Programmation Orientée Objet

Master 1 CS

DR. Sofiane AOUAG
Université De Batna II
Faculté des Mathématique et de l'Informatique
Département d'Informatique

2017/2018

Plan du cours

- Introduction
  - Présentation de java
  - Caractéristique du langage
- Types et Variable
  - Types simples
  - Types Complexes
  - Variables
- Structures de contrôles
  - Composition
  - Sélection
  - Itération
- Les classes en Java
- Organisation
- Exemples

Présentation

Qu'est ce que Java ?

- Significations et origine du nom Java :
  - Une danse
  - Une île
  - Une ville des Etats Unis
  - Une province de Géorgie
  - Certains prétendent qu'il signifie **J**ames Gosling, **A**rthur **V**an Hoff et **A**ndy Bechtolsheim ; ou tout simplement **J**ust **A**nother **V**ague **A**cronym (littéralement « juste un acronyme vague de plus »).
  - Java signifie café en argot américain d'où l'icône de la tasse du café
- **Le langage Java:** est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien deBill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

Présentation

Historique...

1991 "Green Project" projet chez SUN[1] ayant pour but d'embarquer une technologie dans différents appareils (TV, électroménager, ...) permettant de les piloter et de les faire communiquer entre eux

1992 démonstration d'une télécommande (nom: *7) capable de piloter des appareils électroménager, mais pas de marché pour ce système. Un langage plus léger et facile que le C++ fut créé : Oak (chêne) rebaptisé ensuite Java

Entre temps internet se développe rapidement

1995 création d'un navigateur "WebRunner" qui allait s'appeler plus tard "HotJava"

démonstration d'applications tournant dans un navigateur (molécules 3D, ...): les applets

mise a disposition du code source à la communauté des développeurs (succès immédiat)

[1] http://www.sun.com

Présentation

Historique...

... essor très important du langage qui était multiplateforme, petite taille des applications : idéal pour le web
Netscape décide d'intégrer cette technologie dans son navigateur

1996 Java Development Kit 1.0

1997 JDK 1.1 : amélioration de la syntaxe et des exceptions, refonte de l'interface graphique (listener)
Netscape Communicator supporte la totalité de l'API

1998 Java 2 (=J2SE 1.2) : apport du multimédia, une nouvelle interface graphique incorporée : Swing

2004 Java 5 : ajout de fonctionnalités : types génériques, autoboxing, types énumérés, nouveau look, ...

Présentation

Historique...

Le 11 novembre 2006, le code source du compilateur javac et de la machine virtuelle HotSpot (en) ont été publiés en Open Source sous la Licence publique générale GNU8.

Le 13 novembre 2006, Sun Microsystems annonce le passage de Java, c'est-à-dire le JDK (JRE et outils de développement) et les environnements Java EE (déjà sous licence CDDL) et Java ME sous licence GPL d'ici mars 2007, sous le nom de projet OpenJDK9.

En mai 2007, Sun publie effectivement OpenJDK sous licence libre.

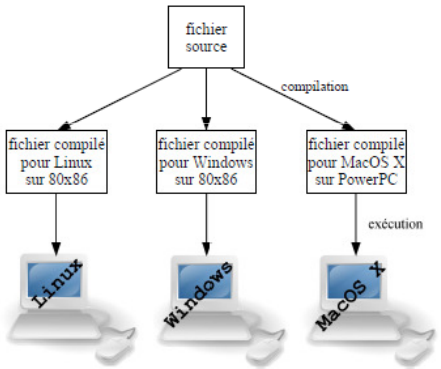
Java SE 8 - Nom de code Wolf.Diverses releases en cours de développement du JDK sont disponibles au téléchargement dés l'automne 201338, et Java 8 sort mi-mars 2014 conformément à une roadmap présentée par Oracle dés mai 201339.
Une des nouveautés majeures de cette version est l'ajout des closures 40.

Java SE 9, initialement prévu pour 2015, a été reporté à 2016 suite aux retards de développement de Java 842; ceux-ci ont déjà conduit au report dans Java 9 du projet Jigsaw, qui devait initialement améliorer la modularité de Java 8.

Caractéristiques

Portabilité...

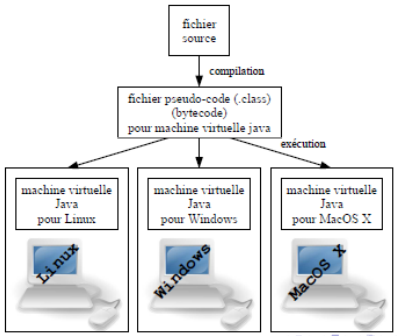
Langage "classique"



Caractéristiques

Portabilité...

Langage Java : "Compile one time, run anywhere"



Caractéristiques

- langage objet (pas d'héritage multiple)
- langage à part entière, pas uniquement pour le web
- nécessite une JVM pour l'exécution
- indépendant de la plateforme
- machines virtuelles incluses dans les navigateurs web (applet)
- sécurisation des exécutions
- ramasse miette (garbage collector)

Part II

Le langage Java

Types et variables

Qu'est ce qu'un type

Definition

Le type d'une donnée détermine

- la taille de son espace de stockage
- l'espace de ses valeurs
- les opérations possibles

Java possède deux catégories de types :

simples : des valeurs "simples" (entiers, caractères, ...)

complexes : des classes et des tableaux

Types et variables

Les types simples de Java

`byte`	entier 8 bits	-128 - 127
`short`	entier 16 bits	-32768 - 32767
`int`	entier 32 bits	-2147483648 - 2147483647
`long`	entier 64 bits	-9223372036854775808 - 9223372036854775807
`float`	décimal[2] 32 bits	$\approx \pm 10^{-45} - \pm 10^{38}$
`double`	décimal 64 bits	$\approx \pm 10^{-323} - \pm 10^{308}$
`char`	caractère[3] 16 bits	
`boolean`	booléen	`true` / `false`

Exemple

```
int a=3,b;
b=a+2;
```

Types et variables

Types complexes de Java

Objet et référence

En Java, les types complexes sont des classes

Vocabulaire

Une variable de type classe contient une "référence" sur un objet résultant de l'instanciation d'une classe (ou `null`).

Exemple

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```


Types et variables

Types complexes de Java

Tableau : vecteur de données de taille fixe...

Le type tableau est un type complexe (objet)

Syntaxe

```
typeelement[] tableauelements1 = {elt1, elt2, ... , eltn};
typeelement[] tableauelements2;
tableauelements2 = new typeelement[taille];
tableauelements2[x]=valeur; // 0 ⩽ x < taille
```

- L'attribut `length` permet d'obtenir la taille du tableau
- La classe `Arrays` permet la manipulation de tableaux a l'aide de méthodes de classe (tri, copie, ...).

Types et variables

Exemples

```
int[] taba = new int[2];
taba[0]=1;
taba[taba.length-1]=2; // le dernier élément
taba[2]=3; // lancement d'une exception
System.out.println("taille de taba : "+taba.length);

int[][] tabb = new int[2][3];
tabb[0][1]=5;

int[][] tabc = new int[4][];
// nécessite l'allocation des 4 sous tableaux
```


Types et variables

Les variables en Java

Manipulation...

Égalité

Le comportement de l'opérateur == dépend du type des variables :

- simple : égalité des valeurs
- objet : égalité des références

Pour la comparaison d'objets Java fournit la méthode `equals`

D'une manière générale, les variables de type :

- simple se manipulent par valeur
- complexe se manipulent par référence

Types et variables

Les variables en Java

Exemples

```
int a=1,b=1;
System.out.println(a==b);                    // true

Integer i1=new Integer(1),i2=new Integer(1);
System.out.println(i1==i2);                  // false
System.out.println(i1.equals(i2));           // true

int[] ta={1,2,3},tb={1,2,3};
System.out.println(ta==tb);                  // false
System.out.println(ta.equals(tb));           // false
System.out.println(Arrays.equals(ta,tb));    //true

Integer[][] tc={{new Integer(1),new Integer(2)}
              {new Integer(1),new Integer(2)}};
Integer[][] td={{new Integer(1),new Integer(2)}
              {new Integer(1),new Integer(2)}};
System.out.println(Arrays.equals(tc[0],tc[1]));  // true
System.out.println(Arrays.equals(tc,td));        // false
System.out.println(Arrays.deepEquals(tc,td));    // true
```


Types et variables

Les variables en Java

Opérateurs

Opérateurs arithmétiques

```
+ - * / %
```

Opérateurs d'affectation

```
= += -= *= /= %= ++ --
```

Opérateurs bit à bit

```
< > <= >= == !=
```

Opérateurs logiques

```
!  && ||
```


Structures de contrôle en java

Les alternances

Conditionnelle

Permet d'exécuter une partie de code en fonction du résultat d'un test

Syntaxe

```
    if (exprBooléenne)
       InstructionOuBloc
[ else
    InstructionOuBloc
]
```


Structures de contrôle en java

Les alternances

Choix multiple

Permet un branchement en fonction d'une valeur entière

Syntaxe

```
switch (exprEntière) {
  case const1 :
    instructionsCas1
    [break;]
  case const2 :
    instructionsCas2
    [break;]
  ...
  default :
    instructionsparDefaut
    [break;]
}
```

Structures de contrôle en java

Différents types d'itération

L'itération permet la répétition d'instructions

Syntaxe

```
do                                  while (exprBooléenne)
  instructionOuBloc                   instructionOuBloc
while (exprBooléenne);

for(initialisation;exprBooléenne;exprVariation)
  instructionOuBloc
```


Structures de contrôle en java

Différents types d'itération

Exemples

```
int [] tableau={1,4,34,25,15,16};
int compteur=0;
do
  compteur++;
while (compteur<tableau.length && tableau[compteur]%3 != 0);
if (compteur!=tableau.length) {
  while (compteur>=0 && tableau[compteur]%2 != 0)
    compteur--;
  for(int i=0; i <= compteur; i++)
    System.out.println(tableau[i]);
}
```


Les classes en java

Qu'est ce qu'une classe

Définition

Une classe (type complexe) est constituée :

- d'attributs (un état interne)
- de constructeurs / destructeurs
- de méthodes dont les accesseurs

Un objet résulte de l'instanciation d'une classe

Manipulation d'un objet

- initialisation à l'instanciation (construction)
- modification / consultation de l'état via l'envoi de messages
- destruction lorsqu'il n'est plus référencé

Les méthodes d'une classe forment l'interface permettant de manipuler les objets.

Les classes en java

Constructeur/Destructeur

Le constructeur

- permet d'initialiser un objet au moment de son instanciation
- garantie un état initial stable de l'objet.

Le destructeur

- permet de libérer l'espace mémoire réservé à l'objet
- nettoie proprement le contenu de l'objet avant sa mort

Les classes en java

Encapsulation

Définition

Cette notion regroupe deux concepts :

- regrouper une structure et des fonctions dans un même module
  ⇒ la classe (attributs + méthodes)
- assurer la cohérence de la classe en empêchant des manipulations
  ⇒ limiter la portée des attributs et des méthodes

Les classes en java

Syntaxe

```
class NomClasse {
  // attributs
  accès1 typeA attributA1, ..., attributAn=initialisation1;
  accès2 typeB attributB1=initialisation2, ..., attributBn;
  ...
  //constructeurs
  accès3 NomClasse([type1 param1, type2 param2, ...) {
    // instructions
  }
  ...
  // méthodes
  accès4 type nomMethodeI([type1 param1, type2 param2, ...]) {
    // instructions
  }
  ...
}
```


Les classes en java

Spécificateurs d'accès des membres

Les membres d'une classe sont ils accessibles à une autre classe

4 spécificateurs :

Les membres

`public`	sont tous visibles
`protected`	ne sont visibles que par les classes filles ou de même package
`private`	ne sont pas visibles
''friendly''	sont visibles pour les classes du même package

Le spécificateur doit se placer devant chaque membre

Les classes en java

Constructeur / Destructeur

Syntaxe : constructeur

```
NomClasse([paramètres]) {
  // instructions
}
```

Si aucun constructeur n'est défini, il y a un constructeur par défaut

Destructeur

- Java ne possède pas de destructeur, un ramasse-miette[a] se charge de libérer la mémoire.
- La méthode `finalize()` appellée avant destruction de l'objet n'est pas un destructeur. Elle n'est utile que pour les composants java natifs qui auraient alloués de la mémoire.

[a] Garbage Collector

Organisation

Utilisation / Création de packages

Importation

- `import nom.du.package.*;`
  Importe l'ensemble des classes du package `nom.du.package`
- `import nom.du.package.UneClasse;`
  Importe uniquement `UneClasse`

Création

Directive en début de fichier : `package` *`nom.du.package`* `;`

- les classes seront placées dans *`chemin`*`/nom/du/package`
- *`chemin`* doit être déclaré dans `$CLASSPATH`
- utilisation par `import nom.du.package.*;`

Si le package n'est pas défini, les classes font parti du package "."

Organisation

Classe publique et fichier

Définition et syntaxe

Seules les classes publiques d'un paquet sont accessibles par les utilisateurs.

```
public class NomClasse {
  ...
}
```

Remarques

- une seule classe publique par unité de compilation (fichier),
- le nom de cette classe doit porter le nom du fichier,
- les autres classes du fichier sont amicales (friendly), elles sont vu comme support de la classe publique.

Organisation

Organisation des classes/fichiers/packages/...

- répertoire de base
  - classes
    - nom/du/package *(automatique)*
      les classes compilées `.class` du package
    - des classes compilées
  - doc
    la documentation générée par `javadoc`
  - lib
    les bibliothèques `.jar`
  - src
    - nom/du/package
      les fichiers sources `.java` du package
    - des fichiers sources `.java`

Part III

Exemple

Exemple

Ampoule.java

```
package fr.insarouen.asi.materiel;

import java.awt.Color;
import java.util.Random;

public class Ampoule {
  private Color couleur=Color.WHITE;
  private boolean etat=false;
  private int dureeVie;

  // constructeurs
  public Ampoule() {
    Random alea=new Random();
    dureeVie=alea.nextInt()%1000;
  }
  public Ampoule(Color _couleur) {
    this();
    couleur=_couleur;
  }

  // méthodes
  public Color getCouleur() {
    return couleur;
  }
  public void setCouleur(Color couleur) {
    this.couleur=couleur;
  }

  public boolean basculer() {
    if (dureeVie>0) {
      user();
      etat=!etat;
    }
    else
      etat=false;
    return etat;
  }

  public boolean getEtat() {
    return etat;
  }

  private void user() {
    dureeVie--;
  }
}
```


Exemple

ClassMain.java

```
public class ClassMain {

  public static void main(String[] args) {

    Ampoule ampoule=new Ampoule();
    int resistance=0;
    boolean etat=ampoule.getEtat();

    while (etat != ampoule.basculer()) {
      resistance++;
      etat=ampoule.getEtat();
    }

    System.out.println("Resistance␣À␣l'utilisation␣:␣"+resistance);
  }
}
```