Programmation 1 bis : introduction

Gilles Trombettoni

IUT de l'Université de Montpellier, département informatique

Novembre 2015

Plan

Module de prog1-bis

- 1 semaine de transition + 5 semaines d'enseignement
- Par semaine : 1h de cours + 3 fois 2h de TD (salle ou machine)
- Contenu : bases du langage Java, algorithmique, classes/objets, structures de données avancées
- Notation divisée en deux parties :

25% ou 30%	participation (chaque enseignant de TD décide)
75% ou 70%	examen final

Plan

The camel has two humps

- Plus de la moitié des métiers offerts à un diplômé de l'IUT en informatique demandent de fortes compétences en programmation.
- Or, entre 20% et 50% des étudiants n'auront jamais cette compétence : *"The camel has two humps"* :

  Contrairement aux autres matières, la distribution des notes en programmation ne suit pas une loi normale.

The camel has two humps (working title)

Saeed Dehnadi and Richard Bornat
School of Computing, Middlesex University, UK

February 22, 2006

Abstract

Learning to program is notoriously difficult. A substantial minority of students fails in every introductory programming course in every UK university. Despite heroic academic effort, the proportion has increased rather than decreased over the years. Despite a great deal of research into teaching methods and student responses, we have no idea of the cause.

It has long been suspected that some people have a natural aptitude for programming, but until now there has been no psychological test which could detect it. Programming ability is not known to be correlated with age, with sex, or with educational attainment; nor has it been found to be correlated with any of the aptitudes measured in conventional 'intelligence' or 'problem-solving-ability' tests.

Quelques enseignements à tirer de cette étude

- Les étudiants ayant obtenu de bonnes notes en prog depuis le début de l'année devraient conserver leur niveau (s'ils continuent à apprendre les nouvelles connaissances).

  Des étudiants en échec jusqu'à présent pourront rejoindre "la bosse des bons en prog".

  (sous réserve que la faculté intellectuelle de "cohérence" soit bien à l'origine des aptitudes en programmation)
- Un langage de programmation est un **langage formel**.
  Vous devez travailler pour avoir un déclic (rapidement).
  Les enseignants doivent faire preuve de pédagogie pour que vous ayez ce déclic.
- Certains n'y arriveront pas et devront admettre qu'ils ne seront probablement jamais doués pour la programmation :
  il y a d'autres débouchés en informatique (ex: technico-commercial) ;
  il y a d'autres débouchés que l'informatique !

Plan

Naissance de Java

- Première version publique : 1995
- Créé par James Gosling, Patrick Naughton, Mike Sheridan à Sun Microsystems (rachetée par Oracle)
- Langage à objets (paradigme supportant la plupart des principes de génie logiciel moderne)
- Syntaxe de base proche du langage C++
- Gestion automatique de la mémoire : ramasse-miettes (*garbage collector*), comme en Lisp, ML, etc
- Portabilité sur tout système d'exploitation et ordinateur

Portabilité et machine virtuelle (JVM)

1. Compilation dans un langage de *bytecode* :
   `javac Temperature.java` crée le fichier `Temperature.class`
2. Bytecode interprété par une machine virtuelle (interprète appelé Java Virtual Machine – JVM) : `java Temperature` (attention pas de suffixe !)

Conséquences

- Facilite aux entreprises le développement d'un logiciel.
- Rend moins efficace l'exécution d'un programme (trois fois plus lent environ qu'un programme exécutable compilé en C++).
- Java a surfé sur le web : applet/servlet :
  - Servlet : classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP.
  - Applet : programme qui s'exécute dans la fenêtre d'un navigateur web (client).

    Une applet Java est fournie sous forme de bytecode Java.

Autres caractéristiques

- Interfaces graphiques : bibliothèques graphiques AWT et Swing.
- Multi-threading :
  - Threads (tâches ou processus légers) : sous-processus partageant le tas (*heap*), mais pas la pile d'appel des fonctions (*stack*).
  - Multi-threading : langage ou plateforme supportant la gestion simultanée de plusieurs tâches concurrentes.

Plan

Principaux types simples

Déclarations de variables

- `<Type de la variable> <Nom de la variable> ;`
- Exemple : `int n` (variable *n* de type entier)
- Principaux types : `int` (4 octets), `float` (4), `double` (8), `char` (1), `boolean` (1 bit), `String`.

Exemples

```
int nb1 = 32, nb2 = -18;
float pi = 3.1416f;
char carac = 'z';
String mot = new String("Je retiens tout !\n");
```

Conditions

Opérateurs différents d'Ada

- égalité: `==`
- *non*: `!`
- différent: `!=`
- *ou* séquentiel: `||`
- *et* séquentiel: `&&`

Instruction conditionnelle `if`

```
if (x == -64 || y != 8) {
   System.out.println("Bel exemple ?");
}
else if (-10 <= nb2 && nb2 <= 10) {
   System.out.println("nb2 appartient à [-10,10]");
}
else {
   System.out.println("Exemple douteux");
}
```

Boucle Tant-que

Syntaxe

```
while (<condition>) {
   <body>
}
```

Exemple

```
i=0;  // Commentaire: ne rentre pas si i==45
while (i < 10) {
   System.out.println(i + " ");
   i++;  // Commentaire:   i := i+1;
}
```

Boucle Répéter...Tant-que

Syntaxe

```
do
   <body>
while <condition>;
```

Exemple

```
i=0;  // Entre une fois si i==45
do {
   System.out.println(i + " ");
   i++;  /* Commentaire sur
            plusieurs lignes:   i := i+1  */
} while (i < 10);
```

Boucle `for`

En Java, comme en C ou PHP, la boucle `for` est une boucle `while` déguisée !

Syntaxe

```
for (<initialisation> ; <condition> ; <incrémentation>) {
  <body>
}
<initialisation>
while <condition> {
   <body>
   <incrémentation>
}
```

Exemples

```
for (int i = 0 ; i < 10 ; i++) {
     System.out.println(i + " ");
}
for(int i = 0, j = 2 ; (i < 10 && j < 6) ; i++, j+=2){
  System.out.println("i = " + i + ", j = " + j);
}
```

Tableaux, matrices

Syntaxe de déclaration d'un tableau

```
<type élément> <nom du tableau> [] = { <contenu du tableau> };
<type> <nom du tableau> [] = new <type> [<taille>];
```

Exemples de déclaration d'un tableau

```
String tableauChaine[] = {"Je", "vois", "bien"};
double tableauDouble[] = new double[5]; // tableau de 5 cases
// Ou encore:
double[] tableauDouble2 = new double[5];
```

Exemples de déclaration d'une matrice

```
int matrice[][] = { {-1,3,-4},{6,-10,8} };
int matrice[][] = new int[2][3];
```

Accéder à un élément d'un tableau

Indiçage des tableaux

Les tableaux sont toujours indicés à partir de l'indice 0 !

Exemple

```
char tab[] = {'a','b','c','d'};
for (int i = 0; i < tab.length; i++) {
    System.out.println("A l'indice " + i +
                       " nous avons = " + tab[i]);
}
```

Plan

Classes et instances

Qu'est-ce qu'une classe et qu'est-ce qu'une instance ?

- Classe : type/moule
- Instance/objet : variable créée à partir de ce moule
- Une classe est un type composé qui contient des attributs/champs/variables d'instance (comme un enregistrement), mais aussi des fonctions appelées méthodes.

Exemple d'une classe Voiture

- Attributs : String modele, float longueur, boolean aCarburantEssence, float cylindree...
- Méthodes : création (constructeur), description...

Exemple utilisant trois premières classes

Une classe contenant le programme et des classes très utiles pour les entrées sorties : `Scanner` et `System`.

Exemple : le fichier MonProg.java

```
import java.util.Scanner; // inclusion de la classe Scanner
                          // du package java.util

class MonProg { // Rmq: une classe commence par une majuscule!

  public static void main (String args[]) {

    System.out.print("Donner la couleur de l'oiseau "
                     + args[0] + ": ");

    Scanner clavier = new Scanner(System.in);
    String couleur = clavier.nextLine();

    System.out.println(args[0] + " a la couleur " + couleur);
  }
}
```

Explications

- Pour compiler: `javac MonProg.java`
- Pour exécuter: `java MonProg aigle`
- `main()` est la fonction principale ne renvoyant rien (`void`).
- `public static` : pour plus tard !
- `String args[]` : tableau de chaînes contenant les arguments entrées derrière le nom du programme (`args[0]` vaut "aigle")
- `java.lang.System` : permet de réaliser des entrées-sorties standards
- `System.out` : attribut de type `PrintStream` : flux de sortie standard
- `System.in` : attribut de type `InputStream` : flux d'entrée standard
- `print()` et `println()` : méthodes d'affichage d'une instance de `PrintStream`
- `Scanner` : classe permettant les saisies/lectures sur un terminal, un fichier, une chaîne de caractères...
- `new` : mot-clé de Java permettant de créer une nouvelle instance à partir d'une classe
- `nextLine()` : méthode de lecture d'une chaîne de caractères