COMP2511

Adapter Pattern

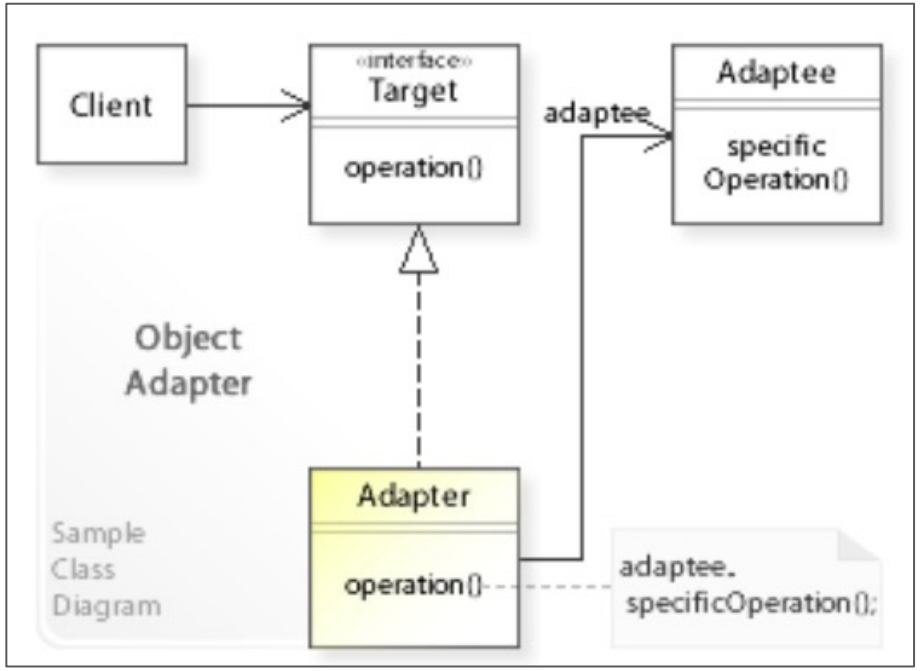
Prepared by

Dr. Ashesh Mahidadia

Adapter Pattern : Intent

- "*Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.*" [GoF]
- The adapter pattern allows the interface of an existing class to be used as another interface, suitable for a client class.
- The adapter pattern is often used to make existing classes (APIs) work with a client class without modifying their source code.
- The adapter class maps / joins functionality of two different types / interfaces.
- The adapter patter offers a wrapper around an existing useful class, such that a client class can use functionality of the existing class.
- The adapter pattern do not offer additional functionality.

Adapter Pattern: Structure



- The adapter contains an instance of the class it wraps.
- In this situation, the adapter makes calls to the instance of the wrapped object.

Adapter: Example

```java
interface LightningPhone {
    void recharge();
    void useLightning();
}

interface MicroUsbPhone {
    void recharge();
    void useMicroUsb();
}
```

```java
class Iphone implements LightningPhone {
    private boolean connector;

    @Override
    public void useLightning() {
        connector = true;
        System.out.println("Lightning connected");
    }

    @Override
    public void recharge() {
        if (connector) {
            System.out.println("Recharge started");
            System.out.println("Recharge finished");
        } else {
            System.out.println("Connect Lightning first");
        }
    }
}
```

```java
class Android implements MicroUsbPhone {
    private boolean connector;

    @Override
    public void useMicroUsb() {
        connector = true;
        System.out.println("MicroUsb connected");
    }

    @Override
    public void recharge() {
        if (connector) {
            System.out.println("Recharge started");
            System.out.println("Recharge finished");
        } else {
            System.out.println("Connect MicroUsb first");
        }
    }
}
```

Adapter: Example

```java
public class AdapterDemo {
    static void rechargeMicroUsbPhone(MicroUsbPhone phone) {
        phone.useMicroUsb();
        phone.recharge();
    }

    static void rechargeLightningPhone(LightningPhone phone) {
        phone.useLightning();
        phone.recharge();
    }

    public static void main(String[] args) {
        Android android = new Android();
        Iphone iPhone = new Iphone();

        System.out.println("Recharging android with MicroUsb");
        rechargeMicroUsbPhone(android);

        System.out.println("Recharging iPhone with Lightning");
        rechargeLightningPhone(iPhone);

        System.out.println("Recharging iPhone with MicroUsb");
        rechargeMicroUsbPhone(new LightningToMicroUsbAdapter(iPhone));
    }
}
```

```java
class LightningToMicroUsbAdapter implements MicroUsbPhone {
    private final LightningPhone lightningPhone;

    public LightningToMicroUsbAdapter(LightningPhone lightningPhone) {
        this.lightningPhone = lightningPhone;
    }

    @Override
    public void useMicroUsb() {
        System.out.println("MicroUsb connected");
        lightningPhone.useLightning();
    }

    @Override
    public void recharge() {
        lightningPhone.recharge();
    }
}
```

Output

```
Recharging android with MicroUsb
MicroUsb connected
Recharge started
Recharge finished
Recharging iPhone with Lightning
Lightning connected
Recharge started
Recharge finished
Recharging iPhone with MicroUsb
MicroUsb connected
Lightning connected
Recharge started
Recharge finished
```

Design Patterns: Discuss Differences

- Creational Patterns
  - Abstract Factory
  - Factory Method
  - Singleton
- Structural Patterns
  - Adapter *discussed*
  - Composite *discussed*
  - Decorator *discussed*
- Behavioral Patterns
  - Iterator *discussed*
  - Observer *discussed*
  - State *discussed*
  - Strategy *discussed*
  - Template
  - Visitor

We plan to discuss the rest of the design patterns above in the following weeks; and many more other topics.

End