
**جامعة دمشق**

**سنة خامسة_ برمجيات**

# مشروع هندسة البرمجيات 3

**إعداد الطلاب :**

**أحمد محمد مريود**

**لين فادي الأشقر**

**ماريمار مأمون رضوان**

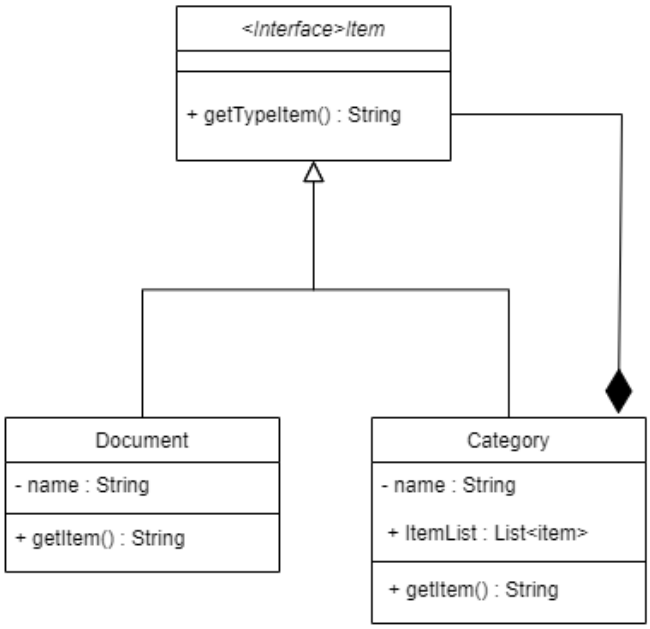
**باشراف المهندس : محمد تقاله**

**تاريخ التسليم :15/ 12/ 2022**

**الطلب الأول:**

الحل باستخدام Composite

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




الكود البرمجي:

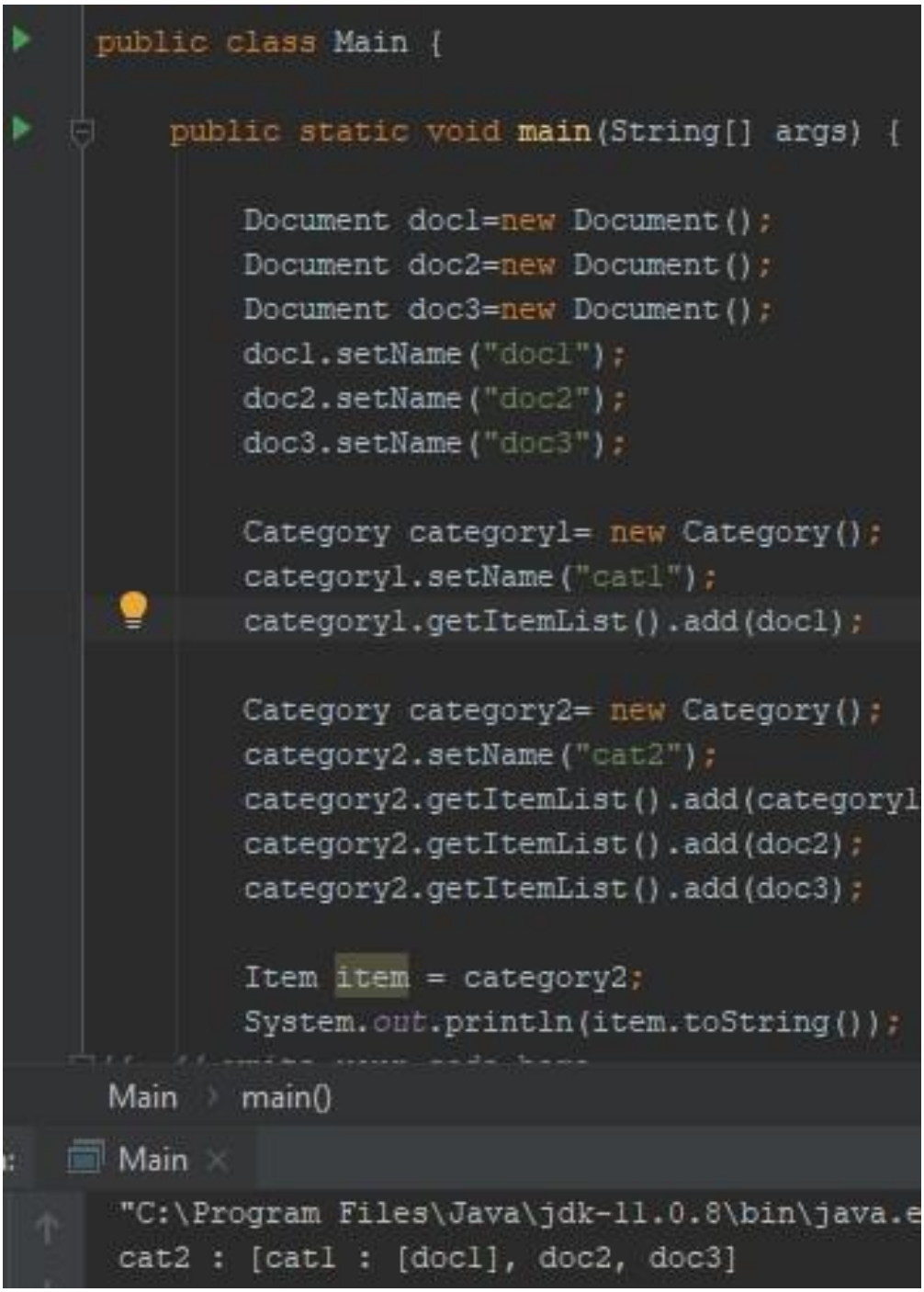
```java
package com.company.step1;

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

```java
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;
    }
```

```java
package com.company.step1;

public class Document implements Item {
    String name;

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

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

تجريب الكود:

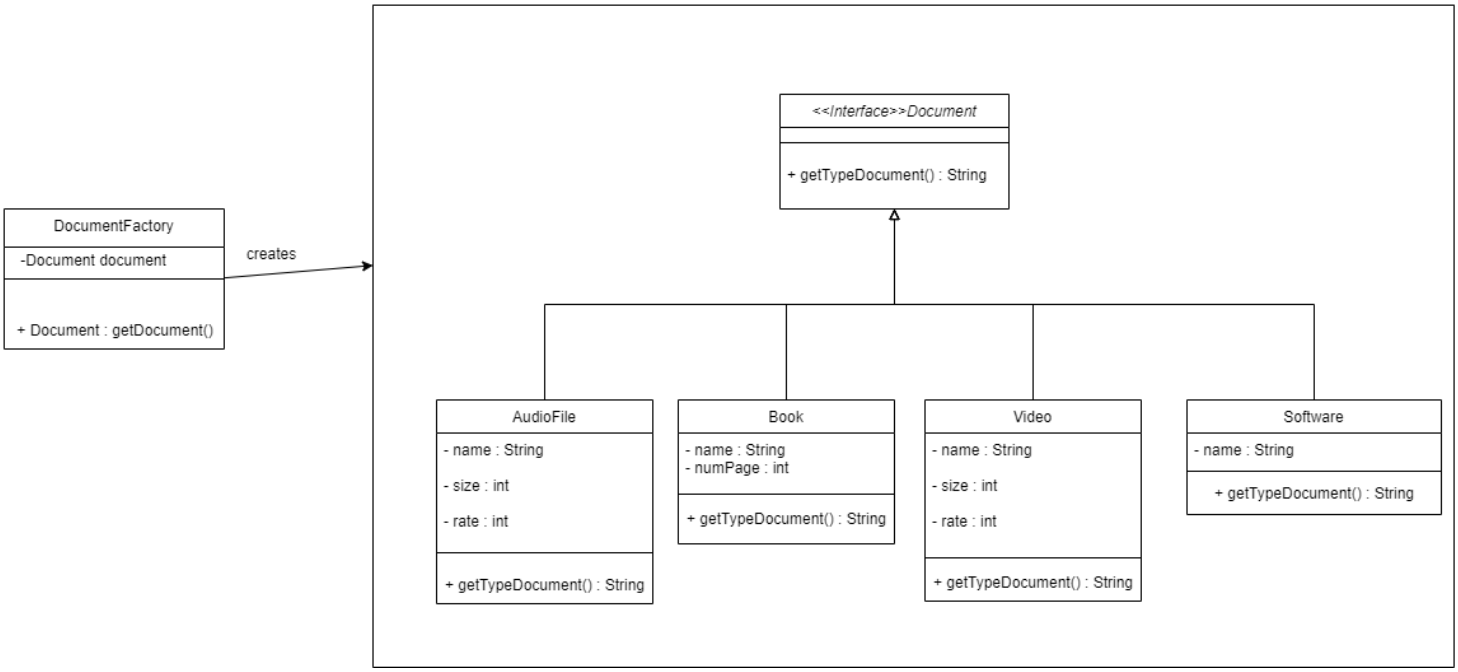
يتم تعريف مجموعة من الفئات بداخل كل فئة وثائق وفئات أخرى، نلاحظ عند طباعة الفئات أن الفئة cat2 تحوي على فئة cat1 ووثيقة doc2 ووثيقة doc3، والفئة cat1 تحتوي على وثيقة doc1



## الطلب الثاني:

الحل باستخدام Factory Method

التعليل: 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:

```java
package com.company.step2;

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

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

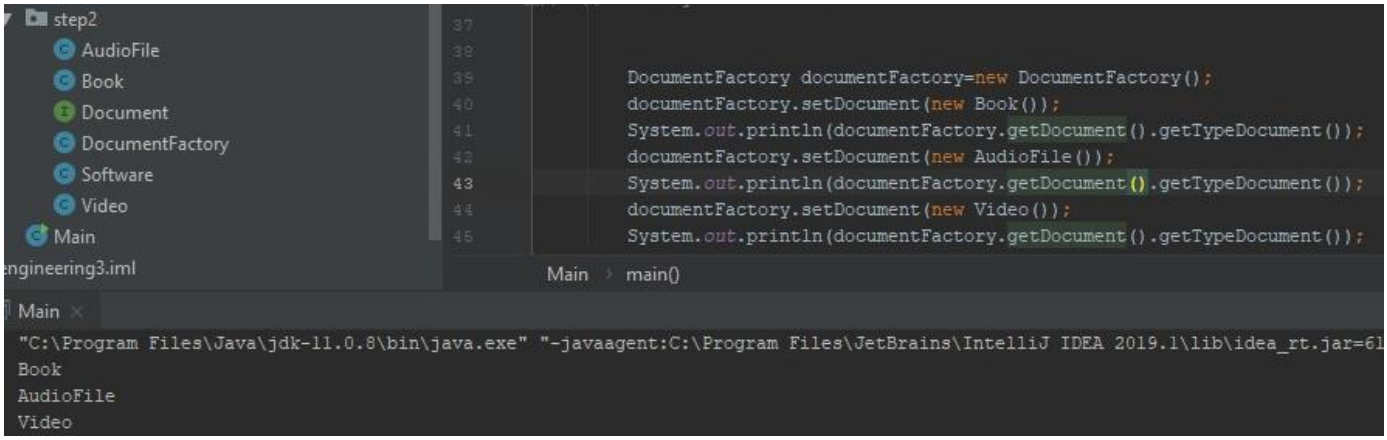
```java
package com.company.step2;

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

```java
package com.company.step2;

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

```java
package com.company.step2;

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

تجريب الكود:

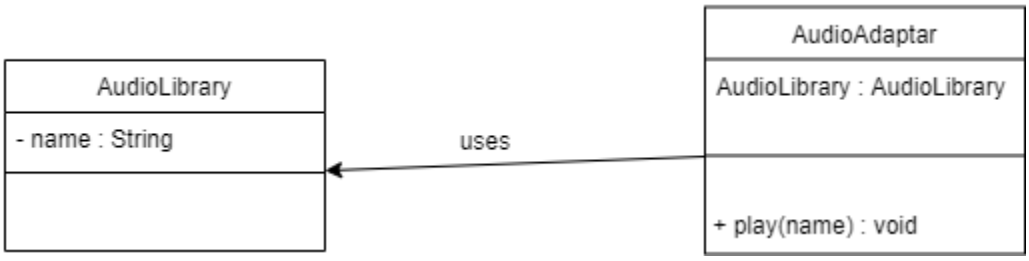
قمنا بتعريف `DocumentFactory` من نوع كتاب وطباعة نوعه ومن ثم ملف الصوتي ومن ثم الفيديو



**الطلب الثالث:**

الحل باستخدام Adapter

التعليل: نحتاج لاستخدام مكتبة الصوت بدون التعديل عليها فلدينا نوعان من الديزاين باتيرن (Bridge - Adapter)، الحل الأنسب هو Adapter



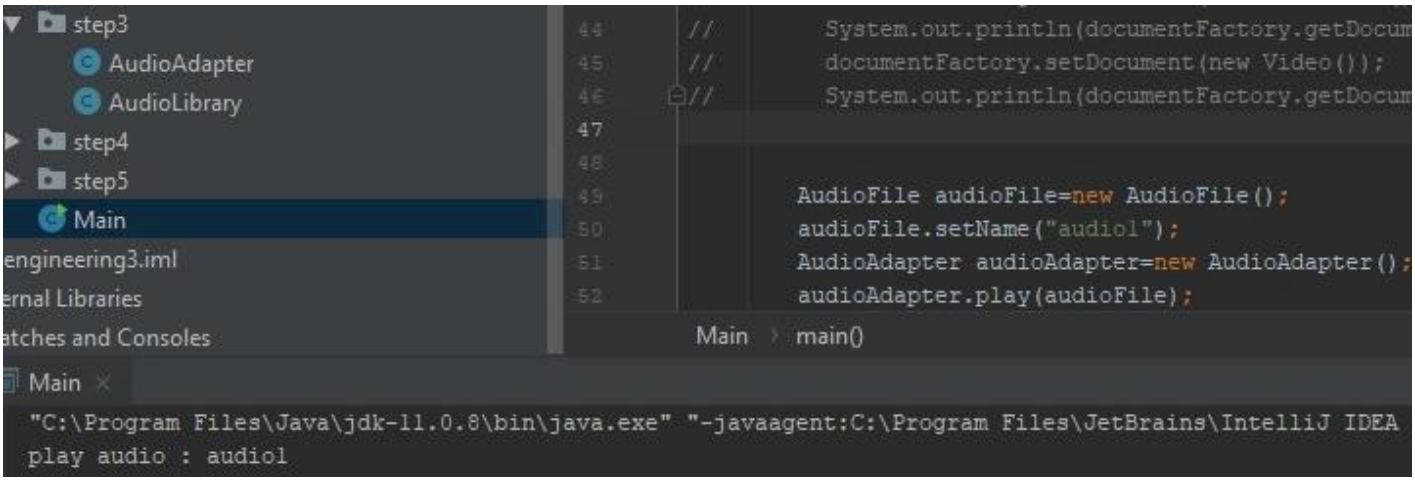

الكود البرمجي:

```java
package com.company.step3;

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

```java
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

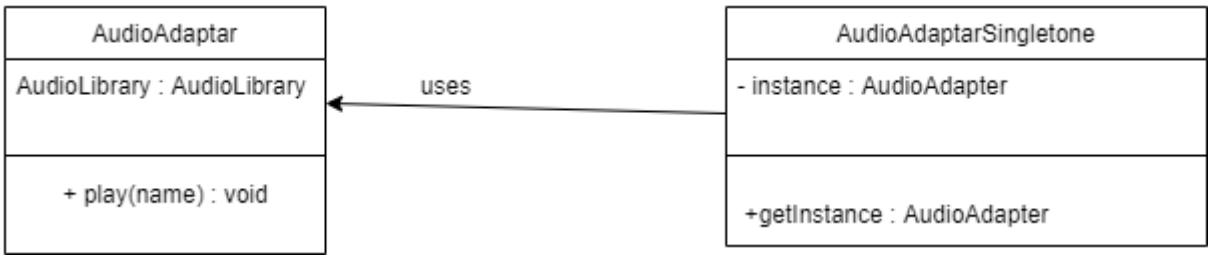
قمنا بتعريف ملف صوتي ومحول من المكتبة الصوتية واستدعاء تابع التشغيل من المحول مع اسناد الملف الصوتي له.



**الطلب الرابع:**

الحل باستخدام Singleton

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

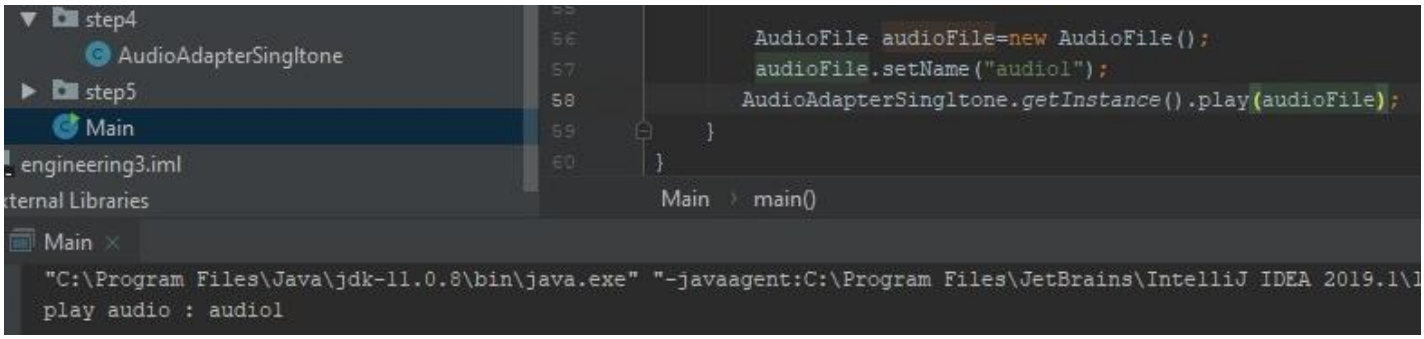



الكود البرمجي:

```
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:

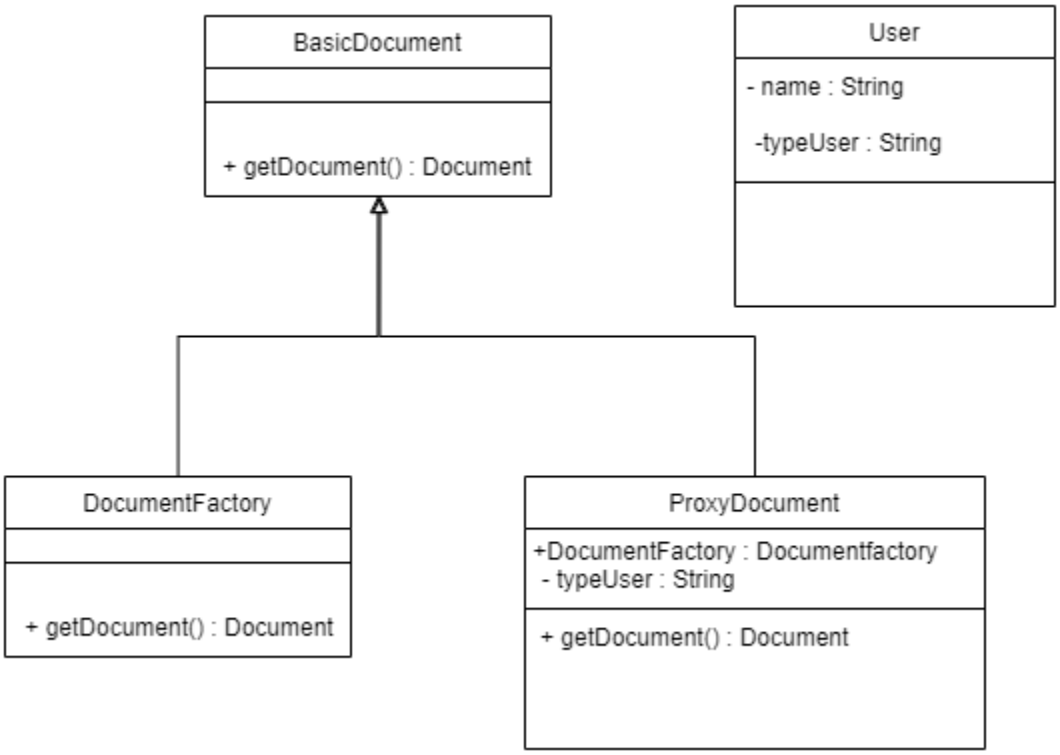
قمنا بتعريف ملف صوتي واستعاء تابع التشغيل من المحول الوحيد في النظام.



**الطلب الخامس:**

الحل باستخدام Proxy

التعليل: نحتاج لتغليف الوثائق بطبقة حماية ليتمكن الأدمن فقط من الوصول اليهم، الحل الأنسب هو Proxy لإضافة طبقة حماية، ومن الممكن استخدام Chain of Responsability، لكن من وجهة نظر مختلفة يمكن تركيب الـ Chain مع الـ 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:

```java
package com.company.step5;

public class User {
    String name;
    String typeUser;

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

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

}
```

```java
lic enum TypeUser
 user,
 admin
```

```java
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 المتسخدم ليس أدمن

```java
        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

التعليل: 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:

```java
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;
    }
```

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

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

```java
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 أخر يحتوي صفحتين وكتاب يحتوي الفصلين، عند طباعة الكتاب نلاحظ أنه قام بطباعة الفصلين مع عدد الصفحات لكل فصل