Objet et classe en java

Structure de base

Classe

Une classe peut être définie en tant que modèle décrivant le comportement/l'état pris en charge par l'objet.

Une classe est composée des :

- Attributs sont des variables typées associées à la classe qui stockent l'état de la classe qui peut évoluer dans le temps.
- Méthodes sont des opérations que la classe peut effectuer ; Elles spécifient le comportement de la classe et impliquent généralement, les attributs de la classe.

Il ne peut y avoir qu'une seule méthode main active par projet ! celle-ci est le point de départ de votre programme. Plusieurs méthodes main peuvent cohabiter dans votre projet, mais une seule sera considérée comme le point de départ de votre programme !

```
Public class Lampe {

        // variable d'instance (attribut)
         boolean isOn;

        // méthode
        public void allumer() {
        isOn = true;
        }

        // méthode
        public void eteindre() {
        isOn = false;
        }
}
```

La classe est précédée du mot clé public, qui correspond à la portée de la classe.

- La portée détermine *qui* peut faire appel à une classe, une méthode ou une variable.
- ***La portée public*** signifie que tout le monde peut faire appel à l'élément.
- ***La portée private*** signifie que la méthode ne peut être appelée que depuis l'intérieur de la classe dans laquelle elle se trouve.

Il en va de même pour les variables. Le principe est le même que pour les méthodes. Ces variables ne seront alors accessibles que dans la classe où elles seront nées…

- **Variables locales** - Les variables définies dans des méthodes, des constructeurs ou des blocs sont appelées variables locales. La variable sera déclarée et initialisée dans la méthode et sera détruite une fois la méthode terminée.
- **Variables d'instance** - Les variables d'instance sont des variables dans une classe mais en dehors de toute méthode. Ces variables sont initialisées lorsque la classe est instanciée. Les variables d'instance sont accessibles depuis n'importe quelle méthode, constructeur ou bloc de cette classe particulière.
- **Variables de classe** - Les variables de classe sont des variables déclarées dans une classe, en dehors de toute méthode, avec le mot clé **static**.

Les constructeurs

Un constructeur est une méthode d'instance qui va se charger de créer un objet et, le cas échéant, d'initialiser ses variables de classe ! Cette méthode a pour rôle de signaler à la JVM (Java Virtual Machine) qu'il faut réserver de la mémoire pour notre futur objet et donc, par extension, d'en réserver pour toutes ses variables.

Chaque classe a un constructeur. Si nous n'écrivons pas explicitement un constructeur pour une classe, le compilateur Java construit un constructeur par défaut pour cette classe.

Chaque fois qu'un nouvel objet est créé, au moins un constructeur sera appelé. La règle principale de constructeurs, c'est qu'ils doivent avoir le même nom que la classe. Une classe peut avoir plusieurs constructeurs.

```
public class Lampe {

   // variable d'instance (attribut)
   boolean isOn;

   // constructeur par défaut
   public Lampe(){
      isOn=false;
   }

   // Ce constructeur a un paramètre, etat.
   public Lampe(boolean etat){
      this.isOn=etat;
   }

   // méthode
   public void allumer() {
      isOn = true;
   }

   // méthode
   public void eteindre() {
```

```
    isOn = false;
  }
}
```

Types de constructeur

Il existe deux types de constructeur en Java

1. **Constructeur sans argument** : Un constructeur sans paramètre est appelé constructeur par défaut. Si nous ne définissons pas de constructeur dans une classe, le compilateur crée un constructeur par défaut (sans argument) pour la classe. Et si nous écrivons un constructeur avec des arguments ou sans arguments, le compilateur ne crée pas de constructeur par défaut.

   Le constructeur par défaut fournit les valeurs par défaut à l'objet, telles que 0, null, etc., en fonction du type.

2. **Constructeur paramétré** : Un constructeur qui a des paramètres est appelé constructeur paramétré. Si nous voulons initialiser les champs de la classe avec nos propres valeurs, on utilise un constructeur paramétré.

```
// constructeur par défaut
   public Personne() {
      System.out.println("Je suis le constructeur");

// constructeur paramétré
         public Personne(String nom, int age) {

          System.out.println("Je suis le constructeur");
          this.nom = nom;
          this.age = age;

         }
```

Objets en Java

Une classe fournit un modèle pour les objets. Donc, fondamentalement, un objet est créé à partir d'une classe. En Java, le mot-clé new est utilisé pour créer de nouveaux objets.

Il y a trois étapes pour créer un objet à partir d'une classe :

- **Déclaration** : Une déclaration de variable avec un nom de variable et un type d'objet.
- **Instanciation** : Le mot clé 'new' est utilisé pour créer l'objet.
- **Initialisation** : Le mot-clé 'new' est suivi d'un appel à un constructeur. Cet appel initialise le nouvel objet.

```java
public class Lampe {

   // variable d'instance (attribut)
   boolean isOn;

   // constructeur par défaut
   public Lampe(){
      this.isOn=false;
      System.out.println("je suis le constructeur par défaut");
   }

   // Ce constructeur a un paramètre, etat.
   public Lampe(boolean etat){
      this.isOn=etat;
      System.out.println("je suis le constructeur d'initialisation");
   }

   // méthode
   public void allumer() {
      isOn = true;
   }

   // méthode
   public void eteindre() {
      isOn = false;
   }

   // méthode principale
   public static void main(String args[]){

      Lampe l1= new Lampe();
      Lampe l2 = new Lampe(false);
   }
}
```

Accès aux variables et méthodes d'instance

Les variables et méthodes d'instance sont accessibles via des objets créés. Vous pouvez accéder aux membres à l'aide de l'opérateur point "**.**".

Syntaxe :

```
// D'abord créer un objet
ReferenceObjet = new Constructor() ;
// appelez une variable comme suit
ReferenceObjet.nomVariable ;
// appelez une méthode de classe
ReferenceObjet.nomMethode() ;
```

Surcharge de constructeur

Comme les méthodes, nous pouvons surcharger les constructeurs pour la création d'objets de différentes manières. Le compilateur différencie les constructeurs en fonction du nombre de paramètres, des types de paramètres et de l'ordre des paramètres.

```
public class Personne {
   String nom;
   int age;

   public Personne() {
      System.out.println("je suis le constructeur par défaut");
   }

   // constructeur paramétré
   public Personne(String nom, int age) {
      System.out.println("Je suis le constructeur paramétré (nom - age)");
      this.nom = nom;
      this.age = age;
   }
   // constructeur paramétré
```

```
    public Personne(int age, String nom) {
        System.out.println("Je suis le constructeur paramétré (age - nom)");
        this.nom = nom;
        this.age = age;
    }
    // méthode principale (main)
    public static void main(String args[]) {
        Personne p1 = new Personne("AMINE", 32);
        Personne p2 = new Personne();
        Personne p3 = new Personne(32, "AMINE");
    }
}
```

En quoi les constructeurs sont-ils différents des méthodes en Java ?

Les constructeurs doivent avoir le même nom que la classe dans laquelle ils ont été définis, ce qui n'est pas nécessaire pour la méthode en java.

Les constructeurs ne renvoient aucun type tandis que les méthodes ont **le type de retour** ou **void** si ne renvoie aucune valeur.

Le constructeur n'est appelé qu'une fois au moment de la création de l'objet, tandis que les méthodes peuvent être appelées n'importe quel nombre de fois.

En résumé

- Une classe permet de définir des objets. Ceux-ci ont **des attributs** (variables d'instance) et **des méthodes** (méthodes d'instance + accesseurs).
- Dans une classe, on accède aux variables de celle-ci grâce au mot clé **this**.
- Une variable de classe est une variable devant être déclarée **static**.
- Il est recommandé de déclarer ses variables d'instance **private**, pour les protéger d'une mauvaise utilisation par le programmeur.
- Le ou les **constructeurs** d'une classe doivent **porter le même nom que la classe** et n'ont pas de type de retour.
- **L'ordre des paramètres** passés dans le **constructeur** doit **être respecté**.
- Les objets permettent d'encapsuler du code et des données.
- On instancie un nouvel objet grâce au mot clé **new**.