Un éditeur orienté ligne

L'objectif de cet exercice est de construire un mini-éditeur orienté ligne qui respecte les commandes de `vi`. On veut qu'il propose un menu textuel donnant accès aux différentes commandes disponibles sur l'éditeur.

Pour simplifier, nous ferons les suppositions suivantes :

1. Nous nous limiterons à l'édition d'une seule ligne.
2. L'interface utilisateur est minimale. Elle contient :
   - le menu listant les opérations que peut réaliser l'utilisateur. L'utilisateur sélectionne une opération en tapant au clavier son numéro dans le menu.
   - la ligne en cours d'édition qui est affichée après chaque réalisation d'une opération par l'utilisateur. Le curseur est matérialisé par deux crochets encadrant le caractère courant. Dans le cas où la ligne est vide, le caractère tilde (~) est affiché et, bien sûr, le curseur n'est pas matérialisé.

```
Je suis la ligne [e]n cours d'édition !
-------------------------------------------------------
 1) Ajouter un caractère au début de la ligne   [I]
 2) Ajouter un caractère à la fin de la ligne   [A]
 3) Placer le curseur au début de la ligne      [0]
 4) Avancer le curseur d'une position à droite  [l]
 5) Reculer le curseur d'une position à gauche  [h]
 6) Remplacer le caractère sous le curseur      [r]
 7) Supprimer le caractère sous le curseur      [x]
 8) Ajouter un caractère avant le curseur       [i]
 9) Ajouter un caractère après le curseur       [a]
10) Supprimer tous les caractères de la ligne  [dd]
 0) Quitter
-------------------------------------------------------
Votre choix :
```

Il est reconnu que pour qu'une interface homme–machine soit conviviale, il faut que les opérations impossibles à réaliser ne puissent pas être sélectionnées par l'utilisateur. Par exemple, si le curseur est sur le dernier caractère de la ligne, il ne doit pas pouvoir sélectionner l'opération qui consiste à avancer le curseur. Généralement, les entrées des menus ne pouvant pas être exécutées sont grisées. Dans notre cas, nous supprimerons le numéro d'accès.

```
~
-------------------------------------------------------
 1) Ajouter un caractère au début de la ligne   [I]
 2) Ajouter un caractère à la fin de la ligne   [A]
 -) Placer le curseur au début de la ligne      [0]
 -) Avancer le curseur d'une position à droite  [l]
 -) Reculer le curseur d'une position à gauche  [h]
 -) Remplacer le caractère sous le curseur      [r]
 -) Supprimer le caractère sous le curseur      [x]
 -) Ajouter un caractère avant le curseur       [i]
 -) Ajouter un caractère après le curseur       [a]
10) Supprimer tous les caractères de la ligne  [dd]
 0) Quitter
-------------------------------------------------------
Votre choix :
```

Dans la suite, nous définissons la notion de ligne, puis nous proposons une modélisation de l'éditeur orienté ligne en définissant des menus textuels réutilisables qui prennent en compte la notion d'opération non réalisables.

Exercice 1

La spécification d'une ligne vous est donnée dans l'interface `Line`.

Une ligne contient un nombre quelconque de caractères. Il est possible de rajouter un caractère au début ou à la fin

de la ligne. La ligne définit un curseur qui peut être déplacé vers la droite (avancer) ou vers la gauche (reculer). Le caractère sous le curseur est appelé caractère courant. Il est possible d'effectuer des opérations relatives au curseur (remplacer le caractère courant, le supprimer, insérer un caractère avant ou après le caractère courant).

On décide de stocker les caractères de la ligne dans un tableau et de représenter la position du curseur par un entier. Écrire une réalisation (concrétisation) `LineTab` de l'interface `Line`.

Exercice 2

Réfléchir à une manière de concevoir l'éditeur capable de prendre en compte toutes les contraintes citées précédemment.

Commencer par s'intéresser au concept de commande.

- Le concept de commande (tout comme celui de menu) doit être indépendant de celui de ligne.
- On doit pouvoir questionner une commande sur sa capacité à être exécutée.
- On doit pouvoir exécuter une commande.

On s'intéresse ensuite au concept de menu à entrées.

- Un menu peut être affiché, c'est à dire afficher l'ensemble de ses entrées.
- On peut lui ajouter une entrée.
- Chaque entrée est associée à une commande.
- Il doit permettre de proposer une sélection conviviale à l'utilisateur.
- Et permettre d'executer ou non la commande associée à la sélection.
- Rappel : les menus doivent rester indépendants d'une application particulière, et donc de l'éditeur orienté ligne.

Finalement, on s'intéresse au concept d'éditeur. Celui-ci est composé d'un menu et permet d'éditer une ligne tant que l'utilisateur ne choisis pas de quitter l'éditeur. Il faudra donc penser à une manière de quitter celui-ci à travers le menu.

Exercice 3

Pour organiser le grand nombre de commandes proposées par une application, il est utile de pouvoir organiser un menu en sous-menus. Par exemple, on décide de réorganiser le menu en regroupant dans un sous-menu spécifique les opérations relatives au curseur.

1. Proposer une manière de réaliser cette notion de sous-menu.
2. Deux comportement sont possible pour le sous-menu :
    1. soit le sous-menu propose la sélection d'une seule opération et il disparaît ;
    2. soit il reste actif jusqu'à ce qu'il soit quitté explicitement par l'utilisateur.

Exercice 4

Il est souvent plus pratique d'accéder à une entrée d'une menu par un raccourci. Le raccourci est généralement plus facile à retenir que le numéro de l'entrée et, dans el cas d'une menu graphique, il permet de conserver les mains sur le clavier et donc d'être plus efficace.

Modifier les menus pour donner la possibilité de préciser un raccourci lorsqu'une nouvelle entrée est ajoutée à un menu. On considère qu'un raccourci ne peut donner accès qu'à une entrée du menu en cours.

(Pensez à bien découper votre travail, faire plusieurs interfaces et plusieurs classes)