

Objets et classes Principes de l'encapsulation

Sommaire

Objectifs :

Découvrir et expérimenter la création de **classes**.

Mettre en oeuvre les mécanismes de base de la P.O.O. :

- l'encapsulation,
- le constructeur,
- les accesseurs.
- Relation d'association entre 2 classes:agrégation, composition

Environnement technique :

- Java SE
- JDK 7
- Un EDI (type *Eclipse*).
- La documentation
 - **Java on line**

<https://docs.oracle.com/javase/tutorial/java/javaOO/index.html>

- les support de cours 01_..... → 05_Classe&Objet.pdf

1. UNE CLASSE «DURÉE»

Cette classe serait très utile dans une application qui comptabilise des temps exemple: cumul des temps (pénalités , bonus) pour déterminer sur plusieurs grands prix, le motard le plus rapide de l'année.

Déclarer une classe Duree possédant 3 attributs privés:

- 1) Le nombre d'heures de la durée.
- 2) Le nombre de minutes de la durée.
- 3) Le nombre de secondes de la durée.

On munira cette classe de méthodes publiques:

- Un constructeur par défaut et un constructeur.
faire les conversions nécessaires pour que les nombres de minutes et de secondes soient strictement inférieurs à 60.
- De méthodes nommées:
 - **afficheDuree** réalisant l'affichage de la durée (sous la forme 3h10m00s).
 - **setDuree** setter pour les 3 attributs privés
 - **equals** compare 2 durées.
 - **conversionEnSec** convertissant la durée en un nombre de secondes.
 - **ajoutSecALaDuree** ajoutant à la durée un nombre de secondes.
 - **somme2Duree** pour ajouter 2 durées
 - **compare2Duree** pour déterminer quel est la plus petite durée

En utilisant la classe durée déterminer le champion du monde marc marquez:

gp du japon: 2h 23 mn 10 sec
gp d'italie: 1h 45 mn
pénalité: 50 secondes (essai du gp du japon)
bonus: 1 mn 10 sec

valentino rossi:

gp du japon: 2h 32 mn 15 sec
gp d'italie: 1h 21 mn 03 sec
pénalité: 15 mn 20 secs (essai du gp du japon)
bonus: 7 secs.

```
if (Marquez.equals(Rossi)) System.out.println("ex-aequo");  
else  
    if (Marquez.compare(Rossi)) System.out.println("Marquez vainqueur");  
    else  
        System.out.println("Rossi vainqueur");
```

page suivante un exemple d'utilisation de la classe:

```



public static void main(String[] args) {
    duree h1 = new duree();
    h1.afficheDuree();
    duree h2= new duree (0, 0, 0 );
    h2.afficheDuree ();

    if (h1.equals(h2) )
        System.out.println("h1 et h2 sont identiques");
    else
        System.out.println("h1 et h2 different");

    h1.afficheDuree();
    h2.setDuree(1, 1, 61);
    h2.afficheDuree();

    if (h1.equals(h2) )
        System.out.println("h1 et h2 sont identiques");
    else
        System.out.println("h1 et h2 different");
    System.out.println( "----- somme de duree -----");
    h1.setDuree(1,60,5);
    h1.somme2Duree(h2); //h1 + h2
    h1.afficheDuree();

```

 Console 

```

h1: 00h00m00s
h2: 00h00m00s
*****appel méthode de classe statique fournit 2 objs à méthode*****
h1 et h2 sont identiques
*****appel méthode d'instance fournit 1 objet à méthode*****
h1 et h2 sont identiques
h1: 00h00m00s
h2: 01h31m01s
h1 et h2 different
----- somme de duree -----
h1:    02h00m05s
h2:    01h31m01s
h1+h2: 03h31m06s

```

2: la calculatrice

Objectifs :

Assimiler les concepts de **classe** et de **méthodes d'instance** en Java.

Déroulement :

Construirez une classe *Calculatrice* permettant d'effectuer les 4 opérations arithmétiques de base, par le biais de méthode d'*instance*.

Etapes de réalisation

1. Créez une classe *Calculatrice* permettant le calcul des 4 opérations arithmétiques de base.

Prévoyez pour cela 4 **méthodes**: **addition**, **soustraction**, **multiplication**, **division**

La méthode appelée dépendra de la saisie utilisateur : '+', '-', '*', '/'

2. Les 4 méthodes déterminent et renvoient le **résultat** de l'opération effectuée.

3. Le choix de l'opérateur, la saisie des réels, l'affichage du résultat se fait dans le main

exemple :

```
operation (+ - * / ) ?
```

```
+
```

```
saisissez 2 réel distinguer par la touche enter :
```

```
1
```

```
2
```

```
somme: 3.0
```

```
operation (+ - * / ) ?
```

```
/
```

```
saisissez 2 réel distinguer par la touche enter :
```

```
22
```

```
3
```

```
division: 7,33
```

```
System.out.println("operation (+ - * / ) ? ");
```

```
op = sc.next().charAt(0);
```

Conseil: pour afficher 2 chiffres apres la virgule

utilisez la méthode: System.out.printf("%2.2f")

```
float pi = 3.141593F;
```

```
System.out.printf("%2.2f", pi); -----> 3.14
```

3. permutation objet passage de paramètres (d'un type objet) par adresse

Ecrire la classe correspondante pour le code suivant:

```
NewInteger IntegerA = new NewInteger(10,20);
```

```
System.out.println("avant permutation");  
IntegerA.affiche();  
IntegerA.permut();  
System.out.println("apres permutation");  
IntegerA.affiche();
```

 Console 

```
avant permutation:  
10,20  
apres permutation:  
20,10
```

4. UNE CLASSE «POINT2D»

Voire SupportCoursGretaProgObjet&Java p 134

Déclarer une classe point avec 2 attributs privés :

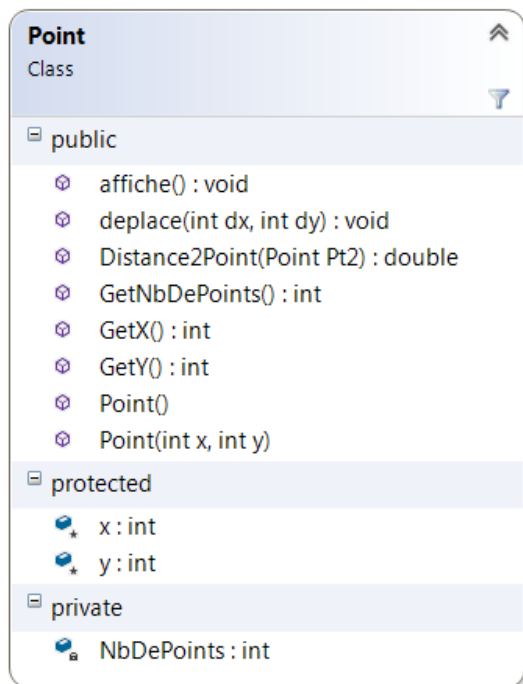
1. L'abscisse du point (x)
2. L'ordonnée du point (y)

On munira cette classe avec:

- 2 constructeurs pour créer des instances de la classe point
 - par défaut
 - pour initialiser les attributs privés d'un objet point.
- Une méthode déplace en 2D: `deplace(int dx, int dy)`
- Une méthode coïncide : `bool coïncide(Point Pt);`
- Une méthode réalisant l'affichage des coordonnées du point.
- Setter/Getter pour chaque attribut. `void setx (...)`
- Un attribut statique `NombreDePoints` à afficher à chaque variation du nombre de points.
- Une méthode distance calculant la distance entre 2 points :
`double Distance2Point(Point Pt2)`
formule de calcul de la distance entre 2 points:

$$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

horizontal distance vertical distance



Point
- x : int - y : int - <u>nbrePoints : int = 0</u>
+ Point() + Point(abs : int, ord : int) + Point(coorEgal : int) + afficher() : void + deplacer(dx : int, dy : int) : void + deplacer(d : int) : void + prodCoor() : int + copie(Point point2) : void + getX() : int + getY() : int + setX(nouvX : int) : void + setY(nouvY : int) : void + diff1Point(point2 : Point ; point3 : Point) : void + diff2Point(point2 : Point) : Point + inverse() : Point + permutation(point2 : Point) : void + <u>getNbrePoints() : int</u>

exo en plus : ajouter les méthodes manquantes correspondant au diagramme de classe ci dessus.

Utilisation de la classe point:

- Créer et afficher les points Pt0(0,0) Pt1(1,0) Pt2(1,1)
- Déterminer la plus grande distance entre 2 points
- Déplacer le point Pt0 en (3,3)
- Déterminer la plus grande distance entre 2 points

déterminer si 2 points 2D coïncident:

*exemple avec la méthode **equals** de la classe objet*

- **Equals** - Compare la référence d'objets

```
Point Pt1 = new Point(1,2);
Point Pt2 = new Point(1,2);

if (Pt1.Equals(Pt2))
    println("Les 2 points coïncident");
else
    println("Les 2 points ne coïncident pas");

    Les 2 points ne coïncident pas
```

les 2 points ne possèdent pas la même référence et la méthode equals indiquera que les 2 points sont différents.

Mais dans la réalité les points coïncident quand ils ont des attributs communs et non quand des objets point référencent le même emplacement mémoire.

surcharger la méthode equals de la classe java.lang.Object pour adapter notre classe point à la réalité géométrique.

utilisation du mot clé this

Surcharger la méthode toString et appeler la méthode affiche pour réaliser l'affichage:

```
Point pt0 = new Point();
    pt0.affiche();
```

```
x:0,y:0
```


5. Relation entre les classes cercle/point

Modélisation:

Un cercle est composé de points (relation de composition entre 2 classes le point centre du cercle est construit en même temps que le cercle).

Les attributs:

le centre du cercle (référence vers la classe point) et un rayon r.

dans le même package que la classe Point: implémenter la classe Cercle

UNE CLASSE « CERCLE »

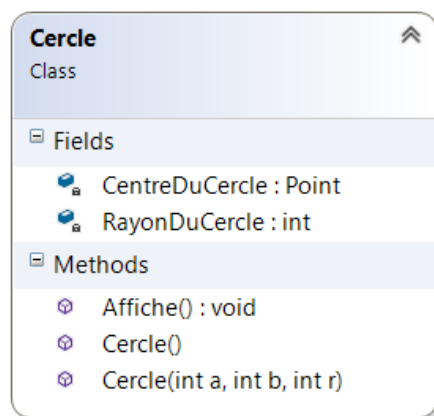
Déclarer une classe Cercle possédant trois attributs privés :

- Le rayon
- L'abscisse du centre
- L'ordonnée du centre.

On munira cette classe de méthodes publiques :

1. Un constructeur par défaut et un constructeur. Ne pas oublier qu'un rayon doit être positif, si le paramètre passé pour initialiser le rayon est négatif, on le remplacera arbitrairement par 0.
2. Une méthode calculant l'aire du cercle.
3. Une méthode calculant le périmètre du cercle.
4. Une méthode réalisant l'affichage des coordonnées du centre et du rayon.
5. Un « setter » pour chaque attribut.
6. Une méthode prenant en paramètre les coordonnées d'un point et retournant un booléen indiquant si ce point est à l'intérieur du cercle ou non.
7. Une méthode prenant en paramètre les coordonnées d'un vecteur et réalisant la translation du cercle par ce vecteur.
8. Une méthode prenant en paramètre un réel et réalisant l'homothétie du cercle par ce réel.

On pourra utiliser la constante **PI** de la classe math



Utilisation de la classe cercle: `Cercle C = new Cercle(3, 3, 2);`
`C.Affiche();`

```
MonCercle:
coordonées du centre : x = 3 et y = 3
rayon : 2
```

Déterminer si un des points Pt0, Pt1, Pt2 est:

- **centre du cercle C**
- **dans le cercle ou en dehors** (préférez la méthode qui manipule un point)

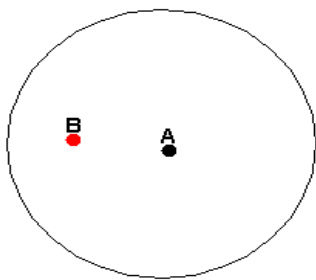
```
bool estDedans(int x, int y)
{
     $((x - x\_centre)^2 + (y - y\_centre)^2) \leq rayon^2$ 
}
```

```
bool estDedans(Point pt)
{
    ..... <= Math.pow(rayon, 2)
}
```

A noter

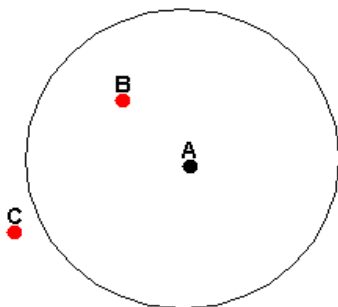
en mode console lors de l'utilisation des classes Cercle, Point
il n'est pas évident de déterminer si le point est dans le cercle.

Plus évident avec une application graphique ..(mais plus complexe à coder)



Le point B est dans le cercle

OK



Le point C est en dehors du cercle

OK

- **implémenter une translation**

```
//deplacement du centre du cercle sur l'axe des x
//deplacement du centre du cercle sur l'axe des y
// le rayon n'est pas modifié
Point ptDebSegment = new Point(1,1);
Point ptFinSegment = new Point(3,3);
Point ptCentreC1 = new Point(0,0);
Cercle C1 = new Cercle(ptCentreC1, 4 );
C1.affiche();
/**translation du cercle sur un segment de droite*****/");
C1.translation(ptDebSegment, ptFinSegment);
C1.affiche();

/***** translation sur un segment de droite *****/
centre : x = 0 y = 0 rayon : 4
/***** translation du cercle sur un segment de droite *****/
centre : x = 2 y = 2 rayon : 4
```

-

implémenter une homothétie:

```
C1.affiche();
C1.homothetie(2);
C1.affiche();
```

```
/***** homothetie*****/
centre : x = 2 y = 2 rayon : 4
/***** homothetie sur C1*****/
centre : x = 4 y = 4 rayon : 8
```