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.

Exercice 1 : Les afficheurs lumineux

Un *afficheur lumineux* (*led displayer*) est un objet en charge de l'affichage d'un message d'une longueur l non nulle. Ce message s'affiche sur un écran de largeur $L(\neq 0)$ qui peut être plus petit ou plus grand que l. L'écran n'affiche donc éventuellement qu'une portion du message.
Un afficheur est donc initialement défini par la largeur L de son écran, le message étant vide.

Il est possible de modifier le texte de l'afficheur (celui qui appparaît sur l'écran) en décalant (*shift*) d'une position vers la gauche le texte du message qui apparaît à l'écran. Quand le dernier caractère du message vient d'entrer dans la partie affichée, au prochain décalage 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 `LedDisplayer` pourrait donc ressembler à :

```
public class LedDisplayer {
   ...
   /** sets the new message to display
   * @param message the new message
   */
   public void setMessage(String message) {...}

   /** shift message by one character on the left */
   public void shift() {...}

   /** returns the text that appears on the screen
   * @return the text that appears on the screen
   */
   public String textOnScreen(){...}
}
```

On peut ensuite utiliser une "horloge" pour cadencer les décalages : à chaque top de l'horloge on fait un appel à `shift()`. Le message défile ainsi en boucle.
Pour fixer les idées, soit la classe `Clock` ci-dessous, l'invocation `new Clock(new LedDisplayer(5)).tryIt(8);` produit alors la trace de droite.

```
public class Clock {
   private LedDisplayer displayer;
   public Clock(LedDisplayer a) { this.displayer = a; }
   public void tryIt(int nbTop) {
      String message = "Abcd";
      this.displayer.setMessage(message);
      for (int i=0 ; i<nbTop ; i++) {
         this.displayer.shift();
         System.out.println("|"+this.displayer.textOnScreen()+"|");
      }
   }
}
```

```
|    A|
|   Ab|
|  Abc|
| Abcd|
|AbcdA|
|bcdAb|
|bdAbc|
|dAbcd|
```

Les afficheurs « simples »

Q 1 . Complétez le code de la classe `LedDisplayer`.

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 `DisplayerWithLatency` cette nouvelle classe d'afficheurs.

Avec l'invocation `new Clock(new DisplayerWithLatency(5,3)).tryIt(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 2 . Définissez la classe `DisplayerWithLatency`.

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 `SpeedDisplayer` cette nouvelle classe d'afficheurs.

Q 3 . Définissez la classe `SpeedDisplayer`.