Proxy Design Pattern

Dreghici Popa Vlad
30432

Description

- **Structural** pattern
- Provides a **placeholder** for another object to control acces to it

When to use?

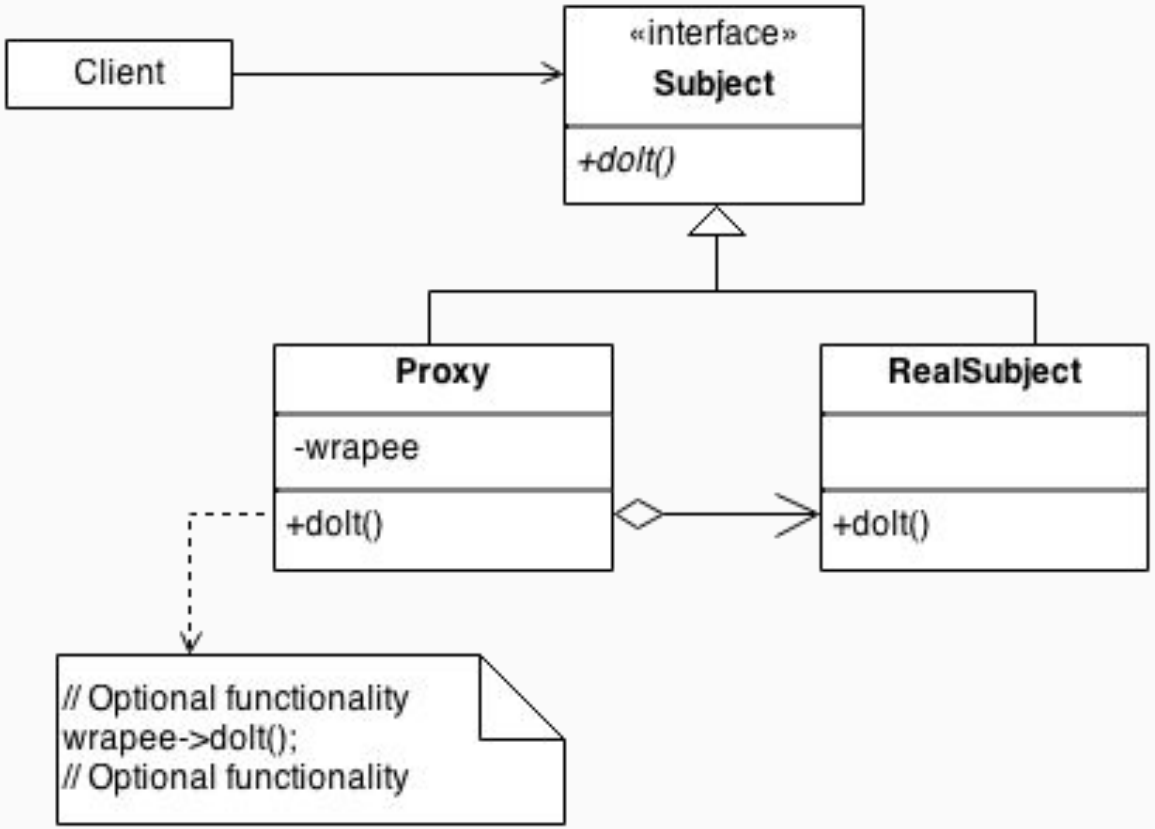
- Proxy pattern is used when we need to create a **wrapper** to cover the main object’s complexity from the client.

Types of proxies

- **Remote**
  - Provides a **local** representative for an object that resides in a different address space.
- **Virtual**
  - Placeholder for "**expensive to create**" objects. The real object is only created when a client **first requests/accesses** the object.
- **Protective**
  - Controls access to a sensitive master object. The "surrogate" object checks that the caller has the **access** permissions required prior to forwarding the request.
- **Smart**
  - Interposes additional **actions** when an object is accessed

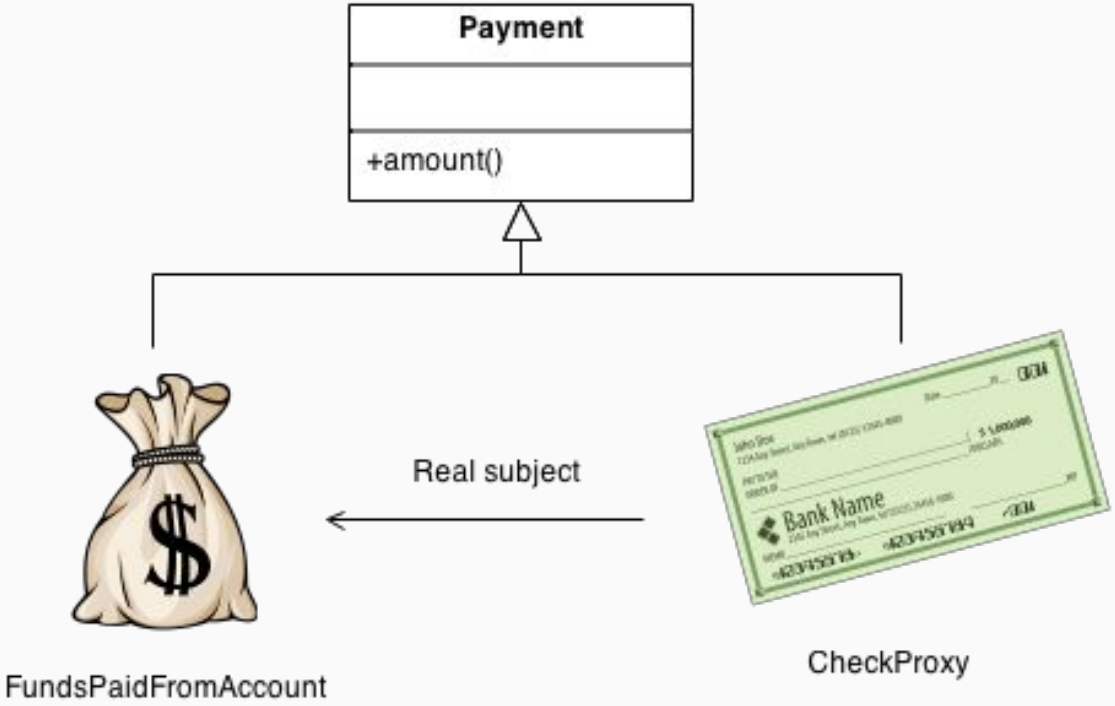
Structure

By defining a **Subject** interface, the presence of the Proxy object standing in place of the RealSubject is **transparent** to the client.

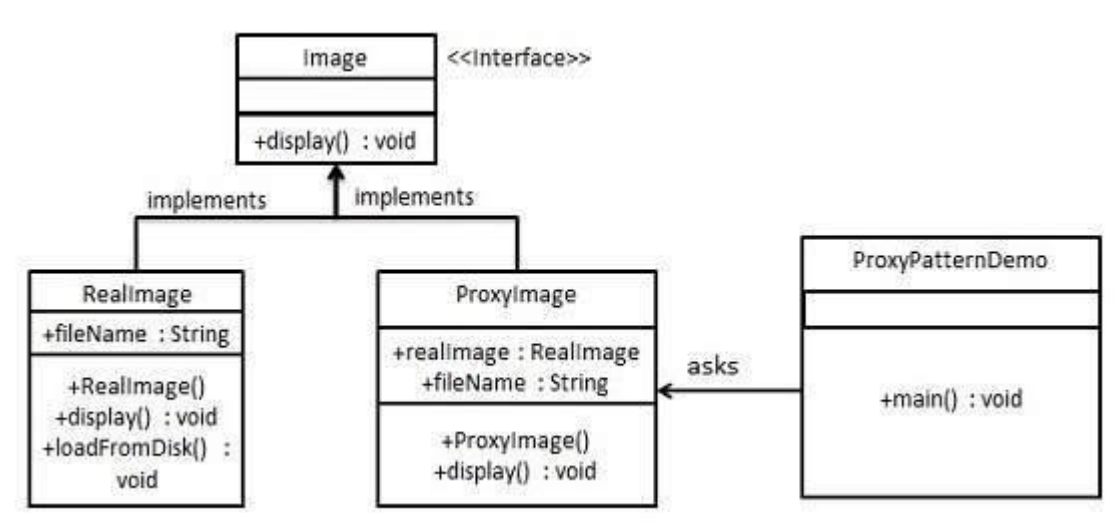


Example

The Proxy provides a surrogate or place holder to provide access to an object. A check or bank draft is a proxy for funds in an account. A check can be used in place of cash for making purchases and ultimately controls access to cash in the issuer's account.



Example



We are going to create an ***Image*** interface and concrete classes implementing the *Image* interface. ***ProxyImage*** is a a proxy class to reduce memory footprint of ***RealImage*** object loading.

ProxyPatternDemo, our demo class, will use ***ProxyImage*** to get an ***Image*****object** to load and display as it needs.

Proxy for loading an image

Interface Image.java

```
public interface Image {

   void display();

 }
```

Concrete Class implementing Interface RealImage.java

```
public class RealImage implements Image {

   private String fileName;

   public RealImage(String fileName){
      this.fileName = fileName;
      loadFromDisk(fileName);
   }

   @Override
   public void display() {
      System.out.println("Displaying " + fileName);
   }

   private void loadFromDisk(String fileName){
      System.out.println("Loading " + fileName);
   }

 }
```

Proxy for loading an image

ProxyImage to get object of RealImage class when required

```
public class ProxyImage implements Image{
    private RealImage realImage;
    private String fileName;
    public ProxyImage(String fileName){
        this.fileName = fileName;
    }
    @Override
    public void display() {
        if(realImage == null){
            realImage = new RealImage(fileName);
        }
        realImage.display();
    }
}
```