ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Filière :
« Génie du Logiciel et des Systèmes Informatiques Distribués »
GLSID

Design patterns

Année Universitaire : 2022-2023

Réalisé par :

Amina MAAKOUL

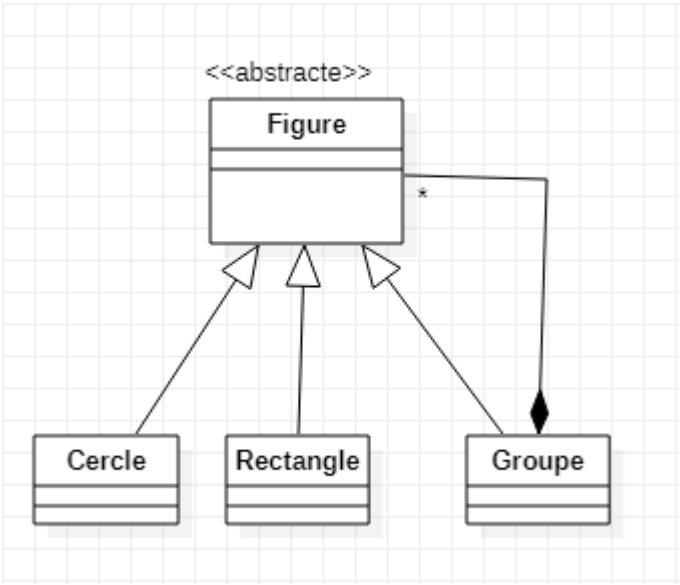
Encadré par :

M. Mohamed YOUSSFI

Exercice 1 :

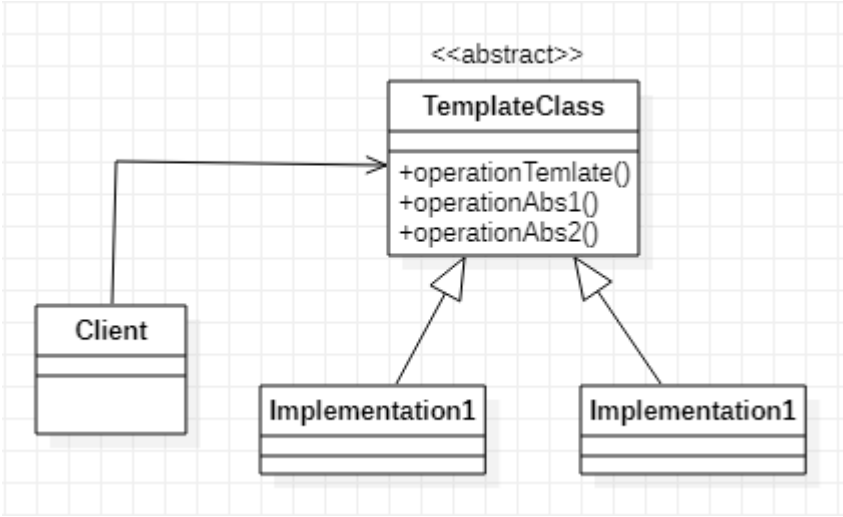
1. Une figure peut être soit un cercle, un rectangle ou un groupe de figures.

Le design pattern approprié à cette situation c'est : Composite



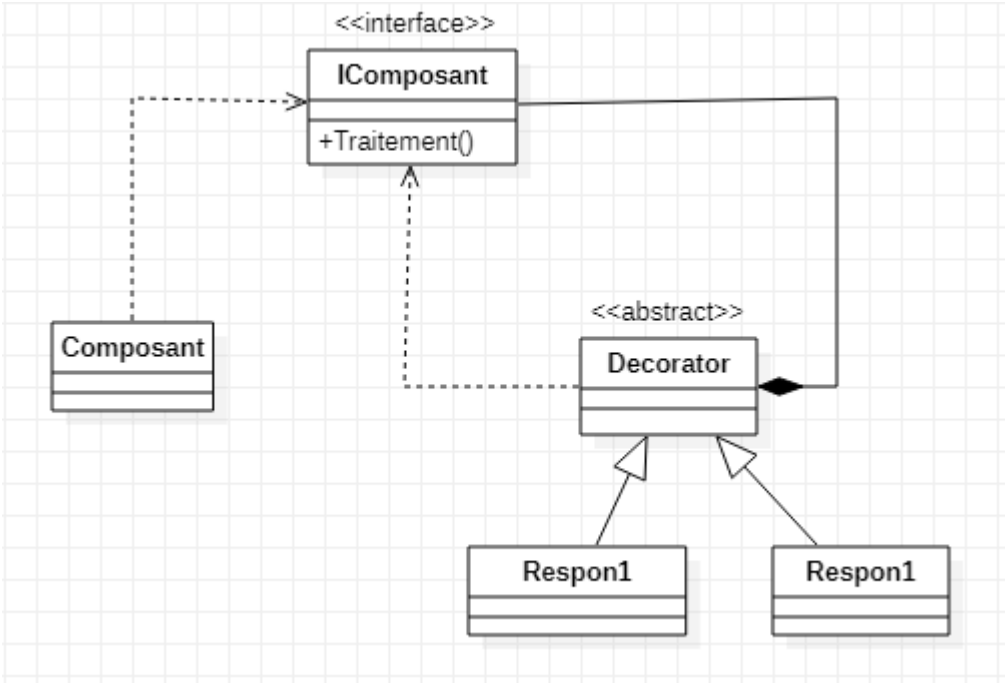
2. Un plugin contient une opération implémentant le squelette d'un algorithme dont deux parties (partie1 et partie2) sont variables. On voudrait laisser le développeur la possibilité d'implémenter les deux parties manquantes de cet algorithme et on voudrait aussi que l'application cliente puisse instancier une implémentation concrète du plugin sans connaitre sa classe d'implémentation.

Le design pattern approprié à cette situation c'est : Template Method



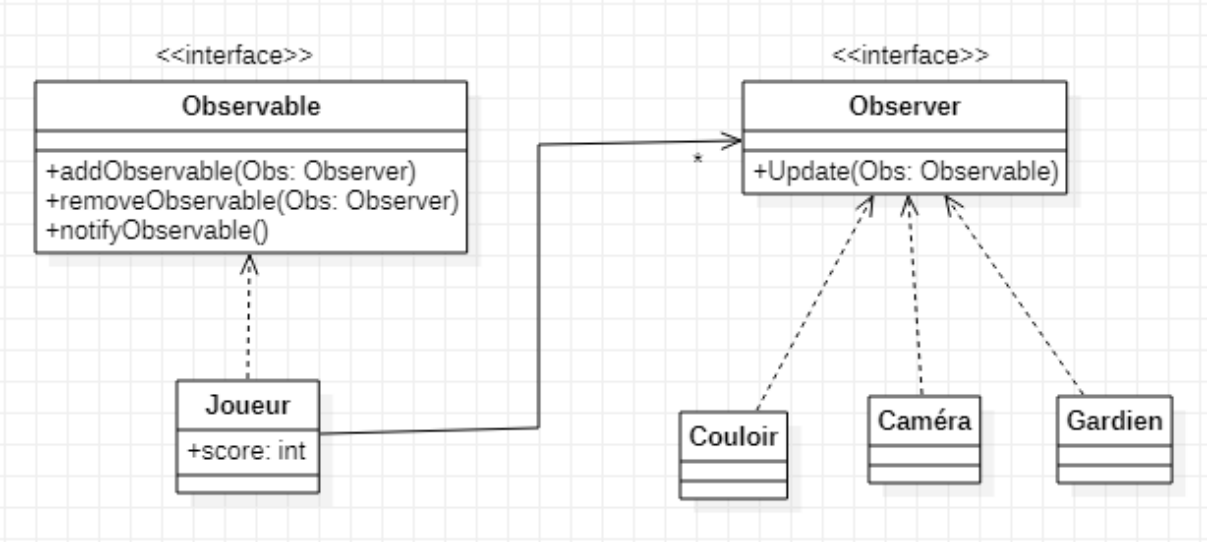
3. On dispose d'un composant implémentant une interface qui définir une opération «traitement()». On voudrait rattacher à ce composant des responsabilités supplémentaires sans modifier son code source. C'est-à-dire envelopper l'exécution de la méthode traitement par d'autres traitements avant et après son exécution.

Le design pattern approprié à cette situation c'est : Decorator



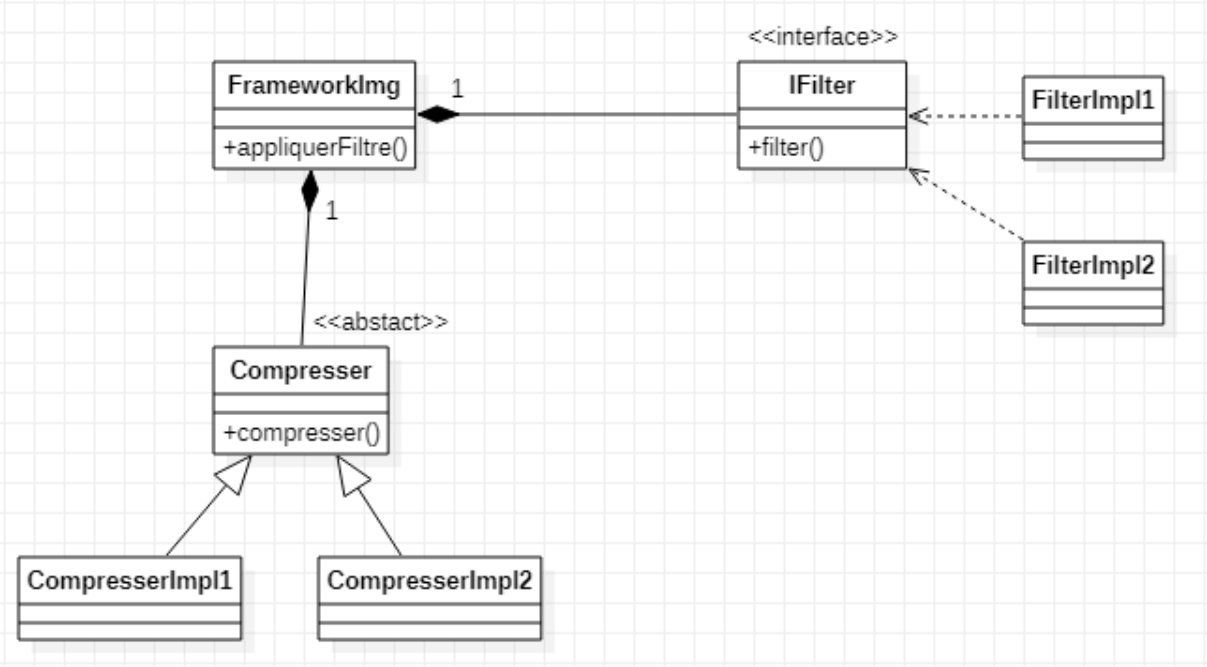
4. On désire créer une classe Joueur ayant un état représenté par une variable score de type int. On voudrait que les objets de l'environnement du jeu (Couloir, Caméra et Gardien) soient informés à chaque fois que le score du joueur change tout en gardant un couplage faible entre la classe Joueur et les autres classes.

Le design pattern approprié à cette situation c'est : Observer



Exercice 2 :

1. Etablir un diagramme de classes de ce Framework en appliquant les design patterns appropriés.



2. Ecrire une implémentation Java de ce Framework.

- La classe « Compresser »

```
Aminamkl
public abstract class Compresser {
    2 usages  2 implementations  Aminamkl
    public abstract int[] compresser (int [] data);
}
```

- Les classes filles

```
Aminamkl
public class CompresserImpl1 extends Compresser {

    2 usages  Aminamkl
    @Override
    public int[] compresser(int[] data) {
        int[] compressedImage = new int [data.length/2];
        for (int i = 0; i < compressedImage.length; i++) {
            compressedImage[i] = data[i];
        }
        System.out.println("\nCompresse using Implementation 1 : ");
        return compressedImage;
    }
}
```

```
Aminamkl
public class CompresserImpl2 extends Compresser {

    2 usages  Aminamkl
    @Override
    public int[] compresser(int[] data) {
        int[] compressedImage = new int [data.length/3];
        for (int i = 0; i < compressedImage.length; i++) {
            compressedImage[i] = data[i];
        }
        System.out.println("\nCompresse using Implementation 2 : ");
        return compressedImage;
    }
}
```

- L’interface « IFilter »

```
1   package filter;
2
3   public interface IFilter {
        2 usages  2 implementations
4       public int[] filter (int [] data);
5   }
6
```

- Les implémentations d’interface

```java
package filter;

import java.util.List;

4 usages
public class FilterImpl1 implements IFilter {
    2 usages
    @Override
    public int[] filter(int[] data) {
        int[] filteredImage = new int[data.length];
        for (int i = 0; i <data.length ; i++) {
            filteredImage[i] = data[i]*45/7;
        }
        System.out.println("\nFilter using Implementation 1 : ");
        return  filteredImage;
    }
}
```

```java
package filter;

public class FilterImpl2 implements IFilter {
    2 usages
    @Override
    public int[] filter(int[] data) {
        int[] filteredImage = new int[data.length];
        for (int i = 0; i <data.length ; i++) {
            filteredImage[i] = data[i]*3-12;
        }
        System.out.println("\nFilter using Implementation 2 : ");
        return  filteredImage;
    }
}
```

- La class « FrameworImg »

```java
2 usages  Aminamkl *
public class FrameworkImg {
    3 usages
    private IFilter iFilter;
    2 usages
    private Compresser Compresser;

    1 usage  Aminamkl *
    public FrameworkImg() {
    }

    2 usages  Aminamkl *
    public int[] appliquerFiltre(int[] data){
        return iFilter.filter(data);
    }
```

```
    4 usages
    public void showImage(int []image)
    {
        for(int i=0;i<image.length;i++){
            System.out.print(image[i]+" ");
        }
    }

    public IFilter getiFilter() {
        return iFilter;
    }

    public void setiFilter(IFilter iFilter) {
        this.iFilter = iFilter;
    }

    2 usages
    public ICompresser getiCompresser() {
        return iCompresser;
    }

    2 usages
    public void setiCompresser(ICompresser iCompresser) {
        this.iCompresser = iCompresser;
    }
}
```

3. Ecrire le code d'une application qui utilise ce Framework en permettant à l'utilisateur de saisir le nom des classes d'implémentation à utiliser pour effectuer les traitements.

```java
public class Main {
    Aminamkl *
    public static void main(String[] args) {
        int [] myimage=new int[10];
        int [] Filterdimage, Compressedimage;
        for (int i=0; i<10; i++){
            myimage[i]=(i+10)*6/2;
        }
        System.out.print("My image is : ");
        for (int i=0; i<10; i++){
            System.out.print(myimage[i] + "\t");
        }

        FrameworkImg framework=new FrameworkImg();

        framework.setiFilter(new FilterImpl1());
        Filterdimage=framework.appliquerFiltre(myimage);
        framework.showImage(Filterdimage);

        framework.setiFilter(new FilterImpl2());
        Filterdimage=framework.appliquerFiltre(myimage);
        framework.showImage(Filterdimage);

        framework.setCompresser(new CompresserImpl1());
        Compressedimage=framework.getCompresser().compresser(myimage);
        framework.showImage(Compressedimage);
```

```java
        framework.setiCompresser(new CompresserImpl2());
        Compressedimage=framework.getiCompresser().compresser(myimage);
        framework.showImage(Compressedimage);


    }
}
```

- Le résultat d'exécution :

```
Main
"C:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
My image is : 30    33  36  39  42  45  48  51  54  57
Filter using Implementation 1 :
192 212 231 250 270 289 308 327 347 366
Filter using Implementation 2 :
78 87 96 105 114 123 132 141 150 159
Compresse using Implementation 1 :
30 33 36 39 42
Compresse using Implementation 2 :
30 33 36
Process finished with exit code 0
```