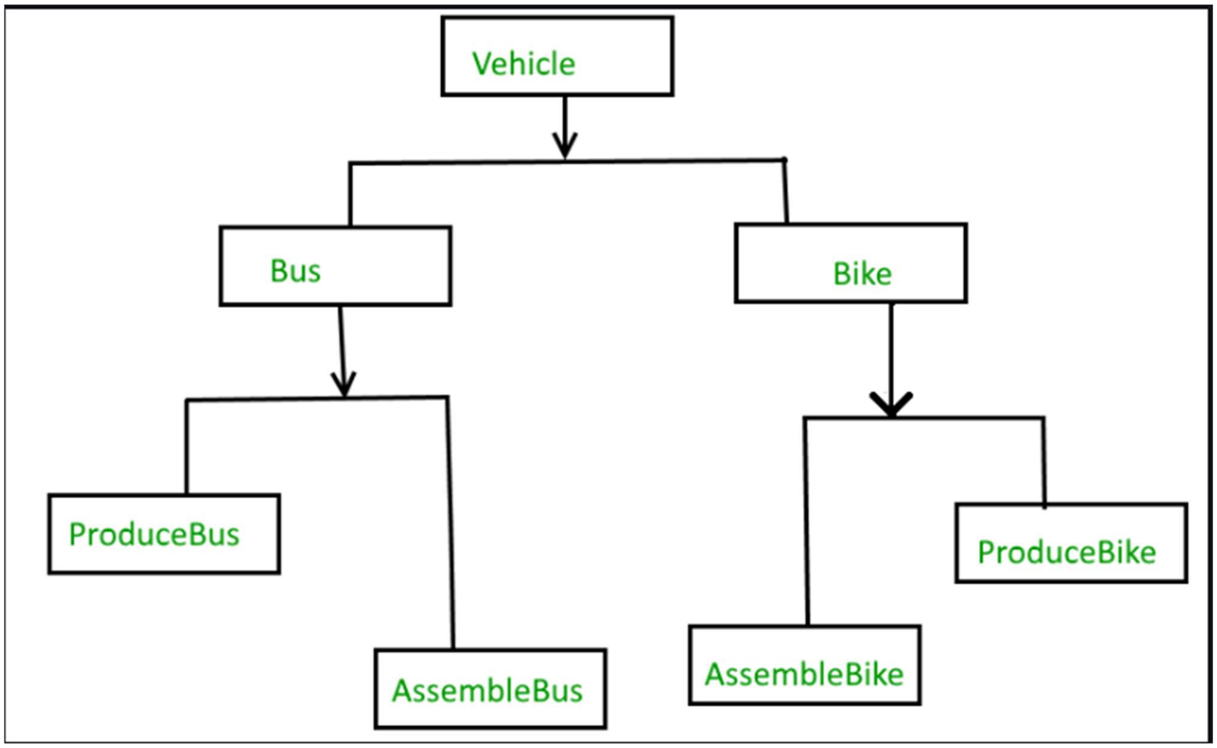
Bridge Design Pattern

Introduction

Bridge is a structural design pattern that is used to separate abstraction from its implementation.

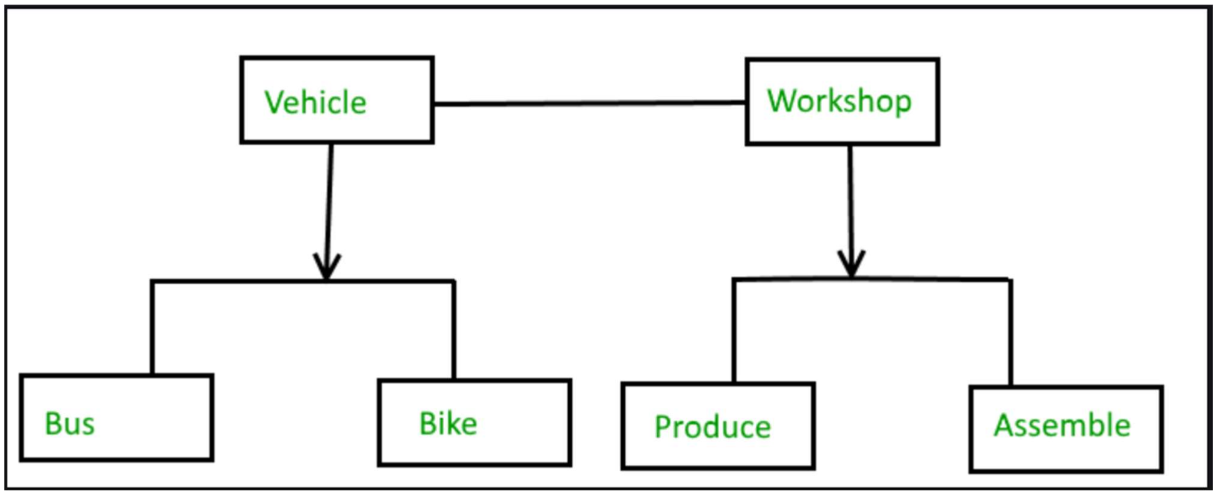
Selected implementation of the design pattern: geeksforgeeks.org/system-design/bridge-design-pattern/

In the selected implementation there was a Vehicle abstract class that was responsible for vehicle abstraction, and the vehicle sub-classes were responsible for their concrete implementations, as seen in picture 1. below. This results in class explosion and is not very scalable.



Picture 1. Before Bridge design pattern.

In the bridge implementation, Vehicle abstraction and implementations were separated as seen in picture 2 below.

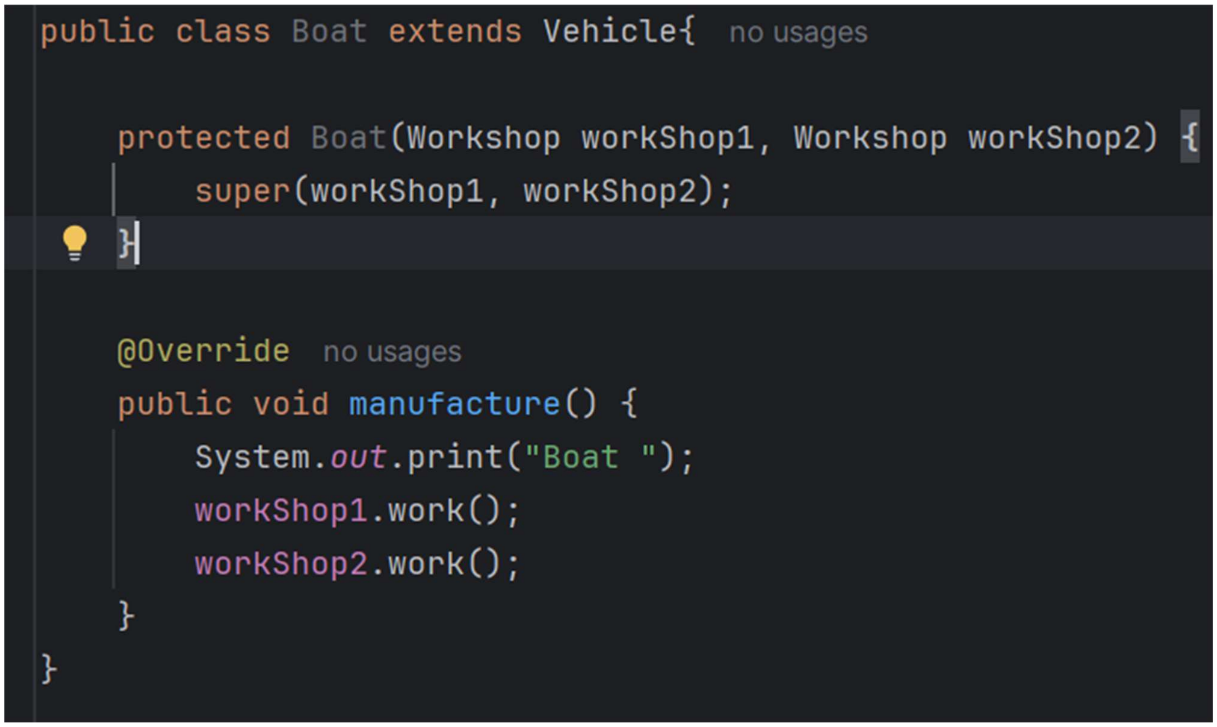


Picture 2. After Bridge design pattern.

As seen in picture 2. Workshop interface is now responsible for the implementations.

New Functionality

Decided to add new Vehicle sub-class Boat, to showcase that there is no need to modify the workshop implementations to start producing and assembling boats.

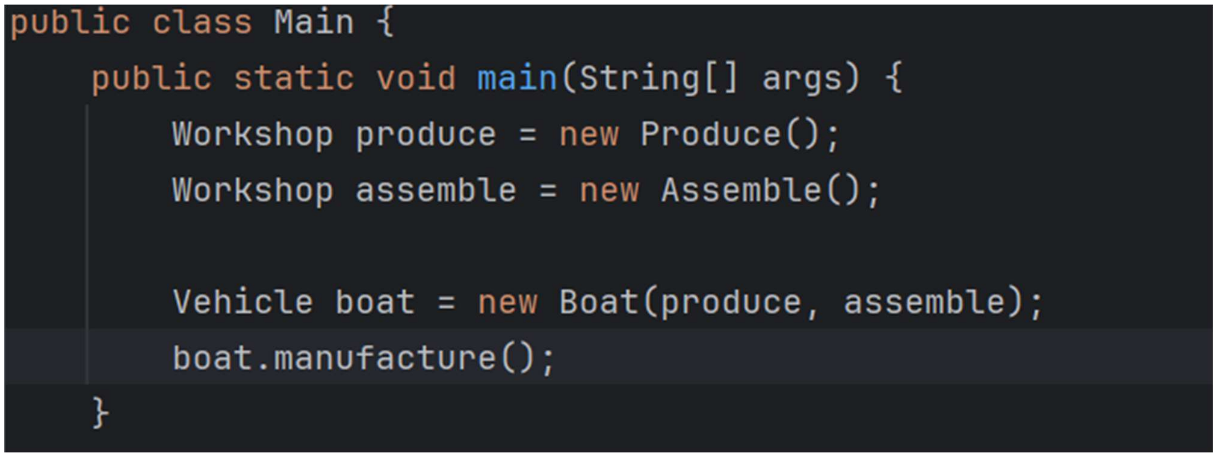


```
public class Boat extends Vehicle{  no usages

    protected Boat(Workshop workShop1, Workshop workShop2) {
        super(workShop1, workShop2);
    }

    @Override  no usages
    public void manufacture() {
        System.out.print("Boat ");
        workShop1.work();
        workShop2.work();
    }
}
```

Picture 3. New Boat sub-class

As seen in picture 3. above we only need to write the Boat class with refined manufacture method for boats and call the Produce and Assemble workshops method work.

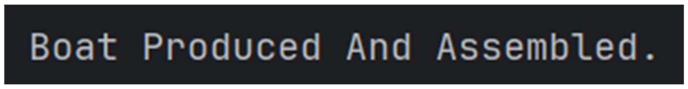
New Vehicle was tested in Main as seen in picture 4. below.



```
public class Main {
    public static void main(String[] args) {
        Workshop produce = new Produce();
        Workshop assemble = new Assemble();

        Vehicle boat = new Boat(produce, assemble);
        boat.manufacture();
    }
```

Picture 4. Testing manufacturing new Boat.

Result of running Main was successful new Boat.



```
Boat Produced And Assembled.
```

Conclusion

In conclusion, bridge is a solid design pattern, it reduces class explosion and improves scalability.

Cons in my case was that all of the Vehicles share the same workshops, adding a new implementation (Workshop) still would require to update all of the existing vehicle constructors.

References

Pictures 1. and 2. : https://www.geeksforgeeks.org/system-design/bridge-design-pattern/