POO : Séance 2
Concepts de base de la POO

Etude de cas

Ce qu'on a vu…

- **POO**
- **Programmation Procédurale**
- **Objet**
- **Classe**
- **Attributs**
- **Méthodes**

Ce qu'on n'a pas vu…

- **Instanciation**
- **Constructeur**
- **Surcharge**
- **Etc.**

Etude de cas : Gestion des comptes bancaires

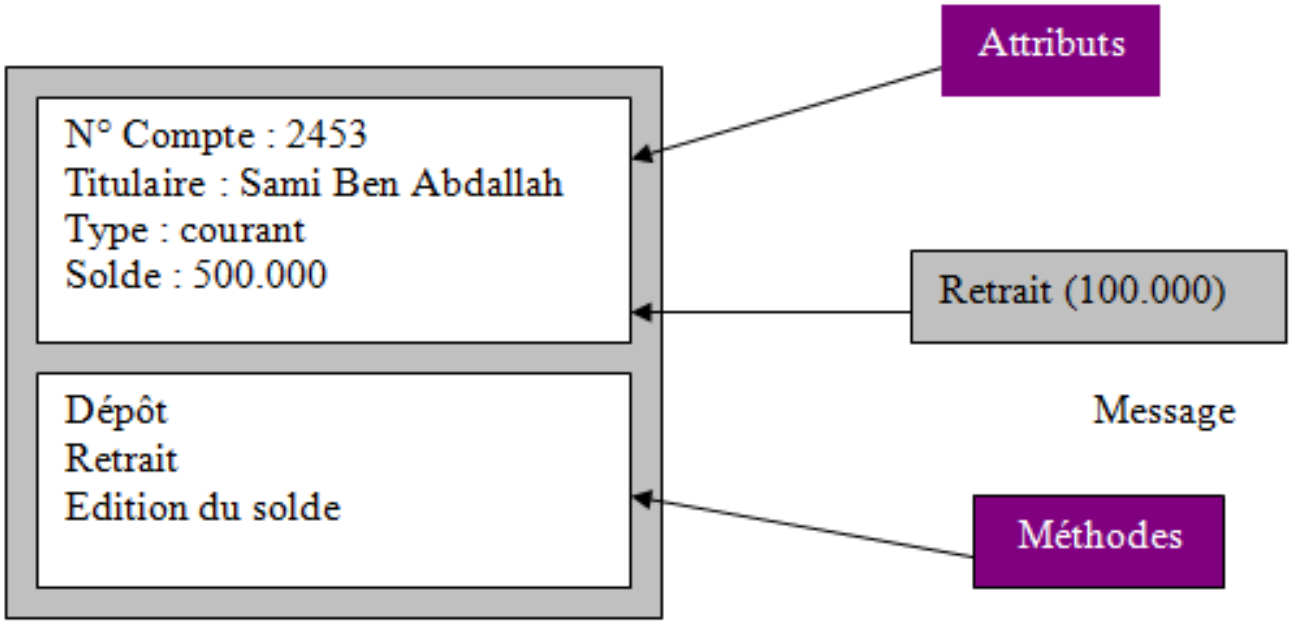
- **But** : Gérer les dépôts et les retraits d'argent.
- Les données qui caractérisent un compte :
  - Nom du titulaire
  - Numéro du compte
  - Type du compte (courant, d'épargne, etc.)
  - Solde du compte
  - Etc.
- Comment faire ?

Modélisation Orientée Objet …

• Programmer cette application consiste à définir des objets (comptes) et à leur transmettre des messages pour leur dire qu'on désire déposer ou retirer de l'argent. A eux de faire le reste.

• Ecrire une application orientée objet ➔ Transmettre des ordres d'actions à des objets (préexistants et autonomes).

Modélisation Orientée Objet …



Définir la classe

```
Class CompteBancaire
{
        //Attributs
          int numCompte;
         String titulaireCompte;
         float soldeCompte;

        //Méthodes
         void Depot (float Somme)
        {soldeCompte += Somme;}

        void Retrait (float Somme)
        {soldeCompte -=Somme;}

        void EditerSolde ()
        {system.out.println (soldeCompte);}
}
```

Rappel : Structure des objets

- Un objet est constitué d'une partie 'statique' et d'une partie 'dynamique'

❖ **Partie ' statique '**

- Ne varie pas d'une instance de classe à une autre
- Permet d'activer l'objet
- Constituée des méthodes de la classe

❖ **Partie ' Dynamique '**

- Varie d'une instance de classe à une autre
- Varie durant le cycle de vie d'un objet
- Constituée d'un exemplaire de chaque attribut de la classe

Cycle de vie d'un objet

- **Création**
  - Usage d'un Constructeur
  - L'objet est créé en mémoire et les attributs de l'objet sont initialisés
- **Utilisation**
  - Usage des Méthodes
- **Destruction et libération de la mémoire lorsque**
  - Usage (éventuel) d'un Destructeur
  - L'objet n'est plus référencé, la place mémoire occupée est récupérée

Définir la classe

```
Class CompteBancaire
{
        //Attributs
           int numCompte;
          String titulaireCompte;
          float soldeCompte;

        //Méthodes
         void Depot (float Somme)
        {soldeCompte += Somme;}

        void Retrait (float Somme)
        {soldeCompte -=Somme;}

        void EditerSolde ()
        {system.out.println (soldeCompte);}
}
```

Utiliser la classe : Création des objets

Une classe est la définition.
L'objet est sa concrétisation physique en mémoire.

```
Class MaBanque
{
        //Méthodes
          public void main (String [] args)
          {
                    CompteBancaire CPT;
                    CPT = new CompteBancaire ();
                    CompteBancaire CPT = new CompteBancaire ();

                    CPT.Depot (100.000);
                    CPT.EditerSolde();
                    CPT.Retrait (25.000);
                    CPT.EditerSolde();
          }
}
```

Déclaration

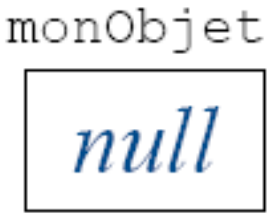
Instanciation

Déclaration + Instanciation (allocation mémoire)

Instanciation

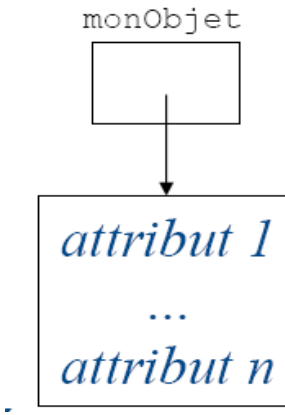
- La création d'un objet à partir d'une classe est appelée instanciation
- L'objet créé est une instance de la classe
- L'instanciation se décompose en trois phases implicites :

1. Déclaration :



- Définit le nom et le type de l'objet
- Un objet seulement déclaré vaut '**null**'

2. Création et allocation de la mémoire



- Appel de méthodes particulières : les constructeurs
- La création réserve la mémoire et initialise les attributs

3. Renvoi d'une référence sur l'objet maintenant créé

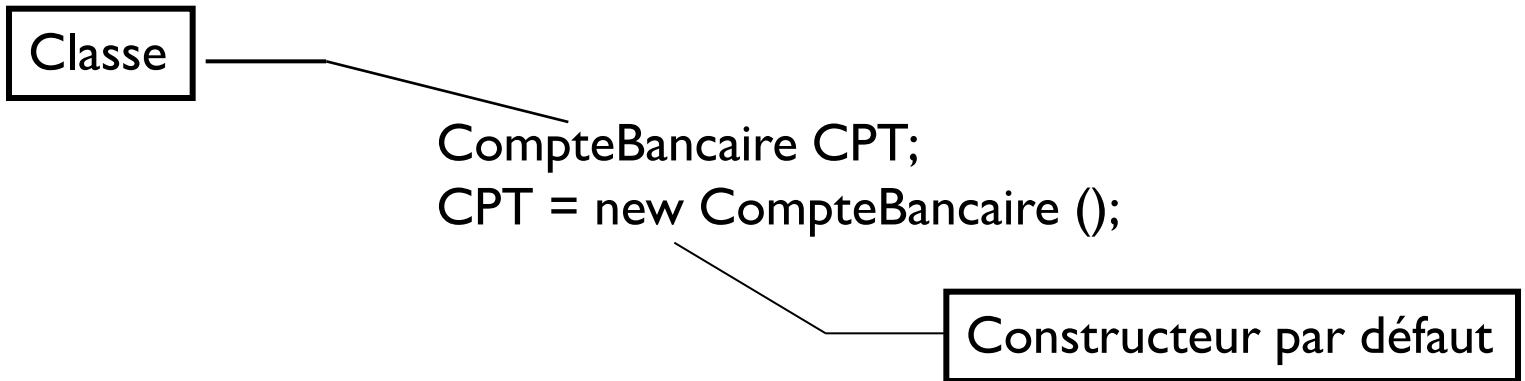
- monObjet != **null**

Constructeur

• La création d'un nouvel objet est obtenue par l'appel à **new** Constructeur (paramètres)

• Un constructeur est une méthode particulière :

1. Son nom correspond exactement au nom de la classe.
2. Il existe un constructeur par défaut qui ne possède pas de paramètre dans toute classe (si aucun autre constructeur avec paramètre n'existe)
3. Elle est automatiquement appelée à chaque fois qu'il y a instanciation d'un objet.
4. Elle n'a aucun type de retour.
5. Elle sert en particulier à initialiser les attributs de l'objet.



Le constructeur de la classe CompteBancaire

➢ **Actuellement**

• On a utilisé le constructeur par défaut sans paramètre
• On ne sait pas comment se construit le ' CompteBancaire '
• Les valeurs des attributs au départ sont indéfinies et identique pour chaque objet (numCompte, soldeCompte, etc.)

➢ **Toute classe Java possède au moins un constructeur**

• Si une classe ne définit pas explicitement de constructeur, un constructeur par défaut sans arguments et qui n'effectue aucune initialisation particulière est invoquée.

➔ **On a besoin d'un constructeur qui initialise les attributs de l'objet selon les paramètres fournis par l'utilisateur**

Exemple de constructeur pour la classe CompteBancaire

```
Class CompteBancaire
{
        //Attributs
         int numCompte;
         String titulaireCompte;
         float soldeCompte;

        //Constructeur
         CompteBancaire (int num, String titulaire, float montant)
          {
             this. numCompte = num;
             this. titulaireCompte = titulaire;
             this. soldeCompte = montant;
          }
}
```

Attention! Le constructeur par défaut n'existe plus!

Appel du constructeur

Instanciation

```
CompteBancaire CPT1 = new CompteBancaire ( 12, 'salma', 50.000)
CompteBancaire CPT2 = new CompteBancaire ( 13, 'Amira', 20.000)
```

Constructeur sans arguments (1)

- **Utilité**

• Lorsque l'on doit créer un objet sans pouvoir décider des valeurs de ses attributs au moment de la création

• Il remplace le constructeur par défaut qui est devenu inutilisable dès qu'un constructeur a été défini dans la classe

Constructeur sans arguments (2)

```
Class CompteBancaire
{
        //Attributs
          int numCompte;
          String titulaireCompte;
          float soldeCompte;

        //Constructeurs
          CompteBancaire (int num, String titulaire, float montant)
          {
             this. numCompte = num;
             this. titulaireCompte = titulaire;
             this. soldeCompte = montant;
          }
        CompteBancaire ( )
         {
             soldeCompte = 0;
        }
}
```

Constructeur sans arguments (3)

```
Class CompteBancaire

{
        //Attributs
         int numCompte;
         String titulaireCompte;
         float soldeCompte;

        //Constructeurs
        CompteBancaire (int num, String titulaire, float montant)
         {
           this. numCompte = num;
           this. titulaireCompte = titulaire;
           this. soldeCompte = montant
         }
        CompteBancaire ( )
        {
           soldeCompte = 0;
        }
}
```

```
Class MaBanque
{
        //Méthodes
        public void main (String [] args)
         {

        CompteBancaire CPT = new CompteBancaire ();
        CompteBancaire CPT1 = new CompteBancaire (12, 'laila', 10.000);

         }
}
```

Constructeurs multiples

❖ Intérêts

- Possibilité d'initialiser un objet de plusieurs manières différentes
- Il doivent seulement être différents par leurs paramètres
- On parle alors de **surchage** (overloaded) de constructeurs
- Le compilateur distingue les constructeurs en fonction :

  - de la position des arguments
  - du nombre d'arguments
  - du type des arguments

Concepts vus

Objet

Classe

Instanciation

Constructeur

Destructeur

Surcharge

Fin de la deuxième séance

Questions ?

Questions (1)

❖ Un constructeur par défaut est un constructeur défini par l'utilisateur

Non, c'est un constructeur créé implicitement

❖ Un constructeur par défaut a plusieurs paramètres

Non, un constructeur par défaut n'a aucun paramètre

❖ Le constructeur par défaut existe même si on définit notre propre constructeur

Faux, un constructeur par défaut n'existe plus si on définit un constructeur pour la classe

❖ Pour instancier un objet, on appelle le constructeur de la classe concerné

Correct, un constructeur sert à instancier une classe et créer un objet

Questions (2)

❖ Chaque objet est détruit après son utilisation

Correct

❖ L'instanciation alloue de l'espace mémoire pour l'objet créé

Correct

❖ Lorsqu'un objet est déclaré, il reçoit la valeur null

Correct

❖ Le constructeur a un type de retour

Incorrect, un constructeur n'a aucun type de retour

Questions (3)

❖ Le constructeur Voiture () est le constructeur de la classe Vehicule

Incorrect, Voiture() est le constructeur de la classe Voiture

❖ This référence l'objet en cours

Correct

❖ Lorsqu'on déclare plusieurs constructeurs, ces derniers doivent avoir le même nombre et type d'attributs

Incorrect