



## مشروع هندسة البرمجيات 3

إعداد الطلاب :

أحمد محمد مريود

لين فادي الأشقر

ماريمار مأمون رضوان

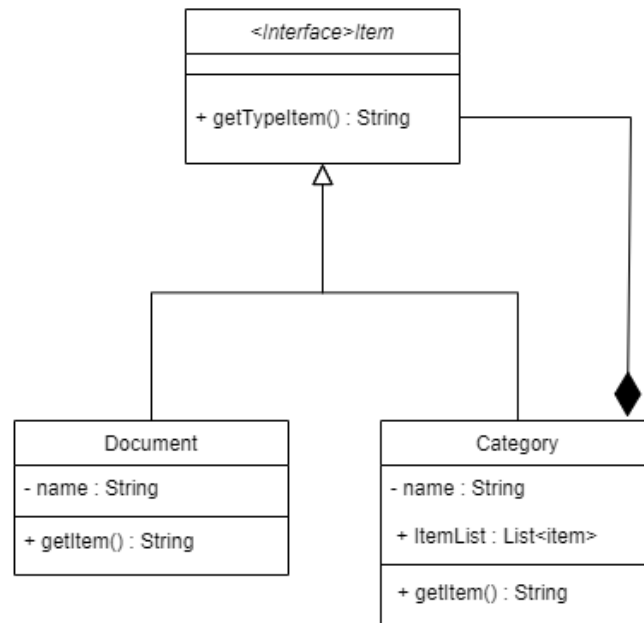
بإشراف المهندس : محمد تقاله

تاريخ التسليم : 2022 /12 /15

## الطلب الأول:

الحل باستخدام Composite

التعليق: نحتاج لفئات تحوي على فئات ووثائق، نلاحظ وجود تعاودية في الفئات فالديزاين الأنسب هو Composite



الكود البرمجي:

```
package com.company.step1;

public interface Item {
    String getTypeItem();
}
```

```
package com.company.step1;

public class Document implements Item {
    String name;

    @Override
    public String getTypeItem() {
        return "Document";
    }

    @Override
    public String toString() {
        return name;
    }
}
```

```
package com.company.step1;

import java.util.ArrayList;
import java.util.List;

public class Category implements Item {
    String name;
    List<Item> itemList=new ArrayList<>();

    @Override
    public String getTypeItem() { return "Ca";
    }

    @Override
    public String toString() {
        return name+" : "+itemList;
    }
}
```

تجريب الكود:

يتم تعريف مجموعة من الفئات بداخل كل فئة وثائق وفئات أخرى، نلاحظ عند طباعة الفئات أن الفئة cat2 تحوي على فئة cat1 ووثيقة doc2 ووثيقة doc3، والفئة cat1 تحوي على وثيقة doc1

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Document doc1=new Document();  
        Document doc2=new Document();  
        Document doc3=new Document();  
        doc1.setName("doc1");  
        doc2.setName("doc2");  
        doc3.setName("doc3");  
  
        Category category1= new Category();  
        category1.setName("cat1");  
        category1.getItemList().add(doc1);  
  
        Category category2= new Category();  
        category2.setName("cat2");  
        category2.getItemList().add(category1);  
        category2.getItemList().add(doc2);  
        category2.getItemList().add(doc3);  
  
        Item item = category2;  
        System.out.println(item.toString());  
    }  
}
```

Main > main()

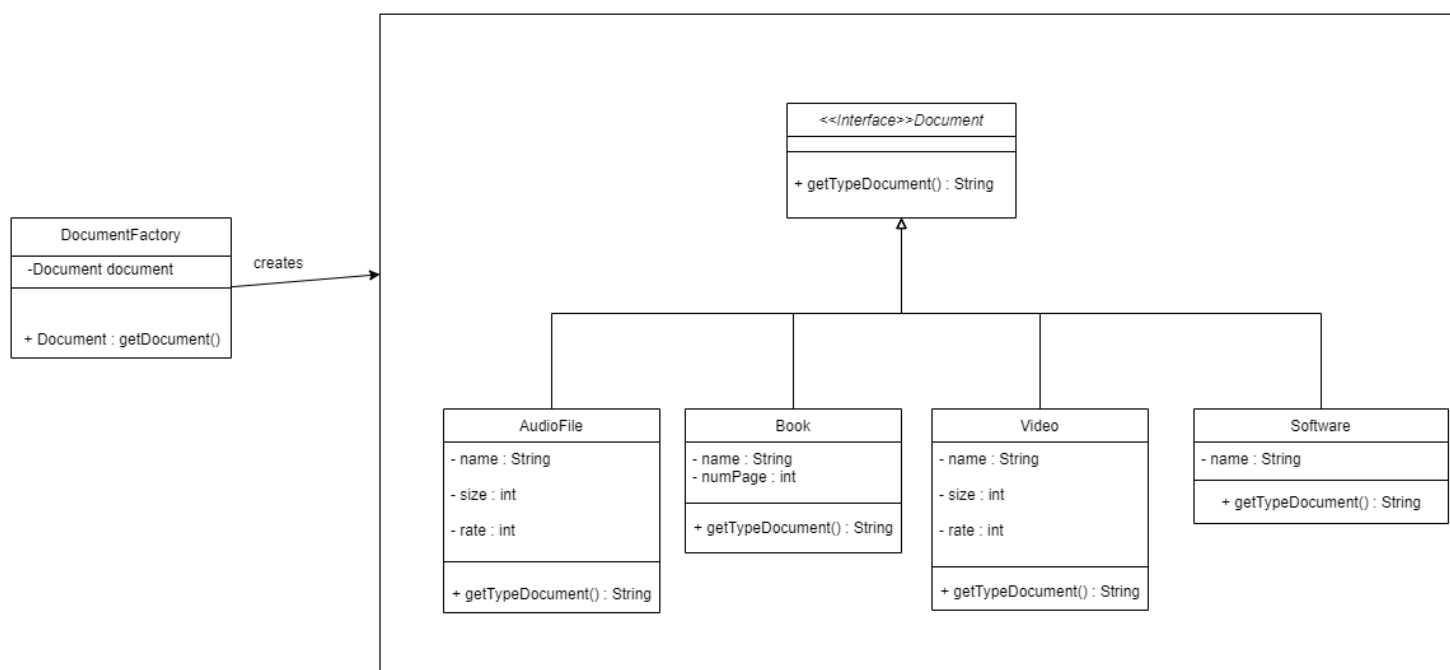
Main x

"C:\Program Files\Java\jdk-11.0.8\bin\java.exe"  
cat2 : [cat1 : [doc1], doc2, doc3]

## الطلب الثاني:

### الحل باستخدام Factory Method

التعليق: نحتاج لأنواع لها خصائص مختلفة من الوثيقة فالحل الأنسب هو Factory، من الممكن أيضا استخدام Bridge لتمثيل الأنواع المختلفة وإمكانية إضافة أنواع مختلفة لاحقا.



## الكود البرمجي:

```
package com.company.step2;

public interface Document {
    public String getTypeDocument();
}
```

```
public class AudioFile implements Document{
    String name;
    int rate;
    int size;
    @Override
    public String getTypeDocument() {
        return "AudioFile";
    }
}
```

```
package com.company.step2;

public class Book implements Document{
    String name;
    int numPage;
    @Override
    public String getTypeDocument() { return "Book"; }
```

```
package com.company.step2;

public class Video implements Document{
    String name;
    int rate;
    int size;
    @Override
    public String getTypeDocument() { return "Video"; }
```

```
package com.company.step2;

public class Software implements Document{
    String name;
    int rate;
    @Override
    public String getTypeDocument() { return "Software"; }
```

تجريب الكود:

قمنا بتعريف DocumentFactory من نوع كتاب وطباعة نوعه ومن ثم ملف الصوتي ومن ثم الفيديو

The screenshot shows the IntelliJ IDEA IDE. On the left, the 'step2' package is expanded, showing classes: AudioFile, Book, Document, DocumentFactory, Software, Video, and Main. The 'Main' class is selected. The editor shows the following code:

```
37
38
39 DocumentFactory documentFactory=new DocumentFactory();
40 documentFactory.setDocument(new Book());
41 System.out.println(documentFactory.getDocument().getTypeDocument());
42 documentFactory.setDocument(new AudioFile());
43 System.out.println(documentFactory.getDocument().getTypeDocument());
44 documentFactory.setDocument(new Video());
45 System.out.println(documentFactory.getDocument().getTypeDocument());
```

The output window at the bottom shows the execution results:

```
Book
AudioFile
Video
```

### الطلب الثالث:

الحل باستخدام Adapter

التعليق: نحتاج لاستخدام مكتبة الصوت بدون التعديل عليها فلدينا نوعان من الـ ديزاين باترن (Bridge - Adapter)، الحل الأنسب هو Adapter



الكود البرمجي:

```
package com.company.step3;

public class AudioLibrary {
    String name;
    void play() {
        System.out.println("play audio : "+name);
    }
}
```

```
import com.company.step2.AudioFile;

public class AudioAdapter {
    AudioLibrary audioLibrary;
    public void play(AudioFile audioFile) {
        audioLibrary=new AudioLibrary();
        audioLibrary.setName(audioFile.getName());
        audioLibrary.play();
    }
}
```

تجريب الكود

قمنا بتعريف ملف صوتي ومحول من المكتبة الصوتية واستدعاء تابع التشغيل من المحول مع اسناد الملف الصوتي له.

```
44 // System.out.println(documentFactory.getDocument());
45 // documentFactory.setDocument(new Video());
46 // System.out.println(documentFactory.getDocument());
47
48
49 AudioFile audioFile=new AudioFile();
50 audioFile.setName("audiol");
51 AudioAdapter audioAdapter=new AudioAdapter();
52 audioAdapter.play(audioFile);

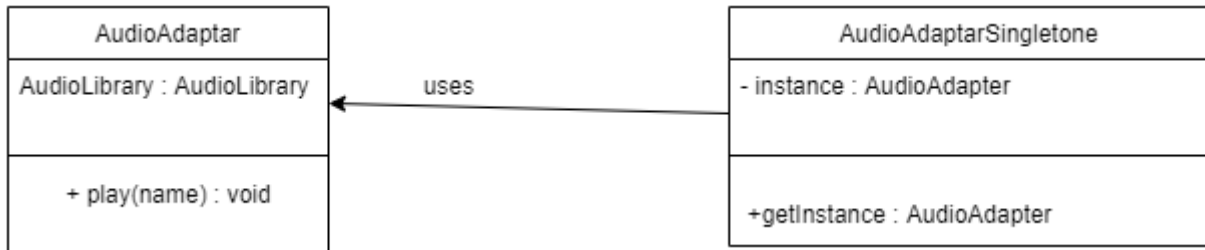
Main > main()

"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
play audio : audiol
```

## الطلب الرابع:

الحل باستخدام Singleton

التعليق: نحتاج لاستخدام مكتبة الصوت لمرة واحدة في كامل النظام فالحل الأنسب هو Singleton على المكتبة المستخدم عليها Adapter بالطلب السابق



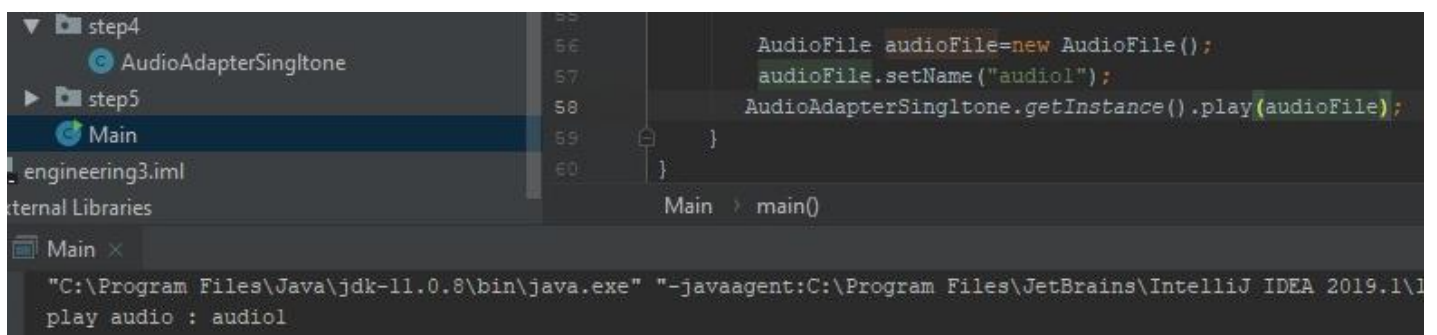
الكود البرمجي:

```
import com.company.step3.AudioAdapter;
import com.company.step3.AudioLibrary;

public class AudioAdapterSingltone {
    static AudioAdapter audioAdapter;
    public static AudioAdapter getInstance() {
        if(audioAdapter==null) {
            audioAdapter=new AudioAdapter();
        }
        return audioAdapter;
    }
}
```

تجريب الكود:

قمنا بتعريف ملف صوتي واستعاء تابع التشغيل من المحول الوحيد في النظام.

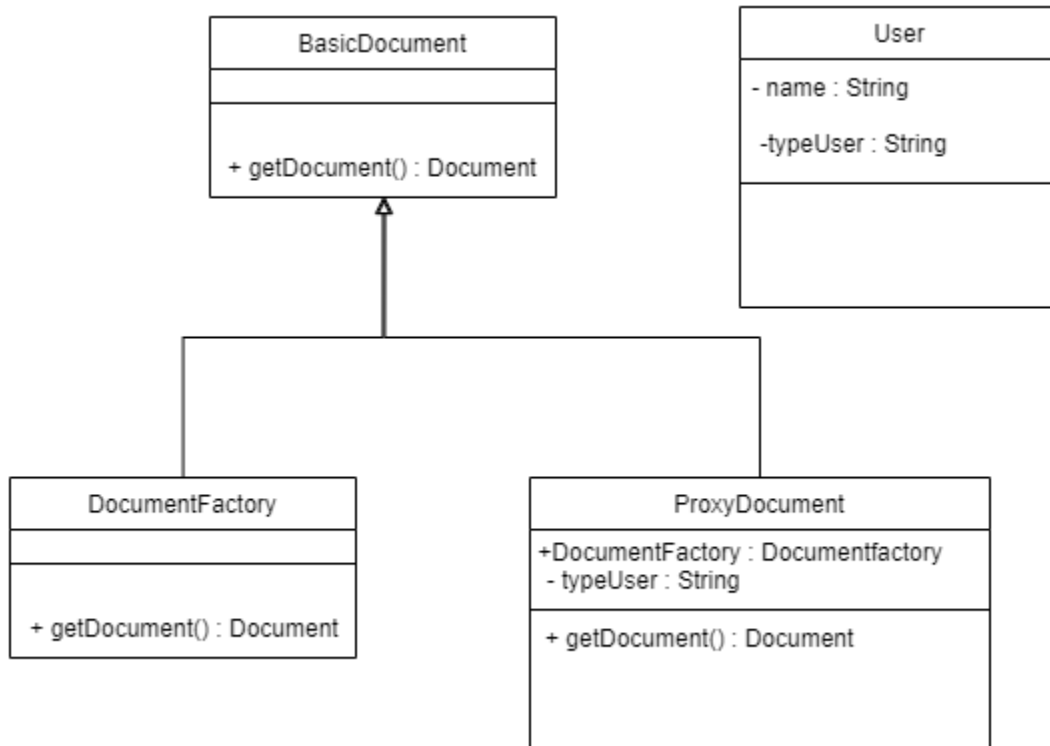




## الطلب الخامس:

الحل باستخدام Proxy

التعليق: نحتاج لتغليف الوثائق بطبقة حماية ليتمكن الأدمن فقط من الوصول اليهم، الحل الأنسب هو Proxy لإضافة طبقة حماية، ومن الممكن استخدام Chain of Responsibility، لكن من وجهة نظر مختلفة يمكن تركيب ال Chain مع ال Proxy في حال تم إضافة شروط لطبقة الحماية.



عندما نقوم بتعريف وثيقة إما من **DocumentFactory** بشكل عادي او من **ProxyDocument** حيث قبل جلب الوثيقة يتحقق من المستخدم ان كان أدمن أم لا

الكود البرمجي:

```
package com.company.step5;

public class User {
    String name;
    String typeUser;

    User(String name,String typeUser){
        this.name=name;
        this.typeUser=typeUser;
    }
}
```

```
public interface BasicDocument {
    public Document getDocument();
}
```

```
public enum TypeUser {
    user,
    admin
}
```



```

public class ProxyDocument implements BasicDocument{

    DocumentFactory documentFactory=new DocumentFactory();
    String typeUser;

    public void setTypeUser(String typeUser) { this.typeUser = typeUser; }

    @Override
    public Document getDocument() {
        if(typeUser.equals(TypeUser.admin.name()))
            return documentFactory.getDocument();
        else return null;
    }
}

```

تجريب الكود :

قمنا بتعريف وثيقة عادية من نوع ملف صوتي ووثيقة محمية من نوع فلم وضعهن في فئة والدخول كمستخدم عادي، نلاحظ عدم اظهار الوثيقة من نوع فيديو لأن المستخدم ليس أدمن

```

Category category1= new Category();
category1.setName("cat1");
BasicDocument basicDocument1=new DocumentFactory()
ProxyDocument proxyDocument=new ProxyDocument();
proxyDocument.setTypeUser(TypeUser.user.name());
BasicDocument basicDocument2=proxyDocument;
basicDocument1.setDocument(new AudioFile());
basicDocument2.setDocument(new Video());
category1.getItemList().add(basicDocument1);
category1.getItemList().add(basicDocument2);
Item item = category1;
System.out.println(item.toString());

```

Main > main()

Main ×

"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagen  
cat1 : [AudioFile:null, ]

## الطلب السادس:

الحل باستخدام Builder

التعليق: نحتاج لتجزئة الكتاب لفصول والفصول لصفحات أي أن الكتاب يتألف من عدة فصول وكل فصل يتألف من عدة صفحات فلدينا عدة أنواع منهم (Composite , Builder)، فيمكن بناء الفصل من صفحات والكتاب من فصول فالحل Builder.

الكود البرمجي:

```
public class Book implements Document{
    String name;
    List<Chapter> chapterList=new ArrayList<>();
    @Override
    public String getTypeDocument() { return "Book"; }

    @Override
    public String toString() {
        return getTypeDocument()+" "+name+chapterList;
    }
}
```

```
public class Page {
    String name;
    String detailsPage ;

    public void setName(String name)
    {
        this.name = name;
    }
}
```

```
import com.company.step6.Chapter;

import java.util.ArrayList;
import java.util.List;

public class Book implements Document{
    String name;
    List<Chapter> chapterList=new ArrayList<>();
    @Override
    public String getTypeDocument() { return "Book"; }

    @Override
    public String toString() {
        return getTypeDocument()+" "+name+chapterList;
    }
}
```

## تجريب الكود:

قمنا بتعريف فصل يحتوي 4 صفحات وفصل آخر يحتوي صفحتين وكتاب يحتوي الفصلين، عند طباعة الكتاب نلاحظ أنه قام بطباعة الفصلين مع عدد الصفحات لكل فصل

The screenshot shows the IntelliJ IDEA IDE with the 'Main' class selected in the project view on the left. The code editor on the right displays the following Java code:

```

DocumentFactory documentFactory=new DocumentFactor
Book book=new Book();
book.setName("book1");

Chapter chapter=new Chapter();
chapter.setName("chapter1");

chapter.getPageList().add(new Page());
chapter.getPageList().add(new Page());
book.getChapterList().add(chapter);

Chapter chapter2=new Chapter();
chapter2.setName("chapter2");

chapter2.getPageList().add(new Page());
chapter2.getPageList().add(new Page());
chapter2.getPageList().add(new Page());
chapter2.getPageList().add(new Page());
book.getChapterList().add(chapter2);

documentFactory.setDocument(book);
System.out.println(documentFactory.getDocument());

```

The output window at the bottom shows the following text:

```

Main > main()

"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2019.
Book:book1[Chapter:chapter1:2, Chapter:chapter2:4]

```