

## TD Simulation d'afficheurs lumineux

Le but de cet exercice est de simuler en Java les afficheurs lumineux qu'on voit un peu partout et qui font circuler un texte en boucle. On réutilisera le travail fait sur les files FIFO.

### Exercice 1 : Les afficheurs lumineux

On s'intéresse maintenant aux afficheurs lumineux.

Ces objets sont en charge de l'affichage d'un message d'une longueur  $l$  non nulle. Ce message déroule en boucle sur un écran de largeur  $L (\neq 0)$  qui peut être plus petit ou plus grand que  $l$ .

Un afficheur est donc initialement défini par la largeur  $L$  de son écran.

Le message défile dans l'afficheur (en fait sur l'écran) en se décalant d'une position vers la gauche à chaque top d'une "horloge". Le message défile ainsi en boucle, quand le dernier caractère du message vient d'entrer dans la partie affichée, au prochain top horloge on reprend le message au début et c'est le premier caractère du message qui y entre à son tour, et ainsi de suite.

Le code de la classe `Afficheur` pourrait ressembler à :

```
public class Afficheur {
    ...
    // fixe un nouveau message a afficher
    public void setMessage(String message) {...}

    // appelle a chaque top d'horloge, decale le message vers la gauche de l'ecran
    public void decale() {...}

    // renvoie ce qui apparait a l'ecran (ce qui est affiche)
    public String toString(){...}
}
```

Pour fixer les idées, soit la classe ci-dessous, l'invocation `new Horloge(new Afficheur(5)).tester(8);` produit alors la trace de droite.

```
public class Horloge {
    private Afficheur afficheur;
    public Horloge(Afficheur a) { this.afficheur = a; }
    public void tester(int nbTop) {
        String message = "abcd";
        this.afficheur.setMessage(message);
        for (int i=0 ; i<nbTop ; i++) {
            this.afficheur.decale();
            System.out.println("<<"+this.afficheur+">>");
        }
    }
}
```

```
<<  a>>
<<  ab>>
<< abc>>
<< abcd>>
<<abcda>>
<<bcdab>>
<<cdabc>>
<<dabcd>>
```

**Q 1 .** Quel est le lien avec l'exercice sur les files FIFO ?

**Q 2 .** Complétez le code de la classe `Afficheur`.

## 1 Les afficheurs avec latence

On remarque que pour les afficheurs de l'exercice précédent il est difficile de voir où se termine le message. Pour éviter ce problème, on veut une nouvelle classe d'afficheurs pour lesquels on pourra spécifier lors de leur création un "*temps de latence*" entre l'entrée du dernier et celle du premier caractère. Ce temps de latence sera exprimé par un nombre positif ou nul d'espaces à insérer entre ces deux caractères. Appelons **Latence** cette nouvelle classe d'afficheurs.

Avec l'invocation `new Horloge(new Latence(5,3)).tester(8);` on crée un afficheur avec une latence de trois espaces et on le teste, ce qui donne :

```
<<  a>>
<<  ab>>
<<  abc>>
<< abcd>>
<<abcd >>
<<bcd  >>
<<cd   >>
<<d    a>>
```

**Q 3 .** Définissez la classe `Latence`.

## 2 Les afficheurs avec latence et vitesse paramétrable

A chaque top d'horloge, les afficheurs précédents font un seul décalage. On voudrait une nouvelle sorte d'afficheur dont on pourrait fixer le nombre de décalages effectués à chaque top. Ce nombre sera un entier positif ou nul. Appelons `Vitesse` cette nouvelle classe d'afficheurs.

**Q 4 .** Définissez la classe `Vitesse` et testez là.