CLASSES ET OBJETS

Ludovic Liétard

Classes

- Une classe est un modèle de définition pour des objets :

  de même structure (même variables d 'instance)
  de même comportement (même méthodes d 'instance)

- Une classe permet d 'instancier (de créer) plusieurs objets

Objets

- Un objet n 'est l 'instance que d 'une seule classe
- Il est référencé par un identifiant (un nom)
- Son état est décrit par les valeurs prises par ses variables d 'instance
- Il peut changer d 'état
- On peut lui appliquer toute méthode d 'instance de sa classe

Variable/Méthode de Classe

- Une variable qui se trouve dans la classe est une variable de classe
- Une méthode qui concerne la classe est une méthode de classe

Déclaration d'une classe

- Les noms des variables et méthodes de classe sont précédés par le mot-clé *static*
- Une méthode de classe ne peut faire appel qu 'à des méthodes de classe ou utiliser des variables de classe
- Les noms des variables et méthodes d 'instance ne sont pas précédés par le mot clé *static*

Déclaration d'une classe

- Le nom d 'une variable d 'instance doit être précédé du mot-clé *private* (oblige l 'encapsulation)
- Le nom d 'une méthode d 'instance est précédé du mot-clé *public* (cf. plus loin)

Déclaration d'une classe

```
class <nom de la classe> {
  <définitions de variables d'instance>
            // utilisez le mot clé private
            // pour forcer l'encapsulation
  <définitions de variables de classe>
            // utilisez les mot clés static
            //  et private
  <définition de constructeurs>
  <définitions des méthodes d'instance>
            // vous pouvez utilisez  public
  <définitions de méthodes de classe>
            // utilisez le mot clé static
 }
```

Déclaration d'une méthode

```
static|public<type retourné><nom méthode>
  ( <liste d'arguments>) {
            <déclarations et instructions>
  }
```

Le mot clé `return` délivre la valeur de la méthode. Si la méthode ne délivre rien (pas d'utilisation du mot clé `return`) le type retourné par la méthode est `void` (ce qui signifie vide).

Déclaration d'une méthode

- Les paramètres des méthodes sont soit des objets, soit des variables de type primitif ou des variables de type tableau, etc… :

  les types primitifs sont passés par valeur (en entrée)

  ce qui n 'est pas de type primitif est passé par adresse (en entrée-sortie)

Envoi de messages

- On utilise la notation pointée (cf. plus loin)

nomObjet.méthode(<liste de paramètres effectifs)

Les constructeurs

- Les constructeurs portent le même nom que la classe
- Pas de type de retour ni de mot-clé void dans la signature
- Retourne implicitement un objet (pas d'instruction return)
- L 'utilisation d 'un constructeur est introduit par le mot-clé *new* :

  nom_objet = *new* nom_constructeur (paramètres)

Objet courant

- Dans les instructions d'une méthode, l'objet courant (à qui on envoie le message) est référencé par le mot-clé *this*
- Il est utilisé pour rendre explicite l'accès aux propres attributs ou pour envoyer l'objet courant en paramètre d'une méthode ou pour lui envoyer un message

Exemple

```
class Personne
{
        // variables d'instance encapsulées donc private

        private String nom;
        private int age;

        // variable de classe
        // donc static

        static private int nb=0;
```

Exemple

```
// constructeurs : de même nom que la classe
// éventuellement plusieurs constructeurs

        public Personne(String lenom, int lage){
                nom=lenom;
                age=lage;
                 nb++;
          }

        public Personne(){
                nom="";
                age=0;
                nb++;
        }
```

Exemple

```
// Méthodes d'instance
        public int obtenirAge(){
                return this.age;
        }
        public String obtenirNom(){
                return this.nom;
        }
        public void modifierNom(String nouveauNom){
                this.nom=nouveauNom;
        }
        public void modifierAge(int nouvelAge){
                this.age = nouvelAge;
        }
```

Exemple

```
// Methode de classe

        static int obtenirNb(){
                return nb;
        }

}
```

La méthode main

- Il existe une méthode de classe particulière: la méthode main
- Elle constitue le point d 'entrée (la commande java exécute cette méthode)
- On peut la définir dans une classe à part

La méthode main

- Signature de la méthode :

public static void main(String[] args)

args : tableau d'objets String des arguments de la ligne de commande

args.length() : nombre des arguments

les arguments sont donc :
args[0], args[1], …, args[args.length()-1]

La notation pointée

- Pour appliquer une méthode : (si cela est possible)

NomDeClasse.MethodeDeClasse(…)
NomD'Objet.MéthodeD'Instance(…)

- Pour obtenir une variable : (non encapsulée)

NomDeClasse.VariableDeClasse
NomD'Objet.VarialeD'Instance

La notation pointée

- Exemple :

System.out.println("Bonjour ");

Classe : System
Variable de classe : out (instance de PrintStream)
Méthode d'instance de la classe PrintStream : println

On envoie le message println("Bonjour") à la variable de classe out (définie dans la classe System).

Gestion de la mémoire

- La libération de la mémoire allouée aux objets est automatique
- Le ramasse miette (garbage collector) est un processus qui s'exécute en tâche de fond et qui gère automatiquement les ressources mémoires