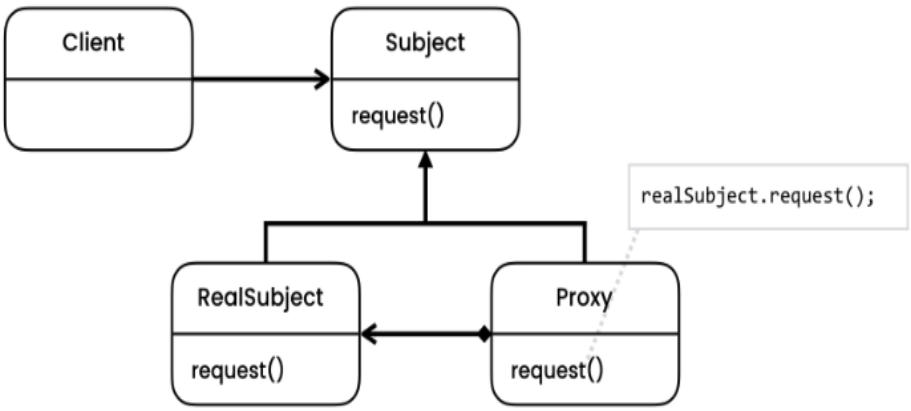
Proxy Pattern:

Definition: Allows providing a substitute for another object. The proxy object delegates all the work to the target object and contains some additional behavior.



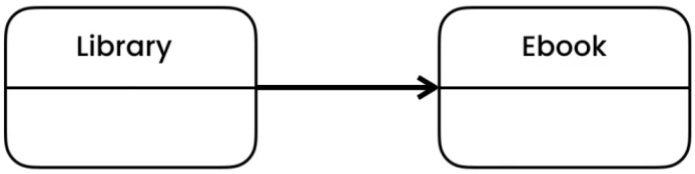
Scenario: We want to build a library for reading ebooks. We first add new ebook in library. Once we add all the books, we load all of them and opens the ebook we want.

Implementation:

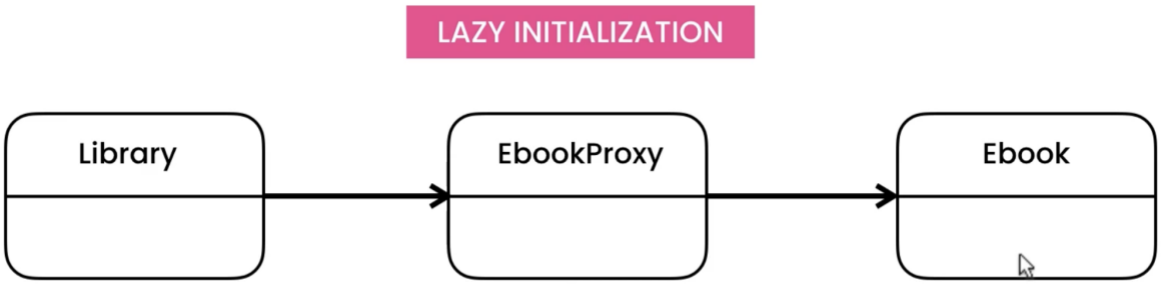
```
public class Ebook {
   private String fileName = new HashMap<>();

   public void Ebook (String fileName){
      this.filName = filename;
            load();
   }
   public void load(){
       //Loads the ebook
   }

   public void show(){
      //Shows the ebook
   }
}
```

```
public class Library {
    private Map<String, Ebook> ebooks = new HashMap<>();

    public void add(Ebook ebook){
        ebooks.put(ebook.getFileName(), ebook);
    }

    public void openEbook(String fileName){
        ebooks.get(fileName).show();
    }
}
```

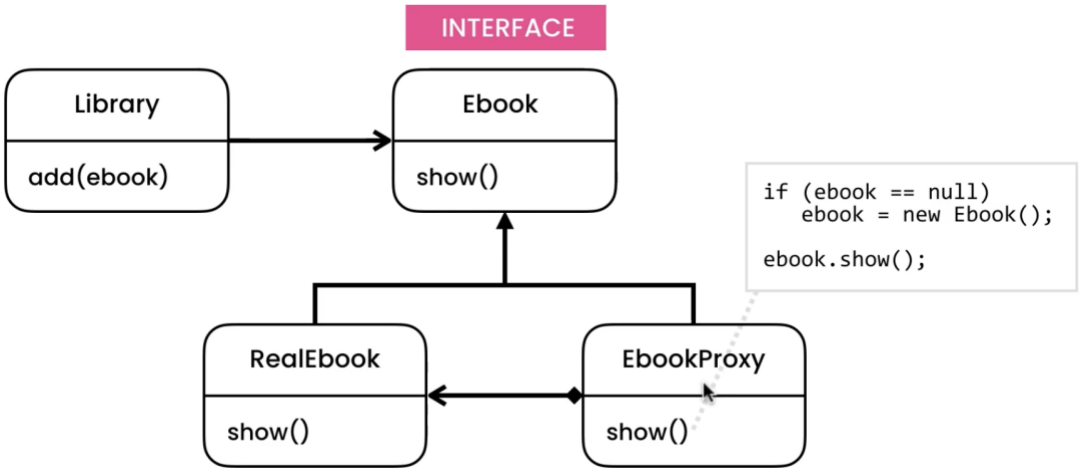
Problem: Creating ebook is costly, because we have to read ebook file from the disk and load it to the memory for a new book addition. We first load all the ebooks from the memory and show the user his required book. If we have 1 million books in library, we first load all and show user for example book one.



Solution: We need a proxy in between, which can implement lazy initialization and talk with ebook.



We will use Realbook instance in EbookProxy but we will only initialize it when it will get called. This means that we will only load that ebook and save in memory, which we want to show. This is also called lazy initialization or on-demand.




```
public interface Ebook {
    void show();
}

public class RealEbook implements Ebook {
    private String fileName;

    public RealEbook(String fileName) {
        this.fileName = fileName;
        load();
    }

    private void load(){
        System.out.println("Loading the ebook: " + fileName);
    }

    @Override
    public void show(){
        System.out.println("Showing the ebook: "+ fileName);
    }

}
public class EbookProxy implements Ebook {
    private String fileName;
    private RealEbook ebook; // Dont initialize it, only initialize when needed
```

```
public EbookProxy(String fileName) {
    this.fileName = fileName;
}

@Override
public void show() {
    if(ebook == null)  // Lazy initialization
        ebook = new RealEbook(fileName);

    ebook.show();
}

@Override
public String getFileName() {
    return fileName;
}
```