**Travaux dirigés n°1**

Java avancé

-M1-

Exercise 1 :

2. **public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

System.out.println();

}

3. I can make a method void :

**private** **void** toStr() {

// **TODO** Auto-generated method stub

}

4. I can make a main method :

**public** **static** **void** main(String[] args) {

}

5. I have several options. When I type set and CTRL + Space, I can make the setter for foo.

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

**public** **void** setFoo(**int** foo) {

**this**.foo = foo;

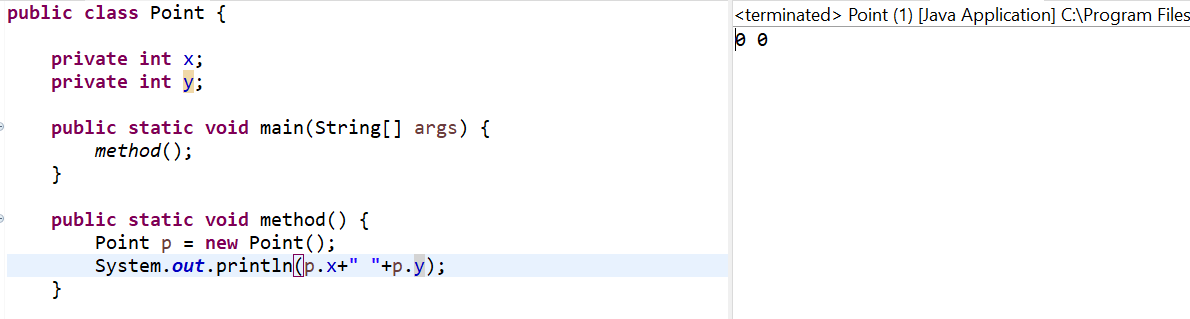
}

6. I can rename the class name and the field foo.

7.

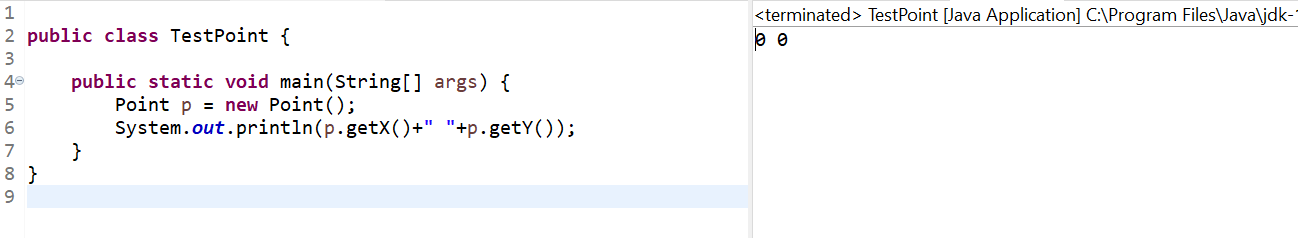
Exercise 2 :

1. When I make a method with this code and I call this method in a main in the class Point, it works because Point.x and Point.y are visible.



1. It doesn’t work because Point.x and Point.y are not visible in TestPoint. We can fix this by creating a getter and a setter for x and y in Point. Thanks to that, we get the values and we print them. Une image contenant texte

   Description générée automatiquement



1. We need to get values accessible by anyone, in all classes and to modify attribute by anyone.
2. On utilise les accesseurs (getter) afin de pouvoir avoir accéder aux champs privés (non accessibles dans toutes les classes) présents dans une autre classe.
3. Dans le main où on exécute la classe Point, cela ne fonctionne pas car ce n’est pas le constructeur mais la classe. Le constructeur est considéré non défini. Il faut alors changer l’appel du constructeur dans le main.

**public** **class** Point {

**private** **static** **int** *x*;

**private** **static** **int** *y*;

**public** Point(**int** px, **int** py) {

**this**.*x* = px;

**this**.*y* = py;

}

**public** **static** **void** main(String[] args) {

*method*();

}

**public** **static** **void** method() {

Point p = **new** Point(*x*, *y*);

System.***out***.println(p.*x*+" "+p.*y*);

}

**public** **int** getY() {

**return** *y*;

}

**public** **void** setY(**int** y) {

**this**.*y* = y;

}

**public** **int** getX() {

**return** *x*;

}

**public** **void** setX(**int** x) {

**this**.*x* = x;

}}

1. Ca marche.

**public** Point(**int** x, **int** y) {

**this**.*x* = x;

**this**.*y* = y;

}

1. On créé une variable de classe qui compte le nombre d’instanciations de la classe Point.

//Compteur du nombre de points créés

{

*nbPoints*++;

}

1. Il saura lequel appeler car les deux constructeurs n’ont pas le même nombre d’arguments.
2. On définit la méthode toString() puis on l’utilise sur les instances de classe : System.***out***.println(*p2*.toString());

**public** String toString() {

**return** "("+*x*+","+*y*+")";

}

Exercice 3 :

1. Ca affiche ceci : . Ca affiche true parce que p1 et p2 font référence au même objet. Ils sont égaux en ce sens. Tandis que p1 et p3 ne sont pas égaux car ce sont deux objets différents.
2. **public** **boolean** IsSameAs(Point p) {

**boolean** b=**true**;

**if**(p.x==**this**.x && p.y==**this**.y) {

b=**true**;

}**else** {

b=**false**;

}

**return** b;

}

1. Ce code renvoie 0 et -1 pour p2 et p3. Il faut créer une méthode equals pour définir l’objet en Point.

@Override

**public** **boolean** equals(Object o) {

Point k = (Point)o;

**return** **this**.x==k.x && **this**.y==k.y;

}

Exercice 5 :

Question 1 : Nous pouvons récupérer les variables x et y courantes dans Point, avec this pour les faire corresponde à dx et dy.

Question 5 : Ca fonctionne car on met New Point dans le constructeur Circle, ce qui réinitialise les coordonnées du point p. Cela ne nous donne pas deux translations pour c et c2.

Question 6 : Je n’ai pas eu de problème.

Question 9 :