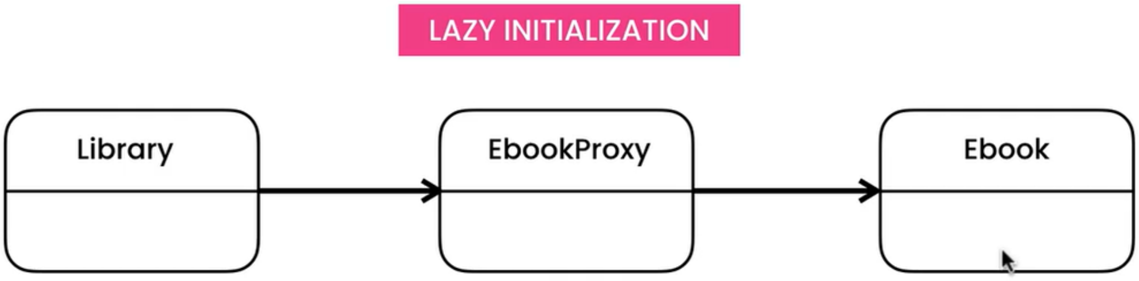
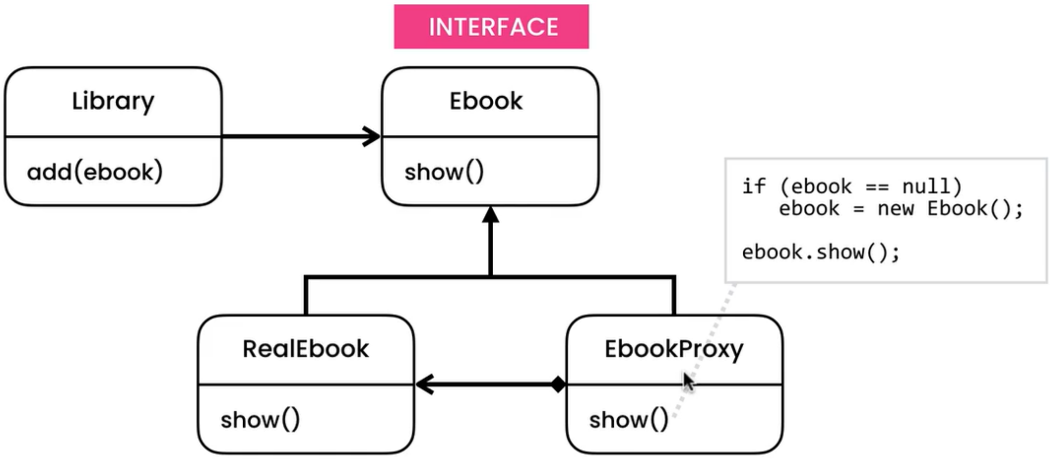
Proxy Pattern

Example Problem - You are developing an ebook application with an Ebook class and a Library class which store, searches and displays the Ebooks. The library class could rad all the Ebooks from a database and then store them all in an array / in moemory. But this would be costly, the proxy pattern looks to solve this by simulating an ebook to avooid them all being stored in memory, then load it on demand only when needed. Known as Lazy Loading :



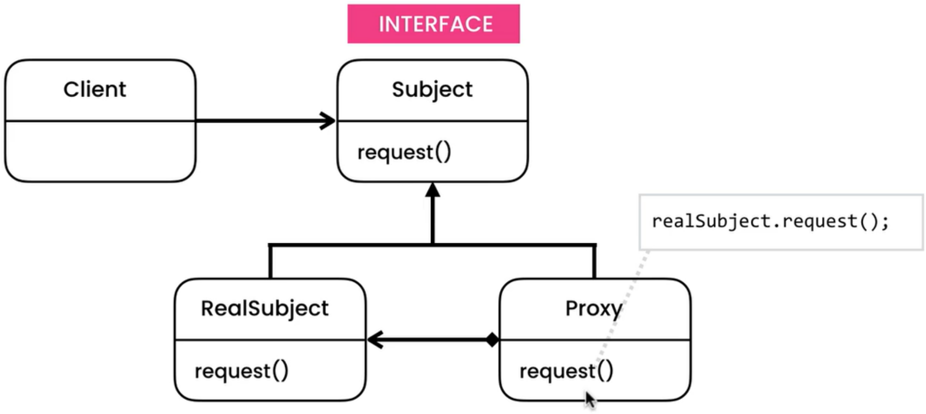
This is achieved with the following structure :



```
if (ebook == null)
   ebook = new Ebook();

ebook.show();
```

The realebook is initial null in the Ebook proxy (which is composed of real ebooks) until it is needed when a new ebook is initialed and the show method is called. As per the GOF book :



```
realSubject.request();
```

As you may notice there is no lazy loading in this example highlighting that it can be used to solve other problems such as logging, authorisation or lazy loading etc. I.e. before calling the RealSubjects Request method we run some code in the proxy's request method.

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

 String getFileName();
}
```

```java
public class EbookProxy implements Ebook {
  private String fileName;
 private RealEbook ebook;

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

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

 ebook.show();
 }

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

```java
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);
 }

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

```java
public class LoggingEbookProxy implements Ebook {
  private String fileName;
 private RealEbook ebook;

 public LoggingEbookProxy(String fileName) {
    this.fileName = fileName;
 }

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

 System.out.println("Logging");
 ebook.show();
 }

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

```java
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();
 }
}
```

```java
public class Main {
    public static void main(String[] args) {
    var library = new Library()'
    String[] fileNames = {"a", "b", "c"}

for (var fileName: fileNames)

libary.add(new RealEbook());

library.open("a")

//outputs adding a and then showing a highlighting the

//lazy loading enabled by the proxy pattern
```