Cours 5 JAVA P.O.O

Mlle AMEUR .K
E-Mail: ameur.khadidja@univ-ouargla.dz

P.O.O

- Le concept de classes et Les objets
- Les modificateurs d'accès
- Les propriétés ou attributs
- Les méthodes
- Getter et Setter (Les accesseurs): Encapsulations
- L'héritage

Le concept de classes et objets

- ✓ C'est un ensemble de données et de fonctions regroupées dans une même Entité
- ✓ Une classe est une description abstraite d'un objet
- ✓ Instancier une classe consiste à créer un objet sur son modèle
- ✓ Une classe comporte sa déclaration, des variables et la définition de ses méthodes.

Le concept de classes et objets

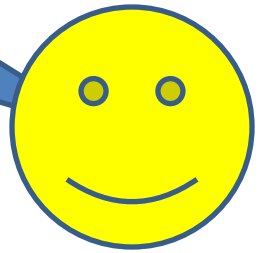
- Les classes Java peuvent posséder des *méthodes et des attributs(propriétés).*
- Les méthodes définissent les actions qu'une classe peut effectuer.
- Les attributs décrivent la classe.

Le concept de classe

La syntaxe de déclaration d'une classe

```
class nom_de_classe
 {
//Dé
    Ty
//Dé
    Ty
 }
```

L'ordre des méthodes dans une classe n'a pas d'importance. Si dans une classe, on rencontre d'abord la méthode A puis la méthode B, B peut être appelée sans problème dans A.



Le concept de classe

Exemple :Jeu de vidéo

```
class JeuDeVidéo
 {
//Déclaration des attributs
      int Score;
      int NbJoueurs ;
      ……….
//Déclarations des méthodes
       DemarerJeu(){}
       ArreterJeu(){}
       Enregistrer(){}
      ……..
}
```

Le concept de « objets »

Un objet est une instance d'une classe. Pour chaque instance d'une classe, le code est le même, seules les données sont différentes à chaque objet.

La création d'un objet : instancier une classe

- Il est nécessaire de définir la déclaration d'une variable ayant le type de l'objet désiré. La déclaration est de la forme:

nom_de_classe nom_d'instance;

Exemple :

JeuDeVidéo sodoku;

Classe: Attributs/ Propriétés

- Pour déclarer les variables constituant les attributs de la classe, nous devons choisir leur type.
  - **Les variables de classes**

    Exemple: int compteur;
  - **Les constantes**

    Exemple: final double pi=3.14 ;

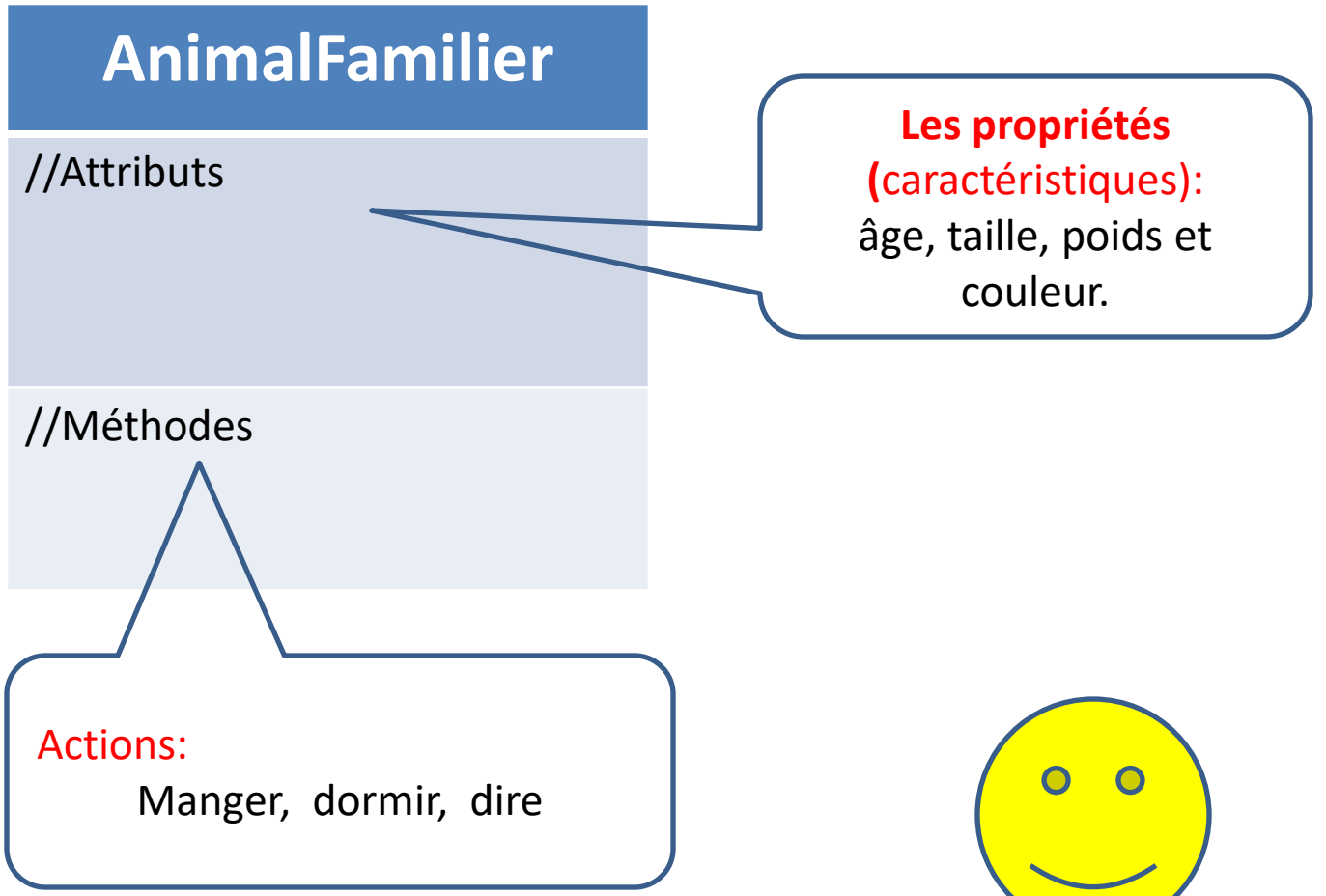
Classe: Méthodes

- Les méthodes sont des fonctions qui implémentent les traitements de la classe.
- **La syntaxe de la déclaration:**

```
type_retourné nom_méthode ( arg1, ... )
 {
// définition des variables locales et du bloc
   d'instructions
}
```

Cas d'étude:

Animaux familiers



Cas d'étude: *Animaux familiers*
Implémentation

1ère étape: création d'une nouvelle classe Java nommée AnimalFamilier

2ème étape: déclaration des attributs et des méthodes de la classe AnimalFamilier.

AnimalFamilier

//Attributs
Âge
Taille
Poids
Couleur

//Méthodes
Manger()
Dormir ()
Dire()

Les Modificateurs d’accès

- Ils se placent avant ou après le type de l'objet mais la convention veut qu'ils soient placés avant.
- Ils s'appliquent aux classes et/ou aux méthodes et/ou aux attributs.
- Ils ne peuvent pas être utilisés pour qualifier des variables locales : seules les variables d'instances et de classes peuvent en profiter.
- Ils assurent le contrôle des conditions d'héritage, d'accès aux éléments et de modification de données par les autres objets.

Les Modificateurs de Classe

Modificateur	Rôle
abstract	la classe contient une ou des méthodes abstraites, qui n'ont pas de définition explicite. Une classe déclarée abstract ne peut pas être instanciée : il faut définir une classe qui hérite de cette classe et qui implémente les méthodes nécessaires pour ne plus être abstraite.
final	la classe ne peut pas être modifiée, sa redéfinition grâce à l'héritage est interdite. Les classes déclarées final ne peuvent donc pas avoir de classes filles.
private	la classe n'est accessible qu'à partir du fichier où elle est définie
public	La classe est accessible partout

Les Modificateurs d'attribut

Modificateur	Rôle
public	L'attribut est accessible aux méthodes des autres classes
private	l'usage de l'attribut est réservé aux autres méthodes de la même classe
protected	L'attribut ne peut être invoquée que par des méthodes de la classe ou de ses sous classes
final	L'attribut ne peut être modifiée (Constante)
static	L'attribut appartient simultanément à tous les objets de la classe (comme une constante déclarée à l'intérieur de la classe). Il est inutile d'instancier la classe pour utilise
synchronized	Permet de gérer l'accès concurrent aux variables et méthodes lors de traitement de thread (exécution « simultanée » de plusieurs petites parties de code du programme)
volatile	Le mot clé volatile s'applique aux variables.

Modificateur	Rôle
public	la méthode est accéssible aux méthodes des autres classes
private	l'usage de la méthode est réservé aux autres méthodes de la même classe
protected	la méthode ne peut être invoquée que par des méthodes de la classe ou de ses sous classes
final	la méthode ne peut être modifiée (redéfinition lors de l'héritage interdite)
static	la méthode appartient simultanément à tous les objets de la classe (comme une constante déclarée à l'intérieur de la classe). Il est inutile d'instancier la classe pour appeler la méthode mais la méthode ne peut pas manipuler de variable d'instance. Elle ne peut utiliser que des variables de classes.
synchronized	la méthode fait partie d'un thread. Lorsqu'elle est appelée, elle barre l'accès à son instance. L'instance est à nouveau libérée à la fin de son exécution.
native	le code source de la méthode est écrit dans un autre langage

Héritage

- L'héritage est un mécanisme qui facilite la réutilisation du code et la gestion de son évolution. Elle définit une relation entre deux classes :
  - une classe mère ou super classe
  - une classe fille ou sous classe qui hérite de sa classe mère

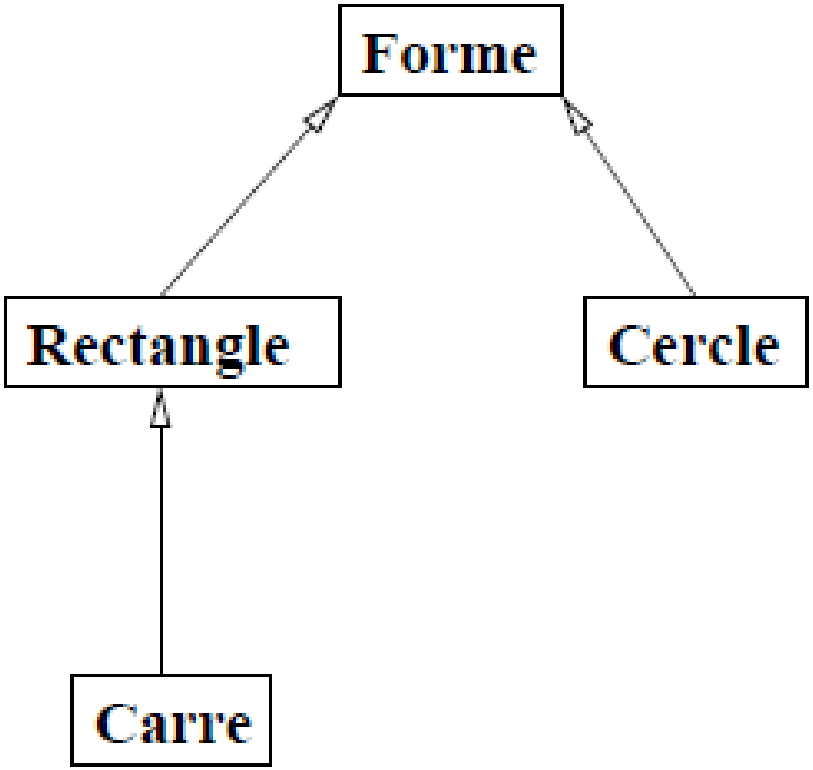
Principe de l'héritage

- les objets d'une classe fille ont accès aux données et aux méthodes de la classe parent et peuvent les étendre.
- Les sous classes peuvent redéfinir les variables et les méthodes héritées.
- Object est la classe parente de toutes les classes en Java.

La mise en œuvre de l'héritage


```
class Fille extends Mere { ... }
```

Exercice



Les concepts

- Les Modificateurs d’ accès (Classe, méthode , attribut)
- La surcharge de méthodes
- Les constructeurs/ Le destructeur
- Héritage /La redéfinition d'une méthode héritée

Classe Forme

```
abstract public class Forme
{
private String nom;
//constructeur
public Forme () {}
public Forme ( String nom) {this.nom=nom;}
//
abstract public double Perimetre() {}
abstract public double Surface () {}
public void dessiner()
{ System.out.println(Je suis+ this.nom); }
}
```

private nécessite la programmation des méthodes Getter et Setter

La surcharge d'une méthode permet de définir plusieurs fois une même méthode avec des arguments différents.

Classe Rectangle

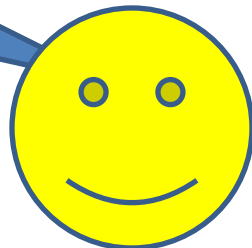

```
public class Rectangle extends Forme
{
private int largeur;
private int longueur;
//Consteructeur
public Rectangle(int x, int y) { this.largeur = x; this.longueur = y; }
public int getLargeur() { return this.largeur; }
public int getLongueur() { return this.longueur; }
public int surface() { return this.longueur * this.largeur; }
public int Perimetre() {return (this.longueur +this.largeur)*2;}
public void affiche() { System.out.println("rectangle " + longueur + "x"
    + largeur);}
}
```

Classe Carre

Remarques :

L'appel au constructeur d'une classe supérieure doit toujours se situer dans un constructeur et toujours en tant que première instruction ;

Si aucun appel à un constructeur d'une classe supérieure n'est fait, le constructeur fait appel implicitement à un constructeur vide de la classe supérieure (comme si la ligne super() était présente). Si aucun constructeur vide n'est accessible dans la classe supérieure, une erreur se produit lors de la compilation.



Des conseils sur l'héritage

Lors de la création d'une classe « mère » il faut tenir compte des points suivants :

- la définition des accès aux variables d'instances, très souvent privées, doit être réfléchie entre **protected** et **private**
- pour empêcher la redéfinition d'une méthode (surcharge) il faut la déclarer avec le modificateur **final**

Des conseils sur l'héritage 2

Lors de la création d'une classe fille, pour chaque méthode héritée qui n'est pas final, il faut envisager les cas suivants :

- la méthode héritée convient à la classe fille : on ne doit pas la redéfinir la méthode héritée convient mais partiellement du fait de la spécialisation apportée par la classe fille : il faut la redéfinir voir la surcharger. La plupart du temps une redéfinition commencera par appeler la méthode héritée(via super) pour garantir l 'évolution du code
- la méthode héritée ne convient pas : il faut redéfinir ou surcharger la méthode sans appeler la méthode héritée lors de la redéfinition.

Interfaces

- Une interface est un type, au même titre qu'une classe, mais abstrait et qui donc ne peut être instancié (par appel à new plus constructeur).
- Une interface décrit un ensemble de signatures de méthodes, sans implémentation, qui doivent être implémentées dans toutes les classes qui implémentent l'interface.
- L'utilité du concept d'interface réside dans le regroupement de plusieurs classes, tel que chacune implémente un ensemble commun de méthodes, sous un même type.

Interfaces

- Elle contient des signatures de méthodes ;
- Elle ne peut pas contenir de variables ;
- Une interface peut hériter d'une autre interface (avec le mot-clé extends) ;
- Une classe (abstraite ou non) peut implémenter plusieurs interfaces. La liste des interfaces
- La liste des interfaces implémentées doit alors figurer après le mot-clé implements placé dans la déclaration de classe, en séparant chaque interface par une virgule.

Interfaces :Example

L'interface Forme s'écrit alors de la manière suivante :

```
public interface Forme {
public int surface() ;
public void affiche() ;
}
```

Pour obliger les classes Rectangle, Cercle et Carre à implémenter les méthodes surface() et affiche(), il faut modifier l'héritage de ce qui était la classe Forme en une implémentation de l'interface définie ci-dessus :

```
public class Rectangle implements Forme
 {
...}
```

Références

- H.M.Deitel et P.J.Deitel, Comment Programmer en JAVA 4ème édition.
- X. BLANC & J. DANIEL , Le langage Java.
- Programmation Java pour les enfants, les parents et les grands-parents