Les patrons de *responsabilités*

Exercice 1 – *Filtrage d'un texte*

On s'intéresse à l'implémentation d'un programme permettant d'appliquer un filtrage configurable sur un flux texte. Le programme suivant ouvre le fichier dont le nom est passé en paramètre, puis applique un filtre sur chacune des lignes avant de l'envoyer sur sa sortie standard :

```
import java.io.*;

public class FilterTest {
        public static IFilter test1() {
                return ...
        }

        public static void main(String[] args) throws IOException {
                BufferedReader in = new BufferedReader(
                        new InputStreamReader(
                                new FileInputStream(args[0])));
                String line;

                // Get the Filter to apply on each line :
                IFilter f=test1();      // Test 1

                // Process each line :
                while ((line = in.readLine()) != null) {
                        System.out.println(f.filter(line)); // <- To be filtered !
                }

        }
}
```

On souhaite pouvoir disposer des filtres suivants :

- Trim : émonde (supprime les caractères d'espacement en début et fin de ligne),
- UpperCase : convertit chaque caractère d'une ligne en majuscule,
- Replace : remplace toutes les occurrences d'un motif représenté par une expression régulière par un autre,
- Numberize : numérote les lignes (le numéro sera précédé d'une tabulation),
- SkipFilter(p ,s) : traite `p` lignes puis ignore `s` ligne et recommence.
- ...

Chaque filtre implémente l'interface :

```
public interface IFilter {
        public String filter(String is);
}
```

L'objectif est de pouvoir combiner ces filtres entre-eux. Par exemple :

Capitaliser deux lignes sur cinq.
ou
Numéroter les lignes et remplacer toutes les occurrences de `public` par `private`

Question 1.1 : Quel patron de conception permet de réaliser ce système ?
Question 1.2 : Donner le diagramme de classes de principe de ce patron.
Question 1.3 : Proposez un diagramme de votre solution (en faisant apparaître les rôles du patron).
Question 1.4 : Créez testez des fonctions `testX()` permettant de produire les filtres suivants :

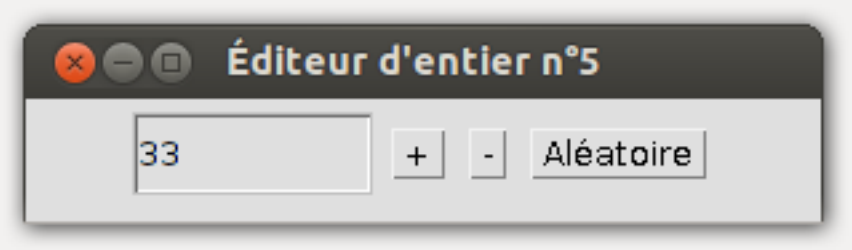
- `test1()` : numéroter une ligne sur cinq,
- `test2()` : émonder et convertir en majuscule,
- `test3()` : numéroter toutes les lignes, émonder une ligne sur deux, et remplacer toutes les occurences de `=` par `<--`.

Exercice 2 – *Vues et modèles*

On dispose d'une classe modèle `IntModel` contenant un entier et implémentant les opérations de l'interface :

```
interface IIntModel {
        public void inc();
        public void dec();
        public void random();
        public int getValue();
}
```

On dispose d'une vue implémentée par la classe `IntEditor` qui crée la boite d'édition suivante :



et qui implémente les opérations suivantes :

```
protected void showValue(int val);
protected void setIncListener(ActionListener al);
protected void setDecListener(ActionListener al);
protected void setRandomListener(ActionListener al);
```

On souhaite pouvoir utiliser un nombre variable d'éditeurs d'entier tels que leur vue reste synchronisée sur l'état du modèle.
Question 2.1 : Quel patron permet de réaliser cela ? Donner le diagramme de principe.
Question 2.2 : Donnez le diagramme de votre solution sachant qu'il n'est pas permis de modifier le modèle ni la vue existante.
Question 2.3 : Écrivez un programme de test qui instancie un tableau de cinq vues connectées à un modèle et montrez que chaque opération sur une vue eset répercutée sur les autres.