

Programmation Orientée Objet

Mathieu RAYNAL

mathieu.raynal@irit.fr

<http://www.irit.fr/~Mathieu.Raynal>

Programmation Orientée Objet

Notion de classe

Notion d'objet

Que voyez-vous ?



4 rectangles



Nom	R1	R2	R3	R4
X	2	11	5	4
Y	5	9	6	2
Longueur	10	7	2	14
Hauteur	5	3	3	2
couleur	bleu	rouge	vert	Jaune

Comment implémenter ces rectangles ?

- Les décrire
- Calculer
 - Périmètre
 - surface
- Tester
 - Une intersection entre deux rectangles
 - Si un rectangle est inclus dans un autre
 - Si deux rectangles sont alignés (haut bas, gauche ou droite)

→ **Comment faire tout ceci en python ?**

- Tous les rectangles ont
 - les mêmes caractéristiques
 - X, Y,
 - longueur, hauteur,
 - nom,
 - Couleur
 - On peut calculer la même chose pour chacun
 - Surface
 - Périmètre
 - Et tester les mêmes choses
 - Intersection
 - Inclusion
 - alignement

Définir un type d'objet

- Une **classe** permet de décrire un type d'objet
 - Ses caractéristiques : Les **attributs**
 - Ses fonctions : les **méthodes**
- Structure d'une classe en JAVA

```
public class NomDeLaClasse
{
    typeAttribut1 nomAttribut1;
    typeAttribut2 nomAttribut2;
    ...
    methode1()
    methode2()
    ...
}
```

Les attributs

- Un attribut est une **variable** affectée à la classe
- Il est défini par
 - le type de données qu'il représente
 - Un type primitif
 - Une référence à un objet
 - Une référence à un tableau
 - Un nom
- Sa **déclaration** se fait en début de classe
- Son **initialisation** se fait dans le **constructeur** de la classe

Les méthodes

- C'est une **fonction** liée à la classe
- Elle décrit un comportement de la classe sous la forme d'un ensemble d'instructions
- Une méthode est définie par, au minimum
 - Un type de retour
 - son nom
 - Des parenthèses

```
typeRetour NomDeLaMethode()
```

Retour d'une méthode

- Une méthode sert à effectuer un ensemble d'instructions
- A la fin de ces instructions, on peut demander à la méthode de retourner un résultat
- Ce résultat est retourné dans une **variable** au moyen du mot clé **return**
- Dans la déclaration de la méthode, il faut définir le type de variable qui sera retournée
 - Si la méthode ne retourne pas de valeur, le type de retour est **void**

Paramètres d'une méthode

- Ce sont des variables **extérieures** à la classe
- Dans la définition de la méthode, les paramètres sont placés entre les parenthèses, sous la forme
 - typeDeLaVariable suivi du nomDeLaVariable
 - S'il y a plusieurs paramètres, ils sont séparés par une virgule

```
typeRetour NomDeLaMethode(type1 param1, type2 param2, ...)
```

- Les paramètres sont utilisables dans la méthode grâce à leur nom

Exemple de méthodes

```
void ditBonjour(String nom)
{
    System.out.println("Bonjour" + nom);
}
```

```
int calculPuissance(int a, int b)
{
    if(b==0)
        return 1;

    int puissance = 1;
    for(int i=0;i<b;i++)
        puissance*=a;

    return puissance;
}
```

Utilisation des membres à l'intérieur de la classe

- Les attributs et méthodes d'une classe peuvent être utilisés dans cette classe par leur nom
- S'il y a une ambiguïté, les membres d'une classe peuvent être précédés du mot **this**

```
public int multiplication(int a)
{
    int c = this.a*a;
    return c;
}
```

```
public class MaClasse
{
    int a;
    int b;

    public int addition()
    {
        int c = a+b;
        return c;
    }

    public void m() { int d=addition();}
}
```

Les instructions

- Une instruction = toute opération que l'on peut effectuer
 - Déclaration d'une variable ;
 - Affectation d'une valeur à une variable ;
 - Opération sur une variable ;
 - Appel à une méthode
- Toute instruction doit terminer par un point virgule

Les blocs d'instructions

- Un bloc délimite l'ensemble des instructions qui sont effectuées pour
 - La définition d'une classe
 - La définition d'une méthode
 - L'utilisation d'une structure de contrôle (if, for, switch, while)
- Il est délimité par des accolades
- Un bloc d'instructions peut contenir d'autres blocs d'instructions
- Une variable déclarée dans un bloc est utilisable jusqu'à la fin de ce bloc

Qu'est-il possible de faire dans chaque bloc ?

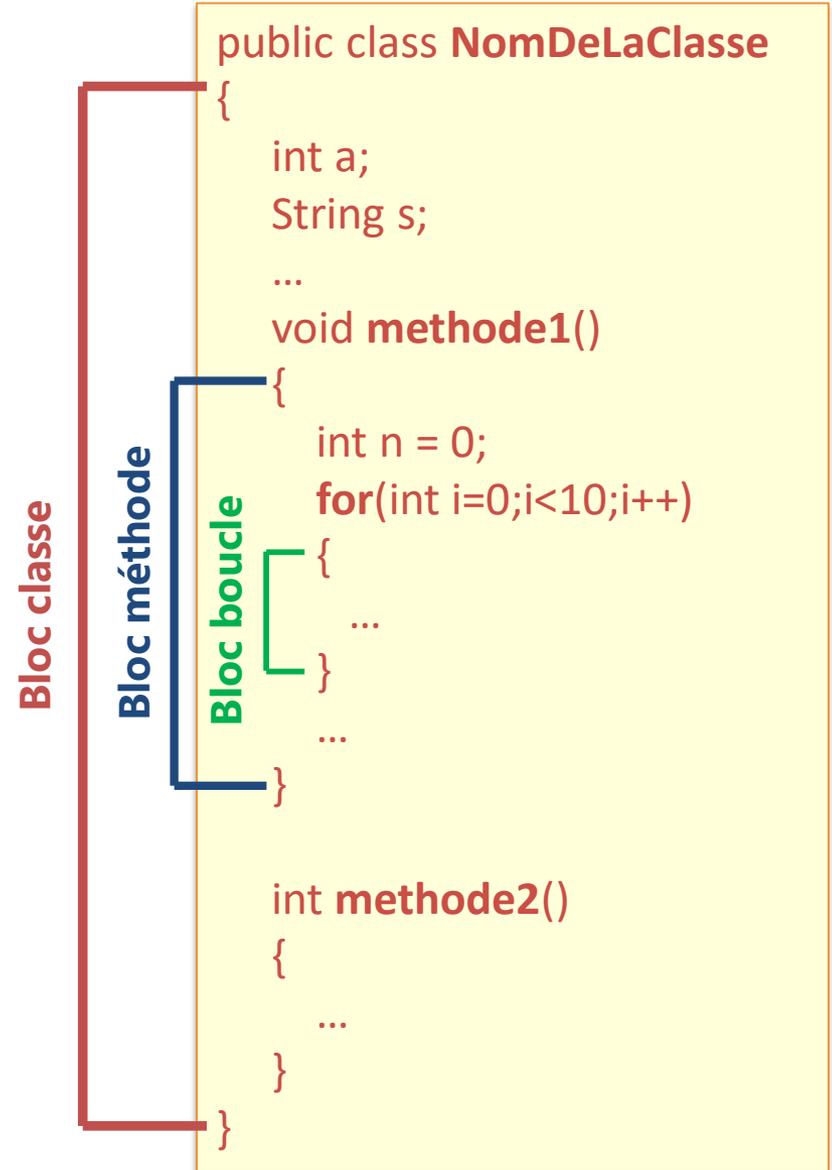
- Au niveau de la classe
 - Seulement des déclarations d'attributs
 - Pas d'affectations ou autres instructions
- Au niveau méthode ou boucle, il est possible d'avoir :
 - déclaration de variable,
 - affectation,
 - instruction,
 - autres boucle

Où positionner les différents éléments ?

- Toutes les méthodes doivent se trouver dans le bloc de la classe
- Toutes les instructions et boucles doivent se trouver dans le bloc d'une méthode, ou d'une autre boucle
- Les variables sont :
 - Dans le bloc de la classe → **Attribut** de la classe
 - Utilisable par toutes les méthodes de la classe
 - Dans le bloc d'une méthode → Variable de la méthode
 - Utilisable seulement par la méthode

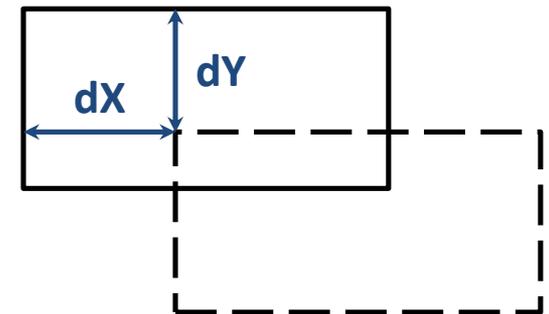
Différents blocs possibles dans une classe

- Les variables **a** et **s** sont utilisables dans toutes les méthodes de la classe
→ Attributs de la classe
- La variable **n** est déclarée dans `methode1`
→ Utilisable que dans la méthode 1
- La variable **i** est déclarée dans le bloc de la boucle `for`
→ Utilisable que dans la boucle `for`



Exercice 1 - Définir une classe Rectangle

- Ses attributs
 - 4 entiers pour
 - les coordonnées du coin supérieur gauche (x, y)
 - la longueur et la hauteur du rectangle
 - Une chaîne de caractères pour le nom
- Ses méthodes pour
 - Calculer sa surface et qui renvoie un entier
 - Calculer son périmètre et qui renvoie un entier
 - Le déplacer de dX et dY , qui sont passés en paramètres
 - Afficher ses caractéristiques sous la forme



Le rectangle R1 est positionné en (x, y) , a une longueur de L et une hauteur de H

Les constructeurs

- Sert à initialiser les attributs d'une classe
- Il a **exactement** le même nom que la classe
- Il n'a pas de valeur de retour
- Il peut avoir des paramètres
 - Les paramètres servent à initialiser les attributs de la classe

La surcharge

- Il est possible d'avoir dans une même classe
 - plusieurs constructeurs
 - plusieurs méthodes ayant le même nom
- Pour les différencier, il faut
 - Un nombre de paramètres différent
 - Ou au moins un type de paramètre différent
- Le type de retour ne suffit pas à distinguer deux méthodes qui ont le même nom

Pourquoi faire plusieurs constructeurs ?

- Un constructeur sert à initialiser les attributs de la classe
- Au moment où on utilise le constructeur, on n'a peut-être pas une valeur à affecter à chaque attribut
 - Plusieurs constructeurs pour initialiser les attributs en fonction de ce que l'on connaît
- Pour ce que l'on ne connaît pas, les attributs doivent être initialiser avec une valeur par défaut

Exercice 2 - Constructeurs de la classe Rectangle

- Ecrire 3 constructeurs
 - Un constructeur par défaut (sans argument)
 - Initialisation des attributs avec des valeurs par défaut
 - Un constructeur avec 3 arguments
 - Deux entiers qui représentent la longueur et la hauteur que l'on souhaite donner au rectangle
 - Une chaîne de caractères pour le nom du rectangle
 - Initialisation de x et y par une valeur par défaut
 - Un constructeur avec 5 arguments
 - 4 entiers pour les attributs x, y, longueur et hauteur
 - Une chaîne de caractères pour le nom du rectangle

Structure d'une classe

- Généralement une classe est structurée de la manière suivante
 - Liste des attributs
 - Constructeurs
 - Méthodes
 - Méthode main s'il y en a une
- **Tous ces éléments doivent être dans le bloc de la classe**

Programmation Orientée Objet

Notion de classe

Notion d'objet

Qu'est ce qu'un objet ?

- Un objet est une instance d'une classe
 - Avec une valeur propre pour chaque attribut
- Pour créer un objet, on utilise un des constructeurs de la classe correspondante précédé du mot **new**
- Pour manipuler cet objet, on déclare une variable dont le type est le nom de la classe dont on veut créer un objet

```
public class Personne
{
    String nom, prenom;
    public Personne(){...}
    public Personne(String nom)
    {
        this.nom=nom;
    }
}
```

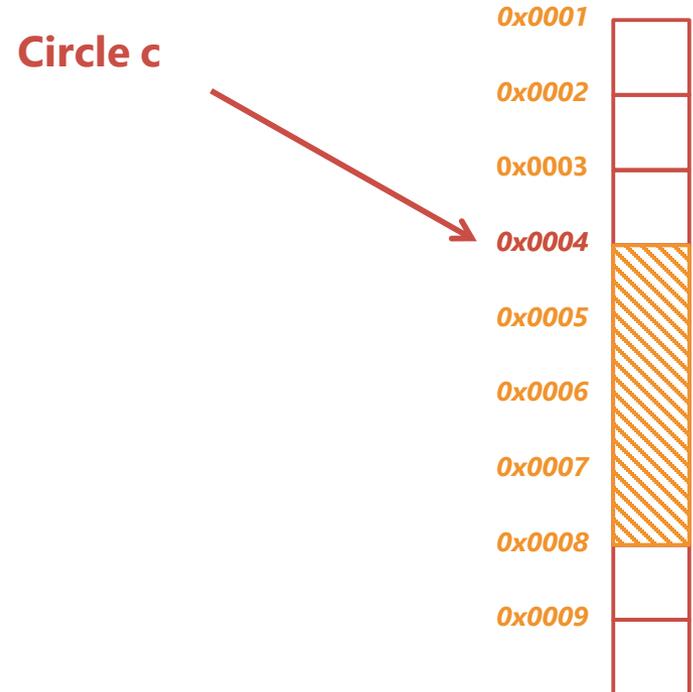
```
Personne p1;
p1 = new Personne();

Personne p2;
P2 = new Personne("Dupont");

Personne p3 = new Personne();
```

Les références

- Ce sont des variables permettant de stocker la référence à un objet
- Aucun objet n'est créé lors de la déclaration
- Il faut faire appel à un des constructeurs de la classe dont on souhaite une instance



```
Circle c;           // Déclaration d'un objet : la variable se nomme c  
c = new Circle();  // Création de l'objet associé à la référence c
```

Utiliser les membres d'un objet

- Quand un objet est créé, on a accès à ses attributs et ses méthodes ... si la visibilité de ses membres le permet.

```
public class Personne
{
    String nom, prenom;
    public Personne(){...}
    public Personne(String nom)
    {
        this.nom=nom;
    }
    public void afficheNom()
    {
        System.out.println(nom);
    }
}
```

```
Personne p = new Personne();
p.nom = "Durant";
p.prenom = "Jean Paul";
p.afficheNom();
```

Exercice 3 – Créer les objets Rectangle

- Dans la méthode main, créer les 4 rectangles

Nom	R1	R2	R3	R4
X	2	11	5	4
Y	5	9	6	2
Longueur	10	7	2	14
Hauteur	5	3	3	2
couleur	bleu	rouge	vert	Jaune

- Afficher les différents rectangles

Conventions

- Une seule classe par fichier. Même nom que la classe
- Noms des classes et interfaces commencent par une majuscule
- Noms des attributs commencent par une minuscule
- Noms des packages en minuscule
- Constantes en majuscule
- Tout doit être dans une classe
- La méthode main dans une classe Main