
JAV - TD 2

Bases du langage
Caractéristiques du langage
Le mot-clé static

Bases du langage

• Types dits primitifs

(pas classes donc pas majuscules!)

- byte : 1 octet
- short : 2 octets
- int : 4 octets
- long : 8 octets
- float : 4 octets
- double : 8 octets
- boolean : true false
- char : 2 octets (Unicode)

• Opérateurs

- Arithmétique : = + - * / %
- Relationnels : < > <= >= == !=
- Logiques : ! && ||
- Incréments : ++ --
- Bit Wise : & | ^ ~ >> << >>>
- Affectation : = += -= *= /= %=
- Conditionnel : ?:

Bloc de programmation

- { } n 'importe où, mais la plupart du temps autour d'une classe ou d'une méthode
- Aucune instruction/déclaration en dehors d'un bloc
- Une variable peut être déclarée n'importe où dans un bloc. Elle possède la portée de ce bloc.

Exemples de blocs de programmation

```
class Test1 {
    private int i;
    public void fonction(){
        int i=5;
        System.out.println("I1:"+i);
    }
    int j=10;
    System.out.println ("J: "+j);
        System.out.println("I2:"+i);
    }
    System.out.println("I3:"+i);
}

    public void fonction2(){
        System.out.println("I:"+i);
    }
}
```

```
class Test2 {
    static {
        System.out.println("coucou");
    }
}
```

```
class Test3 {
    {
        System.out.println("coucou");
    }
}
```

Les structures de contrôle et expressions

- Essentiellement les mêmes qu'en C
 - if, switch, for, while, do while
 - Fonctionnement du switch sur type primitif uniquement

- Constantes et booléens

```
public static final boolean STATE_OK=true;
protected static boolean myState=STATE_OK;
if (MaClasse.myState) {...
}
```

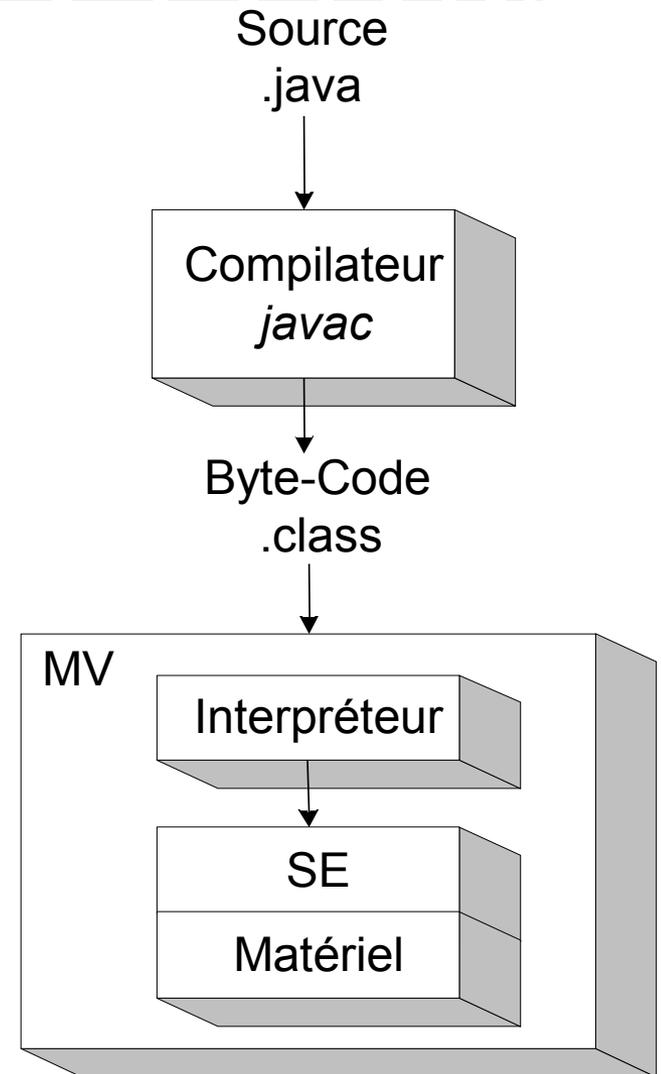
en C

```
if (valeur==0) { /*Etat OK*/
}
```

- Eviter les constantes String (car ce sont des objets) (== ne fonctionne pas, il faut utiliser .equals)

Machine Virtuelle : Emulateur

- Byte-Code/P-Code/J-Code
 - opcode : 1 octet pour l'instruction
 - 0,n opérandes
- MicroProcesseur logiciel
 - Jeu d'instructions
 - Registres (pc, optop, frame, vars)
 - Pile, Heap
 - Ramasse-miettes
 - Espace de stockage des méthodes
 - Tas de constantes
- ==> Compilé ou Interprété



Une première classe MV

- Ecrire une classe Test1 représentant des objets ayant un attribut interne de type entier (int) appelé val.
- Compiler cette classe (le compilateur s'appelle javac)
- Quels sont les fichiers générés ?
- Lancer la commande **javap -c** sur la classe Test1. Que voit-on ?
- Exécuter cette classe dans la machine virtuelle. Que se passe-t'il et pourquoi

Une classe cliente

- Ecrire une classe Test2 cliente de la première (qui utilise des référence sur des objets de la première) qui possède deux attributs (att1, att2) de type Test1.
- Compiler cette classe
- Exécuter cette classe. Que se passe t'il ?
- Lancer la commande javap -p sur la classe Test2. Que voit-on ?

Le Main

- Ajouter dans Test2 une méthode *main(String[])* ayant la signature suivante :
- `public static void main(String [] arg)`
- A quoi sert le paramètre `arg` de cette méthode ?
- Compiler cette classe
- Exécuter cette classe : que se passe t'il ?

Objets impossibles

- Dans la méthode *main(String[])* de Test2, déclarez une variable t1 de classe Test1 mais instanciez un objet de type Test2
- Compiler. Que se passe-t'il ?
- Dans la méthode *main(String [])* déclarez un objet de type Toto, instanciez le.
- -Compiler. Que se passe t'il ?

“l’œuf ou la poule?”

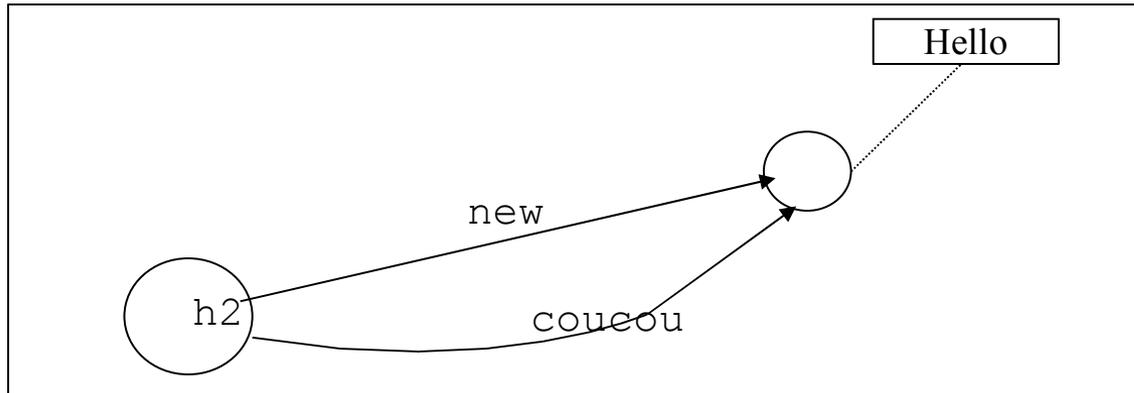
- Chaque objet provient d'une classe dont on fabrique une instance

```
Object o1=new Object();
```

- Mais comme “tout est objet”, comment fait-on pour avoir le premier objet ?
 - Soit il existe un premier objet fourni au démarrage de la machine virtuelle au démarrage
 - Soit il existe des objets toujours disponibles
 - ...

La classe vue comme un objet

- Toutes les classesinstanciées sont connues de la machine virtuelle. Elles pourraient donc être également utilisées comme des objets.



```
...  
Hello h2=new Hello();  
Res r=h2.coucou();  
...
```

```
class Hello{  
    public void coucou(){}  
}
```

Comment écrire une méthode coucou dans la classe hello, mais dont le destinataire ne soit pas une des instances, mais la classe ?

Le mot clé static

- On ajoute un mot-clé à la signature des méthodes

```
public static coucou1(){...} /*Cette méthode est connue de la classe */  
public void coucou2(){ ... } /*Cette méthode est connue des instances-objets*/
```

Comment l'objet h2 fait-il pour invoquer une méthode statique sur la classe Hello?

- On adapte le destinataire du message : Hello.coucou() ==> Indique que coucou() est relatif à la classe et non pas à une de ses instances

Hello
+static coucou1()
+coucou2()

```
public class Hello{  
    public void static coucou1(){...}  
    public void coucou2(){...}  
}
```

```
...  
Hello h2=new Hello();  
h2.coucou2();  
Hello.coucou1();  
...
```

Exercice

- Ecrire le code d'une classe Client, utilisant une classe DivisionO réalisée en utilisant des méthodes d'objets
 - Compiler
- Ecrire une classe DivisionO en objet
 - Compiler, Lancer
- Modifier le code de la classe Client pour utiliser une classe DivisionS qui présente des méthodes statiques
 - Compiler
- Ecrire une classe DivisionS en statique
 - Compiler

Conséquences directes

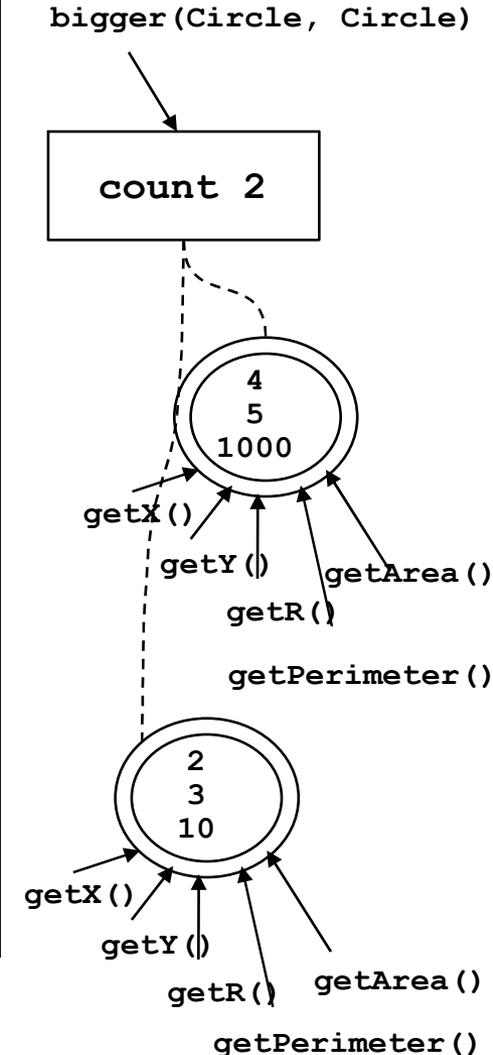
- La classe est également vue comme un objet
- Les attributs et les méthodes statiques sont associés à la classe et non à chaque objet
- Toutes les instances de la classe peuvent être vues comme partageant la valeur des attributs "static"
- Il n'est pas nécessaire d'instancier une classe pour accéder à ses éléments statiques
- La notion de static peut être perçue comme représentant les valeurs communes à toutes les instances d'une même classe

Exemple du mot clé static

```
public class Circle {
    public static int count = 0;
    public static double PI = 3.14;
    protected double x, y, r;
    public Circle(double r){this.r = r; count++;}

    public static Circle bigger(Circle c1, Circle c2){
        if (c1.r > c2.r) return c1; else return c2;
    }
}

public class UneClasseExemple{
    public UneClasseExemple(){
        Circle c1 = new Circle(10);
        Circle c2 = new Circle(20);
        int n = Circle.count; // n = 2
        Circle c4 = Circle.bigger(c1, c2); // c4 = c2
    }
}
```



Rappel sur : “Pourquoi le main est static ?”

```
public class HelloWorld {  
    public static void main(String [] arg){  
        System.out.println("Hello, World !");  
    }  
}
```

Static main(String [] arg)



HelloWorld.java

Compilateur
javac HelloWorld.java

HelloWorld.class

Machine Virtuelle
java HelloWorld

HelloWorld

Exercice

- Décrire en terme d'objets et de messages la commande d'affichage
`System.out.println("coucou");`
- Lancer le programme de gestion des divisions.
 - Quelles sont les différences entre approche statique et l'autre ?
 - A quoi ressemble l'approche statique ?

Pourquoi ?

- Pourquoi ne pas tout faire en statique ?
 - Parce qu'en programmation objet on manipule des objets qui sont beaucoup plus faciles à réorganiser
- Qu'est ce qui est statique ?
 - Le main (on n'a pas le choix, il n'y a pas encore d'objet)
 - Les constantes :

```
public static final int ETAT_STABLE=1;
...
toto.setState(UneClasse.ETAT_STABLE);
```
 - Les fonctions communes de service
 - Dans ce cas on essaye de les regrouper dans une classe de service