Langages objets

Introduction

M2 Pro CCI, Informatique
Emmanuel Waller, LRI, Orsay

Cours 1 : Introduction

- Présentation du module
- Présentation de Java
- Généralités
- Ligne de commande et chaînes de caractères
- Exemples : if, for, while
- Environnement : Unix, Emacs
- Plan du module

présentation du module

logistique

- 12 blocs de 4h + 1 bloc 2h = 50h
  - 1h15 cours, 45mn exercices table, 2h TD machine
- page web du module

contrôle des connaissances

- examen : 3h
  - session 1 : tous documents autorisés
  - session 2 : aucun document autorisé
- contrôle continu (CC) :
  - Assiduité, ponctualité (arrivée, départ)
  - Interrogation 1h vers le bloc 10
  - Éventuellement autres (micro-interrogations, etc.)
- note module : 2/3 examen + 1/3 CC

but du module

- but du CCI :
  - programmation :
    - être autonome dans : spécifier, concevoir, programmer, tester et maintenir des (petites) applications de qualité
    - principes des langages de programmation (C), algorithmique, langages objets, projet C++
  - technologies :
    - utiliser les ordinateurs, avec leur outils matériels et logiciels
    - architecture, bases de données, réseau, etc.

- but du module
  - Java sans objets :
    - Programmation non objet : tout ce qu'on va voir a été vu dans modules précédents
    - on va refaire le point à partir de zéro sur tout, mais très vite
    - = un nouveau langage (partie sans objets), mais pas de nouveaux concepts
    - n'est pas : approfondir algorithmique et structures de contrôle
  - Concepts objets en Java
- prérequis : modules antérieurs

Pourquoi Java ?

- le langage Java (J2SE 1.4) :
  - autres langages : C, Fortran (calcul scientifique : bibliothèques), C++ (objet), Cobol (gestion, fichiers : paie, etc.), ML et Scheme (prototypage rapide), Perl et Python (chaîne de caractères en gros), etc.
  - possible : C++ ; mais... plus technique que Java, demande plus d'approfondissement pour écrire programme fiables (= plus de bugs délicats)

bibliographie

- Claude Delannoy, Programmer en Java, deuxième édition mise à jour, Eyrolles 2002, 35 euros (nouvelle présentation : « best-seller »)
  - pas d'exercices, pas destiné aux débutants objets (parfois complexe, le but n'est pas d'enseigner ou donner du recul sur la programmation objet), comparaisons avec C++
  - pédagogie excellente sur le reste
  - pour les concepts au programme de ce module, on suit la progression ce livre (sauf certains points cause contexte pédagogique CCI-LO)
  - module suit même numérotation des chapitres

- Claude Delannoy, Exercices en Java, Eyrolles 2001, 25 euros
- Java in a nutshell, O'Reilly, au moins 3ème édition, 2000
  - célèbre
  - peut-être plus objet, plus de recul, plus rapide, moins axé sur la pédagogie pour les débutants, moins adpaté à CCI-LO ?

environnement de programmation

- unix
- emacs
- ligne de commande

Fonctionnement d'un bloc

- Page « Fonctionnement » : détailler
  - Cours
  - TD
  - Travail personnel

esprit de la préparation à l'examen

- Vu dans page « Fonctionnement »
- travail personnel hors cours et TD
- entraînement dans le travail personnel :
  - rédiger l'exercice entièrement (sans compiler, ni indentation automatique, ni rien)
  - le tester à la main sur plusieurs tests (pas sur machine)
  - Compiler et exécuter : il doit fonctionner correctement du premier coup
- Examen an passé disponible sur page

Cours 1 : Introduction

- Présentation du module
- Présentation de Java
- Généralités
- Ligne de commande et chaînes de caractères
- Exemples : if, for, while
- Environnement : Unix, Emacs
- Plan du module

présentation de Java

intérêts de Java

- objets
- GUI (graphical user interface) par programmation événementielle
- portablité
- très nombreuses bibliothèques (hors programme module)

historique

- 1991 : Sun, code embarqué, syntaxe proche C++
- « compilation » :
  1. compilateur unique : programme source |-> code intermédiaire (byte codes), indépendant machine
  2. chaque architecture possède machine virtuelle Java (JVM) qui l'interprète (lenteur à l'exécution) : portabilité
- 1995 : HotJava, navigateur Web Sun en Java : exécute « applets » en byte codes

Java et la programmation orientée objets

- POO :
  - (programmation à objets, programmation par objets)
  - fiabilité des logiciels
  - réutilisation (au lieu réécrire)
  - nouveaux concepts (1986) : objets, encapsulation, classes, héritage, etc.
- POO pure : tout est objet, encapsulation, que méthodes (Simula, Smalltalk, Eiffel)
- Java : POO « + » types primitifs, procédures, encapsulation non obligatoire (possible programmation pas du tout objet)

Java et la programmation événementielle

- vu lors introduction à la programmation graphique

Java et la portabilité

- un programme source est portable s'il fonctionne dans différents environnements moyennant nouvelle compilation
- Java : code compilé portable

présentation de Java : récapitulatif

- objets
- GUI (graphical user interface) par programmation événementielle
- portablité
- (Delannoy chapitre 1)

Cours 1 : Introduction

- Présentation du module
- Présentation de Java
- Généralités
- Ligne de commande et chaînes de caractères
- Exemples : if, for, while
- Environnement : Unix, Emacs
- Plan du module

généralités

généralités

- premier exemple
- exécution
- instructions de base
- Déroulement mémoire
- lecture au clavier
- differentes sortes d'instruction
- règles d'écriture

- on va découvrir et manipuler intuitivement un premier programme complet
- on ne rentrera pas tout de suite dans tous les détails
- pas d'objets dans ce chapitre

premier exemple de programme Java

- écrire un programme qui affiche « Bonjour a tous »
- Class Bonjour {

```
    public static void main(String [ ] args) {
        System.out.println("Bonjour a tous");
    }
}
```

- c'est un peu lourd ! on va comprendre pourquoi au fur et à mesure du module

structure générale du programme

- class Bonjour { . . . }
  - définition d'une classe de nom Bonjour
  - { . . . } : le contenu de la classe
- public static void main(String [ ] args) { . . . }
  - définition d'une fonction de nom main
  - String [ ] args : le paramètre, de nom args, est un tableau de chaînes de caractères
  - void : cette fonction ne renvoie pas de valeur
  - { . . . } : le corps de la fonction : des instructions

- contenu du programme :
  - System.out.println("Bonjour a tous");
  - fonction d'impression prédéfinie
  - détails de syntaxe

bref, pour l'instant...

- un programme est une classe
- cette classe est composée d'une fonction main
- on place dans le corps de main les instructions qu'on veut
- il y a plusieurs choses obligatoires qu'on accepte d'écrire sans les comprendre (pour l'instant) : class, public, static, main, String[ ], System.out

exécution d'un programme Java

- fichier source
  - éditer programme source ci-dessus, sauver dans un fichier
  - nom quelconque + extension .java, ex : Ex1.java
- compilation
  - sur la ligne de commande Unix
  - unix> javac Ex1.java
  - génère Bonjour .class

- exécution
  - par la JVM : unix> java Bonjour

    affiche dans la fenêtre : Bonjour a tous
  - sauf en cas d'erreur :
    - messages
    - fichier des byte codes (exécutable) non généré
    - il faut « débuguer » (cf section bientôt)

démonstration : exemple complet

- Tous exemples cours déjà distribués
- ci-joint
- (programmation en direct de cet exemple : selon temps)
- javac, .class, java
- Ex : erreur compilation : pas de .class

exemple 2 : quelques instructions de base

```
class Exemple2 {
    public static void main(String [ ] args) {
        int n;
        double x;
        n = 5;
        x = 2*n + 1.5;
        System.out.print("n = ");
        System.out.print(n);
        System.out.println("Pi + racine de 2 = "
                                    + (Math.PI + Math.sqrt(2)));
        double y;
        y = n*x + 12;
        System.out.println("y vaut " + y);  } }
```

démonstration

- ci-joint

- Même structure qu'exemple 1 :

  class Exemple2 { . . . main . . . }

- int n;

  double x;

  - déclarations de variables
  - obligatoires
  - avant utilisation (pas nécessairement début bloc ou programme)
- n = 5;

  x = 2*n + 1.5;

  - affectation
  - conversion implicite automatique de 2*n en double

- System.out.print("n =");

  System.out.println(n);
  - paramètre : variable entière
  - pas de retour à la ligne : print
- Math.PI, Math.sqrt

- System.out.println("x = " + x);
  - paramètre :

  1. conversion automatique de x double en la suite de caractères représentant x en décimal
  2. concaténation
  3. affichage

  - dès qu'un opérande est une chaîne, l'autre est converti en chaîne
- déclaration « tardive » de y

remarques

- aucun objet, idem C
- System.out.println("x=" + 2*n + 1.5);

  affiche : x = 101.5
- System.out.println("x=" + (2*n + 1.5));

  affiche : x = 11.5
- levée usuelle des ambiguïtés par parenthèses

généralités

- premier exemple
- exécution
- instructions de base
- Déroulement mémoire
- lecture au clavier
- differentes sortes d'instruction
- règles d'écriture

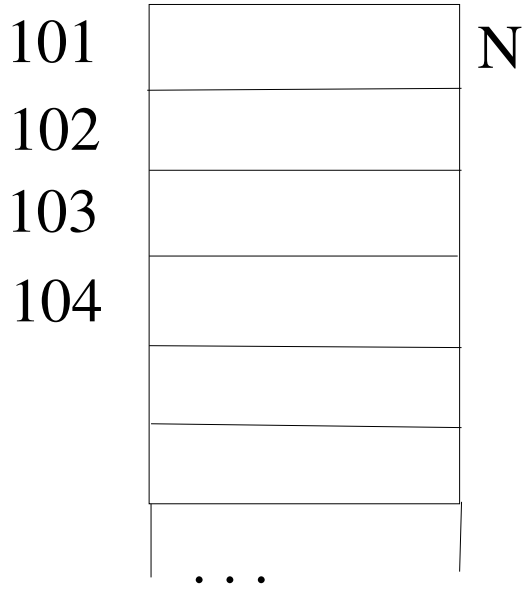
Déroulement mémoire

- Exemple ci-joint : déclarer, initialiser un entier l'incrémenter et l'afficher

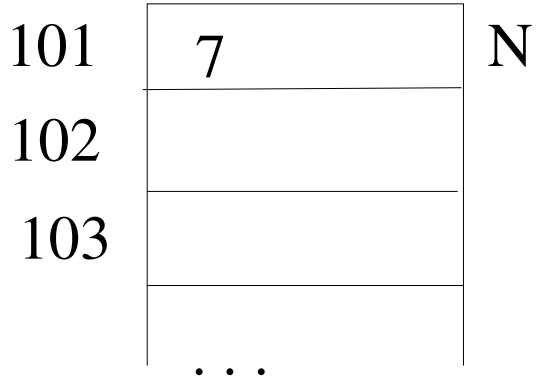
principe

- int n;
  - Réserve une case mémoire pour un entier (4 octets)
  - Alloue 4 octets de la mémoire, associés à la variable n de type int
  - n ne contient aucune valeur et est inaccessible (compilateur)
- n = 7;
  - Écrit 7 dans cet espace de 4 octets

Int n;



n = 7;



Exemple déroulement mémoire

- Au tableau : exemple ci-joint
- (Exercices possibles : dérouler tous les programmes de ce cours)

lecture d'informations au clavier

- moins immédiat que System.out.println
- sera vu en fin de module
- on va voir une manière restreinte mais simple bientôt

Différentes sortes d'instructions

- Instructions de déclaration
- Instructions exécutables
  - Instructions simples, comprenant le point-virgule
  - Instructions de contrôle (ex : for, if)
  - Bloc, délimité par accolades { et } (contient des instructions)

règles de présentation pour écrire un programme

- pouvoir se relire (et les autres aussi)
- noms pertinents
- une instruction par ligne
- indenter, toujours pareil
- commenter les passages délicats
- programme facilement utilisable (messages)

règles générales d'écriture

- Identificateur :
  - Lettre (majuscule, minuscule, souligné, éviter $) + lettre ou chiffre
    - Rq : A n'est pas a
  - Nombre de caractères quelconque

- Tradition (à respecter) :
  - Minuscules sauf suite
  - Première lettre classe majuscule
  - Variable, fonction :
    - Minuscule
    - Si juxtaposition mots : chaque initiale en majuscule (sauf le premier)
    - Ex : valeur, nombreDeValeurs
  - Classes : idem, sauf première lettre majuscule
  - Constantes symboliques : tout majuscule
  - Rq : System est classe, mais pas out
- Mots réservés (mots-clé) :
  - Liste dans documentation ou livre
  - Réservés

- Séparateurs
  - Entre identificateurs
  - Espace, fin de ligne, : , = ; * ( ) [ ] { } + - / < > & |
- Le format libre :
  - Espaces et fins de ligne non séparateurs :
    - Possibles à ajouter
    - Non obligatoires, mais indispensables lisibilité
  - Pas de fin de ligne dans les constantes chaîne
  - Impossible couper identificateurs

- Les commentaires :
  - Usuels (comme C) :
    - /* . . .

      . . . */
    - Non imbriqués
  - De fin de ligne (comme C++) :
    - // . . .
    - Rq :
      - /* a // b */ c
      - A // b /* c */ d
  - Commentaires de documentation (usuel) : /** . . . */ pour extraction automatique

En résumé

- Java reprend beaucoup des concepts et de la syntaxe de C
  - identique jusqu'ici (sauf détail syntaxe programme principal et déclaration fonction)
  - concepts déjà vus dans module Programmation
- on va voir bientôt (cours suivants) pourquoi la syntaxe est différente (ex : objets)
- rem : jusqu'ici : pas d'objets
- autrement dit : « Java sans objets c'est simplement C » (Java conçu pour)

Généralités : récapitulatif

- premier exemple
- exécution
- instructions de base
- Déroulement mémoire
- lecture au clavier
- differentes sortes d'instruction
- règles d'écriture
- (Delannoy chapitre 2)

Cours 1 : Introduction

- Présentation du module
- Présentation de Java
- Généralités
- Ligne de commande et chaînes de caractères
- Exemples : if, for, while
- Environnement : Unix, Emacs
- Plan du module

La ligne de commande et les chaînes de caractères

La classe String

- "Bonjour !" est une chaîne de caractères : String
- String s;

  s = "cou";

  s = s + "cou"; // s += "cou";

  / / on connaît déjà ce +

  System.out.println(s + " " + s);
- Conversion d'une chaîne en entier

  s = "123";

  int n = Integer.parseInt(s);

  System.out.println(n + 1); // affiche 124 ; rem : + int

démonstration

- ci-joint

Les arguments de la ligne de commande

- public static void main(String[] args)
- Fonction main reçoit tableau de chaînes, qui contient les arguments de la ligne (sauf nom du programme)

démonstration

- ci-joints
- Lire un entier et afficher son successeur
- Le bon
- Un mauvais

La ligne de commande et les chaînes de caractères : récapitulatif

- Variable String
- Affectation, concaténation
- Ligne de commande
- (Delannoy chapitre 9.1.1, 9.8)

Cours 1 : Introduction

- Présentation du module
- Présentation de Java
- Généralités
- Ligne de commande et chaînes de caractères
- Exemples : if, for, while
- Environnement : Unix, Emacs
- Plan du module

Exemples : if, for while

démonstration

- ci-joints
- If : tester une valeur lue au clavier
- For : afficher les entiers de 1 à 10 en décroissant
- While : idem

Cours 1 : Introduction

- Présentation du module
- Présentation de Java
- Généralités
- Ligne de commande et chaînes de caractères
- Exemples : if, for, while
- Environnement : Unix, Emacs
- Plan du module

Environnement : unix, emacs

Unix

- Operating system
- Données : fichiers, répertoire, arborescence (dessin)
- Actions sur données:
  - Manipuler : créer, détruire, modifier
  - Consulter
  - Se déplacer dans arborescence
- Démonstration complète : page Environnement

Emacs

- Editeur de texte
- Actions sur un fichier :
  - Création
  - Visualisation
  - Dans contenu :
    - Déplacements
    - Modifications : ajouts, suppressions (caractères)
- Démonstration complète : page Environnement

démonstration

- Démonstration page « Environnement »

Cours 1 : Introduction

- Présentation du module
- Présentation de Java
- Généralités
- Ligne de commande et chaînes de caractères
- Exemples : if, for, while
- Environnement : Unix, Emacs
- Plan du module

plan du module

- introduction, types primitifs, opérateurs et expressions, structures de contrôle
- tableaux
- champs de classe, fonctions, objets
- encapsulation, méthodes dynamiques
- héritage, redéfinition, polymorphisme et classes abstraites
- exceptions, entrées/sorties texte, introduction à la programmation graphique (?)

délégués ?