

Cours1 +Cours 2

- Structure d'un programme et Compilation
- Notions de classe et d'objet
- Syntaxe

Programmation Orientée Objet

Un ensemble d'objet qui communiquent

Pourquoi POO

- Conception

 - abstraction sur les types

 - outils de modélisation

- Réutilisation et extension des composants

 - programmation modulaire

C++, Java,...

- Le langage Java
Charon (Edition Hermes)
- Java 5
R. Chevallier (Edition Pearson)
- La programmation objet en Java
M.Divay (Edition Dunod)
- Exercices en java
C. Delannoy (Edition Eyrolles)
- Effective Java
G. Steele (Edition Addison Wesley)
- Java tutorial
<http://java.sun.com/tutorial>
- Absolute Java
W. Savitch (Edition Pearson)

JAVA: langage orienté objet

portable sans compilation du fichier source

Programme en Java ----> Compilateur Java ----> Bytecode

Bytecode ----> Interpréteur Java ----> Exécution

Interpréteur Java = machine virtuelle Java (JVM)+
bibliothèques de classes (API)

exécution à distance

interprétation est plus lente que l'exécution en langage machine

Il existe 2 types de programmes avec la version standard de Java

- les applications (stand alone program):
exécutées sous le contrôle d'un Système d'Exploitation

méthode main est appelée par JVM

- les applets: applications chargées par un navigateur

*intégrées à une page HTML,
visualisées par un navigateur ou appletviewer*

Java est livré avec un grand ensemble de classes (bibliothèque de classes) appelé **API** (Application Programming Interface).

Le pseudocode est relativement petite, car les bibliothèques de classe ne sont liées à l'exécution

java.lang (les classes plus centrales du langage, classe Object)

java.io (classes pour I/O)

java.net (programmation à travers réseau)

java.awt (composants graphiques de base)

javax.swing (classes pour interfaces graphiques)

java.applet (applet, application téléchargeable du Web)

<http://www.javasoft.com>

trois plateformes d'exécution (ou éditions)

- Java Standard Edition (J2SE / Java SE) :
environnement d'exécution et ensemble complet d'API
sert de base en tout ou partie aux autres plateformes
- Java Enterprise Edition (J2EE / Java EE):
environnement d'exécution reposant intégralement sur Java SE
pour le développement d'applications d'entreprises
- Java Micro Edition (J2ME / Java ME):
environnement d'exécution et API pour le développement
d'applications sur appareils mobiles et embarqués

1996 Java Development Kit JDK 1.0.1(Sun))

....

2006 (déc.) Java SE 6.0

2009(déc.) Java EE 6.0

2010 (janv.) rachat de Sun par Oracle

2011 (juillet) Java SE 7.0

La documentation au format HTML des API de Java est fournie séparément. Elle est indispensable pour obtenir des informations complètes sur les classes

Les outils Sun: javadoc, jar, ...

Les outils libres: netbeans, eclipse, ant,

```
import java.io.*;
public class Program1
{
    public static void main(String[] args)
    {
        System.out.println( "hello");
    }
}
```

Nom du fichier: Program1.java

Compilation: javac Program1.java

Pour chaque classe=un fichier compilé avec le même nom suivi de .class

Exécution: java Program1

String et System sont des classes définies dans java.lang
(inclus par défaut)

import java.io.* :inclure les classes de la bibliothèque I/O

Fichier source avec extension **.java**

Il peut contenir plusieurs classes mais au plus une classe avec modificateur de visibilité **public**

Cette classe doit contenir la méthode main

```
public static void main(String[] args)
{
    //données et instructions
}
```

Le nom du fichier source est celui de cette classe suivi de l'extension **.java**

Une **classe** est définie par un ensemble **d'attributs** (champs, variables, propriétés) et d'un ensemble de **méthodes** (fonctions).

Une classe génère en général un **modèle d'objet**.

Objet = une instance (réalisation) de classe

Il existe des règles de **visibilité** entre les classes et entre les attributs et les méthodes qu'elles contiennent.

Les classes elles-mêmes sont regroupées en unités logiques cohérentes appelées **paquetages** (package).

```

import ...;    //mettre les noms des classes prédéfinies à utiliser
class ...     //mettre le nom de la classe
{....       //mettre les déclarations des attributs
....
.....       //mettre le code des méthodes
}           //fin de la déclaration de la classe
....
public class ...//mettre comme nom de classe le nom du fichier texte
{
    public static void main(String args[]) //écriture obligatoire
    {..... //ici le code de la fonction main
        .....
    }
}

```

Déclaration d'une classe

- Le nom de la classe est spécifié après le mot clé « class »
- Le corps de la classe est délimité par des accolades
- On définit dans le corps les attributs et les méthodes qui constituent la classe

```
class Test {  
    < corps de la classe >  
}
```

Définition d'une méthode

- Une méthode est constituée de :
 - D'un nom
 - D'un type de retour
 - De paramètres (éventuellement aucun)
 - D'un bloc d'instructions
- Un paramètre est constitué :
 - D'un type
 - D'un nom
- « void » est le mot-clé signifiant que la méthode ne renvoie pas de valeur

```
class Test {  
    int calculer (int taux, float delta) {  
        < corps de la méthode >  
    }  
}
```

Déclaration d'une variable

- Une variable possède un type et un nom
- Le type peut être un type de base ou une classe
- L'initialisation d'une variable peut se faire au moment de la déclaration

```
{  
    int compteur;  
    int indice = 0;  
  
    Voiture golf;  
    Voiture twingo = new Voiture();  
}
```

Portée d'une variable

- La portée d'une variable s'étend jusqu'à la fin du bloc dans lequel elle est définie

```
{  
  {  
    int compteur;  
    ...  
    // compteur est accessible  
  }  
  
  // compteur n'est plus accessible  
}
```

```

class Personne
{
    private String nom;
    private int annee_n;           //les attributs
    private int salaire;

    public Personne(String n, int a, int s)
    {
        nom=n;
        annee_n=a;
        salaire=s;
    } //méthode constructeur

    public void affiche()
    {
        System.out.println(nom+" "+ annee_n+" "+salaire);
    }

    public void calcul_age()
    {
        int age=2006-annee_n;
        System.out.println(" age=" +age);
    }
}

```

```
public class Person1
{
public static void main(String args[])
    { Personne p1= new Personne("dupont ",1961, 1700);
    Personne p2= new Personne("bernard",1981, 1400);
    p1.affiche();
    p2.affiche();
    p1.calcul_age();
    p2.calcul_age();
    p1.calcul_age();
    }
}
```

Les attributs sont *private* : ils ne seront accessibles que par les méthodes de l'objet

Les méthodes sont *public* : elles peuvent être appelées depuis une autre classe (Par exemple par la méthode main)

Chaque classe a une méthode constructeur qui

- porte le même nom que la classe
- est déclarée public
- ne retourne rien

```
p1= new Personne("dupont",19561,1700)
```

new est une instruction qui

- alloue la mémoire pour l'objet
- appelle la méthode constructeur
- retourne l'adresse de l'objet ainsi construit

Noms de classes commencent par **une majuscule**

Noms de méthodes commencent par **une minuscule**

2 types de variables

- ordinaires :

entiers: byte(8 bits), short (16) ,int (32), long(64)

réels : float (32) double(64)

caractères : unicode

booléen

- type classe : a pour valeur l'adresse de l'objet

Types de données primitifs

Display 1.2 Primitive Types

TYPE NAME	KIND OF VALUE	MEMORY USED	SIZE RANGE
<code>boolean</code>	<code>true</code> or <code>false</code>	1 byte	not applicable
<code>char</code>	single character (Unicode)	2 bytes	all Unicode characters
<code>byte</code>	integer	1 byte	-128 to 127
<code>short</code>	integer	2 bytes	-32768 to 32767
<code>int</code>	integer	4 bytes	-2147483648 to 2147483647
<code>long</code>	integer	8 bytes	-9223372036854775808 to 9223372036854775807
<code>float</code>	floating-point number	4 bytes	$-3.40282347 \times 10^{+38}$ to $-1.40239846 \times 10^{-45}$
<code>double</code>	floating-point number	8 bytes	$\pm 1.76769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$

le transtypage (cast):

byte -->short-->int-->long-->float-->double

```
byte b; short p; int n; long q; float x; double y;
```

```
b=n; erreur
```

```
b=25;
```

```
b=500; erreur
```

```
x=2*q;
```

```
p=b*b;
```

(int) 2.9 *vaut* 2

```
int i= (int) 5.5;
```

Syntaxe similaire au C

- **Commentaire** : // commentaire jusqu'à la fin
/* Commentaire */
- **Constantes** : variables précédées par *final*
final double PI = 3.1415926535;
- **Opérateurs arithmétiques** : +, -, *, /, +=, -=, *=, /=
%(modulo), ++, --
- **Opérateurs de comparaison** := >, >=, <, <=, ==, !=
- **Opérateurs logiques** : &&, ||, !
- **Opérateurs de décalage** : <<, >>
- **Règles de précedence** :
 - , ++, -- (opérateurs avec 1 seul argument)
 - *, /, % (opérateurs multiplicatifs avec 2 arguments)
 - +, - (opérateurs additifs avec 2 arguments)

Classe String

- Il n'y a pas de type primitif (ordinaire) string
- String: une classe prédéfinie

String a = "bonjour";

- concaténation: +

a+ "Bonjour"; *bonjourBonjour*

```
String greeting = "Hello";  
int count = greeting.length();  
System.out.println("Length is " + greeting.length());
```

Objet System.out

Méthodes : **println**, **print**

```
System.out.println("The answer is " +  
42);
```

à partir de la version 5.0 **printf**

```
double prix = 19.8;  
System.out.print("$");  
System.out.printf("%6.2f", prix);
```

\$ 19.80

Display 2.1 Format Specifiers for System.out.printf

CONVERSION CHARACTER	TYPE OF OUTPUT	EXAMPLES
d	Decimal (ordinary) integer	%5d %d
f	Fixed-point (everyday notation) floating point	%6.2f %f
e	E-notation floating point	%8.3e %e
g	General floating point (Java decides whether to use E-notation or not)	%8.3g %g
s	String	%12s %s
c	Character	%2c %c

POO

Classe Scanner

à partir de la version 5.0

```
import java.util.Scanner
Scanner keyboard = new Scanner(System.in);
ou
Scanner x = new Scanner(System.in);

int n = keyboard.nextInt();
double d1 = keyboard.nextDouble();
String word1 = keyboard.next(); (délimiteur: espace)
String line = keyboard.nextLine();
String line = keyboard.nextLine();
```

```
Scanner keyboard = new Scanner(System.in);  
int n = keyboard.nextInt();  
String s1 = keyboard.nextLine();  
String s2 = keyboard.nextLine();
```

si on tape

2

bonjour ca va

salut

quelles sont des valeurs de n, s1, s2 ?

if (*Expression booléenne*)

{ }

else

{ }

else facultatif

max = (n1 > n2) ? n1 : n2;

switch (i){

case 1: a=5; break;

case 2: case 3: a=6; break;

default : a=7; //facultatif

break; //facultatif

}

- *for (i=0;i<10;i++){ }*

- *while (Expression booléenne)*
{ }

- *do{*
} while (Expression booléenne)

break: *sortir de la boucle ou switch le plus proche*

continue: *annule l'itération en cours*

System.exit(n): *sortir du programme*
n=0 *exécution normale*