Analyse et Programmation Orientées Objets / Java

III- Présentation du langage

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 de l'environnement**
  - JDK 1.0 : 212 classes prédéfinies en 8 packages
  - JDK 1.1 : 504 classes en 23 packages
  - JDK 1.2 : 1520 classes en 59 packages (Java 2)
  - 2000 : Java 2 et J2SE version 1.3
  - 2002 : Java 2 et J2SE version 1.4
  - 2005 : Java 5 (J2SE version 5.0)

Caractéristiques techniques (1)

➡ **Langage orienté objets**

- langage de classes avec possibilité de sous-classes
- fortement typé
- objets désignés par référence uniquement
- syntaxe proche du langage 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"

Exécution d'un programme

- **Compilation**
  - Societe.java -------> Societe.class
  - Test.java -------> Test.class
- **Exécution**

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

Deux outils de base du JDK

- **Compilation**

  > *javac* Test.java

  produit un fichier **Test.class**

- **Exécution**

  > *java* Test

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

Notions complémentaires (1)

➡ **Attributs de classe**

Attribut partagé par toutes les instances (objets) d'une même classe. Sa valeur peut être modifiée par chaque instance.

Utilisation du spécificateur *static*

Notions complémentaires (2)

➡ **Langage Java**

public class **Societe** {

private String nom, localité**;**

static String nomDuGroupe;

}

Notions complémentaires (3)

➡ **Méthode de classe**

Méthode invocable en l'absence de toute instance.

- Utilisation du spécificateur *static*
- Invocation par l'opérateur d'application
- Invocation par le biais du nom de la classe (en l 'absence du nom d 'instance !)
- Notion "évidente" pour la méthode *main*

Notions complémentaires (4)

➡ **Langage Java**

public class **Societe** {

---

static void m1 (…) {…}

}

Notions complémentaires (5)

➡ **Règles applicables aux méthodes de classe**

Une méthode de classe ne peut pas référencer un attribut d'instance.

Une méthode d'instance de la classe XXX peut référencer à son gré toute variable de classe de XXX.

Arrêt programmé de l'exécution

➡ **Classe prédéfinie System**

La méthode de classe *exit* de la classe *System* interrompt l'exécution de la méthode appelante et retourne au système d'exploitation le code d'erreur spécifié par le programmeur.

.

System.exit(1);