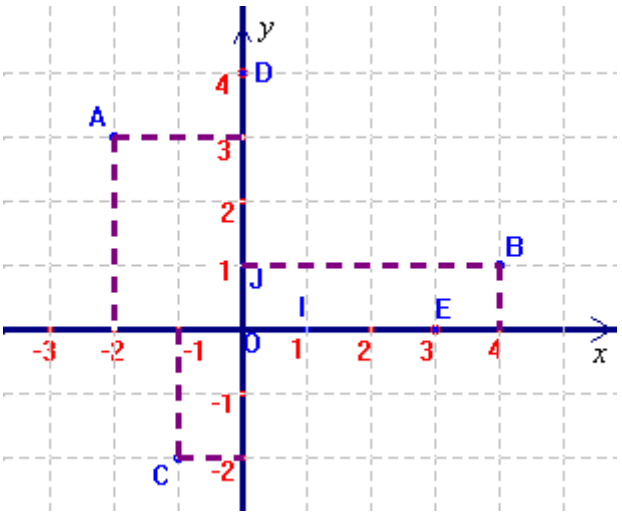
TP N°2

Objectifs	Temps alloué	Outils
- Maitriser la définition de classe et la création d'objets. - S'initier à la déclaration des membres d'une classe. - Manipuler les constructeurs, l'appel de méthode, les attributs objets.	3h00	Eclipse

Exercice 1 : La classe Point

Déclaration de classe, d'attributs, de méthodes

1) Ecrire une classe **Point** qui permet de représenter un point sur un plan cartésien.



2) Chaque point est caractérisé par son **abscisse** et son **ordonnée** de type **int** et son **nom** de type **char**.

a. Déclarerez, dans un premier temps, ces attributs sans leur attribuer de modificateur de visibilité.

b. Ajoutez une méthode **afficher()** imprimant sur l'écran les coordonnées d'un point

c. Définir une méthode **deplacer(int a, int b)** effectuant son déplacement selon les **pas** communiqués à la méthode comme arguments (déplacement par rapport à la position courante).

d. Ajoutez la méthode **reset()** permettant de ramener un point à l'origine du repère.

Instanciation et appel de méthode

3) Ecrire une classe **TestPoint** décrivant un petit programme utilisant la classe Point. Ce programme permet de
   a. créer un point p (grâce au constructeur par défaut)

   La création d'un objet entraîne systématiquement l'initialisation par défaut de ses données membres, comme le montre le tableau suivant :

Types de données	Valeur par défaut
boolean	False
char	Ø
byte, short, int, long	0
float, double	0.0
Objet	Une référence

   b. Afficher les coordonnées du point p sous la forme suivante :

      Nom(abscisse, ordonnée)

   c. Le déplacer (abscisse +1 et ordonnée -2) et en afficher de nouveau ses coordonnées.
   d. Faire appel à la méthode **reset()** puis afficher de nouveau les coordonnées du point.

Question :

- Peut-on déplacer un point à partir de **TestPoint** sans appeler la méthode deplacer()?
- Est-ce que l'appel de **deplacer()** (à partir de **TestPoint**) est toujours possible? Faire une discussion.

Constructeurs et création d'objets

4) Reprendre la définition de la classe **Point** en ajoutant un constructeur à 2 paramètres recevant en arguments les coordonnées d'un point (Faites en sorte que les paramètres passés aient le même nom que les attributs). Puis ré-exécuter la classe TestPoint.
   Que se passe-il à présent ? Comment expliquez-vous cela ?
5) Complétez la classe **Point** par :
   a. un constructeur sans paramètres permettant d'initialiser l'abscisse et l'ordonnée à zéro et le nom du point à 'O'.
   b. un troisième constructeur qui prend un point en paramètre et utilise les coordonnées de celui-ci pour s'initialiser.

6) Modifier la classe TestPoint de façon à créer trois objets Point, le premier (p1) avec le constructeur paramétré (abscisse =3 et ordonnée =5), le deuxième (p2) avec le constructeur sans paramètres et le troisième (p3) en passant en paramètre le point p1.

Méthode toString()

- ✓ La méthode toString est définie dans la classe Object ; en conséquence toutes les classes Java en héritent.
- ✓ elle renvoie le **nom de la classe** de l'objet concerné suivi de **l'adresse** de cet objet.
- ✓ L'instruction System.out.println(O.toString()) est équivalente à System.out.println(O). O étant un objet.

Tester le code suivant dans votre classe TestPoint :

```
System.out.println(p1);
System.out.println(p2);
System.out.println(p3);
```

Quel résultat cela produit ?

- ✓ La méthode toString, définie dans la classe Object, admet pour prototype :
  public String toString()
- ✓ Quand on redéfinit la méthode toString, on fait en sorte qu'elle renvoie une chaîne de caractères servant à décrire l'objet concerné.

7) Définir dans la classe Point une méthode **toString()** permettant de retourner une chaîne de caractères (un String) décrivant un point (dans ce cas son nom et ses coordonnées).
8) Utilisez la méthode toString() dans la classe TestPoint afin d'afficher les caractéristique des 3 objets.

Comparaison d'objet

Nous souhaitons faire un test d'égalité sur les objets précédemment créés.

- ✓ L'opérateur **==** vérifie si deux objets sont **identiques** : il compare que les deux objets possèdent la **même référence mémoire** et sont donc en fait le même objet.

- ✓ Deux objets **identiques** sont **égaux** (possède les même valeurs) mais deux objets égaux ne sont pas forcément **identiques** (n'ont pas la même référence mémoire).

Question : Que donne d'après vous le code suivant :

```
System.out.println(p1==p2);

System.out.println(p1==p3);
```

9) Définir dans la classe Point une méthode **coïncide_V1 (Point p)** qui retourne un résultat **booléen** selon que les attributs de l'objet point communiqué en argument possèdent les mêmes valeurs que ceux de l'objet courant ou non. (comparaison entre l'objet ayant fait appel à la méthode **coïncide_V1** et l'objet passé en paramètre).
10) Faire appel à la méthode **coïncide_V1** dans la classe **TestPoint** afin de vérifier l'égalité des points.

Méthode de classe

11) Il serait plus judicieux de faire en sorte que la méthode permettant de tester l'égalité de deux points prenne en paramètre les deux points à tester. Trouvez une solution pour réaliser une telle méthode que vous nommerez **coincide_V2(Point a, Point b).** Testez cette méthode dans la classe **TestPoint**.