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. Sawitch (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

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

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 téléchargable du Web)

http://www.javasoft.com


```
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 ficher 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 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
        …..
        }
}
```



```
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)
char : unicode
boolean

- 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
`boolean`	`true` or `false`	1 byte	not applicable
`char`	single character (Unicode)	2 bytes	all Unicode characters
`byte`	integer	1 byte	−128 to 127
`short`	integer	2 bytes	−32768 to 32767
`int`	integer	4 bytes	−2147483648 to 2147483647
`long`	integer	8 bytes	−9223372036854775808 to 9223372036854775807
`float`	floating-point number	4 bytes	$-3.40282347 \times 10^{+38}$ to $-1.40239846 \times 10^{-45}$
`double`	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 : <<, >>

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*

Classe String

•Il n'y a pas de type primitive 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

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 ?