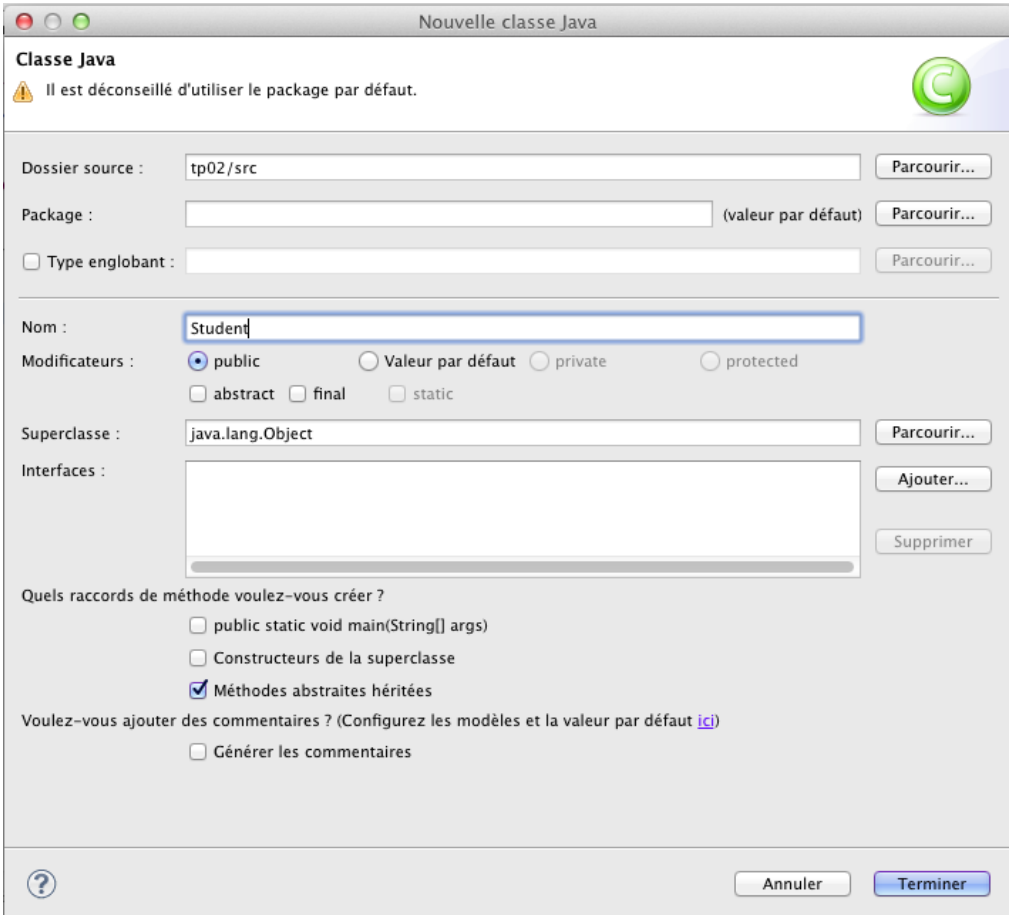
TP : Programmer une classe Student

Dans ce TP, nous allons apprendre à coder une première classe que nous nommerons *Student* et qui nous servira à modéliser une promotion d'étudiants. Cette classe sera également utilisée dans le prochain TP.

Parce que les logiciels connaissent mieux la langue anglaise que les autres, il peut être compliqué de travailler avec des noms de classe, de méthodes ou d'attributs écrits en français. C'est pourquoi nous utiliserons des mots anglais, ou plutôt globish. Internet est une référence pour la programmation en Java. Vous y trouverez des exemples de programmation pour tout ce que vous voudrez. Il suffit d'utiliser un moteur de recherche en utilisant les bons mots pour votre recherche. Vous y trouverez également des tutoriels sur l'utilisation d'Eclipse. Et la grande majorité de ces tutoriels est en anglais. Si votre version d'Eclipse est en français, cela vous demandera parfois quelques efforts pour vous y retrouver.

Lancer Eclipse à partir des menus de votre environnement Linux ou Windows. Créer un projet nommé *tp02*. Dans le dossier *src*, créer une classe **`Student`**. Cette classe servira à modéliser la notion d'*élève*.



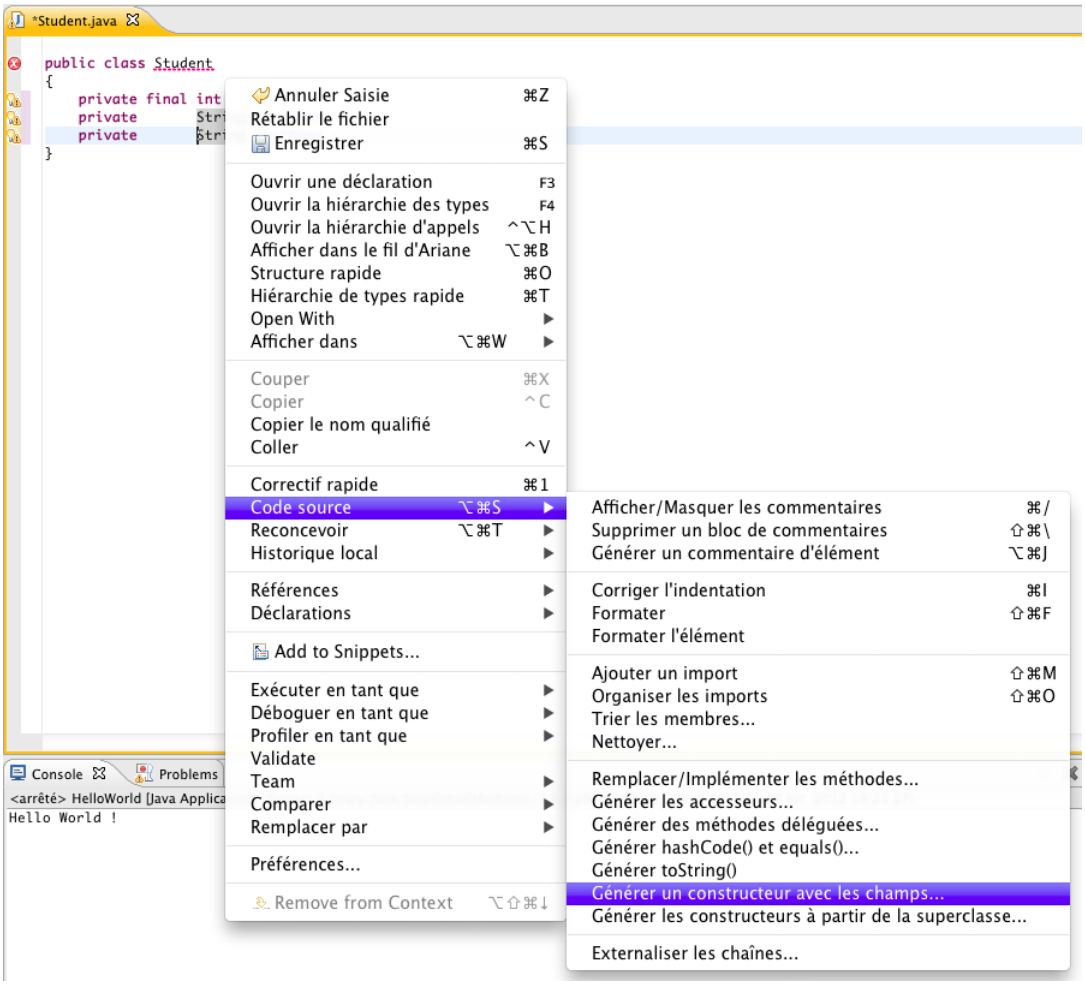
Parmi les attributs de la classe **`Student`**, on doit avoir :

- Un numéro d'identification appelé **`id`** de type **`int`**.
- Le prénom, **`firstName`** en globish, de type **`String`**[1].
- Le nom, **`lastName`**, en globish, de type **`String`**.

Le numéro d'identification ne changera jamais tandis que le nom et le prénom sont susceptibles de changer si l'état-civil de l'élève change. Déclarez ces trois attributs dans la classe **`Student`**. Lorsque c'est fait, cliquer ici pour vérifier votre code.

Nous allons maintenant écrire un **constructeur** pour cette classe. Ce constructeur devra initialiser tous les champs. Eclipse vous aide :

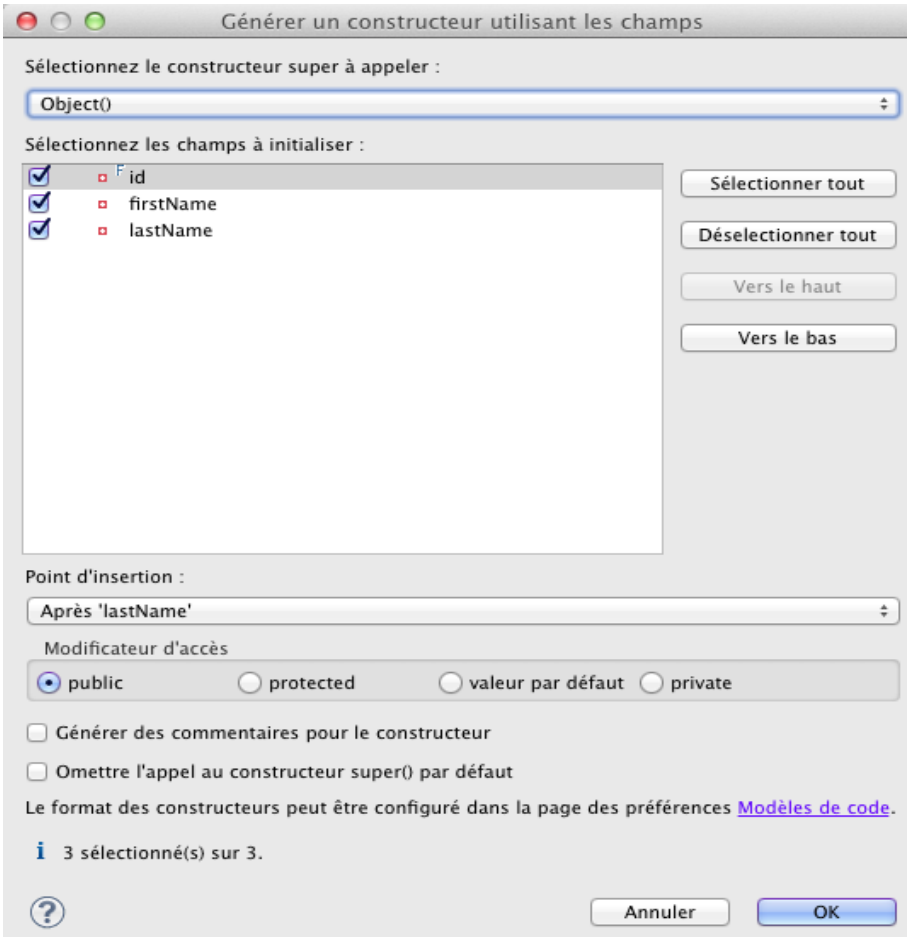
1. Clic droit sur la fenêtre d'édition de la classe **`Student`**.
2. Un menu apparaît ; amenez la souris sur *Code source*.
3. Un second menu apparaît ; cliquez sur *Générer un constructeur avec les champs*.



Une fenêtre apparaît qui vous propose de personnaliser le constructeur :

---

[1] La classe **String** est prédéfinie dans le JDK de Java et modélise une chaîne de caractères. Elle sera présentée plus en détails dans un cours ultérieur.

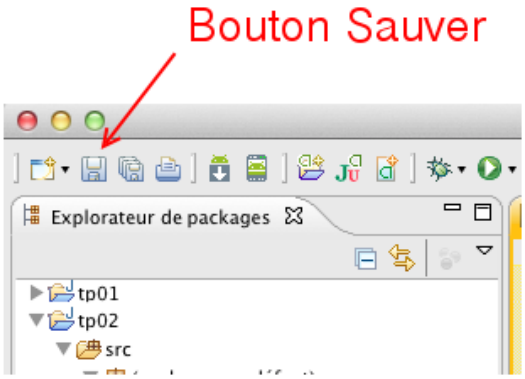


Ne sélectionnez pas le constructeur de la classe super. Vous n'êtes pas encore prêts pour cela. Vérifier que les trois attributs sont sélectionnés. Le point d'insertion du constructeur dans la classe peut être laissé tel qu'il est. L'accès au constructeur doit être **`public`**. Cliquer sur le bouton *OK*.

On obtient alors une classe modifiée :

```
public class Student {

    private int id;

    private String firstName;

    private String lastName;

    public Student( int id,
                    String firstName,
                    String lastName ) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
    }
```

Avant de quitter une classe pour aller en éditer une autre, il est prudent et fortement conseillé de sauvegarder la classe en cours d'édition. Le bouton de sauvegarde est situé dans la barre de menu.



Avant de continuer, sauvegardons notre classe **`Student`**. A présent, générons une seconde classe appelée **`Test`** et contenant une méthode **`main`** comme dans le premier TP. Dans cette méthode **`main`**, céez un objet de type **`Student`** avec l'instruction :

```
Student student = new Student(1024,"Robert","Tartempion");
```

Puis demandez l'affichage de l'objet dans la console avec l'instruction :

```
System.out.println(student);
```

Puis exécutez le programme. Vérifiez votre travail ici.

Nous avons vu dans le corrigé que Java ne sait pas afficher les objets de la classe **`Student`** avec la fonction **`System.out.println(…)`**. Tout simplement, Java ne connaît pas cette classe que nous avons inventée. Si l'on veut que Java soit capable d'afficher un objet de la classe **`Student`**, il faut lui fournir une méthode qui donne l'affichage sous la forme d'une chaîne de caractères que la fonction **`System.out.println(...)`** utilisera pour afficher l'objet.

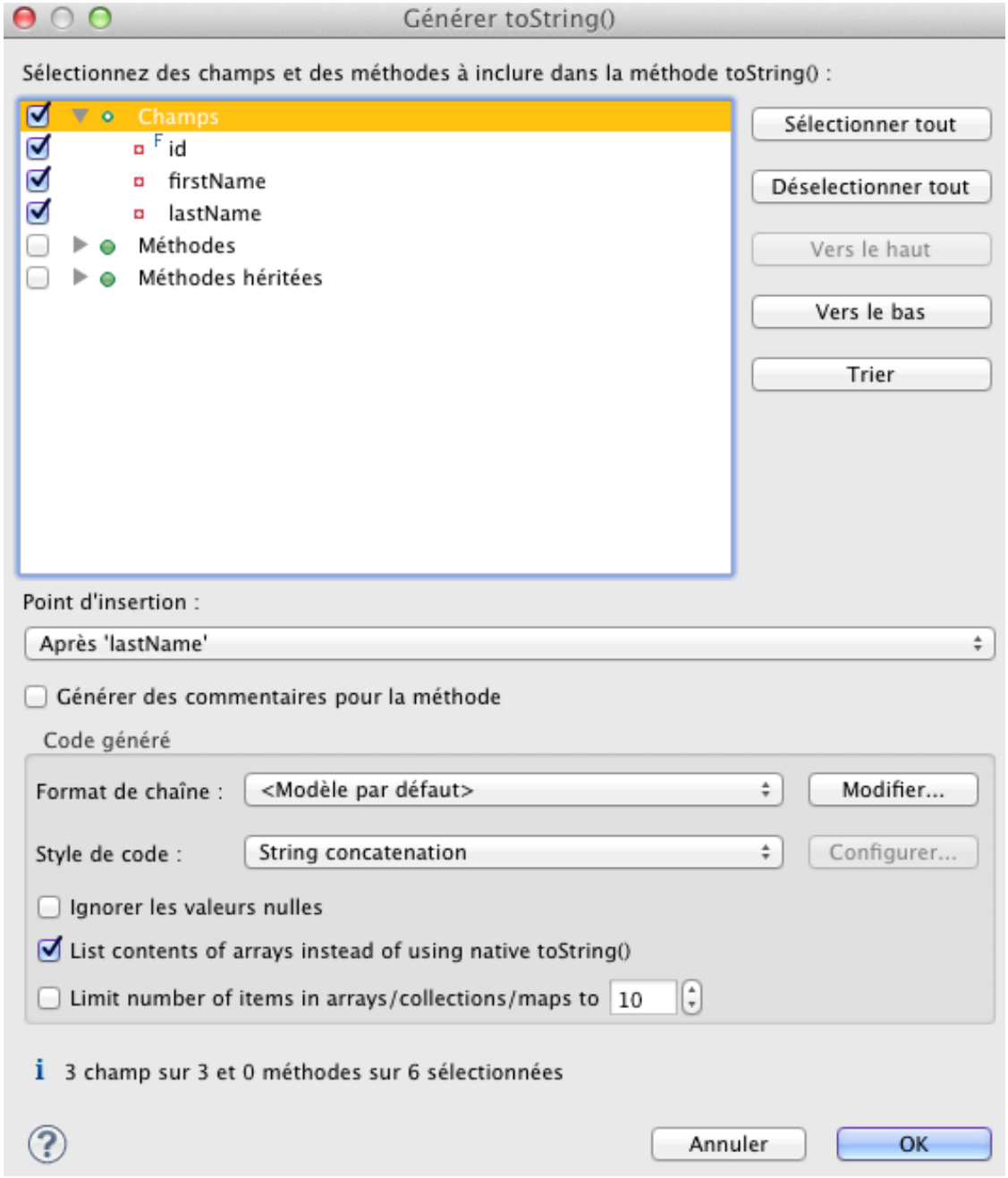
Cette méthode a l'en-tête suivant:

```
public String toString()
```

Pour générer cette méthode dans la classe,

1. Clic droit sur la fenêtre d'édition de la classe **`Student`**.
2. Un menu apparaît. Amenez la souris sur *Code source* .
3. Un second menu apparaît ; cliquez sur *Générer toString()*.

Une fenêtre apparaît pour vous proposer la génération d'une méthode **`toString`** :



La méthode qui sera générée calculera une chaîne de caractères qui comprendra le nom de la classe de l'objet (**`Student`**) suivi de la valeur des attributs. Sur cette fenêtre, il est possible d'ajouter d'autres propriétés à l'affichage. Cliquer sur le bouton *OK*.

La classe est modifiée :

Student.java

```java
public class Student {

    @Override
    public String toString() {
        return "Student [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + "]";
    }

    public Student(String id, String firstName, String lastName) {
        super();
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    String id;

    String firstName;

    String lastName;

    public Student() {
        // TODO Auto-generated constructor stub
    }
}
```

L'annotation **`@Override`**, placée juste avant la méthode, indique que cette méthode est la redéfinition d'une méthode héritée. Nous verrons cela en cours plus tard.

Re-éxécuter le programme. Contrôler votre travail ici.

Dans ce TP, nous avons finalement peu écrit d'instructions Java nous-même. Ce ne sera pas toujours le cas. L'environnement de développement Eclipse nous permet de réaliser des tâches standards (constructeurs, getters, setters, affichage, etc.) en quelques clics de souris. Eclipse vous signale également les erreurs et les possibilités d'erreurs dans vos programmes. Il peut même vous proposer des corrections, ou encore de renommer des éléménts du code (refactoring). Toutes ces fonctionnalités spnt très utiles en environnement de développement industriel car cela permet d'améliorer grandement la productivité du programmeur. Mais Eclipse ne programmera jamais d'algorithmes pour vous!