Introduction au langage Java

Valvassori Moïse

Master IMA 1 – Paris 8

11 octobre 2005

Plan du cours

1 –	Introduction à Java (rappels)	MV
2 –	Java embarqué, MIDP et CLDC	NJ
3 –	Java API Games	NJ
4 –	XML	MV
4 –	XML pour J2ME	MV
6 –	Java et réseau	NJ

+ séances de TP

Programmation impérative

- état dans des variables
- exécution séquentielle

Découpage fonctionnel

- programme découpé en fonctions
- évite les répétitions de code
- variables locale

Programmation structurée et modulaire

- variable $\longrightarrow$ structure
- regroupement de fonctions en modules

Objectifs

Améliorer:

- modularité
- sécurité
- conception
- portabilité

Objet

- état
- comportement
- communique par envoie de message

Classe

- modèle d'objet.
- peut être instanciée
- contient:
  - des attributs
  - des méthodes
  - des classes

Encapsulation

- attributs et méthodes inclus dans classes
- accès restreints suivant les cas.

Example

Objet *passager* n'a pas accès à l'attribut *pression* des objets *roues*.

Instance

- objet créé à partir d'une classe
- créé par un *constructeur*
- détruit par un *destructeur*

Méthode de classe

- est invoquée sans faire référence à une instance
- affecte la classe dans son ensemble

Héritage

- partager des attributs et méthodes communs
- spécialise une classe
- surcharge des méthodes

Interface

- “contrat” d’implémentation
- collection de signatures de méthodes
- met en relation des classes distantes conceptuellement

Classe abstraite

- classe ne contenant pas toutes ses méthodes implémentés
- ne peut être instanciée
- les sous-classes doivent fournir une implémentation
- permet de définir des comportements génériques

Résumé

- modularité: classe, objet
- sécurité: encapsulation
- conception: héritage, interface
- portabilité: tout

Java...

- créé par Sun
- langage objets “impurs”
- multi-plateforme
- industriel

Composant de Java

- langage
- JVM
  - machine virtuelle
  - bytecode
  - compilation JIT
- bibliothèque

Version de Java

- J2SE
- J2EE
- J2ME
- JavaCard

Sur le WEB

- http://java.sun.com/
- http://java.sun.com/tutorial/
- http://java.sun.com/j2se/1.5.0/docs/
- http://java.sun.com/j2se/1.5.0/docs/api/
- http://java.sun.com/j2me/docs/

Premier Programme

Hello.java

```
/** Affiche "Bonjour le monde" */
public class Hello {
  public static void main (String a[]) {
    System.out.println("Bonjour le monde");
  }
}
```

Premier Programme

Hello.java

```
/** Affiche "Bonjour le monde" */
public class Hello {
  public static void main (String a[]) {
    System.out.println("Bonjour le monde");
  }
}
```

- compilation:
  `$ javac Hello.java`

Premier Programme

Hello.java

```
/** Affiche "Bonjour le monde" */
public class Hello {
  public static void main (String a[]) {
    System.out.println("Bonjour le monde");
  }
}
```

- compilation:
  `$ javac Hello.java`
- compilation produit `Hello.class`

Premier Programme

Hello.java

```
/** Affiche "Bonjour le monde" */
public class Hello {
  public static void main (String a[]) {
    System.out.println("Bonjour le monde");
  }
}
```

- compilation:
  `$ javac Hello.java`
- compilation produit `Hello.class`
- exécution:
  `$ java Hello`

Types (1)

type	taille/format
`boolean`	`true` / `false`
`char`	unicode 16 bits
`byte`	8 bits
`short`	16 bits
`int`	32 bits
`long`	64 bits
`float`	32 bits IEEE 754
`double`	64 bits IEEE 754

Types (2)

N'existe pas:

- `enum`
- `struct`
- `union`
- `typedef`

Variable

- se déclare où on veut
- possèdent toujours une valeur par défaut

$n! \longrightarrow 0$?

```
public static int fact(int n) {
  int result;
  for (int i=1; i<=n; i++)
    result *= i;
  return result; // renvoie tj 0
}
```

Structure de contrôle

- pas de coercition entre booléens et nombres.

Incorrect

```
int n;
while (n) {
...
}
```

Correct

```
if (n != 0) {
...
}
```

Mémoire

- pas de pointeurs
- (presque) pas de `malloc`/`free`
- garbage collector

Divers

- pas d'argument variable
- pas de `goto`
- `argv[0]` $\not\rightarrow$ nom du programme

Déclaration d'une classe

Simple

Fichier Voiture.java

```
package fr.univ-paris8.mime.voiture;
import fr.univ-paris8.mime.vehicule.Vehicule;
public class Voiture {
  // Attributs
...
  // Méthodes
...
  // Classes internes
...
}
```

Instanciation d'une classe

Quelques part...

```
voiture = new Voiture();
Voiture voiture2 = new Voiture("Marque", 1723);
String maChaine = new String("une chaine de char");
```

Déclaration d'un attribut

Fichier `Voiture.java`

```
public class Voiture {
  private Volant volant;
...
}
```

Modifieurs
Contrôle d'accés

Modifieur	Classe	Package	Sous-classe	Partout
private	☑	☐	☐	☐
rien	☑	☑	☐	☐
protected	☑	☑	☑	☐
public	☑	☑	☑	☑

Modifieurs

Autres

- `final`
- `static`
- `transient`
- `volatile`
- `synchronized`
- `abstract`

Accès à un attribut

Fichier Voiture.java

```
public class Voiture {
  public Volant volant;
...
  volant= ...;
  tourner(volant);
  Voiture v = new Voiture();
  v.volant= ...;
  v.tourner(volant);
}
```

Déclaration et accès à un tableau

Quelques part dans le code

```
int a[];
a = new int[10];

String b[] = new String[n];
for (int i=0; i<b.lenght; i++)
  System.out.println(b[i]);
```

Déclaration d'une méthode

Fichier Voiture.java

```
public class Voiture {
  protected double deplacer(double vitesse) {
    ...
  }
  public void deplacer() {
    ...
  }
  public String toString(){
    return nomVoiture;
  }
}
```

Appel d'une méthode

Fichier Voiture.java

```
public class Voiture {
...
  deplacer(20.0);
...
  Voiture v = new Voiture();
  v.deplacer();
}
```

Déclaration d'un constructeur

Fichier Voiture.java

```
public class Voiture {
  public Voiture(String marque){
    ...
  }
  public Voiture (String marque, int modele) {
    ...
  }
}
```

Traitement des erreurs

Récupération

- capture des exceptions

Division par 0

```
try{
  resultat = x / 0;
} catch (ArithmeticException e) {
  System.err.println(e+"division par 0");
}
```

Traitements Erreurs

Transmission

Dans la classe Voiture

```
void deplacer() throws VitesseExcessiveException {
  if (vitesse > VITESSE_MAX)
    throw new VitesseExcessiveException("Flash");
}

void conduire () {
  try {
    deplacer();
  } catch (VitesseExcessiveException e) {
    e.printStackTrace();
  }
}
```

Déclaration d'une classe

Héritage

- héritage par défaut: classe `Object`

Fichier `Voiture.java`

```
public class Voiture extends Vehicule {
...
}
```

Déclaration d'une classe

Classe abstraite

Fichier Vehicule.java

```
public abstract class Vehicule {
  public abstract void deplacer();
...
}
```

Fichier Voiture.java

```
public class Voiture extends Vehicule {
  public void deplacer() {
...
  } }
```

Déclaration d'une interface

- contient des méthodes abstraites
- contient des constantes (`static final`)
- peut hériter d'une interface

Fichier `Deplacable.java`

```
public interface Deplacable {
  public void deplacer();
}
```

Implémenter une interface

Fichier Vehicule.java

```
public class Vehicule implements Deplacable {
  public void deplacer() {
  }
}
```

Fichier Animal.java

```
public class Animal implements Deplacable {
  public void deplacer() {
  }
}
```

Variables spéciales

- `this`
- `super`

Programme Java

- ensemble de
  - *package*
  - classes
  - ressources
- peut être regroupé dans des archives `jar`

Chemin de classes

Où trouver les classes ?

- variable d'environnement `CLASSPATH`
- argument `-cp` ou `-classpath`
- séparateur de chemin:
  - ":" (UNIX)
  - ";" (Windows)

Compilation / exécution

- `javac [ options ] [ sourcefiles ] [ @argfiles ]`
- `java [ options ] class [ argument ... ]`
  `java [ options ] -jar file.jar [ argument ... ]`

Options:

- `-cp`
- `-D`*`property=value`*

Archives

- `jar cf jarfile [-C dir] inputfiles`
- `jar cmf manifest jarfile [-C dir] inputfiles`
- `jar xf jarfile [inputfiles]`

manifest:

- `Class-Path:`
- `Main-Class:`

Ant

- `make` pour java
- configure par fichier XML
- `http://ant.apache.org/`

Documentation du code

Fichier Voiture.java

```
/**
 * Documentation de la classe Voiture
 * @author "Moise" <moise@valvassori.org>
 * @version $Id$
 * @since Sun Oct 9 15:54:22 2005
 */
public class Voiture {
```

Documentation du code

Fichier Voiture.java

```
   /**
   * Déplace la voiture.
   * @param vitesse la vitesse de déplacement de la
voiture
   * @return renvoie la distance parcourue
   * @exception VitesseExcessiveException si on roule
trop vite
   */
  public int seDeplacer(double vitesse) throws
VitesseExcessiveException {
```

Documentation du code

javadoc

```
javadoc [ options ] [ packagenames ] [
sourcefilenames ] [ -subpackages pkg1:pkg2:...  ]  [
@argfiles ]
```

- -sourcepath
- -d
- -private

Fichiers

- 1 classe $\longrightarrow$ 1 fichier
- nom de classe $\longrightarrow$ nom de fichier

Nommage

- NomDeClasse
- nomDeVariable
- nomDeMethode
- NOM_DE_CONSTANTE

Package java.lang

- Object
- Integer, Boolean, String, ...
- Math
- Thread, Runnable
- System

Package java.util

- Vector
- Hashtable
- Collection
- Enumeration, Iterator

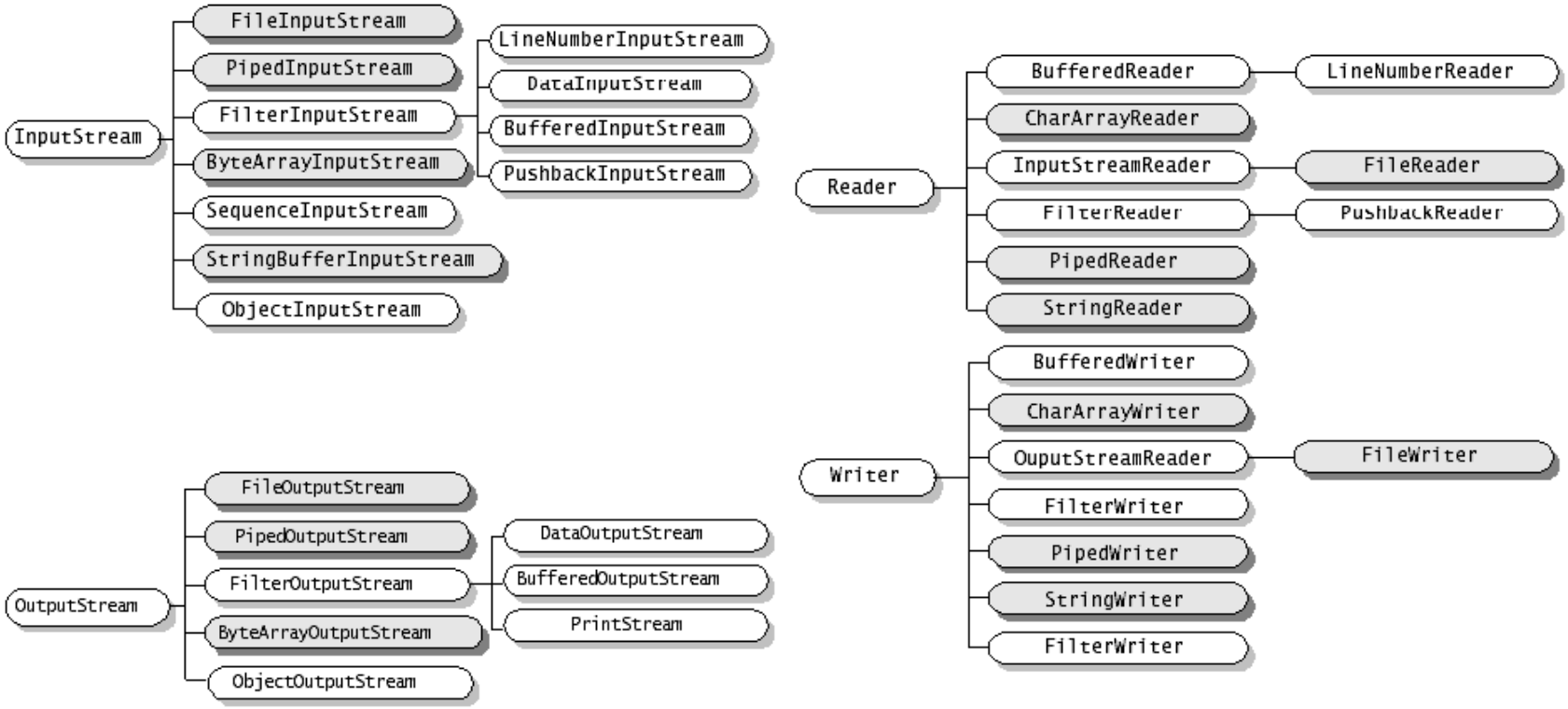
Enumeration d'un vecteur

```
for (Enumeration e = v.elements() ;
    e.hasMoreElements() ;)
  System.out.println(e.nextElement());
```

Entrées / Sorties

- flux
- 2 types:
  - flux de caractères: `Reader` / `Writer`
  - flux d'octets: `InputStream` / `OutputStream`

Hiérarchie des flux



Exemple: Copie de fichiers

Fichier Copy.java (1/2)

```
import java.io.*; // * → beurk

public class Copy {
  public static void main(String[] args)
      throws IOException {
    File inputFile = new File(args[0]);
    File outputFile = new File(args[1]);

    FileReader in = new FileReader(inputFile);
    FileWriter out = new FileWriter(outputFile);
    int c;
```

Exemple: Copie de fichiers

Fichier Copy.java (2/2)

```
    while ((c = in.read()) != -1)
     out.write(c);

   in.close();
   out.close();
  }
}
```