UEF 1 : Informatique & Programmation

Faculté des Sciences de Nice

DEUG 2000-2001

Jérôme DURAND-LOSE

Sandrine JULIA

Jean-Paul ROY

COURS 4

Les « objets »

et leurs

« méthodes d'instance »

♦ Jusqu'à présent, les **classes** représentaient des programmes pouvant communiquer entre eux. Ces classes contenaient des *méthodes de classe* [« `static` »].

♦ Second point de vue : une classe peut être conçue comme un « moule à objets », avec cette fois des *méthodes non statiques*, les « *méthodes d'instance* »...

Programmation « orientée objet »

Mais qu'est-ce qu'un « objet » ???

Approche intuitive du vocabulaire objet

♦ Exemple d'*objet* de la vie courante : un poste TV.

♦ Il appartient à la *classe* de tous les postes TV. On dit que c'est une *instance* de cette classe.

→ *une concrétisation particulière*

♦ Un poste TV est un *objet* ayant une *identité*, un *état*, ainsi que du *savoir-faire* : changer de chaîne, baisser le son...

méthodes d'instance *données privées*

♦ Pour activer les *méthodes* d'un poste TV, il suffit de lui envoyer un *message* [par la télécommande]...

Exemple simple : la classe des machines

♦ Décrivons la classe `Machine`. Une instance de cette classe sera donc un objet de type `Machine`.

♦ Chacun de ces objets aura un seul champ « nom » sous la forme d'une variable privée à l'objet, et dont la valeur sera ici une instance de la classe `String`.

↓

oui, les chaînes sont des objets...

des machines plus complexes :

nom	en marche ?	prix	poids
`"HAL 90"`	`oui`	`350000`	`1284`
`"inconnu"`	`non`	`0`	`0`
`"FUJI Z4"`	`non`	`15.35`	`1.8`

♦ Dans ces exemples, l'état d'une machine est décrit par 4 « champs » [données privées].

♦ Soit `comp` la machine dont le nom est `"HAL 90"`, qui est allumée, vaut `350000` et pèse `1284`.

♦ La description [simplifiée] d'une classe d'objets comprend 3 sections :

```
/* fichier Machine.java */
class Machine
{
    les données privées d'une machine

    les constructeurs

    les méthodes d'instance
}
```


Les données privées [*l 'état*] d'un objet

♦ Chaque objet de la classe `Machine` dispose d'un nom qui lui appartient en propre. Ce nom est contenu dans une variable privée de type `String` :

↓

```
private String nom;
```

♦ Il sera éventuellement possible de modifier la valeur de cette variable `nom` par la suite [comment ?]...

↓

en le demandant poliment à l'objet via une méthode d'instance...

Les constructeurs

- Et oui, il faut créer, engendrer, les instances avant de les utiliser !...
- Ils vont permettre de construire un nouvel objet, nouvelle instance de la classe Machine.

new

```
Machine (String n)
{   nom = n;
}

Machine ( )
{   nom = "inconnu";
}
```

- Exemple d'utilisation : TestMachine ligne n°22

Rappel : les méthodes « static »

- Méthode *static* = méthode *de classe* . Exemples :

```
r = Math.sqrt(x);
```

on envoie le message sqrt à la classe Math avec un argument x de type « double »

```
Console.println("r vaut " + r);
```

on envoie le message println à la classe Console avec un argument de type « String »

Les méthodes d'instance (1)

- Les méthodes d'instance s'adressent directement à un objet et non à sa classe :

```
System.out.println("Salut !");
```

on envoie le message println à l'objet System.out avec un argument de type « String »

Les méthodes d'instance (2)

- Les premières vont agir sur l'état [variables privées] de l'objet :

retourne le nom courant

```
String leNom ( )
{   return nom;
}
```

change le nom !

```
void changeNom (String nouveauNom)
{   nom = nouveauNom;
}
```

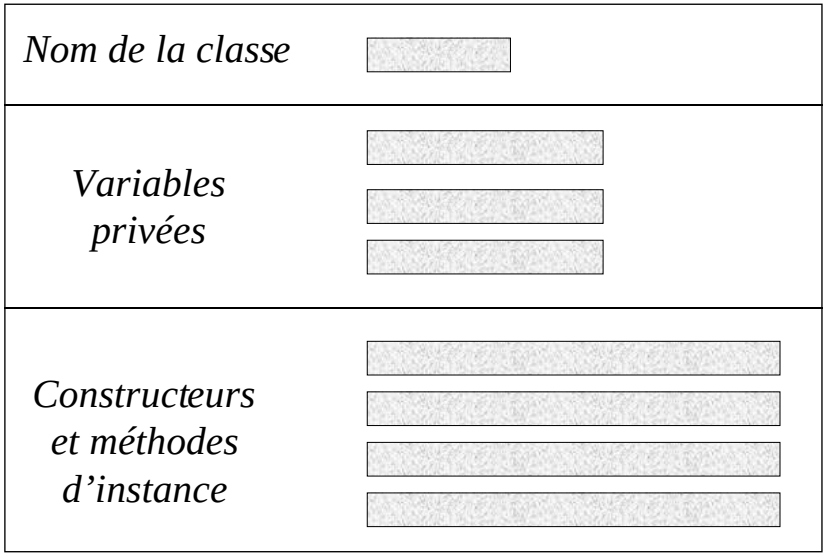
Les méthodes d'instance (3)

♦ Diverses méthodes permettront de demander à l'objet d'effectuer une action, de procéder à un calcul :

```
void décrisToi ( )
{   Console.print("Je suis une machine");
    Console.print(" et je me nomme ");
    Console.println(nom);
}
```

sous-entendu : mon nom (celui de l'instance courante)...

- Un objet -



Exemple d'utilisation : TestMachine.java

```
import unsa.Console;

class TestMachine
{  public static void main(String[] args)
   {  Machine comp = new Machine("HAL 90");
      comp.décrisToi( );
      Console.println("Nouveau nom !");
      comp.changeNom("Pokémon");
      comp.décrisToi( );
      Console.print("comp se nomme : ");
      Console.println(comp.leNom( ));
      System.exit(0);
   }
}
```


Compilation puis exécution

```
Je suis une machine et je me nomme HAL 90
Nouveau nom !...
Je suis une machine et je me nomme Pokémon
comp se nomme : Pokémon
```

```
/*
     Machine.java  --- TC DEUG octobre 2000 --- Cours n°4
     La classe Machine. Une machine est pour l'instant seulement pourvue
     d'un nom, que l'on peut consulter et modifier via des méthodes d'instance,
     et sachant se décrire.
*/

1    class Machine
2    {    private String nom;                    /* l'unique donnée privée */

3         Machine(String n)                      /* constructeur */4
4         {    nom = n;
5         }

6         Machine( )                             /* un autre constructeur */
7         {    nom = "inconnu";
8         }

9         String leNom( )                        /* un accesseur */
10        {    return nom;
11        }

12        void changeNom(String nouveauNom)      /* un modificateur */
13        {    nom = nouveauNom;
14        }

15        void décrisToi( )                      /* un descripteur */
16        {    System.out.println("Je  suis  une machine nommée : " + nom);
17        }
18   }
```

```
/*
     TestMachine.java  --- TC DEUG octobre 2000 --- Cours n°4
     La classe TestMachine. Cette classe ne décrit pas d'objets, elle sert
     à tester la classe Machine, et ne contient qu'une méthode de classe,
     la méthode principale main(...).
*/

19   import unsa.Console;                       /* pas vraiment utile ici... */

20   class TestMachine
21   {    public static void main(String[] args)
22        {    Machine comp = new Machine("HAL 90");
23             comp.décrisToi( );
24             Console.println("Nouveau  nom !...");
25             comp.changeNom("Pokémon");
26             comp.décrisToi( );
27             Console.println("comp  se nomme : ") ;
28             Console.println(comp.leNom( ));
29             System.exit(0);        /* à cause de la classe Console */
30        }
31   }
```