Les classes et les objets

Université Virtuelle de Côte d'Ivoire

Table des matières

Objectifs

- Définir les concepts de classe et d'objet
- Illustrer les concepts d'attributs et de méthodes
- Écrire du code java pour manipuler les objets

Introduction

Java utilise la **P**rogrammation **O**rientée **O**bjet (POO). Ce type de programmation consiste en la manipulation d'objet, chaque objet est une **unité indépendante** avec une **identité unique**, des caractéristiques et un comportement juste comme les objets du monde réel.

Afin de faire usage d'un objet dans un programme, il doit exister une **classe** qui le définit. Dans cette leçon, l'apprenant aura connaissance du principe de la POO et pourra manipuler les objets sous Java

Les classes

I

Objectifs

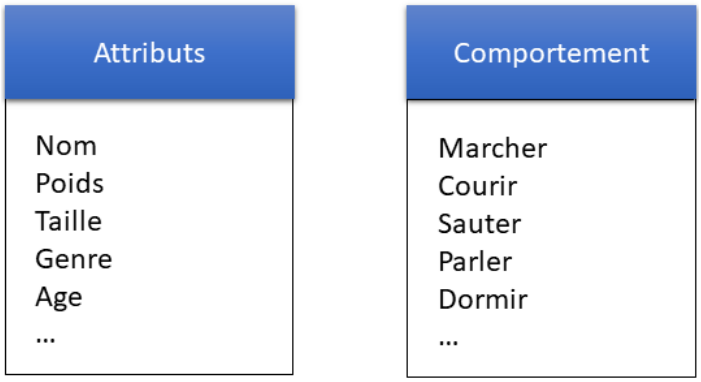
- Définir le concept de classe
- Illustrer les concepts d'attributs et de méthodes
- Faire usage des accesseurs pour protéger son code.

1. description d'une classe

Une classe décrit comment sera un objet ; c'est-à-dire que la classe est le plan, la description, ou **la définition de l'objet**. Une classe est utilisée pour créer plusieurs objets.

Ainsi la 1ere étape de la manipulation d'objet est la définition ou la création de la classe. Chaque classe possède un **nom**, des **attributs** et un **comportement**

Exemple : exemple d'attribut et de comportement



attributs et comportement d'une classe Personne

2. Le comportement ou les méthodes

en POO, les méthodes définissent le comportement. Une méthode est une collection d'instructions qui permet de faire une opération bien précise (comme les fonctions dans la programmation procédurale ou fonctionnelle).

Il est possible pour le programmeur d'utiliser les méthodes fournies par java ou de définir ses propres méthodes pour exécuter une tâche donnée.

Exemple

```
1 class maClasse{
2
```

```
    static void direBonjour(){
        System.out.println("Bonjour cher tous!");
    }

     public static void main(String[] args){
      direBonjour();
   }
}
```

Le résultat de ce programme sera : `Bonjour cher tous!`

Le code précédent déclare une méthode direBonjour() qui affiche un texte à l'écran. Pour faire appel à une méthode, il suffit d'écrire le nom de la méthode suivit des parenthèses.

méthode avec paramètre

Une méthode peut aussi avoir des paramètres. par exemple on peut modifier la méthode précédente pour qu'elle affiche le nom de celui qui reçoit le bonjour

Exemple

```
class maClasse{

    static void direBonjour(String nom){
        System.out.println("Bonjour " + nom);
    }

     public static void main(String[] args){
      direBonjour("Marius");
   }
}
```

Le résultat de ce code sera : `Bonjour Marius`

Remarque

Une méthode peut être appelé plusieurs fois dans un programme

3. Types de retour d'une méthode

Le mot clé **return** est utilisé au sein d'une méthode pour spécifier sa valeur de retour. par exemple, on peut définir une méthode nommé multiplication qui renvoie la multiplication de ses deux paramètres

```
static int multi(int val1, int val2){
    return val1 * val2
}
```

pour définir une méthode qui retourne une valeur, la **signature** de la méthode doit contenir le type de retour de la méthode. Dans notre cas **int**.

Remarque

- Si la méthode ne renvoie aucune valeur, alors la signature contiendra le mot clé **void**
- La valeur retournée par la méthode doit correspondre au type défini pour la méthode.

4. Les modificateurs d'accès

Les modificateurs d'accès permettent de définir le niveau d'accès aux classes, attributs et méthodes. Il en existe 4:

- **public** : La variable ou la méthode est accessible par n'importe quel autre classe, c'est-à-dire que toutes le classes d'un programme peuvent faire appel à cette variable ou méthode.
- **default** : Le modificateur par défaut. La variable où la méthode déclarée sans modificateur d'accès est accessible par toutes les classes d'un **package**.
- **private** : La variable ou la méthode est accessible seulement à l'intérieur de la classe où elle est définie
- **protected** : Fournis les même accès que default, en plus d'être accessible par les classes dérivées et les classes mères

Pour les classes, seulement les modificateurs d'accès public ou default peuvent être utilisées.

Exemple

```
1 public class maClass{}
2 private int variable_entiere;
3 private String chaine;
4
5 protected void methode(){}
```

5. création de classe

classe vide

La syntaxe suivante **modificateur_d_acces** *class* `nom_class` permet de créer une classe. La 1ere lettre du nom de la classe doit toujours être en majuscule

```
1 public class Voiture{
2 }
3
4 class Voiture{
5 } //crée une classe avec le modificateur d'accès default
```

classe avec attributs et méthodes

Les attributs définissent les caractéristiques de la classe et les méthodes son comportement. Les attributs et méthodes sont encapsulés dans la classe

```
1 public class voiture{
2    private String marque;
3    private int vitesse_max;
4    private String couleur;
5
6    public void demarrer(){
7       tourner_cle();
8    }
9    protected void claxonner(){
10       System.out.println("pin pin pin");
11   }
12 }
```

6. Les accesseurs (les getters et les setters)

Les **getters** et les **setters** sont des **méthodes** utilisées pour protéger les données d'une classe. **En effet il est fortement recommandé de déclarer les attributs d'une classe avec le modificateur private**.

la **méthode getter** retourne la valeur de l'attribut et la **méthode setter** prend un argument en paramètre qu'il assigne à l'attribut.

La syntaxe des getters débutent avec **get**, suivi du nom de la variable à laquelle il accède en mettant la 1ere lettre en majuscule.

La syntaxe des setters débutent avec **set**, suivi du nom de la variable à laquelle il accède en mettant la 1ere lettre en majuscule

```
1 public class voiture{
2    private String marque;
3
4    //getter
5    public String getMarque(){
6       return marque;
7    }
8
9    //setter
10   public void setMarque(String marq){
11      this.marque = marq;
12   }
13 }
```

Remarque

Le mot clé **this** est utilisé pour faire référence à l'objet courant. Ainsi this.marque récupère l'attribut marque de l'objet manipulé. Cela est d'autant plus utile si le nom du paramètre du setter est le même que celui de l'attribut.

```
public void setMarque(String marque){
      this.marque = marque ;
}
```

7. Exercice

[solution n°1 p.13*]*

Quels modificateurs d'accès sont valides pour les classes

- ☐ private
- ☐ public
- ☐ protected
- ☐ default
- ☐ hidden

8. Exercice

[solution n°2 p.13*]*

Une classe définit :

- ☐ les aléas
- ☐ les valeurs
- ☐ les attributs
- ☐ le comportement

9. Exercice

[solution n°3 p.13*]*

Complétez le code suivant pour faire appel à la méthode bonjour dans la méthode main

```
public static void main(String[] args){
    ;
}
static void bonjour(){
    System.out.println("Bonjour !") ;
}
```

10. Exercice

[solution n°4 p.13*]*

Compléter le code suivant pour définir les méthodes set et get de la class

```
Class A{
   private int x ;
   public                    (){
        return x ;
   }
   public                    (int x){
        this.x = x ;
   }
}
```

Les objets

II

Objectifs

- Définir le concept d'objet
- Définir la notion de constructeur
- Écrire du code java pour manipuler les objets

1. Définition d'un objet

Un objet est une instance d'une classe; c'est-à-dire qu'il est créé à partir de la définition de la classe. L'objet se crée à l'intérieur de la méthode main de la classe principale.

Exemple : usage de la classe Voiture dans un programme

```
1 class MaClasse{
2
3   public static void main(String[] args){
4      Voiture mercedes = new Voiture();  //création de l'objet mercedes
5      mercedes.setMarque("C180 Kompressor");
6   }
7 }
```

- **Voiture** est le type de l'objet que l'on souhaite créer
- **mercedes** est la référence sur l'objet qui sera créé. Par abus de langage, on dit qu'il est l'objet créé.
- **new Voiture()** permet d'instancier ou de créer un nouvel objet en mémoire. **Voiture()** est appelé le constructeur de la classe Voiture. Ce constructeur est créé par défaut par Java, c'est grâce à lui qu'il est possible d'instancier un nouvel objet d'une classe.
- **mercedes.setMarque()** est un accesseur de type setter qui permet de modifier l'attribut marque de notre objet nouvellement créé.

2. Le constructeur

Un **constructeur** est une méthode spéciale d'une classe qui permet d'instancier un nouvel objet d'une classe et aussi de l'initialiser.

Un **constructeur** peut être utilisé pour fournir des valeurs initiales aux attributs d'un objet.

Un **constructeur** doit avoir le même nom que sa classe.

Un **constructeur** ne doit avoir aucune valeur de retour.

Par défaut, il existe un constructeur qui permet de créer un objet vide. Le programmeur peut aussi définir un constructeur qui initialise un objet à sa création

Exemple

```
1 public class Voiture{
2
3    private String marque;
4    private int vitesseMax;
5
6    public Voiture(){}   //Le constructeur par défaut, pas besoin de le spécifier
7
8    public Voiture(String marque, int vitesseMax){
9
10       this.marque = marque;
11       this.vitesseMax = vitesseMax;
12
13    }//ce constructeur initialise le nouvel objet avec les valeurs fournies en
  paramètre
14 }
```

Remarque

- Il est possible de créer autant de constructeur en fonction des besoins du programmeur. On peut créer seulement des constructeurs qui initialise certains attributs et pas d'autres.

3. Exercice

[solution n°5 p.14]

Compléter le code suivant pour créer un objet de la classe A dans la classe B et de faire appel à sa méthode test

```
public                A{
    public void test(){
      System.out.println("Bonjour") ;
    }
}
class B{
    public static void main(String[] args){
                      obj =                A() ;
      obj.                ;
    }
}
```

4. Exercice

[solution n°6 p.14]

Complétez le code suivant pour créer un constructeur valide

```
class Personne{
      private int age ;
      public                (                age){
        this.age = age ;
     }
}
```

* *

*

Au terme de cette leçon, l'apprenant s'est familiarisé avec les concepts de classe et d'objet en java grâce auxquels il pourra construire du code pour renforcer ses capacités en POO. La prochaine leçon portera sur les classes abstraites et les interfaces qui sont des notions de la POO.

Solutions des exercices

> **Solution** n°1

Exercice p. 8

Quels modificateurs d'accès sont valides pour les classes

- ☐ private
- ☑ public
- ☐ protected
- ☑ default
- ☐ hidden

> **Solution** n°2

Exercice p. 9

Une classe définit :

- ☐ les aléas
- ☐ les valeurs
- ☑ les attributs
- ☑ le comportement

> **Solution** n°3

Exercice p. 9

Complétez le code suivant pour faire appel à la méthode bonjour dans la méthode main

```
public static void main(String[] args){
    bonjour() ;
}
static void bonjour(){
    System.out.println("Bonjour !") ;
}
```

> **Solution** n°4

Exercice p. 9

Compléter le code suivant pour définir les méthodes set et get de la class

```
Class A{
   private int x ;
   public int getX(){
       return x ;
   }
   public void setX(int x){
       this.x = x ;
   }
}
```

> **Solution** n°5

Exercice p. 11

Compléter le code suivant pour créer un objet de la classe A dans la classe B et de faire appel à sa méthode test

```
public class A{
      public void test(){
        System.out.println("Bonjour") ;
     }
}
class B{
      public static void main(String[] args){
        A obj = new A() ;
        obj.test() ;
     }
}
```

> **Solution** n°6

Exercice p. 12

Complétez le code suivant pour créer un constructeur valide

```
class Personne{
      private int age ;
      public Personne(int age){
        this.age = age ;
     }
```

```
}
```