Foundation of Software Engineering Lab Report

Your Student Name: Wang Yilin

Your Student Number: 3170103095

Accepted format: PDF

Section 1: Class Relationships

1. Brief Introduction of adapter pattern:

   Typically, a client can access the services it provides through the target class's interface. Sometimes, the existing class can meet the functional needs of the customer class, but the interface it provides is not necessarily what the customer class expects.

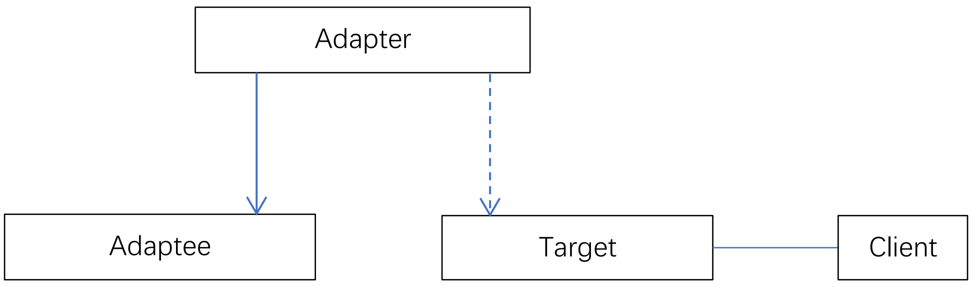
   In this case, the existing interface needs to be transformed into the interface expected by the customer class, which ensures the reuse of the existing class. Adapter pattern is created to do that.

   In the Adapter pattern, you can define a class that wraps objects with incompatible interfaces. Such class refers to the Adapter, and the object it wraps is the Adaptee, the class that is adapted.

   The adapter provides the interface required by the customer class, and the implementation of the adapter is to convert the request of the customer class into a call to the corresponding interface of the adapter. That is, when the client class calls the adapter's method, the adapter class's method is called inside the adapter class, and this process is transparent to the client class, which does not directly access the adapter class. As a result, the adapter enables classes that cannot interact with each other due to incompatible interfaces to work together. This is the pattern motivation for the adapter pattern.

The class Adapter implements the Target interface and wraps the Adaptee class to implement the interface transformation. The target interface needs an operation of operation 1, but the Adaptee class can only provide one operation of operation 2. Thus, an incompatibility occurs, and an operation 1 function, implemented through Adapter, converts Adaptee's operation 2 into the operation required by Target for compatibility.

2. UML Class Diagram:



Section 2: Java Source Code of Furutech

```
public final class FurutechPlug implements UKPlugConnector {

    Public void giveElectricity() {
        System.out.printIn(“giving electricity to a Furutech plug.”);
    }
}
```

Section 3: Creation of UKToGerman Adapter

In order to make German electrical sockets work together with UK plugs, the

UKPlugConnector is wrapped in a new class, which implements the GermanPlugConnector interface.

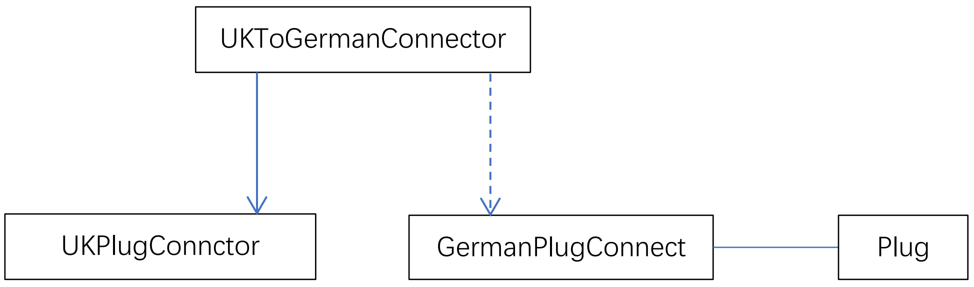
```
public class UKToGermanPlugConnectorAdapter implements GermanPlugConnctor
{
    private UKPlugConnector plug;
    public UKToGermanPlugConnectorAdapter(UKPlugConnector plug) {
        this.plug = plug;
    }
    @Override
    Public void provideElectricity() {
        Plug.giveElectricity();
    }
}
```

The adapter, UKToGermanConnectorAdapter wraped UKPlugConnector to satisfy the plugs from UK; besides, the adapter implements the GermanPlugConnector interface, which satisfy the sockets from Germany.

I can use the adapter like this:

```
UKPlugConnector plug = new FurutechPlug();
GermanElectricalSocket socket = new GermanElectricalSocket();
GermanPlugConnector germanAdapter = new UKToGermanPlugConnectorAdapter(plug);
Socket.plugIn(germanAdapter);
```

UML Class Diagram:


UKToGermanConnector
UKPlugConnctor
GermanPlugConnect
Plug