Objectifs

"Être capable d'analyser un problème, de proposer une stratégie pour le résoudre par ordinateur, et de programmer correctement cette solution."

- Analyse de problème
- Données et Algorithmes
- Approche orientée objet
- Java

Bref...

Programmation $\neq$ Java

Objectifs

"Être capable d'analyser un problème, de proposer une stratégie pour le résoudre par ordinateur, et de programmer correctement cette solution en Java."

- Analyse de problème
- Données et Algorithmes
- Approche orientée objet
- Java

Bref...

Programmation $\neq$ Java

Prérequis

- Commandes de base d´ordinateur
  - Environnement Unix (Linux)
  - Éditeur de texte (`vi`, `emacs`, ...)
  - Gestion des fichiers (`ls`, `cp`, `mv`, `rm`, ...)
- Aptitude en résolution de problèmes
- Mathématiques de base
- Aucun expérience en programmation requise

Séminaires Unix

`http://www-etud.iro.umontreal.ca/~semunix`

Introduction à Linux, aux commandes, et à l'usage des environnements informatiques du DIRO.

Vous devez y participer!

Prérequis

- Commandes de base d´ordinateur
  - Environnement Unix (Linux)
  - Éditeur de texte (`vi`, `emacs`, ...)
  - Gestion des fichiers (`ls`, `cp`, `mv`, `rm`, ...)
- Aptitude en résolution de problèmes
- Mathématiques de base
- Aucun expérience en programmation requise

Séminaires Unix

`http://www-etud.iro.umontreal.ca/~semunix`

Introduction à Linux, aux commandes, et à l'usage des environnements informatiques du DIRO.

Vous devez y participer!

Références

Il n'y a aucun livre obligatoire. Nous suggérons:

- Anne Tasso, *Le livre de Java premier langage*, Eyrolles, 2002 (ISBN 2-212-11100-2).
- C. Delannoy, Programmer en Java, Eyrolles, 2002 (ISBN 2-212-11119-3).

Un livre sur la **programmation** et les principes, pas les détails.
Pas besoin de livre sur l'API (applet, netneans, ...)

Références

Autres livres utiles:

- J. Niño et F. A. Hosch, An Introduction to Programming and Object Oriented Design, Wiley, 2002 (ISBN 0-471-35489-9).
- Cay Horstmann, Big Java. John Wiley
- Anne Tasso, Apprendre Java et C++ en parallèle, Eyrolles, 2002 (ISBN 2-212-11152-5)

Évaluation

Théorie		
Examen Intra	20%	16 octobre
Examen Final	30%	13 décembre
Pratique		
Examen Pratique	10%	1er novembre
10 Exercices	10%	1 par semaine
2 Travaux pratiques	30%	

Seuil

Conditions pour que les travaux pratiques et exercices comptent:

- Moyenne pondérée Intra et Final $\geq 40/100$
- Examen Pratique $\geq 40/100$

Intra et Final

Examen Intra

Examen écrit, durée 2 heures, matière théorique du début à mi-session, **aucune** documentation est permise

Examen Final

Examen écrit, durée 3 heures, matière théorique du début à la fin de la session, **aucune** documentation permise

L'emphase est placée sur la compréhension,
pas sur les détails ou le *par coeur*.

Examen Pratique

Évaluation directe de l'aptitude à réaliser un programme.

- se fait devant un ordinateur, en présence d'un évaluateur
- 2-3 problèmes simples programmés au complet
- durée à déterminer (autour de 30-45 minutes)
- un seul examen, à la mi-session
- droit de reprise
- doit être réussi avec une note supérieure ou égale à 40%.

Exercices

Visent à développer les aptitudes en programmation.

- Problèmes simples reliés à la matière vue chaque semaine
- doivent être réalisé **seul**
- remis sous forme de rapport écrit ou de programme
- 10 exercices (1/semaine)

Travaux pratiques

Vise à développer les aptitudes en programmation et la capacité de résoudre de plus gros problèmes.

- À réaliser **seul** ou en **équipe de deux**
- 2 travaux pratiques (15% chacun) dans la session

Plagiat

N'oubliez pas...

Faites vos travaux vous-même!

- Première délit: Zéro pour le travail
- Seconde délit: Échec au cours

En cas de doute sur l'origine d'un travail pratique, nous nous réservons le droit de questionner un étudiant sur les détails du travail remis.

Tous les membres d'une équipe doivent participer à **tous** les travaux pratiques.

Le plus important

Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code,
Taper du code, Taper du code, Taper du code, Taper du code.

Pourquoi Java?

Pourquoi pas?

- langage moderne (moins de détails étranges que le C++)
- utilise l'approche orientée objet (OO)
- syntaxe de base similaire au C et C++
- librairie standard très riche
- ... et c'est utilisé dans les cellulaires ...

Important : Le langage a peu d'importance

- un bon programmeur est bon dans tous les langages
- il peut les apprendre au besoin

Au programme

- Définir *"Informatique"*
- Qu'est-ce qu'un algorithme?
- Qu'est-ce qu'un ordinateur?
- Qu'est-ce qu'un programme?
- Les langages de programmation
- Les erreurs de programmation
- Mon premier programme Java

Informatique

"L'informatique est la science de l'abstraction. Elle vise à créer le bon modèle pour un problème et conçoit une technique ¨méchanisable¨ pour le résoudre."

(Aho & Ullman)

Construire un algorithme

Un ordinateur peut ...

- calculer
- compter
- trier
- rechercher

... à condition qu'on lui dise quoi faire.

Construire un algorithme

Un ordinateur est rapide, .

L'algorithme définit la marche à suivre des tâches à exécuter.

Construire un algorithme

Un ordinateur est rapide, mais stupide.

L'algorithme définit la marche à suivre des tâches à exécuter.

Construire un algorithme

Un ordinateur est rapide, mais stupide.

L'algorithme définit la marche à suivre des tâches à exécuter.

Programmer = Cuisiner

Problème

Comment faire une crème anglaise?

- Ingrédients
  - 8 jaunes d'oeufs
  - 250g de sucre
  - 500cl de lait
  - vanille
- Algorithme
  - battre les jaunes et le sucre en un mélange lisse
  - ajoutez graduellement le lait porté à ébullition au mélange en brassant
  - laissez cuire 12 minutes en remuant et en évitant de faire bouillir

Programmer = Cuisiner

Question

Quel est le bon niveau de détail?

- Ingrédients
  - 8 oeufs
  - ...
- Algorithme
  - pour chacun des 8 oeufs, faire:
    - casser la coquille sur le bord d'un bol
    - séparer le jaune du blanc

Dans une recette normale

- beaucoup d'ingrédients implicites
- beaucoup d'algorithmes plus simples supposés connus

Programmer = Cuisiner

Question

Quel est le bon niveau de détail?

- Ingrédients
  - 8 oeufs
  - ...
- Algorithme
  - pour chacun des 8 oeufs, faire:
    - casser la coquille sur le bord d'un bol
    - séparer le jaune du blanc

Dans une recette normale

- beaucoup d'ingrédients implicites
- beaucoup d'algorithmes plus simples supposés connus

Programmer = Cuisiner

Une recette complexe doit utiliser des ingrédients complexes et des algorithmes connus.

Problème

Ccomment faire une charlotte aux poires?

- Ingrédients
  - 750cl de crème anglaise (voir page 73)
  - 500cl de crème fouettée
  - 1kg de poires pochées (voir page 22)
  - ...
- Algorithme
  - Mélanger la crème anglaise et la crème fouettée
  - Étendre en couches successives le mélange et les poires
  - ...

Qu'est-ce qu'un programme?

Un programme est une séquence d'instructions à effectuer.

- séquence → l´ordre est important
- langage ≡ un ensemble d'instructions élémentaires

Niveaux de programmation

- langages machine et assembleur
- langages évolués (C, C++, Java, ...)
- langages spécialisés (Oracle, SPSS, Mathematica, PHP, Tcl/Tk, Perl, Python, Awk, ...)

Qu'est-ce qu'un ordinateur

Ordinateur

Appareil d'usage général pour traiter de l'information

- information qui entre et qui sort (entrée-sortie)
- stockage et traitement de l'information

Stockage de l'information

- Physiquement: *interrupteur* à deux états, *on* ou *off*
- Au niveau abstrait le plus bas: nombres binaires, 1 ou 0
- Au niveau abstrait plus haut: nombres décimaux, réels, caractères
- Au niveau abstrait le plus haut: mp3, video, document, ...

Qu'est-ce qu'un ordinateur

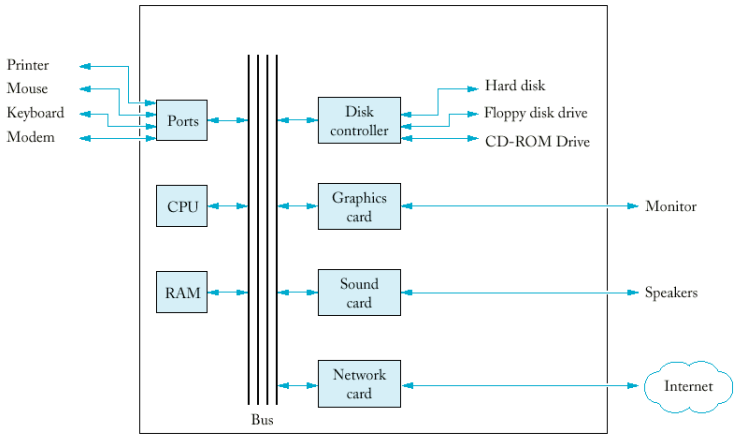
Stockage des données

- nombres entiers dans un intervalle donné: $0 = 0000$, $9 = 1001$, $15 = 1111$
- caractères (code ASCII): 'A'$= 01000001$, 'B'$= 01000010$
- images noir et blanc: 1 bit/pixel
- etc.

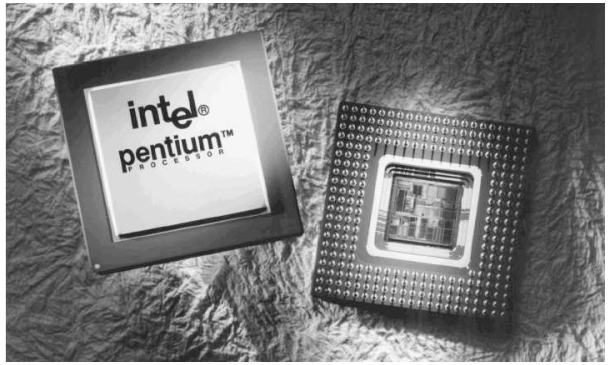
Architecture de l'ordinateur

- Unité centrale (central processing unit – CPU)
  - le ¨cerveau¨ qui exécute les instructions
- Mémoire (random access memory – RAM)
  - une collection des tiroirs numérotés contenant des nombres
- Périphériques
  - clavier, écran, carte de réseau, etc.

Diagramme schématique d'un ordinateur



Le CPU



- Exécute des instructions très simples
- Exécute des instructions très rapidement

Programme de style "recette"

Tâche

Additionner deux nombres et afficher le résultat à l'écran

Le CPU a un accumulateur: mémoire "locale" pour les paramètres et les résultats du calcul

Programme de style "recette"

Programme

1. Lire le premier nombre du clavier et le placer dans la mémoire.
2. Lire le deuxième nombre du clavier et le placer dans la mémoire.
3. Charger le premier nombre de la mémoire dans l'accumulateur.
4. Ajouter le deuxième nombre de la mémoire au nombre qui se trouve dans l'accumulateur, et placer le résultat dans l'accumulateur.
5. Stocker le nombre qui se trouve dans l'accumulateur dans la mémoire.
6. Afficher le résultat sur l'écran.

Ordinateur imaginaire simplifié

Mémoire (RAM)

- taille de 32 octets
- adresses de 5 bits

CPU

- accumulateur: mémoire "locale" de 8 bits
- 5 instructions élémentaires: `read`, `write`, `load`, `save`, `add`
- chaque instruction a un paramètre

Ordinateur imaginaire simplifié

Instructions élémentaires du CPU

- `read` n: "lire un nombre d'un octet du clavier et le placer dans la mémoire à l'adresse n"
- `write` n: "afficher sur l'écran le nombre qui se trouve dans la mémoire à l'adresse n"
- `load` n: "mettre le nombre qui se trouve dans la mémoire à l'adresse n dans l'accumulateur"
- `save` n: "mettre le nombre qui se trouve dans l'accumulateur dans la mémoire à l'adresse n"
- `add` n: "ajouter n au nombre qui se trouve dans l'accumulateur, et placer le résultat dans l'accumulateur"

Programme assembleur

Code

```
1. read  20
2. read  21
3. load  20
4. add   21
5. save  22
6. write 22
```

Programme en langage machine

Codes binaires des instructions:

Code

```
read = 000
write = 001
load = 010
save = 011
add = 100
```

Programme:

Code

```
1. read  20  =  00010100
2. read  21  =  00010101
3. load  20  =  01010100
4. add   21  =  10010101
5. save  22  =  01110110
6. write 22  =  00110110
```

Programme en langage machine

Exécution du programme:

- écrire le programme dans un fichier
- charger le programme du fichier dans la mémoire à l'adresse 5
- commander au CPU d'interpréter le contenu de la mémoire à partir de l'adresse 5 jusqu'à l'adresse 10 comme un programme
- exécuter le programme instruction par instruction

Programme en langage machine

Le cycle d'exécution:

1. $i \leftarrow 5$
2. lire le contenu de l'adresse i de la mémoire
3. décoder les premiers 3 bits pour obtenir l'instruction `inst`
4. décoder les derniers 5 bits pour obtenir le paramètre `n`
5. exécuter `inst n`
6. $i \leftarrow i + 1$
7. **Si** $(i == 11)$ **Alors** STOP **Sinon** GOTO 2

Langage machine

Avantages

- très rapide
- accès direct aux composantes matérielles

Inconvénients

- difficile à écrire et lire (comprendre)
- détection des erreurs très difficile

Langage de haut niveau

Avantages

- isole le programmeur du langage machine
- instructions plus riches et plus compréhensibles
- concepts plus abstraits (variable, structure, ...)

Inconvénients

- incompréhensible pour l'unité centrale (CPU)
- doit être traduit (ou interprété) en langage machine pour s'exécuter $\longrightarrow$ compilation

Langages de haut niveau

Code

```
int num1,num2,sum;
num1 = read();
num2 = read();
sum = num1 + num2;
print(sum);
```

- instructions plus compréhensibles
- instructions de niveau plus élevé
- variables: références abstraites au contenu de la mémoire

Erreurs de programmation

Ordinateur = exécutant

- l'ordinateur est incapable de distinguer une "bonne" instruction d'une "mauvaise"
- l'ordinateur exécute tout ce qu'on lui demande d'exécuter
- c'est la responsabilité du programmeur de s'assurer que l'ordinateur fait vraiment ce qu'on veut qu'il fasse

Stratégies de programmation

Approche procédurale

PROGRAMME = ensemble de méthodes (recettes)

Exemples

- PASCAL, FORTRAN, C

Inconvénients

- technologie "pré-industrielle"
- les morceaux de programmes ne sont pas interchangeables
- besoin d'experts très compétents pour le développement et la maintenance
- le développement est long

Stratégies de programmation

Approche orientée objet

PROGRAMME = ensemble d'objets collaborants

Exemples

- Ada, C++, Java

Avantages

- permettre la conception de haut niveau
- le développent est moins long, plus sûr
- les programmes sont plus faciles à comprendre
- plus facile d'échanger les morceaux de différents programmes

Le langage Java

Historique

- Sun Microsystems
- 1991: conception d'un langage indépendant du hardware
- 1994: browser de HotJava, applets
- 1996: Microsoft et Netscape commencent à soutenir
- 1998: l'édition Java 2: plus stable, énorme librairie
- ...

Le langage Java

Avantages

- simplicité, régularité, sécurité
- portabilité (indépendance du matériel et du système d'exploitation)

Désavantages

- les programmes simples sont tout de même assez compliqués
- vitesse d'exécution (de moins en moins problématique)

Le premier programme

Le fichier `Hello.java`

Code

```
public class Hello
{
    public static void main(String[] args)
    {
        // affiche un message
        System.out.println("Hello, World!");
    }
}
```

Le premier programme

Le fichier `Hello.java`

Compilation

```
javac Hello.java
```

Éxécution

```
java Hello
```

ce qui s'affiche

```
Hello, World!
```

La syntaxe

Grammaire du langage

- sensible à la casse (aux majuscules/minuscules)
- indépendant de la mise en page

Éléments syntaxiques

- mots réservés: `public`, `class`, `static`, `void`
- identificateurs: `args`, `Hello`, `main`, `String`, `System`, `out`, `println`
- littéraux: `"Hello World!"`
- commentaires: `// Affiche ...`

La définition d'une classe

L'entête

```
public class NomDeClasse
```

- `public` ≡ tout le monde peut utiliser la classe
- `class` ≡ élément de base des programmes OO
- une classe par fichier
- `class NomDeClasse` doit être dans le fichier `NomDeClasse.java`

La définition d'une classe

Corps

```
{
  ...
  ...
}
```

- contient un ensemble d'instructions, délimitées par { }

Conventions

- le nom de classe: `NomDeClasse`
- l'indentation des { } et du contenu

La définition d'une méthode

Entête

```
public static void main(String[] args)
```

- machine d'entrée-sortie
- `main`: nom de méthode
- `void`: aucune sortie
- `String[] args`: le paramètre (entrée)
- `String[]`: le type du paramètre
- `args`: le nom du paramètre

L'appel d'une méthode

Entête générale

`nomDObjet.nomDeMethode(<liste des paramètres>)`

- `System.out`: l'objet qui représente le terminal (l'écran)
- `println`: la méthode qui imprime son paramètre
- `"Hello, World!"`: le paramètre de `println`

La méthode `main`

- "`java Hello`" exécute la méthode `main` dans la classe `Hello`

La suite...

Il ne reste plus qu'à expliquer en détail...

... et à mettre tout ca en pratique!

La suite...

Il ne reste plus qu'à expliquer en détail...

... et à mettre tout ca en pratique!

FIN