UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Le langage JAVA et la Programmation Orientée Objet (POO)

Mathieu RAYNAL

mathieu.raynal@irit.fr

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

Qu'est ce qu'un objet ?

- Dans le monde réel
  - Quelque chose qui a
    - Des caractéristiques
    - Des comportements
  - Chaque objet a des caractéristiques qui lui sont propres
    - Couleur
    - Position
    - …
  - Les objets de même type ont généralement les mêmes fonctions/comportements

Qu'est ce qu'un objet ?

- En programmation
  - Parallèle avec le monde réel
  - Un objet a ses propres caractéristiques
    - Rectangle 1 : coordonnées (10,10), longueur=10, hauteur=20 ;
    - Rectangle 2 : coordonnées (30,50), longueur=30, hauteur=15 ;
  - Des objets de même type ont les mêmes fonctions
    - Déplacer un rectangle
    - Calculer sa surface, son périmètre
    - Agrandir, rétrécir un rectangle

Qu'est ce qu'une classe ?

- Une classe décrit un type d'objet
  - Les caractéristiques communes à ce type d'objet
    - Pas la valeur de l'objet !
  - Les fonctions utilisables par tous les objets de ce type

- Exemple : Rectangle est un type d'objet
  - Caractéristiques
    - Coordonnées en x et y
    - Longueur
    - Hauteur
  - Fonctions
    - Déplacer, Agrandir / Rétrécir, estimer sa surface, estimer son périmètre

Une classe en JAVA

- Quelques mots clés
  - Caractéristiques = Attributs
  - Fonctions/Comportements = Méthodes
- La structure d'une classe

```
public class NomDeLaClasse
{
    typeAttribut1 nomAttribut1;
    typeAttribut2 nomAttribut2;
    …

    méthode1()
    méthode2()
    …
}
```

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 / while / switch)
- Il est délimité par des accolades

- Un bloc d'instructions peut contenir d'autres blocs d'instructions

- Une variable déclarée dans un bloc est valable jusqu'à la fin de ce bloc

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

- Décrit un comportement de la classe sous la forme d'un ensemble d'instructions
- Une méthode est définie par un nom
- Possibilité de lui passer des paramètres
  - Sous forme de variable : type + nom
  - Les paramètres sont utilisables dans la méthode grâce à leur nom
- Elle peut retourner une valeur en fin d'exécution
  - La valeur à retourner doit être précédée du mot clé return
  - Si pas de valeur retour attendue : void

Exemple de méthodes

```
public typeRetour NomDeLaMethode(type1 param1, type2 param2, …)
{
   instructions …

   return valeurRetour;
}
```

```
public void affiche(int n)
{
   System.out.println(n);
}
```

```
public int division(int n, int d)
{
    if(d==0)
        return 0;
   return n/d;
}
```

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 ambigüité, les membres d'une classe peuvent être précédés du mot **this**

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

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

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

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

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

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

Comment utiliser 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 référence définie par le type de classe et un nom

```
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();
```

Utiliser les membres d'un objet

- Quand un objet est crée, 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();
```

Des membres un peu particuliers …

- Les membres **static**
  - Ils sont communs à toutes les instances de la classe
  - Pas besoin d'instancier un objet pour les utiliser
  - Pour les utiliser : nom de la classe . nom du membre

- Les membres précédés de **final**
  - On ne peut pas modifier la valeur d'un attribut **final**