

# Programmation Orientée Objet Java

Samuel Peltier

[samuel.peltier@univ-poitiers.fr](mailto:samuel.peltier@univ-poitiers.fr)

# Bibliographie

- Livres

- Introduction à Java *O'Reilly*
- Java in a nutshell *O'Reilly*
- Penser objet avec UML et Java *Dunod*
- Programmer en Java *Eyrolles*
- Effective Java *Addison-Wesley*
- UML 2 en action *Eyrolles*
- UML 2 par la pratique *Eyrolles*
- ...

- Internet

- <http://www.oracle.com/technetwork/java/javase/documentation/index.html>
- <https://stackoverflow.com/>
- <http://java.developpez.com/>
- <http://docs.oracle.com/javase/specs/>
- ...

# Notation de l'UE (6 ECTS)

## SESSION 1

```
float cc_1 // écrit sur table;  
float cc_2 // production écrite (projet);  
float cc_3 // écrit sur table;
```

```
float poo_mark = (3*cc_1+4*cc_2+5*cc_3) / 12 ;
```

## SESSION 2

```
float session_2 // écrit sur table (3h);
```

```
poo_mark = (4*cc_2 + 8*session_2) / 12 ;
```

# Contenu du cours

- Chapitre 1 : POO & Java  
Principaux paradigmes de programmation, Bref historique de la POO, structures de contrôle, types en Java
- Chapitre 2 : classes et objets  
Les objet, notion de classe, encapsulation, abstraction, modélisation UML, constructeurs, méthodes, visibilité, static, classes de base en Java, les tableaux
- Chapitre 3 : associations de classes
  - Associations de classes, unidirectionnelles, bidirectionnelles, agrégations vs compositions
- Chapitre 4 : héritage et ses implications  
Principes de l'héritage, redéfinition et surcharge, contrôle de l'héritage, polymorphisme, classes abstraites, interfaces

# Contenu du cours

- **Chapitre 5 : Types Avancés**  
Enum, generics, collections, Streams.
- **Chapitre 6 : Documenter avec UML**  
diagrammes d'états, diagrammes de séquences.
- **Chapitre 7 : Les tests**  
JUnit, tests structurels, tests fonctionnels, critères de tests.
- **Chapitre 8 : Java avancé**  
Refactoring, Gestion des dépendances, Création et utilisation de bibliothèques, Javadoc, Exceptions, Entrées/Sorties, les Threads, ...

# Chapitre 1 : POO & Java

- Principaux paradigmes de programmation
- Objectifs et bref historique de la POO
- Fonctionnement d'un programme Java
- Les types en Java

# Principaux paradigmes de programmation

- Programmation impérative (Ada, C) :  
*Un programme peut être représenté par une machine à états. Décomposition d'un problème en sous-problèmes (sous-programmes).*
- Programmation fonctionnelle (LISP, OCaml) :  
*Un programme est un enchainement de fonctions mathématiques. Pas d'opérateur d'affectation, pas de boucle.*

# Principaux paradigmes de programmation

- Programmation logique (Prolog) :

*Programmation déclarative. Un programme est un ensemble d'axiomes et de règles, à partir desquels on peut lancer des requêtes.*

- Programmation Orientée Objet (C++, Java) :

*La notion d'objet (brique de base de la programmation orientée objet) regroupe à la fois des données et les opérations qui les manipulent.*



# Bref historique de la POO

- 1967 : origine de la POO *Simula*
  - modélisation de simulations
    - Description naturelle par des objets changeant d'état
    - Programmation impérative inadaptée (séparation des données et opérations)
- Années 70 : développement de la POO *Smalltalk*
  - *langage tout objet*
  - *avènement des environnements fenêtrés WIMP (Window, Icon, Menu, Pointing device)*

# Bref historique de la POO

- Années 80 : apparition de langages OO
  - *C++, Object Pascal, Delphi, Objective C, C#, Python, Php...*
- Années 90 : refondation des concepts OO
  - *1996 : Java JDK 1.0*
  - *1998 : norme C++*

# Bref historique de Java

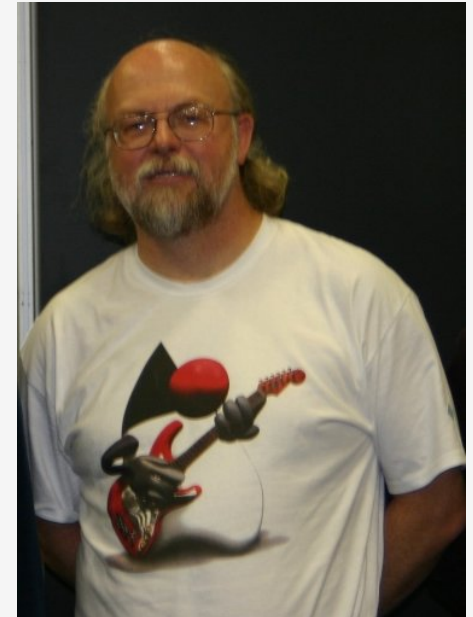
- 1990 : La *green team* est constituée chez Sun
- 1991 : Création d'un nouveau langage : *oak*
- 1992 : Création d'un PDA avec interface graphique (star 7)
- 1995 :
  - navigateur HotJava capable d'exécuter des applications web Java ;
  - Netscape embarque du Java

# Bref historique de Java

- Historique des Versions de Java
  - 1996 : JDK 1.0
  - 1997 : JDK 1.1
  - 1998 : J2SE 1.2 (Playground)
  - 2000 : J2SE 1.3 (Kestrel)
  - 2002 : J2SE 1.4 (Merlin)
  - 2004 : J2SE 5.0 (Tiger)
  - 2006 : SE 6 (Mustang)
  - 2011 : SE 7 (Dolphin)
  - 2014 : SE 8 (Spider)
  - 2017 : SE 9
  - 03/2018 : SE 10
  - 09/2018 : SE 11
  - 03/2019 : SE 12
  - 09/2019 : SE 13
  - 03/2020 : SE 14

# Bref historique de Java

- De 2006-2008 : Passage de Java sous licence libre
- 2009 : rachat par Oracle (qui est également propriétaire de MySQL)
- 2010 : James Gosling démissionne d'Oracle
- Actuellement
  - Java SE 14 (JDK + JRE)
  - Java EE (applications server)
  - Java ME (applications embarquées)



[https://commons.wikimedia.org/wiki/File:James\\_Gosling\\_2005.jpg](https://commons.wikimedia.org/wiki/File:James_Gosling_2005.jpg)

[CC BY-SA 3.0](#)

# Objectifs de la POO

- Motivations de la POO
  - Répondre aux limitations de la prog. structurée
- La POO ?
  - Modélisation d'un problème par :
    - Les objets qui interviennent
    - Comment ils interagissent
  - Exemple : modélisation du fonctionnement d'une TV
    - Objets :
      - Ecran (taille, résolution...)
      - Télécommande (couleur, poids...)
    - Fonctionnalités :
      - allumer/éteindre
      - augmenter/diminuer le son
      - changer de chaîne...

# Objectifs de la POO

- Caractéristiques du langage Java :

source : <https://www.oracle.com/java/technologies/introduction-to-Java.html>

- Simple, Object Oriented, and Familiar
  - Simplification de certains aspects (gestion mémoire, héritage contraint...)
- Robust and Secure
  - Langage fortement typé (gestion mémoire, détection d'erreurs à la compilation, exécution via JVM)
- Architecture Neutral and Portable
  - Bytecode interprété par une JVM
- High Performance
  - Compilation à la volée de certaines parties du code (**hot spot**) par les VM
- Interpreted, Threaded, and Dynamic
  - Bytecode interprété, exécution de plusieurs tâches en parallèle, classes chargées dynamiquement

# Fonctionnement d'un programme Java

- code source : `MonProgramme.java`
- compilation : `javac`
- byte code : `MonProgramme.class`
- exécution de la JVM : `java`



# Fonctionnement d'un programme Java

- Byte code (code objet) java :
  - résultat de la compilation du code source
  - interprété par toute JVM.
- Machine virtuelle Java (JVM) :
  - programme interprétant le byte code Java
  - Interprète du byte code quelle que soit sa provenance
- Les outils de développement :
  - Éditeur de texte + javac + java
  - Netbeans
  - Eclipse
  - ...

# Fonctionnement d'un programme Java

- The Java Application Program Interface
  - Description des packages et classes disponibles
  - Très riche
  - Utilisation simple
- Un premier exemple Java : HelloWorld.java

# Fonctionnement d'un programme Java

- Commentaires :
  - `// un commentaire`
  - `/* un autre  
  * commentaire  
  */`
- Les structures de contrôle :
  - **séquence** : `{instruction1 ; instruction2 ; instruction 3;}`
  - **conditionnelle** :

```
if (condition) {  
    instructions;  
}  
//optionnel  
else{  
    instructions;  
}
```

# Chapitre 1 : POO & Java

## – boucles :

- `while (condition) {  
    instruction1;  
}`
- `for (init ; condition ; incrémentation) {  
instructions;  
}`
- `do {  
    instructions;  
}  
while (condition);`

Les instructions **continue** et **break** permettent de ne pas suivre le déroulement normal d'une boucle

# Fonctionnement d'un programme Java

## – Tests multiples :

```
• switch (var) {  
    case val1 : instructions;  
               break;  
    case val2 : instructions;  
               break;  
    default  : instructions:  
               break;  
}
```

**ATTENTION** : sans instruction `break`, les instructions des cas suivant le `case` sont également effectuées jusqu'à la prochaine instruction `break`, ou s'il n'y en a pas, jusqu'à la fin du bloc `switch`.

# Les types en Java

- Deux types de données en Java :
  - types primitifs (ne sont pas des objets)
  - classes d'objets

ATTENTION : contrairement aux variables de type primitif, les objets sont des références en java.
- Déclaration : `type name`  
`int x;`  
`Watch myWatch;`
- Déclaration + initialisation : `type name = value`  
`int x = 2;`

# Les types en Java

- Portée locale

```
for (int i=0 ; i<5; i++) {  
    instructions;  
}  
// ici i n'existe plus
```

- Constantes : mot clé **final**

```
final int UNE_CONSTANTE;
```

- Les types primitifs :

- Entiers signés :

- byte (1o) ; short (2o) ; int (4o) ; long (8o)

- Flottants :

- float (4o) ; double (8o)

- Booléens :

- boolean valeurs : true , false

- Caractères :

- char (codage unicode sur 2o)

# Les types en Java

- Les opérateurs de calcul
  - Calcul arithmétique :
    - +, -, \*, /, %
  - Incrémentation, décrémentation :
    - ++, --
  - Affectation :
    - =, +=, -=, \*=, /=
  - Comparaison :
    - ==, !=, >, >=, <, <=
  - Opérateurs booléens :
    - &&, ||, !
  - Opérateurs bit à bit :
    - &, |, ~
  - Transtypage :
    - `double d ;`
    - `int i = (int)d;`



# *Les tableaux*

- Déclaration :

```
type[] tab;
```

- Instanciation :

```
tab = new type[taille];
```

- Déclaration + instanciation :

```
type[] tab = new type[size];
```

Exemple :

```
int[] tab = new int[5];
```

// un tableau de 5 cases indicé de 0 à 4

- Accès à une case du tableau :

```
myArray[i];
```

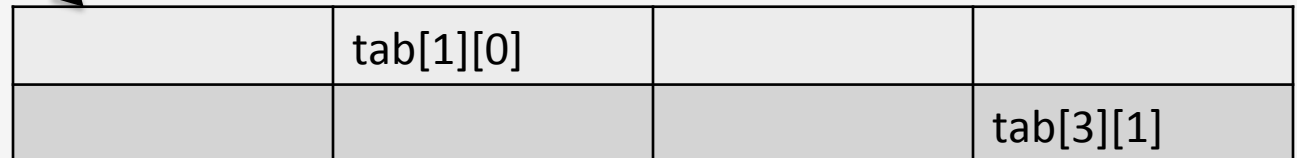
# *Les tableaux*

- Tableaux multidimensionnels :

```
int[][] tab;
```

```
tab = new int[4][2];
```

tab



	tab[1][0]		
			tab[3][1]

# *Les tableaux*

- Parcours :

- Itérer sur chaque case

```
for (int i=0 ; i<monTableau.length;i++) {  
    ...  
}
```

- Itérer sur chaque valeur

```
for (int val : monTableau) {  
    ...  
}
```

# Les Collections en 2 mots

- Manipulation de « paquets » d'objets
  - Structures de données :  
listes, ensembles, ...
  - Opérations sur les structures :  
ajout/suppression d'un élément, taille de la collection...

*exemple :*

```
List<Card> hand = new ArrayList<>();
```

# Conventions de nommage

- `ClassName`
- `varName`
- `methodName(...)`
- `CONSTANT_NAME`