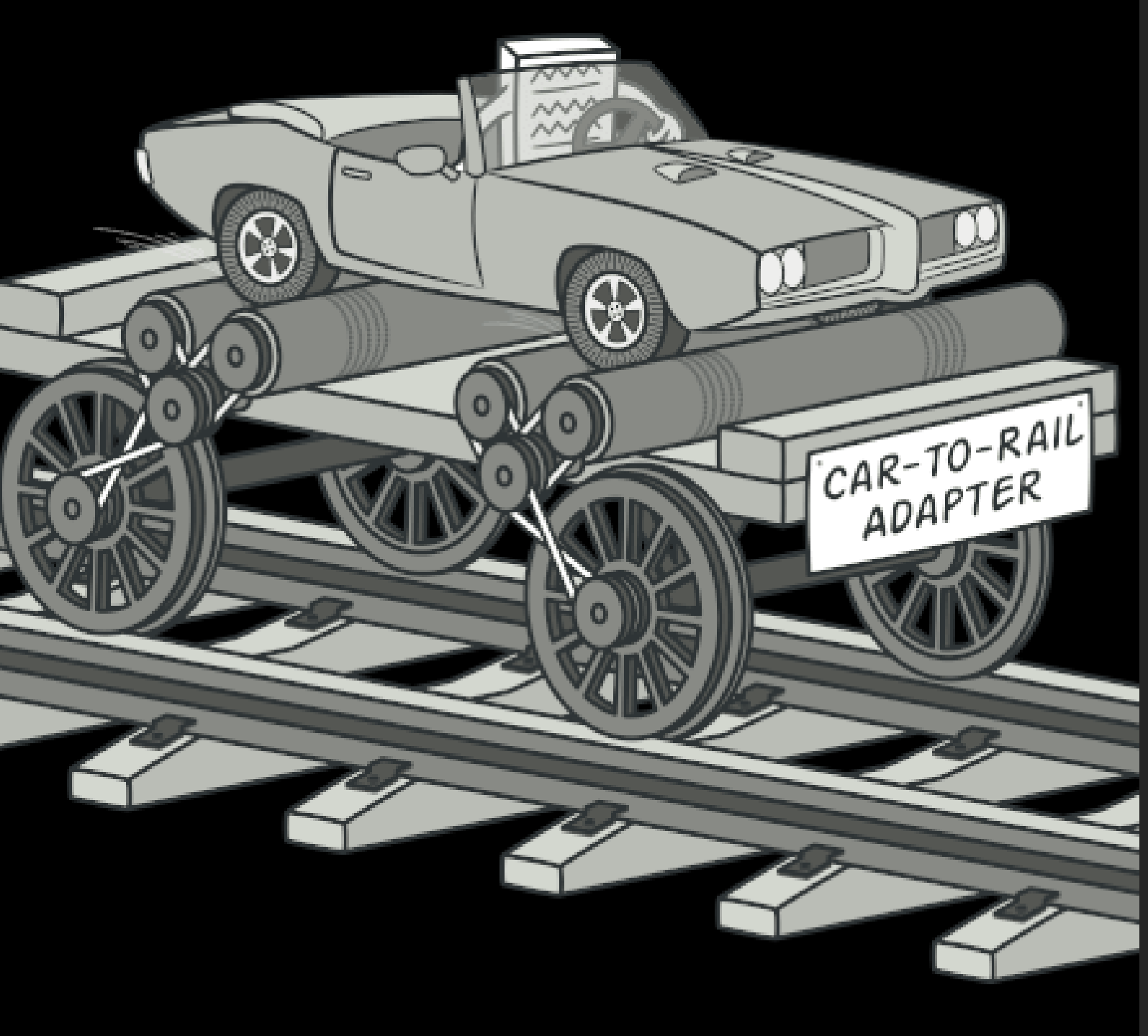



# ADAPTER DESIGN PATTERN

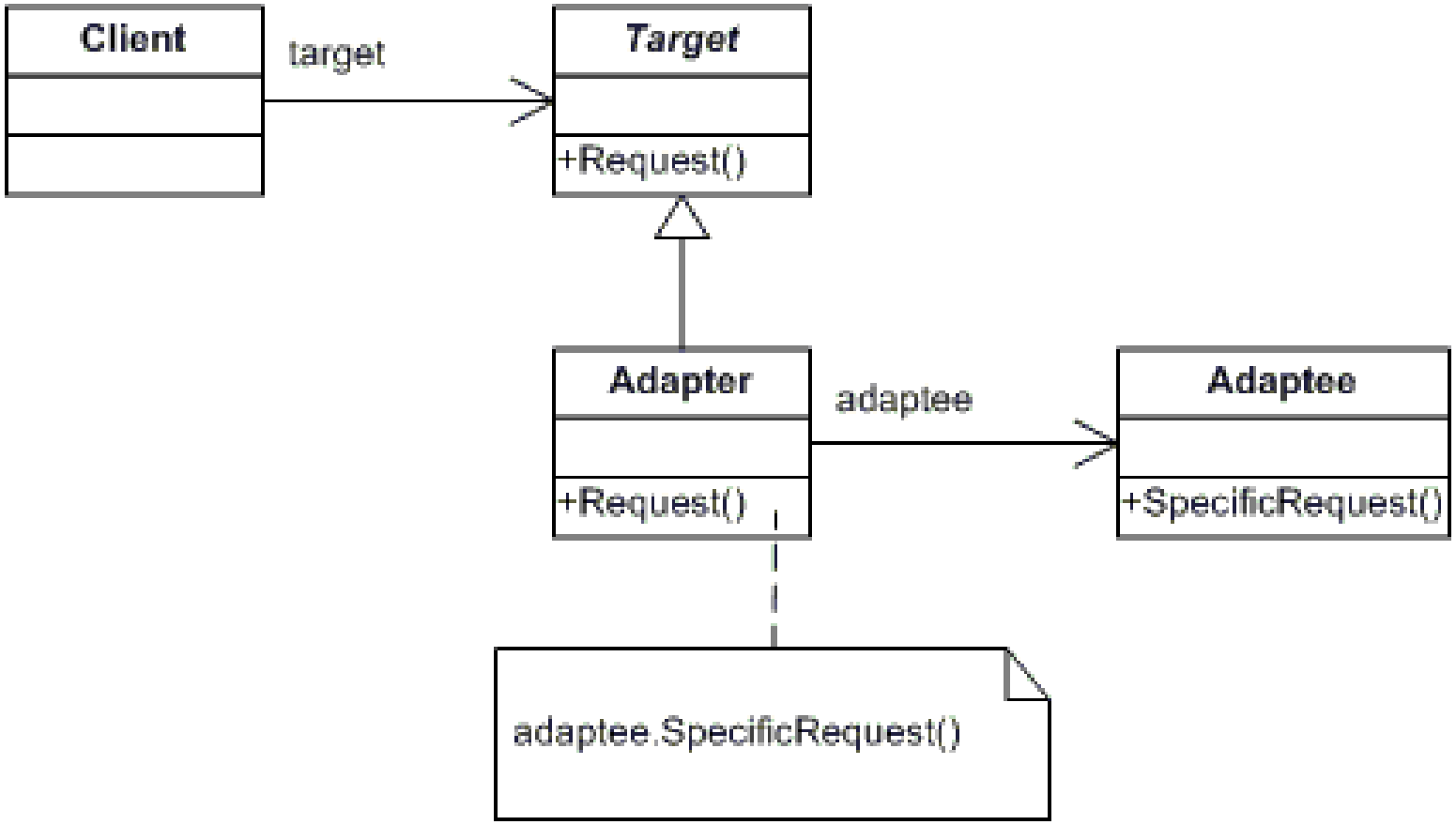
STRUCTURE PATTERN

# WHAT IS ADAPTER DESIGN PATTERN

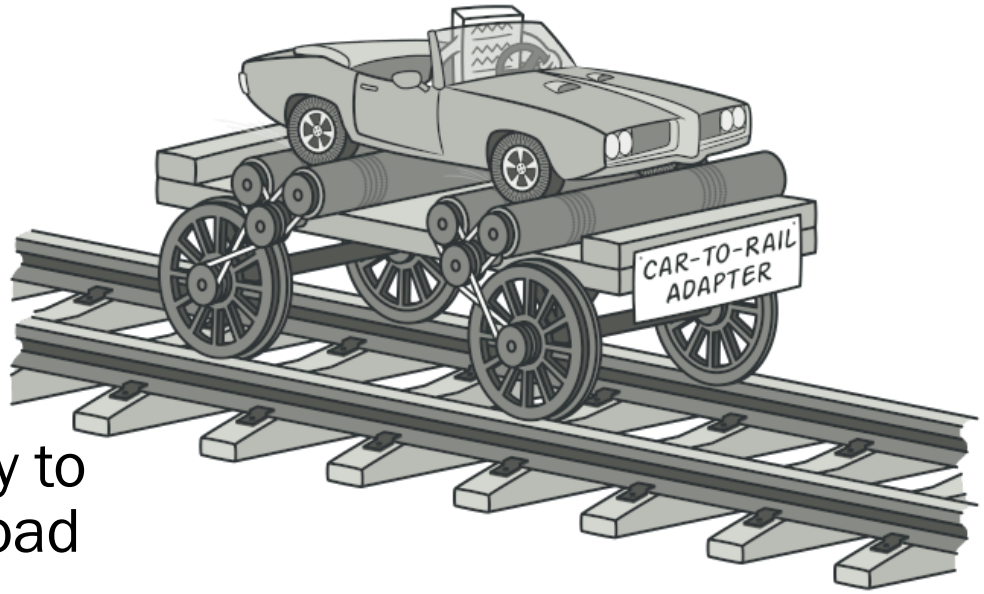
- The adapter pattern is a structure pattern that convert the called method to expect result by using
  - Client class ( by calling the method)
  - Adapter class ( by turning Client class to result class)
  - Result class ( the expect out put of the application)

- Advantage/ disadvantage of Adapter pattern
  - Advantage
    - Reduce the complex of code relation if using
    - If using this pattern with factory it will also reduce the complex on concreate factory
    - Easy to change rater and easy to add more adapter
  - Disadvantage
    - You cannot directly use adapter
    - It reduce complex of code but increase the complex of maintenance

# THE CLASS DIAGRAM OF THE ADAPTER PATTERN



# THE SIMPLIFY OF ADAPTER PATTERN (EXAMPLE 1)




The Normal car that try to travel though the railroad

The car that can travel though the railroad

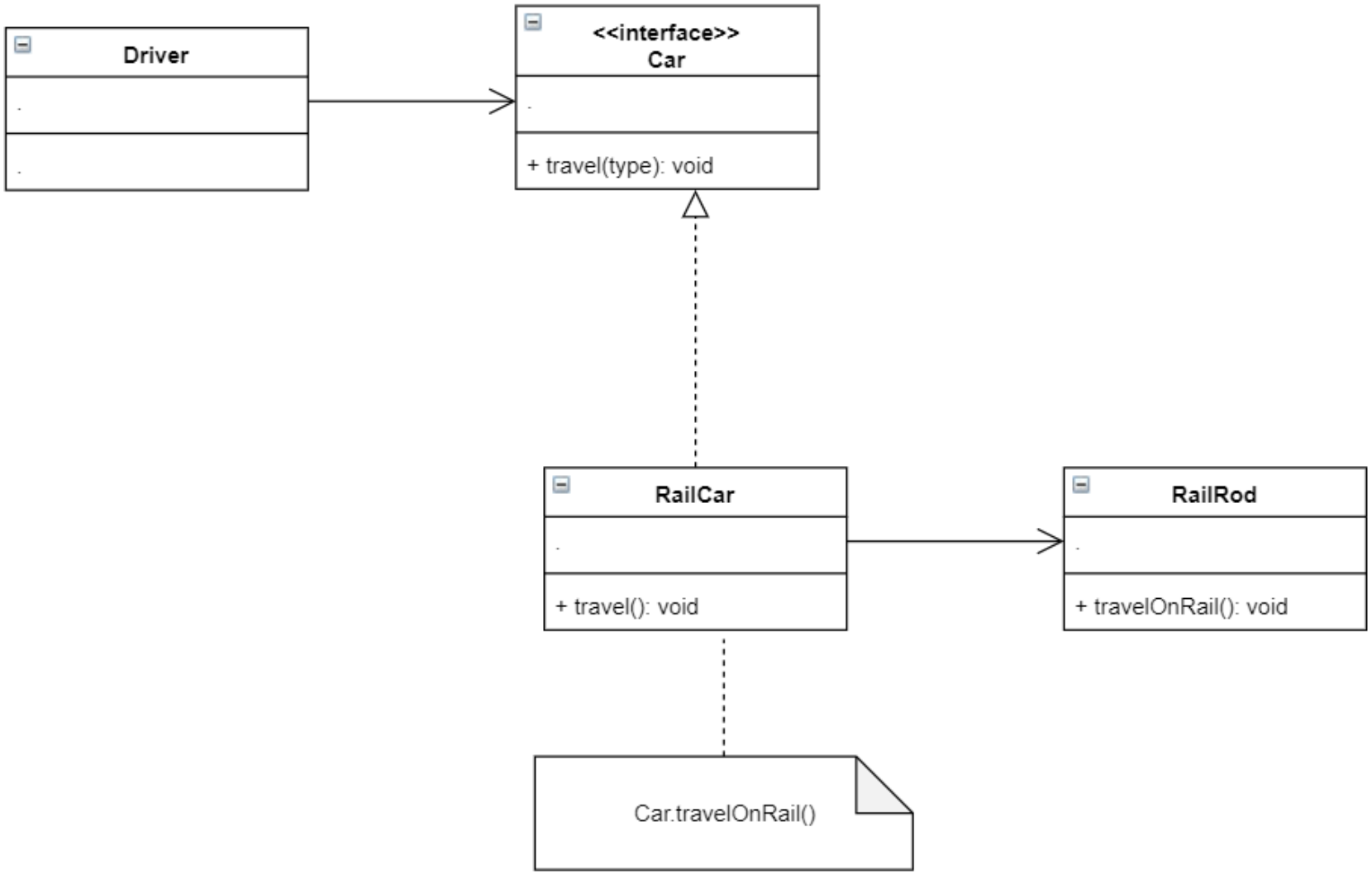
Client

Adapter

Result

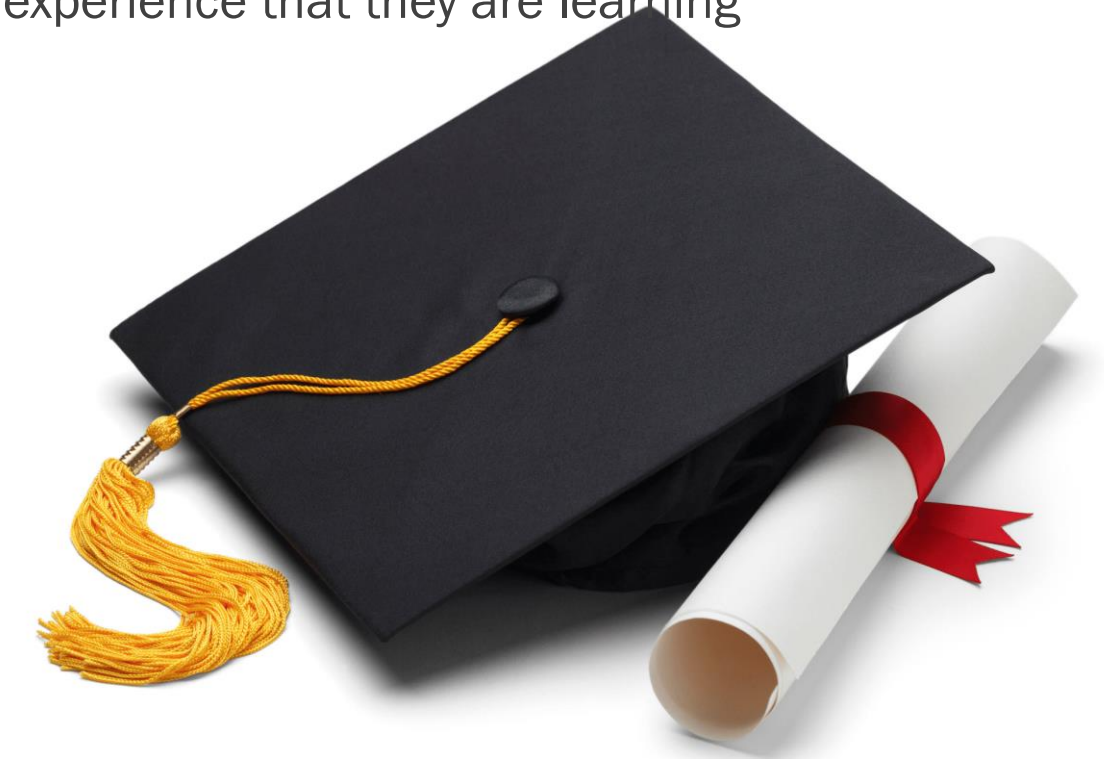
The rail car that can plug the whole car

# FROM EXAMPLE 1 TO CLASS DIAGRAM

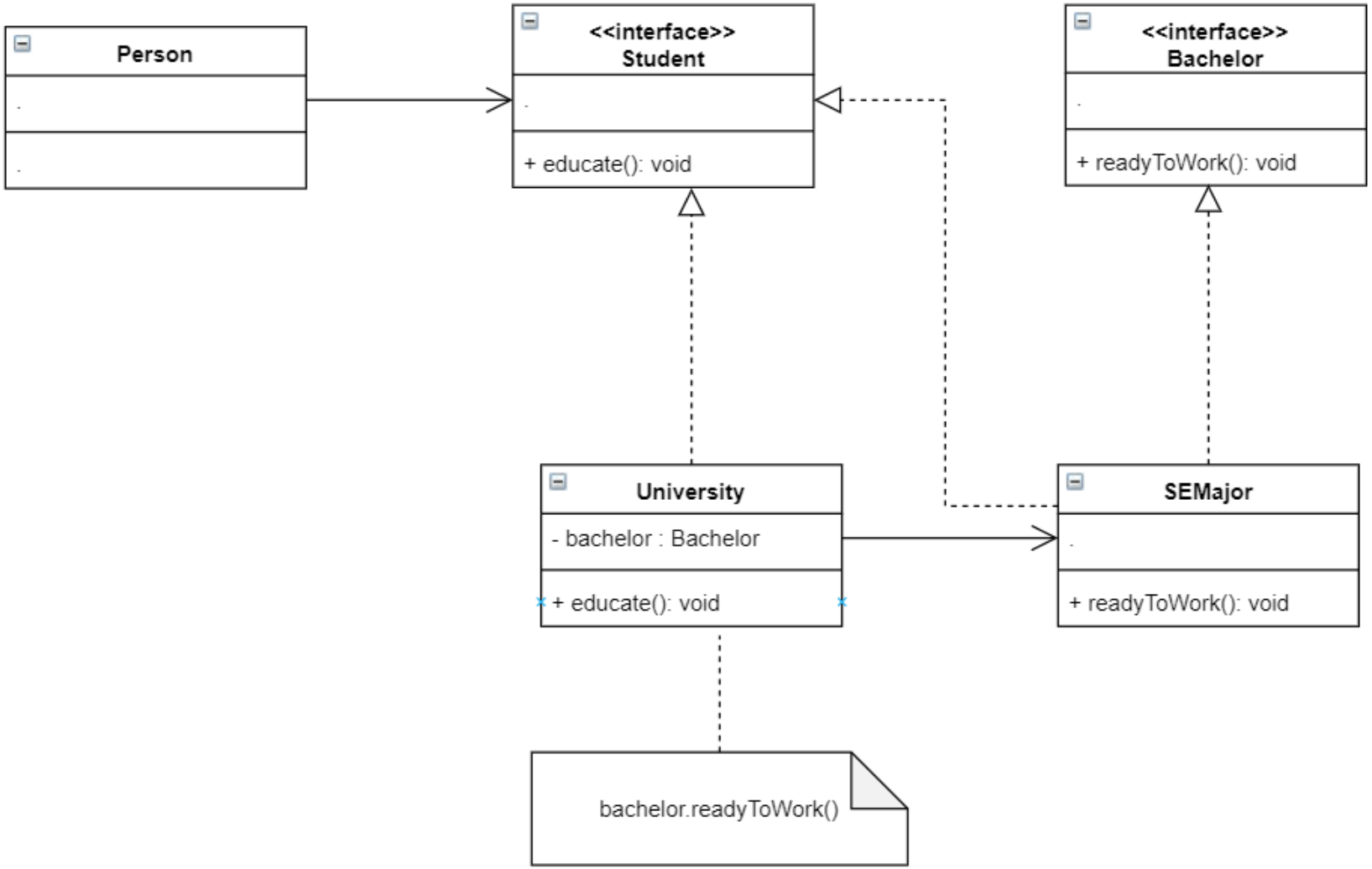


# EXAMPLE 2 STUDENT IN WORK SITUATION

- The capitalist world requires a degree to work. The degree must be at least a bachelor's degree, and the bachelor's degree comes from the university. The student that performs self-learning will not have a chance to get the degree but gain the experience that they are learning

# THE CLASS DIAGRAM OF EXAMPLE 2



# THE MODEL

## University

```java
public class University implements Student {

    private Bachelor bachelor;

    public University(Bachelor bachelor){
        this.bachelor = bachelor;
    }

    @Override
    public void educate() {
        System.out.println("We make the Great Bachelor");
        this.bachelor.readyToWork();
    }
}
```

## SE major

```java
public class SEmajor implements Bachelor,Student {
    @Override
    public void readyToWork() {
        System.out.println("I am a Software Architecture and Ready to work");
    }

    @Override
    public void educate(){
        System.out.println("I am a software engineering student that study for future");
    }

}
```

# THE SERVICE (INTERFACE)

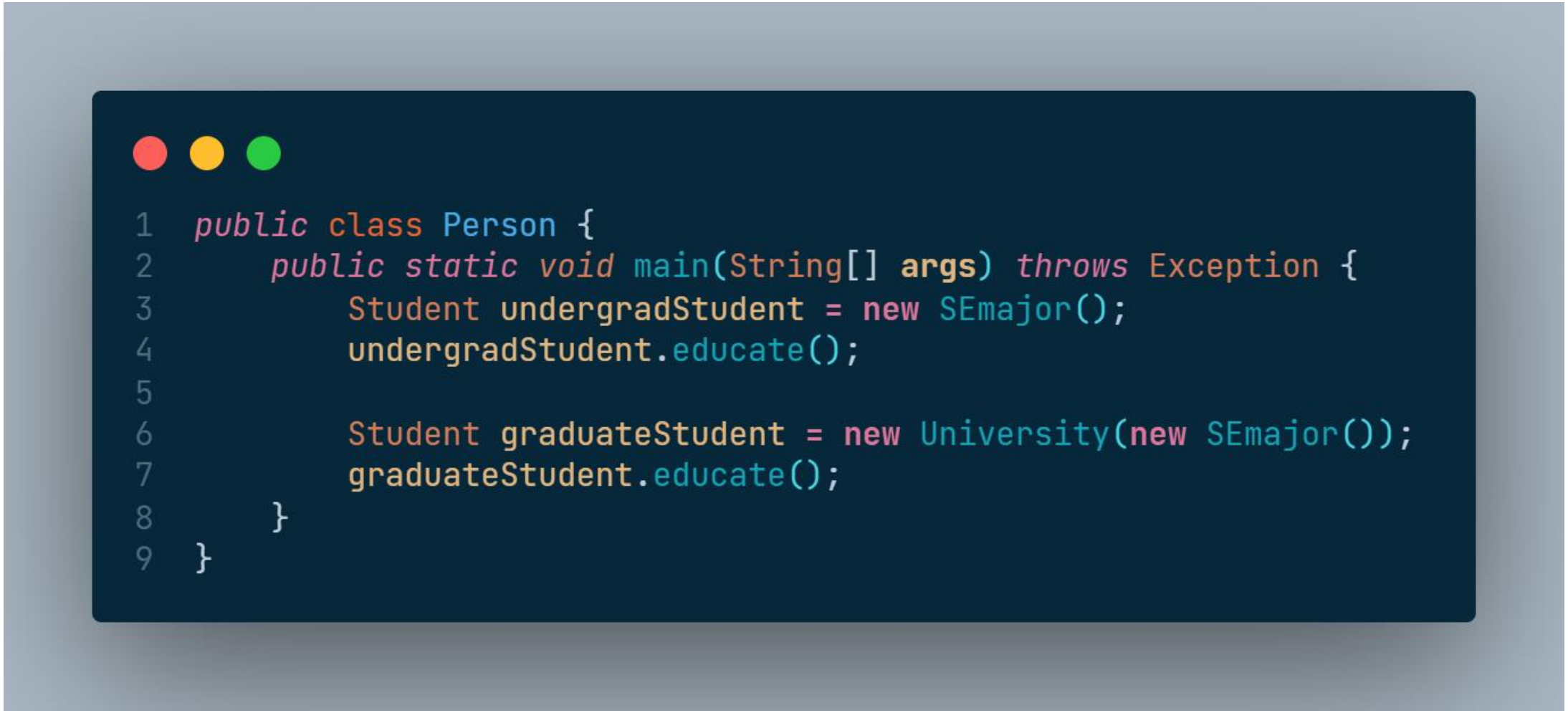
Student

```java
public interface Student {
    void educate();
}
```

Bachelor

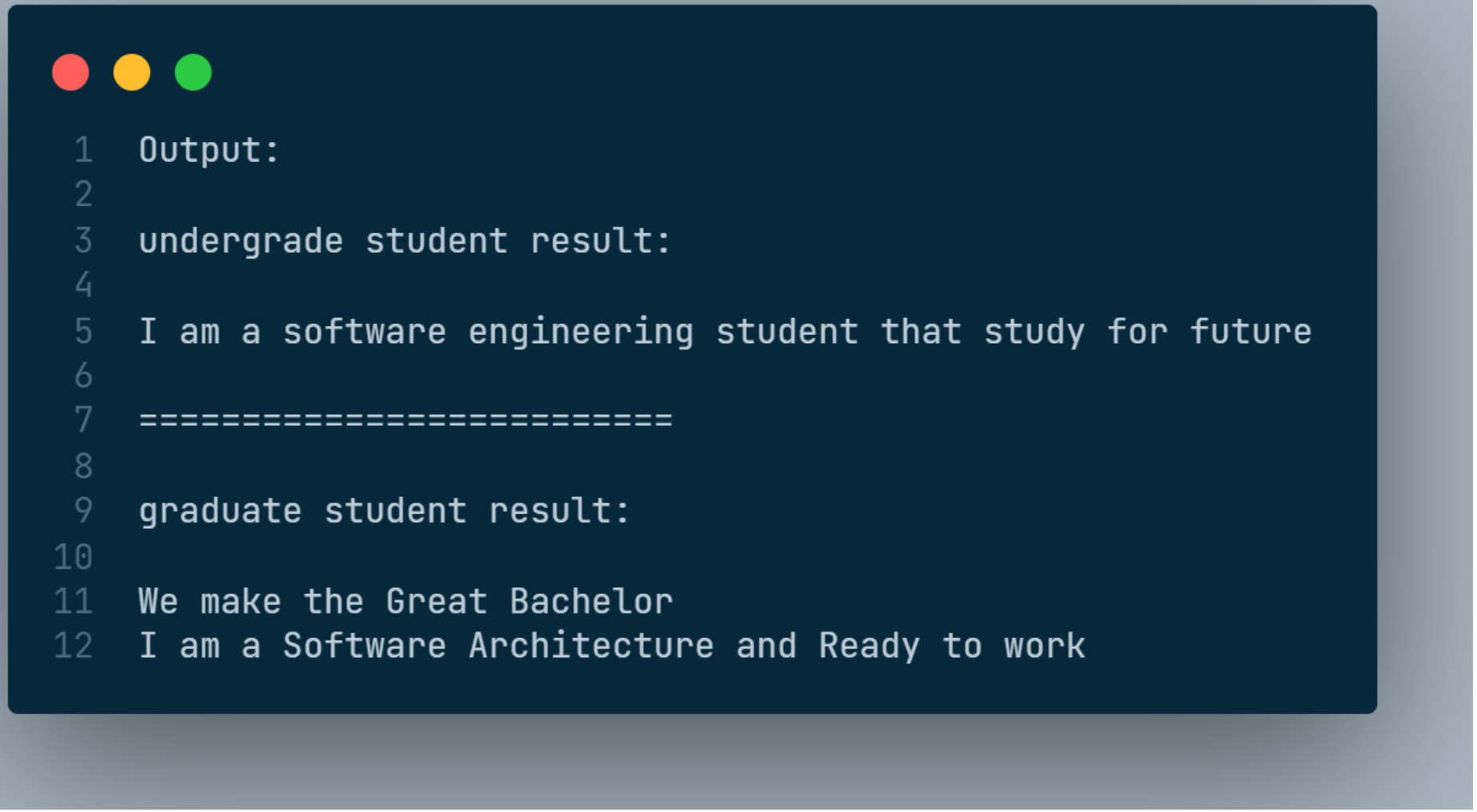
```java
public interface Bachelor {
    void readyToWork();
}
```

# THE CLIENT



```
1  public class Person {
2      public static void main(String[] args) throws Exception {
3          Student undergradStudent = new SEmajor();
4          undergradStudent.educate();
5
6          Student graduateStudent = new University(new SEmajor());
7          graduateStudent.educate();
8      }
9  }
```

# THE RESULT



```
Output:

undergrade student result:

I am a software engineering student that study for future

==========================

graduate student result:

We make the Great Bachelor
I am a Software Architecture and Ready to work
```

# THANK YOU!!!

HERE MY REFERENCE

- Adapter Pattern – GeeksforGeeks
-  Adapter Pattern - Saladpuk.com
- Adapter pattern – Wikipedia

- My code
  -HomeworkCollection/SoftDes322/Adapter at main · oat431/HomeworkCollection (github.com)