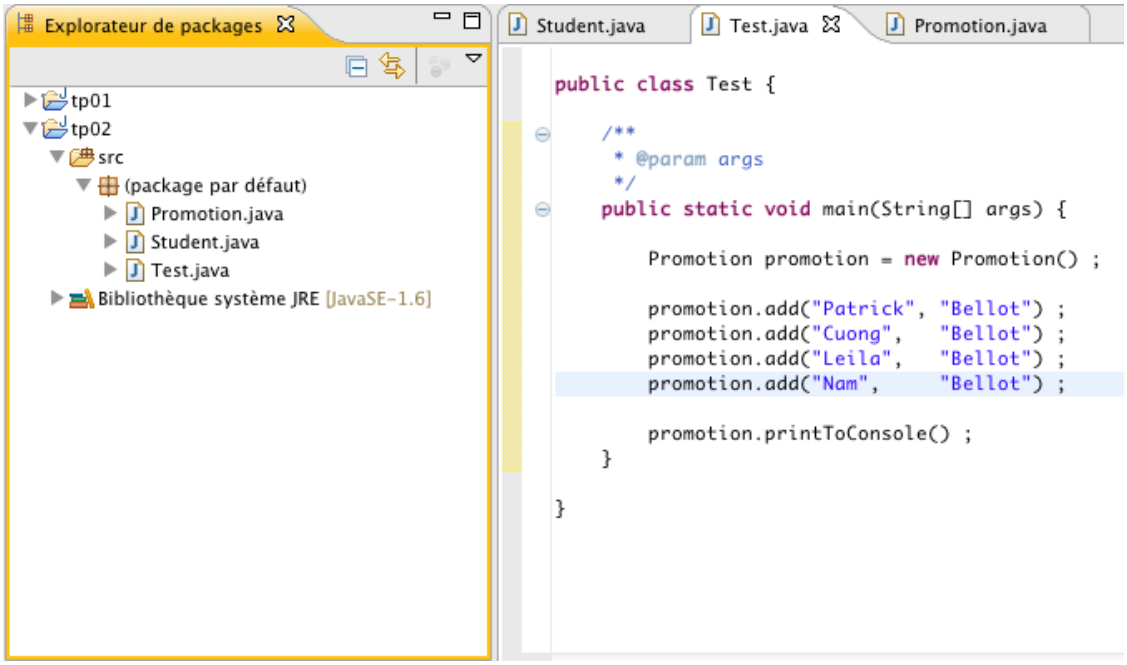
TP Modéliser et trier une promotion d'étudiants

Dans ce TP, nous allons apprendre à programmer les méthodes de classe en codant un algorithme de tri d'étudiants par ordre alphabétique.

Partie 2 : Trier une promotion d'étudiants

Lancer Eclipse à partir des menus de votre environnement Lunix ou Windows et ouvrir le projet tp02.



Exécutez le programme pour vérifier que tout fonctionne toujours bien.

Nous allons trier une promotion, c'est-à-dire ordonner les élèves du premier au dernier selon un critère permettant de comparer deux élèves afin de définir leur **ordre**.

Comparer deux élèves

La comparaison de deux élèves utilisera l'**ordre alphabétique** de leurs noms, prénoms et identifiants. Pour comparer deux élèves, on commence par comparer leurs noms de famille (`lastName`) :

1. Si le nom de famille du premier élève précède celui du deuxième élève dans l'ordre alphabétique, le premier élève sera considéré comme étant << plus petit >> que le second.
2. Si le nom de famille du premier élève arrive après celui du deuxième élève dans l'ordre alphabétique, le premier élève sera considéré comme étant << plus grand >> que le second.
3. Si les noms de famille des deux élèves sont égaux, on compare alors leurs prénoms (`firstName`)

3.1. Si le prénom du premier élève précède le prénom du deuxième élève dans l'ordre alphabétique, le premier élève sera considéré comme étant << plus petit >> que le second.
3.2. Si le prénom du premier élève arrive après celui du deuxième élève dans l'ordre alphabétique, le premier élève sera considéré comme étant << plus grand >> que le second.
3.3. Si les prénoms des deux élèves sont égaux, on compare alors les identifiants (**`id`**) qui sont des nombres entiers :
    3.3.1. Si l'identifiant du premier élève est strictement plus petit que l'identifiant du deuxième élève, le premier élève sera considéré comme étant << plus petit >> que le second.
    3.3.2. Si l'identifiant du premier élève est strictement plus grand que l'identifiant du deuxième élève le premier élève sera considéré comme étant << plus grand >> que le second.
    3.3.3. Chaque élève ayant un identifiant unique, les identifiants des deux élèves ne pourront jamais être égaux.

Pour comparer deux élèves, nous devrons être capable de comparer des chaînes de caractères (**`String`**) et des nombres entiers (**`int`**). La comparaison de nombres entiers est immédiate avec les opérateurs **`<`**, **`<=`**, **`>`**, **`>=`** et **`==`**. Mais rappelons que ces opérateurs ne fonctionnent qu'avec les types scalaires tel que vu en cours.

Comparer deux chaînes de caractères

Pour savoir comment comparer deux chaînes de caractères, allons voir la Javadoc de la classe **`String`** sur le Web :

- Ouvrez un navigateur sur la page de votre moteur de recherche préféré.
- Rechercher **Java SE String**.
- Trouver le lien sur la Javadoc de la classe **`String`**.

C'est la méthode **`int compareTo(String anotherString)`** de la classe **`String`** qui permet de comparer deux chaînes de caractères. **Lire la documentation de cette méthode.**

En résumé, si **`string1`** et **`string2`** sont deux objets de type **`String`**, la méthode **`int compareTo(String anotherString)`** retournera :

- Un entier strictement négatif si **`string1`** est strictement plus petite que **`string2`**
- Zéro si **`string1`** est égale à **`string2`**
- Un entier strictement positif si **`string1`** strictement plus grande que **`string2`**

Une méthode pour comparer deux élèves

Dans la classe **`Student`**, écrire une méthode :

```
int compareTo(Student anotherStudent)
```

Cette méthode permettra de comparer deux élèves sur le même modèle que la méthode **`compareTo`** de la classe **`String`**.

Comparez votre programme avec celui qui est proposé ici.

Mise en garde !!!!

Attention ! Les algorithmes sont souvent présentés en supposant que les éléments d'une liste ou d'un tableau contenant **`n`** éléments sont indiçés par des valeurs comprises entre **`1`** et **`n`**. Or, dans la plupart des langages de programmation, et en Java en particulier, les éléments sont indicés par des valeurs comprises **`0`** et **`n-1`**. Il faudra donc tenir compte de cela lorsque vous transcrirez ces algorithmes en Java.

L'algorithme de tri sélection

Maintenant que nous savons comparer deux élèves, nous allons pouvoir trier les élèves d'une promotion. Ceux-ci sont stockés dans l'attribut **`studentList`** de la classe **`Promotion`**. Rappelons l'algorithme du **tri sélection** :

Tri sélection

Les données sont rangées dans un tableau T entre les indices 1 et n.

On utilise trois variables entières notées i, j et *indicePetit*, et une variable *min* du même type que les données de la liste. Pour chaque itération de la boucle portant sur i, on cherche le plus petit élément de la liste qui se trouve entre les indices i et n.

- pour i qui varie de 1 à $n - 1$;
  - *indicePetit* ← i ;
  - *min* ← $T[i]$;
  - pour j qui varie de $i + 1$ à n, faire
    - si $T[j] < min$
      - *indicePetit* ← j ;
      - *min* ← $T[j]$;
  - *échanger*(T, i, *indicePetit*).

N.B. Nous allons travailler sur la variable **`studentList`** qui est de type **`ArrayList<Student>`**. Vous devriez donc avoir sous les yeux le Javadoc de la classe **`ArrayList`**.

En globish, << échanger >> se dit << swap >>. Ecrivez une méthode :

```
private void swap(int i, int j)
```

dans la classe **`Promotion`** qui échange les élèves d'indices **`i`** et **`j`** dans **`studentList`**.

Comparez votre programme avec celui montré ici.

En globish, << tri >> se dit << sort >>. Ecrivez une méthode :

```
public void selectionSort()
```

dans la classe **`Promotion`**. Cette méthode appliquera l'algorithme du **tri sélection** décrit plus haut. Elle utilisera bien sûr la méthode **`swap`** que vous venez de réaliser.

Comparer votre programme avec celui montré ici.

Dans la méthode **`main`** de la classe **`Test`**, utilisez la méthode **`selectionSort()`** pour trier la promotion avant de l'afficher.

Exécuter le programme. Contrôler ici.