- Méthode qui affiche "Tu as (n) ans" en ayant reçu en argument une année de naissance.

```
public class Exemple2
{
        static void afficheAge(int anaiss)
        {
                int age = 2007 - anaiss ;
                System.out.println("Tu as " + age + " ans") ;
        }
        public static void main (String args[])
        {
                afficheAge(1983) ;
        }
}
```


1. Règle d'écriture des méthodes

b) Méthodes fournissant un résultat

Méthodes qui effectuent des actions et renvoient une valeur.

Syntaxe

```
static type nom-méthode (liste d'argument )
    {
        instructions ;
        return … ;
    }
```

Appel … = nom-méthode (arg 1, arg2,…) ;

Exemple :

- Méthode qui calcule la somme de 2 entiers reçus en arguments.

```
public class Exemple3
{
    static int somme ( int a, int b )
    {
        int calcul = a + b ;
        return calcul ;
    }

    public static void main (String args[])
    {
        int resultat = somme (22, 8) ;   // resultat réceptionné dans une variable
        ...
    }
}
```

Exercices

1) Méthode qui renvoie le pourcentage d'un élève (un entier) en ayant reçu en argument ses points de TJ, d'interro et d'examen (des entiers) exprimés chacuns sur 20 points. (Pondération : 10 - 20 - 70)

2) Méthode qui affiche la factorielle d'un nombre entier reçu en argument (n ! = 1 * 2 * ... * n)

2. Notion de classe

```
public class NomDeClasse
{
    // Attributs et méthodes de la classe...
}
```

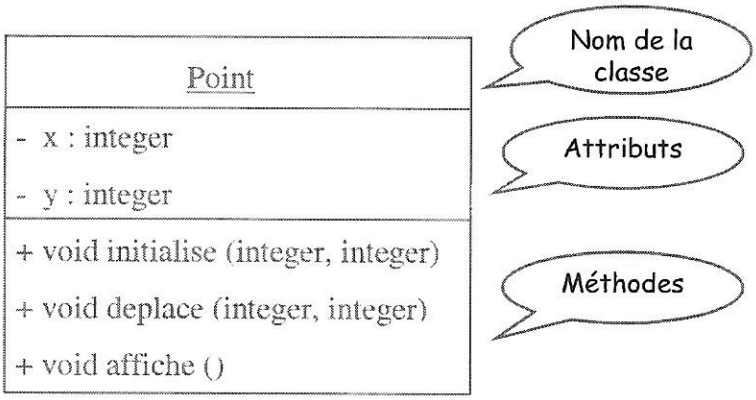
Exemple : soit une classe Point, destinée à manipuler les points d'un plan.

- Un objet de type Point sera représenté par 2 coordonnées entières : x et y (= attributs) ;

Il disposera aussi des 3 méthodes suivantes :

- initialise : attribue les valeurs de départ aux coordonnées du point ;
- deplace : modifie les coordonnées du point ;
- affiche : affiche les coordonnées du point ;

En notation UML, le diagramme de classe se présenterait comme ceci :




```
class Point
{  private int x ;                    // abscisse
   private int y ;                    // ordonnée
   public void initialise (int abs, int ord)
      x = abs ;
      y = ord ;
   public void deplace (int dx, int dy)
      x += dx ;
      y += dy ;
   public void affiche ()
      System.out.println ("Mes coordonnees:" + x + " " + y) ;
}
```

(*)

- Cette classe Point sert à instancier (= créer) des objets de type Point et à leur appliquer à volonté les méthodes disponibles ;

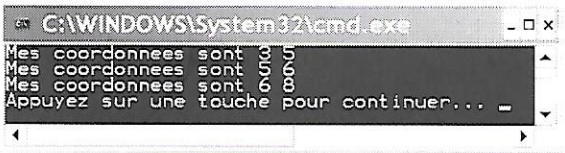
- Pour faire appel à une méthode de la classe :

  objet.méthode()

- Nous pouvons utiliser la classe Point depuis la méthode main d'une autre classe ;


```
public class TstPoint
{  public static void main (String args[])
        Point a ;
        a = new Point() ;
        a.initialise(3, 5) ;
        a.affiche() ;
        a.deplace(2, 1) ;
        a.affiche() ;
        Point b = new Point() ;
        b.initialise(6, 8) ;
        b.affiche() ;
}
```

Que donnera l'exécution de la classe TstPoint ?



3. Notion de constructeur

- Dans la classe Point, il faut toujours commencer par la méthode 'initialise' ;
- Peu pratique ;

- Un constructeur automatise ce mécanisme d'initialisation ;
- Il est automatiquement appelé lorsqu'on instancie un objet de la classe ;
- En Java, il porte le même nom que la classe ;

Constructeur qui remplace la méthode 'initialise'

```
public Point (int abs, int ord)
        {
                x = abs ;
                y = ord ;
        }
```

Changements dans la classe TstPoint

La commande suivante ne convient plus :

```
Point a = new Point( ) ;
```

Refusée par le compilateur : la création d'un objet fait appel au constructeur.

Celui prévu requiert 2 arguments, la commande devient :

```
Point a = new Point (3, 5) ;
```


Autre changement

Il n'est plus nécessaire maintenant de faire appel à la méthode initialise...

On obtient donc :

```
public class TstPoint
{
    public static void main (String args[])
    {
        Point a = new Point(3, 5) ;
        // Cette commande fait appel au constructeur...

        a.affiche( ) ;
        a.deplace(2, 1) ;
        a.affiche( ) ;

        Point b = new Point(6, 8) ;
        b.affiche( ) ;
    }
}
```


Règles concernant les constructeurs

- Un constructeur ne renvoie aucune valeur, aucun type ne doit figurer devant son nom, pas même void ;
- Lorsqu'une classe ne comporte pas de constructeur, Java considère un constructeur vide sans arguments ;

Règles concernant les constructeurs

- Un constructeur doit être déclaré 'public' ;
- Contrairement aux variables déclarées dans un 'main', les champs d'un objet sont initialisés par défaut :

  une booléenne à false
  une donnée numérique à 0
  un char à vide

- La création d'un objet (commande 'new') s'effectue en 3 phases :

  1°) attribution de l'espace mémoire

  2°) appel du constructeur

  3°) renvoi de la référence de l'objet créé

Exercice

- Ecrire le code d'une classe 'Calcul' dont le constructeur initialise les attributs n1 et n2 avec des infos reçues en argument ;

- Elle dispose aussi des méthodes 'somme' et 'produit' qui renvoient le résultat demandé et une méthode 'affiche' qui affiche la valeur de n1 et n2 ;

- Ecrire enfin le code d'une méthode main qui instanciera 2 objets de type Calcul et les manipulera grâce aux méthodes disponibles ;

4. Accès aux membres d'une classe

- Java permet plusieurs types d'accès pour les attributs, méthodes et constructeurs d'une classe :

  private : accessible uniquement ds la classe courante ;

  public : accessible dans toutes les classes ;

  (néant) : accessible par les classes du même package ;