

# Travaux Dirigés numéro1

## Java Avancé -M1-

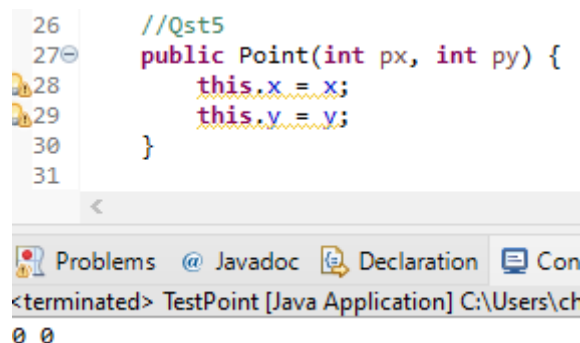
---

### Exercice 1. Eclipse

1. Lorsqu'on tape `syso` et appuie sur `Ctrl+espace` dans la méthode `main`, "`System.out.println()`" s'affiche.
2. Lorsqu'on tape `toStr` et appuie sur `Ctrl+espace` dans la classe, la méthode "`toString()`" s'affiche.
3. Lorsqu'on tape `main` et appuie sur `Ctrl+espace` dans la classe, la méthode "`main`" s'affiche.
4. Lorsqu'on tape `Ctrl+espace` dans la classe, une liste de recommandation des signatures des méthodes, des types de variable etc s'affiche. Et quand on tape `set` après on appuie sur `Ctrl+espace` dans la classe, la méthode setter "`setFoo(int foo)`" s'affiche.
5. Lorsqu'on sélectionne le nom de la classe et appuie sur `Alt+Shift+R`, une fenêtre "`Rename Type`" s'affiche pour changer le nom de cette classe. De même pour l'attribut `foo`, une fenêtre "`Rename field`" s'affiche pour changer le nom de l'attribut.

### Exercice 2. Point

1. Lorsqu'on tape le code, une erreur s'est produite, car "`System.out.println()`" ne fonctionne pas directement dans le corps de la classe, elle doit être dans une méthode, un constructeur ou un bloc d'initialisation statique.
2. Lorsqu'on tape le code dans la classe `TestPoint`, une erreur s'est produite car les attributs `x` et `y` sont déclarés **private**. Alors pour corriger l'erreur, il faut changer la visibilité de ces attributs "**public**" ou bien fournir la classe `Point` par les méthodes **get** et **set** pour accéder à les valeurs `x` et `y`.
3. On utilise le modificateur de visibilité "**private**" pour rendre les attributs, les méthodes et les constructeurs accessibles uniquement dans la classe déclarée, c'est à dire pour protéger les données de l'extérieur.
4. Les **accesseurs** sont des méthodes particulières qui servent à accéder à des données privées. Il y a deux types de méthodes : les accesseurs en lecture dit les **Getters** (`get`) et les accesseurs en modification dit **Setters** (`set`). Les accesseurs qu'on doit mettre dans la classe **Point** sont `getX()`, `getY()`, `setX(int x)` et `setY(int y)`.
5. Lorsqu'on fait un constructeur avec deux paramètres `px` et `py` et on exécute la classe `TestPoint`, "`0 0`" imprime sur l'écran (voir figure1). Pour résoudre ce problème on doit respecter l'utilisation de mot clé **this** car le `this` fait la référence à l'objet courant dans une méthode ou un constructeur. En outre, elle permet de faire la différence entre les paramètres de la méthode et les attributs de classe (Au cas où il a deux variables de même noms).



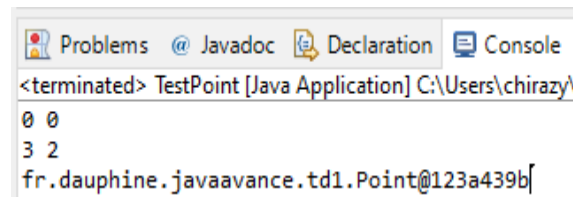
```
26      //Qst5
27      public Point(int px, int py) {
28          this.x = x;
29          this.y = y;
30      }
31
```

<terminated> TestPoint [Java Application] C:\Users\ch  
0 0

Figure 1

6. Lorsqu'on modifie les paramètres de constructeur à x et y, 0 0 s'affiche sur l'écran, car dans la classe TestPoint il y a l'affichage d'un seul point p. Donc pour imprimer le point p1 on doit taper le code suivant "System.out.println(p1.getX() + " "+p1.getY()); "

Lorsqu'on fait l'appel de System.out.println avec le nom d'objet, l'écran imprime le nom de package avec le nom de la classe Point et @ pour ensembler la chaîne avec le hashcode de l'objet (voir Figure 2).



```

<terminated> TestPoint [Java Application] C:\Users\chirazy\
0 0
3 2
fr.dauphine.javaavance.td1.Point@123a439b

```

Figure 2

7. Ask
8. Le compilateur peut déterminer le constructeur à utiliser, grâce au nombre et de type des arguments qui sont transmis au constructeur.

### Exercice 3. Equality

1. Le code affiche true et false car "==" compare les objets par leurs adresses.

==> **True** car les deux objets p1 et p2 ont la même référence car on affecte p2 par p1 et **false** car les deux objets p1 et p3 ont la même valeur mais pas la même adresse, se sont deux objets différents.

2. Voir classe Point.
3. IndexOf( object o) : cette méthode renvoie l'index de la première occurrence de l'objet o, sinon -1 si l'élément spécifié n'existe pas dans la liste.

==> Dans notre cas cela signifie que "System.out.println(list.indexOf(p3));" imprime -1 car p3 n'existe pas dans la liste, puisque p3 et p1 ne sont pas égaux (c'est vrai qu'ils ont le même contenu mais pas la même référence : faux) par contre p1 et p2 sont égaux (ils ont le même contenu et la même référence : vrai) cela explique bien l'affichage de 0 sur l'écran. Par conclusion, indexOf() joue le rôle d'égalité equals() (==).

Pour résoudre ce problème, on doit instancier un point p2 (1,2) et annuler l'affectation de p1 à p2 pour avoir une différente référence mais de même contenu (voir le code).

### Exercice 4. Polyline