Environnement de compilation et d'exécution

Mise à niveau Java

Y. Boichut & F. Moal

packages

Définition des packages

- Les classes Java sont regroupées en paquetages (packages en anglais)
- Les paquetages offrent un niveau de modularité supplémentaire pour
  - réunir des classes suivant un centre d'intérêt commun
  - la protection des attributs et des méthodes
- Ils sont définis de façon hiérarchique (comme les répertoires), le séparateur étant le .
- Le langage Java est fourni avec un grand nombre de paquetages
- Un développeur DOIT mettre ses classes dans des packages qu'il définit

Quelques paquetages du SDK

- java.lang : classes de base de Java
- java.util : utilitaires, collections
- java.io : entrées-sorties
- java.awt : interface graphique
- javax.swing : interface graphique avancée
- java.applet : applets
- java.net : réseau
- java.rmi : distribution des objets

Nom complet d’une classe

- Le nom complet d'une classe (qualified name dans la spécification du langage Java) est le nom de la classe préfixé par le nom du paquetage :

  java.util.ArrayList

- Une classe du même paquetage peut être désignée par son nom « terminal » (les classes du paquetage java.util peuvent désigner la classe ci-dessus par simplement « ArrayList »)
- Une classe d’un autre paquetage doit être désignée par son nom complet

Importer une classe d’un autre package

- Pour pouvoir désigner une classe d'un autre paquetage par son nom terminal, il faut l’importer :

```
import java.util.ArrayList;

public class Classe {

    …

    ArrayList liste = new ArrayList();
```

- On peut utiliser une classe sans l’importer ! l’importation permet seulement de raccourcir le nom d’une classe dans le code

Importer toutes les classes

- On peut importer toutes les classes d'un paquetage :

```
import java.util.*;
```

- Les classes du paquetage java.lang sont automatiquement importées dans tout code Java

=> si une erreur de classe non trouvée apparaît sur des classes de java.lang (eg String), c’est que votre IDE ou compilateur ne trouve pas le JDK de Java !

Ambiguïté des noms de classes

- On aura une erreur à la compilation si
  - 2 paquetages ont une classe qui a le même nom
  - ces 2 paquetages sont importés en entier
  - la classe commune aux 2 paquetages est désignée court
- Exemple (2 classes List) :

```
import java.awt.*;
import java.util.*;
...
List l = getListe();
```

- Pour lever l'ambiguïté, on devra donner le nom complet de la classe. Par exemple, java.util.List l = getListe();

Import static

- Depuis le JDK 5.0 on peut importer des variables ou méthodes statiques d'une classe ou d'une interface avec « import static »
- On allège ainsi le code, par exemple pour l'utilisation des fonctions mathématiques de la classe java.lang.Math
- Exemple :

```
import static java.lang.Math.*;
public class Machin {
. . .
x = max(sqrt(abs(y)), sin(y));
```

- On peut importer une seule variable ou méthode :

```
import static java.lang.Math.PI;
```

Mettre sa classe dans un package

```
package nom-paquetage;
```

- doit être la première instruction du fichier source définissant la classe (avant même les instructions import)
- toutes les classes du fichier sont alors dans ce package
- Par défaut, une classe appartient au « paquetage par défaut » qui n'a pas de nom, et auquel appartiennent toutes les classes situées dans le même répertoire (et qui ne sont pas dans un paquetage particulier)

! A NE PAS UTILISER !

sous-package

- Un paquetage peut avoir des sous-paquetages

  Par exemple, java.awt.event est un sous-paquetage de java.awt

- Attention ! L'importation des classes d'un paquetage n'importe pas les classes des sous-paquetages

  on devra écrire par exemple :

```
import java.awt.*;

import java.awt.event.*;
```

Nom des packages

- Le nom d’un paquetage est hiérarchique :

  java.awt.event

- Pour éviter les conflits de noms identiques quand on diffuse son code, il est conseillé de préfixer ses propres paquetages par son nom de domaine inversé :

  fr.miage.orleans.event

Placement d’un package

- Les fichiers .class doivent se situer dans l’arborescence d’un des répertoires du classpath
- Le nom relatif du répertoire par rapport au répertoire du classpath doit correspondre au nom du paquetage
- Par exemple, les classes du paquetage

  fr.miage.orleans.liste doivent se trouver dans un répertoire fr/miage/orleans/liste relativement à un des répertoires du classpath

Visibilité des classes dans les packages

Si la définition de la classe commence par public class, la classe est accessible de partout

Sinon, la classe n’est accessible que depuis les classes du même paquetage

Compiler les classes d'un package

javac -classpath rep-class -sourcepath rep-source -d rep-compil Classe.java

- -d défini le répertoire racine où sera rangé le fichier compilé
- -classpath précise où sont les classes déjà compilées (dépendances)
- –sourcepath indique des emplacements pour des fichiers sources

Complexe de le faire à la main ! Respecter une structure simple :

- un répertoire racine pour le projet
- les sources dans le sous-répertoire src
- les .class dans un sous-répertoire classes (ou out, ...)
- lancer DANS LE REPERTOIRE racine du projet :
  javac -classpath classes -sourcepath src -d classes package/Classe.java

Exécuter une classe d'un package

On lance l'exécution de la méthode main d'une classe en donnant le nom complet de la classe (préfixé par le nom de son paquetage)

Par exemple, si la classe C appartient au paquetage p1.p2 :

java p1.p2.C

Le fichier C.class devra se situer dans un sous-répertoire p1/p2 d'un des répertoires du classpath (option -classpath ou variable CLASSPATH)

Jeu :P

Dans un terminal :

- créer une classe publique vehicule.locomotion.Roue qui possède un diamètre entier, et une classe PetiteRoue dans le même fichier
- créer une classe vehicule.Velo qui possède 2 roues et un main qui créer un Velo
- compiler la classe Velo à l’aide de javac
- lancer le main de Velo

Recherche des classes par la JVM

Chemin de recherche des classes

Les outils java et javac recherchent toujours d'abord dans les fichiers système qui sont placés dans le répertoire dit <java-home>, le répertoire dont le nom commence par jre

Ces fichiers système sont :

- fichiers rt.jar et i18n.jar dans le répertoire jre/lib où java a été installé,
- fichiers .jar ou .zip dans le sous-répertoire jre/lib/ext

Ils regardent ensuite dans le classpath

Classpath

- Le classpath contient par défaut le seul répertoire courant (« . »)
- Si on donne une valeur au classpath, on doit indiquer explicitement le répertoire courant si on veut qu'il y soit
- Le classpath indique les endroits où trouver les classes et interfaces :
  - des répertoires (les classes sont recherchées sous ces répertoires selon les noms des paquetages)
  - des fichiers .jar ou .zip
  - depuis le JDK 6, « * » indique que tous les fichiers .jar placés directement sous le répertoire sont inclus ; par exemple, lib/*
- La recherche d'une classe se termine dès qu'elle a été trouvée dans une des entrées, en commençant par les entrées les plus à gauche

Exemple de classpath

- Sous Unix, le séparateur est « : » :

  .:~/java/mespackages:~/mesutil/cl.jar

- Sous Windows, le séparateur est « ; » :

  .;c:\java\mespackages;c:\mesutil\cl.jar

Variables d’environnement

PATH : chemin d’accès vers les utilitaires java (java, javac, ...)

CLASSPATH : à éviter !

JAVA_HOME : vers le répertoire d’install de Java ; utilisée par certaines applications java (Tomcat, ...)