TP 3b : Trier les robots de l'usine

Dominique Blouin, Télécom Paris, Institut Polytechnique de Paris
dominique.blouin@telecom-paris.fr

Dans ce TP, nous allons apprendre à programmer les méthodes de classe en codant un algorithme de tri de robots par ordre alphabétique de leur nom.

Trier les robots

Lors du dernier TP, vous avez créé une classe *Factory* qui contenait un attribut contenant une liste de robots. Puis, vous avez créé une méthode addRobot() permettant d'instancier un robot en lui donnant le nom passé en paramètre de la méthode. Dans cet exercice, nous allons ajouter une méthode à la classe *Factory* permettant de trier les robots, c'est-à-dire ordonner les robots du premier au dernier selon un critère permettant de comparer deux robots afin de définir leur **ordre**.

Comparer deux robots

La comparaison de deux robots utilisera l'**ordre alphabétique** de leurs noms. Pour comparer deux robots, on commence par comparer leurs noms :

Si le nom du premier robot précède celui du deuxième robot dans l'ordre alphabétique, le premier robot sera considéré comme étant << plus petit >> que le second. Si les noms des deux robots sont égaux, on comparera alors leurs vitesse (*speed*). Si la vitesse du premier robot est strictement plus petite que la vitesse du deuxième robot, le premier robot sera considéré comme étant << plus petit >> que le second.

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

Comment 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 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*

Créer une méthode pour comparer deux robots

Dans la classe *Robot*, écrire une méthode :

```
int compareTo(Robot anotherRobot)
```

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

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 indicés 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 entre *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 robots, nous allons pouvoir trier les robots d'une usine (*Factory*). Ceux-ci sont stockés dans l'attribut *robots* de la classe *Factory*. 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 *robots* qui est de type *ArrayList<Robot>*. Vous devriez donc avoir sous les yeux le Javadoc de la classe *ArrayList*.

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

```
private void swap(int index1, int index2)
```

dans la classe *Factory* qui échange les positions des robots d'indices *index1* et *index2* dans la liste des *robots*.

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

```
public void selectionSort()
```

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

Dans la méthode *main* de la classe *TestRobotSim* que vous avez créée au TP précédent, utilisez la méthode *selectionSort()* pour trier les robots avant de les afficher à la console.

Exécuter le programme et vérifier que les robots s'affichent bel et bien dans le bon ordre.