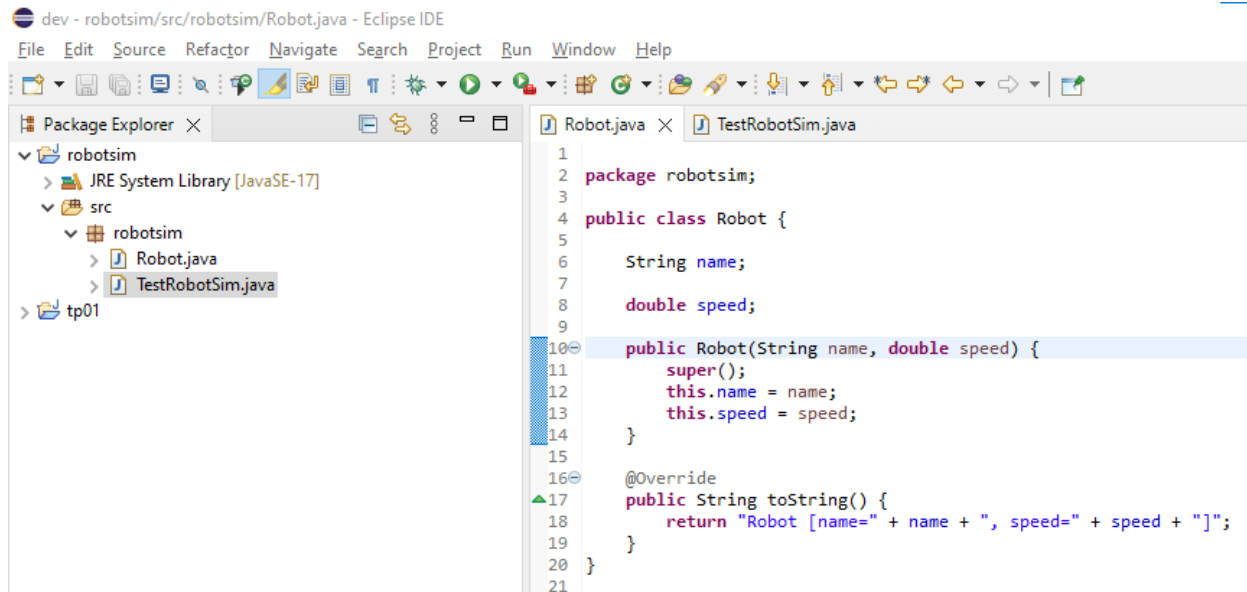


TP 3 : Modéliser une usine contenant des robots

Pour ce TP, vous allez continuer à travailler dans le même projet que celui du TP précédent, (nommé *robotsim*), et qui à terme, contiendra toutes les sources votre projet de développement. Ce projet sera à rendre en fin de cours.

Vous allez donc rajouter une nouvelle classe nommée *Factory* pour modéliser une usine de production de biens. Lancez Eclipse pour travailler sur le projet *robotsim*.



```
dev - robotsim/src/robotsim/Robot.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
  robotsim
    JRE System Library [JavaSE-17]
    src
      robotsim
        Robot.java
        TestRobotSim.java
    tp01

Robot.java
1
2 package robotsim;
3
4 public class Robot {
5
6     String name;
7
8     double speed;
9
10    public Robot(String name, double speed) {
11        super();
12        this.name = name;
13        this.speed = speed;
14    }
15
16    @Override
17    public String toString() {
18        return "Robot [name=" + name + ", speed=" + speed + "]";
19    }
20 }
21
```

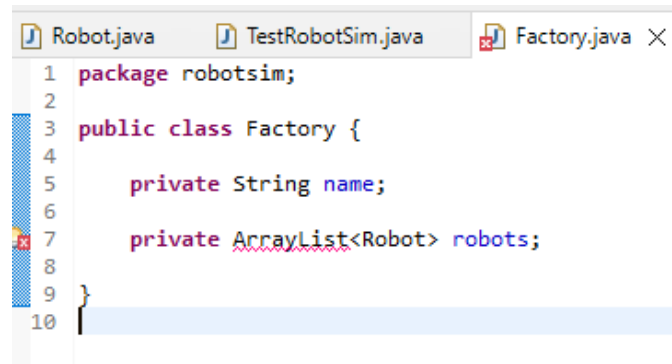
Encapsuler les données de la classe Robot

Nous avons vu en classe l'importance d'encapsuler les données. Celle-ci s'effectue en utilisant le mot clé de visibilité *private* en Java. Modifiez la visibilité des attributs de votre classe robot puis générer les accesseurs pour ces attributs, en considérant que le nom d'un robot ne pourra pas changer au cours de son existence, ce qui n'est pas le cas pour sa vitesse. Tout comme pour la génération de la méthode *toString()* du TP précédent, l'IDE peut automatiquement générer ces accesseurs pour vous.

Créer une classe Factory

En faisant un clic droit sur le package *robotsim* puis en sélectionnant le menu *New>>Class*, créer une classe nommée *Factory*.

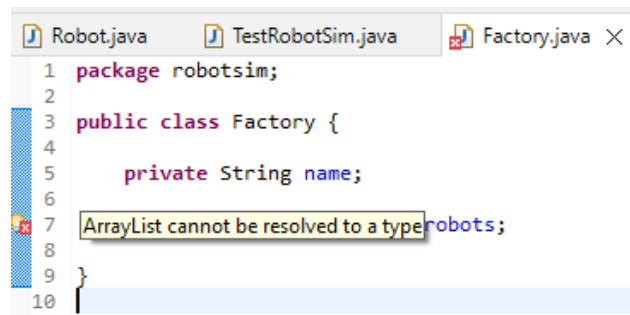
Une usine ayant un nom et devant contenir plusieurs robots, déclarer dans cette classe un attribut *name* comme pour votre classe *Robot* et un attribut nommé *robots* de type *ArrayList<Robot>* pour contenir les robots de l'usine. N'oubliez pas d'encapsuler ces attributs en utilisant le qualificateur *private*. Vous devriez obtenir ceci :



```
1 package robotsim;
2
3 public class Factory {
4
5     private String name;
6
7     private ArrayList<Robot> robots;
8
9 }
10
```

Déclaration d'importation

Une marque rouge indiquant une erreur est apparue dans la marge gauche de l'éditeur de code. En survolant ce marqueur avec la souris, l'erreur s'affiche :



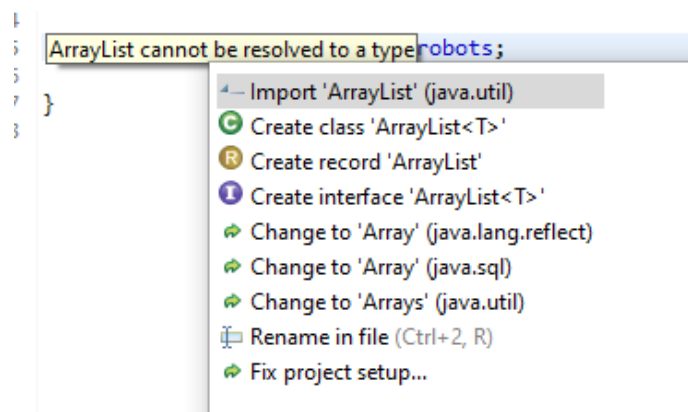
```
1 package robotsim;
2
3 public class Factory {
4
5     private String name;
6
7     private ArrayList<Robot> robots;
8
9 }
10
```

ArrayList cannot be resolved to a type

En effet, la classe *ArrayList* n'est pas connue par défaut. Il faut alors, tel que vu en cours, importer cette classe grâce à la déclaration :

import java.util.ArrayList;

Cette déclaration doit être placée au début du fichier de la classe. Ici encore, l'IDE peut vous aider. En cliquant sur la marque rouge d'erreur, l'IDE vous proposera différentes solutions possibles :



```
1
2
3
4
5
6
7 }
8
9 }
```

ArrayList cannot be resolved to a type

- Import 'ArrayList' (java.util)
- Create class 'ArrayList<T>'
- Create record 'ArrayList'
- Create interface 'ArrayList<T>'
- Change to 'Array' (java.lang.reflect)
- Change to 'Array' (java.sql)
- Change to 'Arrays' (java.util)
- Rename in file (Ctrl+2, R)
- Fix project setup...

La première solution est bien sûr la bonne. Déplacer la souris et cliquer sur *Import ArrayList*. La déclaration d'importation est alors automatiquement ajoutée à la classe.

Écrire le constructeur

Comme nous l'avons vu en cours, par défaut l'attribut *robots* est initialisé avec la valeur *null*. Il faut spécifier un constructeur afin d'initialiser les attributs d'une classe avec des valeurs convenables. Consulter la Javadoc de la classe *ArrayList* pour connaître les constructeurs de cette classe. Écrire un constructeur pour votre classe *Factory*.

Ajouter un robot à l'usine

Ecrire une méthode dans la classe *Factory* qui permettra d'ajouter des robots à l'usine. Elle aura la signature suivante :

```
public boolean addRobot(String name)
```

Cette méthode devra d'abord vérifier que le nom du nouveau robot nommé *name* est unique parmi les noms des robots qui ont déjà été ajoutés à l'usine de production. Si le nom est unique, alors le robot sera ajouté à l'usine et la méthode retournera la valeur booléenne *vraie*. Dans le cas contraire, le robot ne sera pas ajouté à l'usine et la valeur booléenne *faux* sera retournée. La vitesse du robot créé sera de 0.0.

Vérifier l'unicité des noms des robots

Ecrire une méthode dans la classe *Factory* qui vérifiera l'unicité d'un nom de robot passé en paramètre. Cette méthode, qui sera appelée par la méthode *addRobot*, aura la signature :

```
private boolean checkRobotName(String name)
```

Elle devra parcourir la liste des robots pour vérifier qu'aucun d'entre eux n'a le même nom que celui passé en paramètre.

Afficher l'usine et ses robots à la console

Ecrire une méthode dans la classe *Factory* ayant la signature suivante :

```
public void printToConsole()
```

Cette méthode affichera le nom de l'usine ainsi que la liste de ses robots à la *Console*.

Tester les méthodes

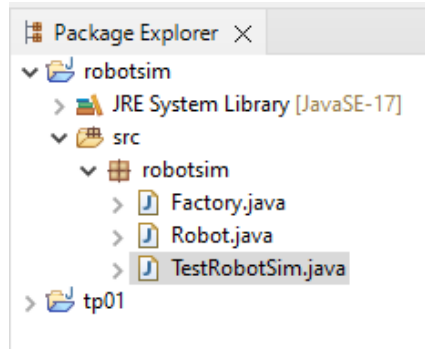
Dans la méthode *main* de la classe *TestRobotSim* créée au TP précédent, ajouter des instructions qui permettront de :

- Créer un objet de la classe *Factory*.
- Ajouter quelques robots à cet objet. Faites différents essais avec quelques robots de noms identiques afin de vérifier que votre programme fonctionne correctement.
- Afficher l'objet de la classe *Factory* à la *Console*.

Exécuter votre programme avec différents jeux de robots et vérifier que ce qui s'affiche à la console est correct.

Organiser les classes en packages

Vous avez sans doute remarqué lorsque vous avez créé les classes *Robot* et *Factory* avec Eclipse que celui-ci les a mises dans un package nommé *robotsim*, dont le nom est le même que celui du projet.

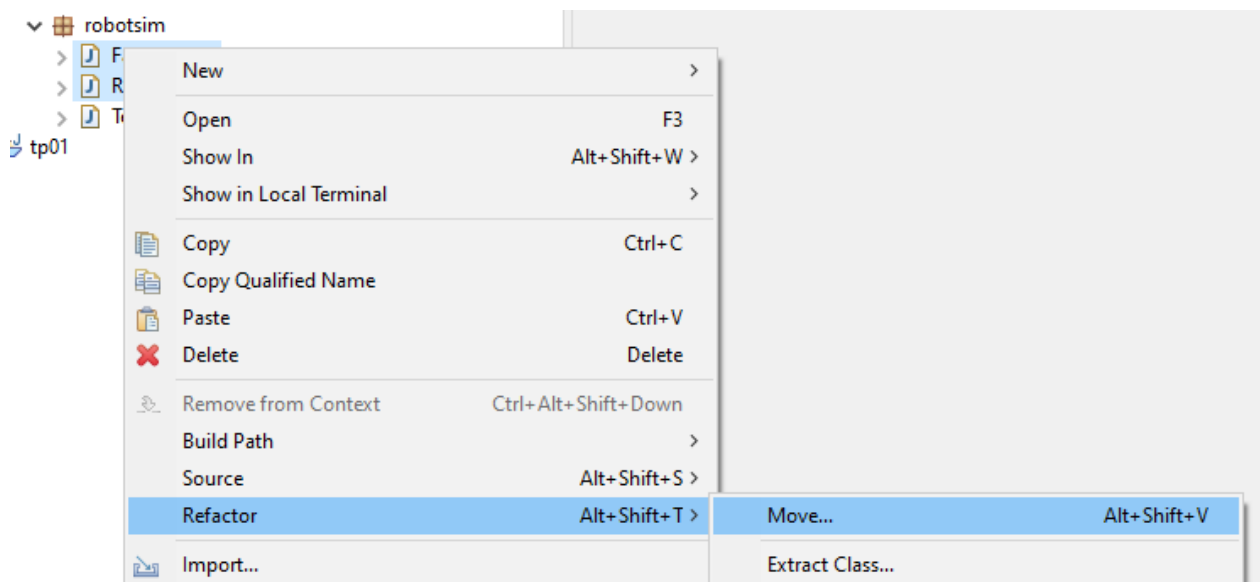


Cette organisation des classes n'est pas idéale car tel que vu en classe, il est préférable de regrouper les classes réalisant une fonctionnalité commune de l'application au sein d'un même package.

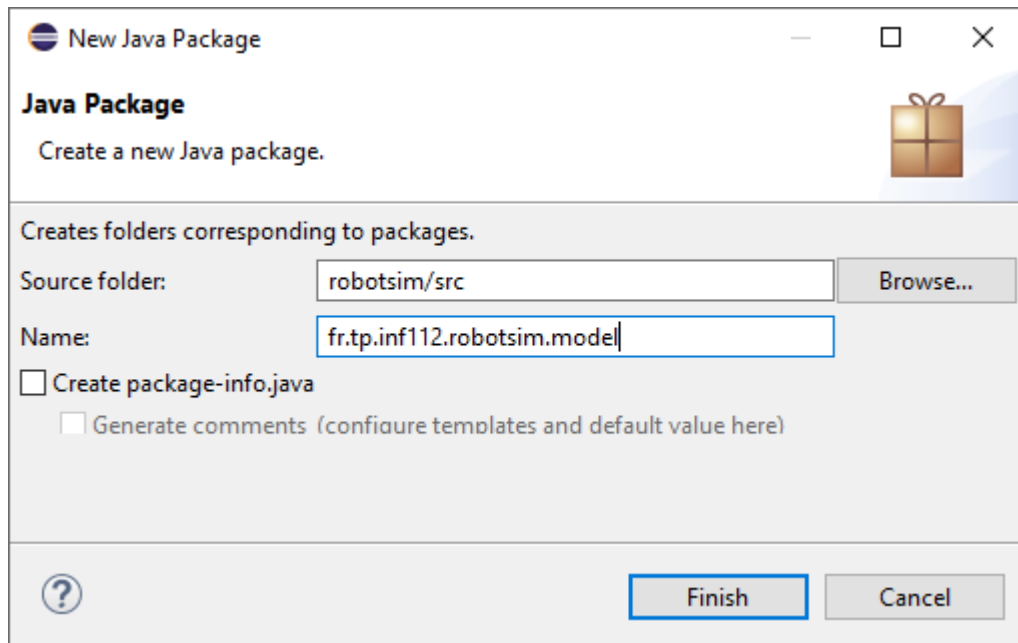
Les classes *Factory* et *Robot* réalisent la fonction de *modélisation* de l'usine de production. Elles seront donc regroupées au sein d'un package nommé *fr.tp.inf112.robotsim.model*. Qu'en est-il de la classe *TestRobotSim* ?

Pour changer le package d'une classe, il ne suffit pas de renommer la déclaration du package dans la classe. En effet, Java impose que le fichier de la classe soit localisé dans une arborescence de dossiers du système de fichier équivalente au nom du package de la classe, après conversion des caractères « . » du nom du package en caractères « / ».

Encore une fois, l'IDE pourra automatiquement changer la déclaration de package de la classe et déplacer son fichier dans le bon sous-répertoire correspondant. Pour ce faire, sélectionner le (ou les) classe(s) dans l'explorateur de package et faire un clic-droit. Dans le menu qui s'affiche, sélectionner *Refactor* > > *Move...*



Dans la boîte de dialogue qui s'affiche, saisir le nom de package souhaité et cliquer sur *Finish*.



N'oubliez pas de changer également le package de la classe *TestRobotSim*.

Relancer la classe *TestRobotSim* afin de vérifier que votre programme fonctionne toujours correctement après la réorganisation de vos classes.