

TP 1

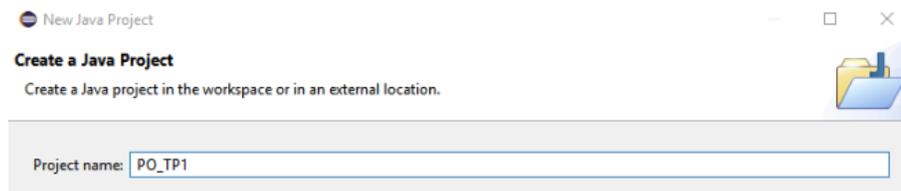
Programmes de base et utilisation Eclipse

Nous allons dans ce TP écrire quelques programmes simples en Java, en exploitant l'environnement de développement intégré Eclipse.

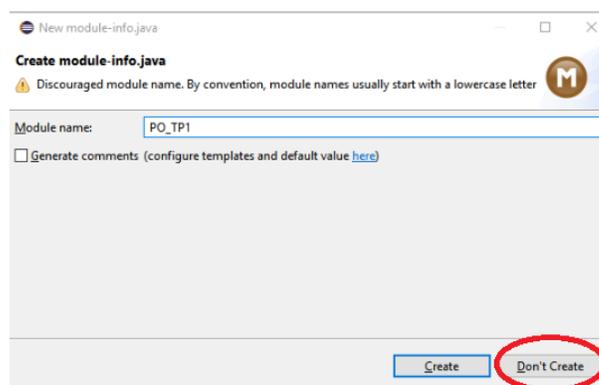
1 Préparation - initiation à Eclipse.

Exercice 1. Lancement d'Eclipse et création d'un projet 'vide' dans Eclipse

- Commencez par démarrer Eclipse, soit en allant le chercher dans les menus, soit en tapant eclipse dans un terminal.
- Eclipse va vous demander quel workspace vous voulez utiliser. Un workspace est répertoire où Eclipse stockera les fichiers de vos projets. Vous pouvez accepter ce qu'il vous propose par défaut (en général `$HOME/workspace`, où `$HOME` est votre répertoire personnel), ou lui indiquer un autre répertoire de votre convenance (par exemple `$HOME/P0`).
- Nous allons créer un nouveau projet pour le TP. Un projet Eclipse est un regroupement de fichiers de code, de configuration et de ressources (ex : images) qui permettent de réaliser une tâche particulière.
 - Dans le menu File, cliquez sur New, et choisissez Java Project.
 - Une boîte de configuration apparaît, vous avez juste à y entrer le nom du projet à côté de Project name, par exemple `P0_TP1`.

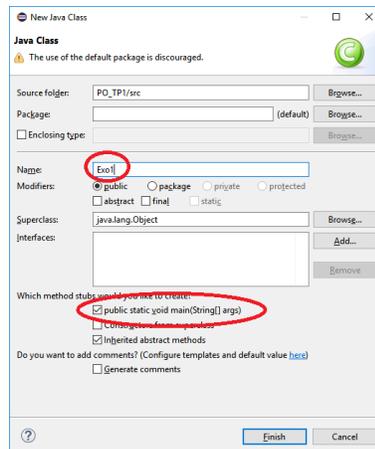


- Cliquez sur *Finish*, si Eclipse vous propose de créer un module-info (comme sur la figure ci-dessous), déclinez la proposition. Votre projet est prêt!!!



Exercice 2. Exécution d'un programme et debug dans Eclipse Le but de cet exercice est d'apprendre à écrire et exécuter des programmes dans Eclipse, puis d'utiliser le debugger pour découvrir et corriger une erreur.

- Dans la vue Package Explorer à gauche d'Eclipse, faites un clic-droit sur votre projet, puis choisissez New et Class.
- Le dialogue de construction de classe s'affiche :



- Donnez un nom à votre classe (= programme ici), par exemple Exo1. **Attention il faut toujours que la première lettre soit en majuscule.** Puis cochez la case *public static void main(String[] args)*. Cf. figure ci-dessus.
- Eclipse vous ouvre le programme Exo1.java, il y a déjà un peu de code dedans.
- Vérifions que tout marche bien : dans la méthode main, à la ligne 7, écrivez :

```
System.out.println("Hello world");
```

Raccourci : si vous ne voulez pas taper autant, tapez juste `sysout` puis `Ctrl + Espace`. Eclipse vous propose plusieurs choix de *snippets*, le premier convient donc tapez Entrée.

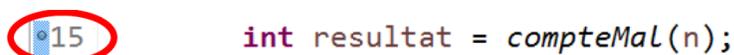
- Sauvez votre programme (raccourci : `Ctrl + s`) puis exécutez le : bouton  ou raccourci `Ctrl + F11`. Vous verrez s'afficher en bas dans la vue *Console* le texte `Hello world` : tout va bien !
- Maintenant, modifiez votre code pour avoir le code suivant :

```
public class Exo1 {

    public static int compteMal(int n) {
        int compteur = 0;
        for (int i=0; i<n ; i++) {
            compteur += compteur+1;
        }
        return compteur;
    }

    public static void main(String[] args) {
        int n = 5;
        int resultat = compteMal(n);
        System.out.println("La somme des "+n+" premiers entiers est : "
            + resultat + " ?");
    }
}
```

- La fonction `compteMal(int n)` doit faire la somme des entiers de 1 à n , c'est-à-dire $1+2+\dots+n$. Exécutez le programme, et vérifiez la somme à la main : est-ce correct ?
- Comme son nom l'indique, la fonction précédente est buggée. Nous allons donc la débogger !
 - On veut que l'exécution du programme s'arrête avant l'appel à notre fonction suspecte, `compteMal`. Pour cela, double cliquez juste à côté du numéro de ligne où cette fonction est appelée. Un point bleu va apparaître, comme dans l'image ci-dessous. Cela s'appelle un point d'arrêt (*breakpoint* en anglais).



- Démarrez ensuite le debugger : bouton  ou raccourci F11.
- Votre interface Eclipse se modifie : dans le jargon Eclipse, il est passé dans la perspective de débogage. Le programme est arrêté à la ligne demandée, qui est sur-lignée en vert. Nous allons faire un « pas » (*step* en anglais) pour rentrer dans la fonction `compteMal` : faites un *step into* soit avec le bouton , soit avec le raccourci F5.
- Vous voyez que la ligne sur-lignée est maintenant la première ligne de `compteMal`. En haut à droite, une vue vous indique les variables disponibles localement : pour l'instant il y a juste le paramètre $n = 5$.
- Nous allons faire avancer l'exécution pas à pas, cette fois avec des *step over*, bouton  ou raccourci F6. Faites deux pas : vous êtes maintenant arrêtés sur la ligne `compteur += compteur + 1`. Remarquez que deux nouvelles variables locales ont été ajoutées en haut à droite : `compteur = 0` et `i = 0`.
 - **Remarque :** *step into* vous fait rentrer dans les fonctions, alors que *step over* les exécute sans vous faire rentrer dedans. Si une fonction n'est pas censée avoir de bug, vous pouvez donc « sauter » son exécution détaillée grâce à *step over*.
- Continuez à faire des *step over*. Observez l'augmentation du compteur quand i augmente, comparez là à ce que vous obtenez quand vous faites la somme à la main : n'y a-t-il pas un problème ?
- Quand vous avez compris, arrêtez le débogage (bouton  ou raccourci Ctrl + F2). Revenez à la perspective Java standard en cliquant sur *Java* au lieu de *Debug* en haut à droite. Corrigez l'erreur et relancez l'exécution.
- S'il y a toujours un problème, recommencez le débogage.

Ceci conclut l'initiation à Eclipse. Pratiquez ce que vous avez appris dans les exercices suivants !

2 Boucles et itérations - manipulation de chaînes de caractères et dessin

Exercice 3.

1. Rédigez la fonction suivante qui indique si une chaîne de caractère `s1` (non vide) apparaît en position i dans une chaîne de caractères `s`. Sa spécification est :

```
public static boolean estPresentEn(int k, String s1, String s)
```

Exemples :

- `estPresentEn(0, "bon", "bonjour bonsoir")` vaut true
- `estPresentEn(5, "bon", "bonjour bonsoir")` vaut false
- `estPresentEn(8, "bon", "bonjour bonsoir")` vaut true
- `estPresentEn(56, "bon", "bonjour bonsoir")` vaut false

2. En utilisant `estPresentEn`, rédigez la fonction suivante qui rend en résultat l'indice de la première occurrence d'une chaîne `s1` (non vide) dans une chaîne `s`. Si `s1` n'est pas présente dans `s` le résultat est `-1`. Sa spécification est :

```
public static int indiceDe(String s1, String s)
```

3. Observez la documentation de la classe `String` de Java sur internet, à l'adresse :

<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>

A quelle méthode de cette classe correspond la fonction `indiceDe` ?

Exercice 4.

- Téléchargez depuis Internet la classe `StdDraw.java`, à l'adresse :

<http://introcs.cs.princeton.edu/java/stdlib/StdDraw.java>

- Dans Eclipse, faites un clic-droit sur votre projet, et choisissez *Import...* puis *File System*.

Indiquez le répertoire où vous avez téléchargé `StdDraw.java` et cliquez sur *Next>* : Eclipse va vous montrer tous les fichiers de ce répertoire. Cochez `StdDraw.java` et cliquez sur *Finish*. Bravo, vous avez maintenant une librairie graphique simplifiée à votre disposition !

- La librairie `StdDraw` vous permet de dessiner des figures géométriques simples, dont les coordonnées sont des double valant entre 0.0 et 1.0.

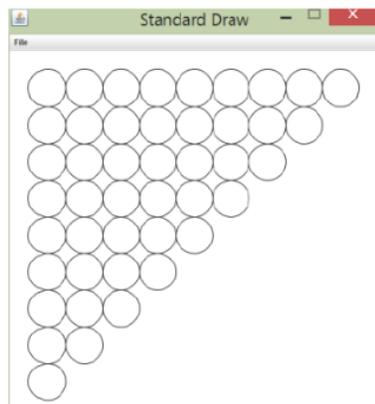
- Essayez par exemple :

```
public class Exo3 {
    public static void main(String[] args) {
        StdDraw.circle(0.5, 0.5, 0.2);
    }
}
```

- La documentation complète est ici : <http://introcs.cs.princeton.edu/java/stdlib/javadoc/StdDraw.html>

- Plus d'infos ici : <http://introcs.cs.princeton.edu/java/15inout/>

- Écrivez un code qui reproduit la figure suivante :



Exercice 5. On veut écrire une fonction `public static int imageMiroir(int n)` qui étant donné un entier `n`, renvoie une « image miroir » de `n`, c'est-à-dire que tous les chiffres sont renversés.

Exemple : `imageMiroir(1234)` vaut 4321.

Écrivez la fonction `imageMiroir`, soit dans un style itératif, soit dans un style récursif.