Bases de la Programmation Orientée Objet / Java

DUT / Module M213

Langage et plateformes Java

Caractéristiques techniques (1)

➔ **Langage orienté objets**

- langage fortement typé
- classes avec possibilité de classes internes
- objets **désignés par référence** uniquement (**tas**)
- syntaxe proche du langage C/C++
- gestion automatique de la mémoire (ramasse miettes)
- gestion intégrée des exceptions
- héritage simple et composition

Caractéristiques techniques (2)

➔ **Langage semi-compilé et interprété**

- Pas de pré-processing
- contrôles syntaxiques et d'initialisation des variables
- compilation en code intermédiaire (byte-code)
- Machine virtuelle d'exécution (JVM)
- chargement de code dynamique
- exécution multi-threads
- erreurs d'exécution contrôlées par la JVM ("no trap !")

Caractéristiques techniques (3)

➔ **Règles lexicographiques et syntaxiques**

- un fichier par classe et une classe par fichier
- le fichier porte le nom de la classe qu'il contient
- définition d'une classe encadrée par {...}
- définition d'une méthode encadrée par {...}
- lignes logiques terminées par le séparateur ;
- description textuelle "case sensitive"

Caractéristiques techniques (4)

➔ Allocation mémoire

- **tous les objets sont alloués dans le tas**
- ***new*** analogue à ***malloc*** (retour d'une adresse dans le tas)
- **les variables sont allouées sur la pile**
- toute variable qui désigne un objet contient en fait une référence (adresse) sur cet objet
- les pointeurs sont donc cachés !
- plusieurs variables peuvent pointer sur le même objet (notion de multi vue et de compteur de vues)

Le répertoire JDKxxx / bin

➔ **Exécutables & DLL**

- Compilateur **(**exécutable **javac)**
- Nombreuses options de compilation
- Java Virtual Machine (**JVM -** exécutable **java**)
- Toutes les DLL associées
- Outil de gestion des librairies **(**exécutable **jar**)
- Outil de production de la documentation (exécutable **javadoc**)

Mise en oeuvre d'un programme (1)

➔ **Compilation**

- Societe.java -------> Societe.class
- T_Societe.java -------> Test_Societe.class

➔ **Exécution**

Fournir à la JVM une classe compilée en bytecode (xxx.class) qui définit la méthode de classe *main*

Mise en oeuvre d'un programme (2)

➔ **Compilation**

> *javac* Societe.java

produit un fichier **Societe.class**

> *javac* T_Societe.java

produit un fichier **T_Societe.class**

➔ **Exécution**

> *java* T_Societe

Autres outils du JDK

➔ ***javadoc***

Pour générer automatiquement la documentation technique de la classe cible, en format HTML

➔ ***jar***

Pour créer une archive

➔ ***appletviewer***

Pour tester une *applet* sans navigateur Web

La variable **CLASSPATH**

- ➔ Conditionne l'accès aux fichiers compilés
- ➔ Indispensable pour les packages prédéfinis
- ➔ Séparateur ; entre les différents chemins
- ➔ Placer le meta caractère . en tête de la valeur de cette variable (pas de valeur implicite !)

Plateformes Java

➔ **Plusieurs technologies**

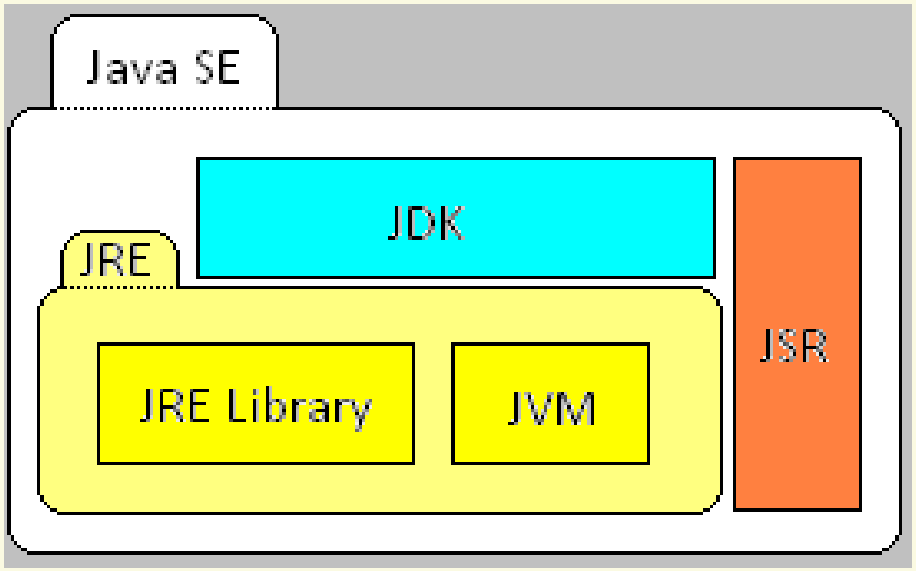
http://www.oracle.com/fr/technologies/java/overview/

- Java Entreprise Edition (Java EE)
- **Java Standard Edition (Java SE)**
- Java Embedded
- Java Micro Edition (Java ME)
- Java Cloud Service

Java Standard Edition (1)

➔ **Destinée aux applications du poste client**

http://fr.wikipedia.org/wiki/Java_SE



Java Standard Edition (2)

➔ Description générale

http://www.oracle.com/technetwork/java/javase/tech/index.html

- **Java Development Kits (JDK)**
- **Java RunTime Environment (JRE)**
- Application Programming Interfaces (**API**) spécialisées pour le poste client (**JFC** par exemple)
- API d'usage général (**JAXP** par exemple)
- Java DataBase Connectivity (**JDBC**)

Origine et historique (1)

➔ **SUN Microsystems (constructeur US)**

- fabricant de stations de travail sous SUN_OS (Unix)
- création d'un groupe de travail sur la définition d'un environnement de conception indépendant du matériel
- Toolkit indépendant du système d'exploitation
- intégration de la programmation réseau
- le langage OAK (James GOSLING) présenté en 1992
- Java 1.0 en téléchargement libre en 1995

Origine et historique (2)

➔ **Evolutions (1)**

http://en.wikipedia.org/wiki/Java_version_history

- **1996** : Java 1 et JDK 1.0

  212 classes prédéfinies en 8 packages
- **1997** : Java 1 et JDK 1.1

  504 classes en 23 packages
- **1998** : Java 2 et JDK 1.2

  1520 classes en 59 packages

Origine et historique (3)

➔ **Evolutions (2)**

Java Community Process (**JCP**)

Java Specification Requests (**JSR**)

Java Language Specification (**JLS**)

http://en.wikipedia.org/wiki/Java_Community_Process

https://www.jcp.org/en/jsr/all

- **2000** : Java 2 et J2SE version 1.3

  1839 classes en 76 packages

Origine et historique (4)

➔ **Evolutions (3)**

- **2002** : Java 2 et J2SE version 1.4

  2757 classes en 135 packages
- **2004** : Java 5 et J2SE version 5.0

  Evolutions majeures du langage de base

  (notamment **JSR 14** - Ajout des types génériques !)
- **2009** : Java 6, avec notamment

  Rachat de SUN par Oracle

Origine et historique (5)

➔ **Evolutions (4)**

- **2009 : Java 6, avec notamment**

  JSR 223 : couplage avec des langages de scripts (Jython, JRuby, Groovy, Scala, …)

  JSR 224 : couplage avec les web services

- **2011** : Java 7, avec notamment

  JSR 334 : « petites améliorations du langage » !

- **2014** : Java 8

Caractéristiques techniques (1)

➔ **Langage orienté objets**

- langage fortement typé
- classes avec possibilité de classes internes
- objets **désignés par référence** uniquement (**tas**)
- syntaxe proche du langage C/C++
- gestion automatique de la mémoire (ramasse miettes)
- gestion intégrée des exceptions
- héritage simple et composition

Caractéristiques techniques (2)

➔ **Langage semi-compilé et interprété**

- Pas de pré-processing
- contrôles syntaxiques et d'initialisation des variables
- compilation en code intermédiaire (byte-code)
- Machine virtuelle d'exécution (JVM)
- chargement de code dynamique
- exécution multi-threads
- erreurs d'exécution contrôlées par la JVM ("no trap !")

Caractéristiques techniques (3)

➔ **Règles lexicographiques et syntaxiques**

- un fichier par classe et une classe par fichier
- le fichier porte le nom de la classe qu'il contient
- définition d'une classe encadrée par {...}
- définition d'une méthode encadrée par {...}
- lignes logiques terminées par le séparateur ;
- description textuelle "case sensitive"

Caractéristiques techniques (4)

➔ Allocation mémoire

- **tous les objets sont alloués dans le tas**
- ***new*** analogue à ***malloc*** (retour d'une adresse dans le tas)
- **les variables sont allouées sur la pile**
- toute variable qui désigne un objet contient en fait une référence (adresse) sur cet objet
- les pointeurs sont donc cachés !
- plusieurs variables peuvent pointer sur le même objet (notion de multi vue et de compteur de vues)

Bibliographie

➔**Polycops et présentation en ligne**

- T. Leduc (ENS Cachan)
- P. Itey – Introduction a Java (INRIA Sophia Antipolis)
- T.T. Dang Ngoc (Université de Versailles & CNAM)