Chapitre 10

Les packages

Définition et utilisation des packages en JAVA

1. Les packages

1.1. Définitions

Le package est une unité de programmation permettant de regrouper et architecturer les classes du langage Java (prédéfinies ou développées) dans des répertoires et accessibles aux autres unités de programmation (programme Java, Applet, …).

Introduire la notion de unité de programmation (librairie, module, packages)
Couches logiciels,
Nommage
Hiérarchie
Accessibilité
Réutilisation
Equipe de développement

Un package a donc deux aspects :
- le répertoire qui contient les sources .java des classes du package
- le répertoire qui contient les "binaires" .class des classes du package.

Il y a donc deux utilisations des packages :
- lors de la compilation des fichiers sources
- lors de l'exécution des fichiers class

Un package est un **répertoire** de fichier .java ou de fichier .class.

Un répertoire de package peut contenir d'autres répertoires qui peuvent contenir eux-mêmes d'autres packages.

On obtient donc une architecture d'arborescence de packages et donc une arborescence des fichiers .java et .class.

L'accès d'un package se fait donc par son chemin d'accès dans le système de fichier de l'OS utilisé.
Le nom du package est le nom du répertoire ou le nom d'accès au répertoire.

Un package est transportable, déplaçable, compressible.

Une API (Application Programming Interface) est une arborescence de package.

1.2. Création et utilisation simple des packages

Simple = les fichiers .java et les fichiers .class sont côte à côte, et les packages sont dans le répertoire de compilation.

Pour créer un package de .java, il faut :

1/ Créer un répertoire dont le nom est celui du package
2/ Commencer chacun des fichiers .java du répertoire par la ligne :

```
package <nom du package> ;
```

Et c'est tout !!

Exemple : (**Exemple39_Packages**)
Soit l'architecture des répertoires suivante :

Exemple39_Packages
exemple1
MainPP.java
pkg1
A.java
B.java
HorsPackage.java
bin

Le fichier A.java

```
package pkg1;   //nom du package

public class A
{
    public A()
    {
        System.out.println("Création d'un A");
    }
}
```

Le fichier B.java

```
package pkg1;   //nom du package

public class B
{
    public B()
    {
        System.out.println("Création d'un B");
    }
}
```

Le fichier HorsPackage.java

```
public class HorsPackage
{
    public HorsPackage()
    {
        System.out.println("Création d'un HorsPackage");
        //HorsPackage h = new HorsPackage();  // Erreur de compilation
    }
}
```

Dans cette exemple le répertoire **pkg1** est un package car il contient au moins une classe qui est déclaré comme appartenant au package pkg1 : A.java et B.java

Un répertoire de package peut contenir un fichier .java qui n'est pas déclaré comme appartenant au package mais comme on le verra cette classe ne sera pas accessible à travers le package. Ici : HorsPackage.java.

Compilation du package :
Dans le répertoire pkg1, avec la commande :

```
javac –d ../bin *.java
```

bin :

pkg1/A.class
pkg1/B.class
HorsPackage.class

Utilisation du package :
Soit le programme suivant qui se trouve dans le répertoire Exemple1 : MainPP.java

```
import pkg1.A;
import pkg1.B;

public class MainPP
{
    static public void main(String[] args)
    {
        A a = new A();
        B b = new B();
        //HorsPackage h = new HorsPackage();  // Erreur de compilation
    }
}
```

Commande d'exécution :
Dans le répertoire Exemple1 : **java –classpath "." MainPP**

Les commandes d'import sont essentielles. Elles permettent d'indiquer les classes appartenant au package que le programme veut utiliser.

Remarque :
Les commandes d'import ne sont pas essentielles mais sont bien pratiques car sinon on serait obligé de préfixer les classes par le path d'accès à leur package.

```
//import pkg1.A;
//import pkg1.B;

public class MainPP
{
    static public void main(String[] args)
    {
        pkg1.A a = new pkg1.A();
        pkg1.B b = new pkg1.B();
        //HorsPackage h = new HorsPackage();  // Erreur de compilation
    }
}
```

La syntaxe de la commande d'import :
import <path d'accès>.<class>
ou
import <path d'accès>.*

La 1ère n'importe que la classe désignée.
La deuxième importe toutes les classes du package.

Le path d'accès est ici simple car le package est dans le même répertoire.

Créons un répertoire **monapi** et déplaçons le packages dedans. On obtient l'arborescence suivante, dans un deuxième exemple :

Exemple2
MainPP.java
monapi
pkg1
A.java
B.java

Si on exécute le programme, on obtient l'erreur de compilation :

```
java -classpath "." MainPP
Exception in thread "main" java.lang.NoClassDefFoundError: pkg1/A
 at MainPP.main(MainPP.java:8)
Caused by: java.lang.ClassNotFoundException: pkg1.A
 at java.net.URLClassLoader$1.run(Unknown Source)
 at java.security.AccessController.doPrivileged(Native Method)
 at java.net.URLClassLoader.findClass(Unknown Source)
 at java.lang.ClassLoader.loadClass(Unknown Source)
 at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
 at java.lang.ClassLoader.loadClass(Unknown Source)
 ... 1 more
```

Pour résoudre le problème, il faut faire deux choses :
- changer le nom d'appartenance des .java du package
- changer les commandes d'import

Fichier A.java :

```
package monapi.pkg1;   //nom du package

public class A
{
    public A()
    {
        System.out.println("Création d'un A");
    }
}
```

Fichier B.java :

```
package monapi.pkg1;   //nom du package

public class B
{
    public B()
    {
        System.out.println("Création d'un B");
    }
}
```

Fichier MainPP.java :

```
import monapi.pkg1.*;

public class MainPP
{
    static public void main(String[] args)
    {
        A a = new A();
        B b = new B();
        //HorsPackage h = new HorsPackage();  // Erreur de compilation
    }
}
```

La règle :

Le path d'accès au package est précisé dans la création du package et dépend de la **racine**..

Ici la racine est le répertoire courant Exemple1.

1.3. Le classpath

Déplaçons notre api dans un répertoire en obtenant l'architecture suivante :

Exemple39_Packages
- Exemple3
  - MainPP.java
- home
  - perso
    - **monapi**
      - **pkg1**
        - **A.java**
        - **B.java**
        - **A.class**
        - **B.class**

Notre api de package est dans le répertoire /home/perso/.
Les sources ne sont pas modifiés mais, en étant dans le répertoire Exemple3, on compile avec la commande :
javac –d bin -classpath ..\home\perso MainPP.java

Pour exécuter:
java -classpath "..\home\perso;." MainPP

Attention au point : . d'accès au répertoire courant.
Dans le doute mettez-le tout le monde, y compris dans la commande de compialtion.

La règle :

Le classpath désigne le ou les répertoires qui contiennent les packages (ou api).

S'il existe plusieurs répertoires alors :
javac -classpath "..\home1\perso;..\home2\perso;." MainPP.java

En WINDOWS : **; et **

En Linux : **: et /**

1.4. La variable d'environnement CLASSPATH

Au lieu d'utiliser dans toutes les commandes de compilation, l'option -classpath, on peut une bonne fois pour toute définir le classpath dans une variable d'environnement : $CLASSPATH.

On utilise cela quand l'arborescence est stable. Cela est bien adapté dans un projet informatique.

Dans nos arborescences de TP, on préfère utiliser le -classpath.

1.5. Exemple d'arborescence

Soit un projet informatique partagé en différentes applications informatique et d'une partie commune à toutes les applications. Chaque application informatique est composé de packages.

/home/jl/dev/cvs123/fr/cnam/**commun**/Terminal.java
package fr.cnam.commun;

/home/jl/dev/cvs123/fr/cnam/nfa002/**commun**/Fichiers.java
package fr.cnam.nfa002.commun;

/home/jl/dev/cvs123/fr/cnam/nfa002/**bibliotheque**/Bibliotheque.java
package fr.cnam.nfa002.bibliotheque;

/home/jl/dev/cvs123/fr/cnam/nfa002/bibliotheque/**data**/Livre.java
package fr.cnam.nfa002.bibliotheque.data;

/home/jl/dev/cvs123/fr/cnam/nfa002/bibliotheque/data/Adherant.java
package fr.cnam.nfa002.bibliotheque.data;

/home/jl/dev/cvs123/fr/cnam/nfa002/bibliotheque/**ihm**/AccueilBiblio.java
package fr.cnam.nfa002.bibliotheque.ihm;

/home/jl/dev/cvs123/fr/cnam/nfa002/exemple3/**geom**/FigureGeometrique.java
package fr.cnam.nfa002.exemple3.geom;

/home/jl/dev/cvs123/fr/cnam/nfa002/exemple3/geom./Carre.java
package fr.cnam.nfa002.exemple3.geom;

/home/jl/dev/cvs123/fr/cnam/nfa001/**commun**/Tableaux.java
package fr.cnam.nfa001.commun;

/home/jl/dev/cvs123/fr/cnam/nfa001/exemple1/**algo**/Trier.java
package fr.cnam.nfa001.exemple1.algo;

/home/jl/dev/cvs123/fr/cnam/nfa001/exemple1/algo/Rechercher.java
package fr.cnam.nfa001.exemple1.algo;

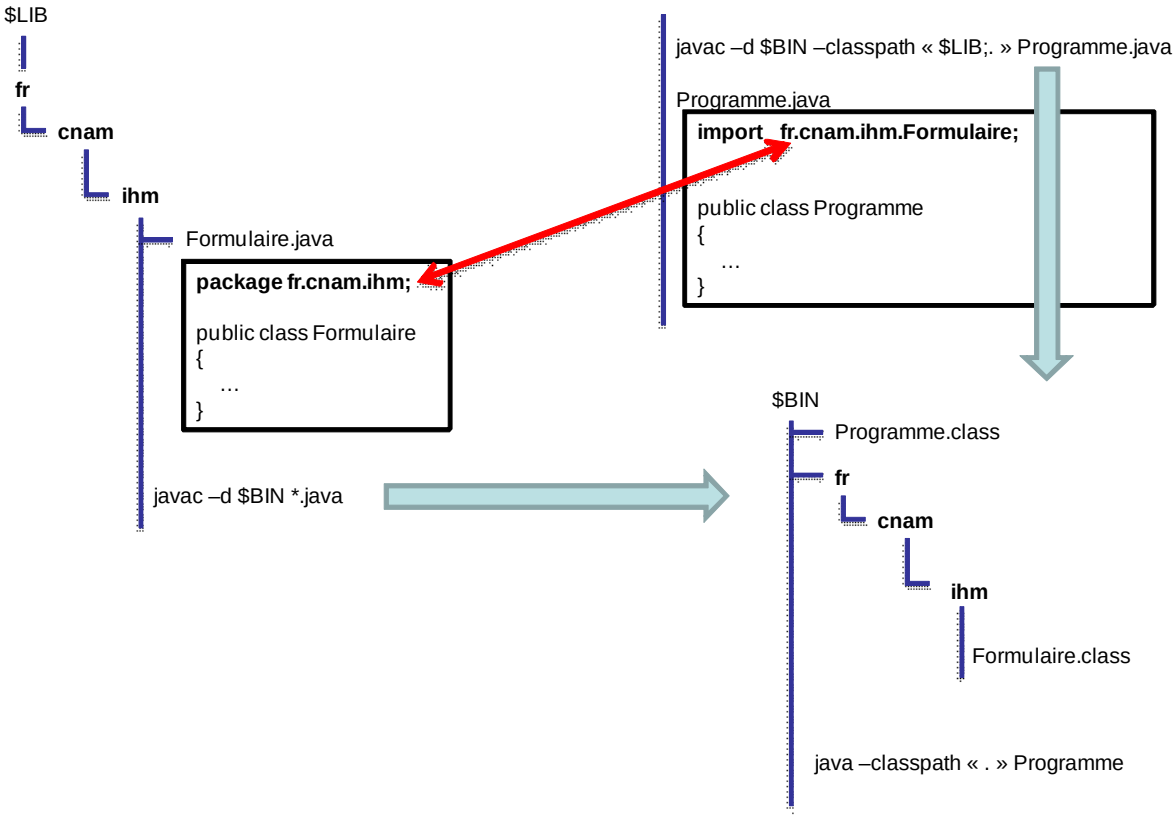
Le classpath est ici : /home/jl/dev/cvs123/.

Les commandes d'import sont par ex **import fr.cnam.nfa002.exemple3.geom.*;**

On appelle **chemin d'accès**, le path nécessaire pour accéder au package à partir du classpath.

1.6. Synthèse

ACCES	classpath
DECLARATION :	chemin d'accès + package
IMPORTATION :	chemin d'accès + package + classe (ou *)



2. La séparation des java et des class

2.1. Le main n'est pas dans un package

Pour les raisons suivantes, on sépare les sources et les fichiers générés dans un programme informatique :

- protection des sources
- livrer et exécuter que les fichiers compilés
- faciliter le nettoyage des fichiers compilés
- transporter les éléments exécutables à travers le réseau
- compression, compactage des api

L'option **-d** de la commande de compilation permet de désigner le répertoire dans lequel les fichiers seront générés par la commande de compilation.

On a donc deux classpath à définir :

- un pour la compilation

- un pour l'exécution

Exemple : Exemple39_Packages

Exemple4/
- src/
  - MainPP.java
  - home1/
    - perso/
      - **monapi/**
        - **pkg1/**
          - A.java
  - home2/
    - perso/
      - **monapi/**
        - **pkg2/**
          - B.java
- bin/

Compilation :

On est dans le répertoire : Exemple5/src
On réalise la commande suivante :

javac -d ..\bin -classpath "./home1/perso;./home2/perso;." MainPP.java

On obtient :

Exemple5/
- bin/
  - MainPP.class
  - **monapi/**
    - **pkg1/**
      - A.class
    - **pkg2/**
      - B.class

Les chemins d'accès de tous les packages se retrouvent dans le même répertoire.

Exécution :

L'exécution doit se faire alors dans le répertoire bin :

java –classpath "." MainPP

Si vous utilisez une version au moins 1.6, on n'a pas besoin de préciser de classpath puisque ils sont dans le répertoire courant.

2.2. Cas du main dans un package :

Soit l'arborescence suivante où le MainPP. java est dans le package **apitest**.**testPP**.

Exemple5/
- src/
  - test
    - **apitest/**
      - **testPP/**
        - MainPP.java
        - (avec en entête du fichier : ***package apitest.testPP;***)
  - home1/
    - perso/
      - **monapi/**
        - **pkg1/**

A.java
home2/
perso/
monapi/
pkg2/
B.java

Compilation :

On est dans le répertoire : Exemple5/src
On réalise la commande suivante :
javac -d ..\bin -classpath "home1\perso;home2\perso" test\apitest\testPP\MainPP.java

On obtient :

Exemple5/
bin
apitest/
testPP/
MainPP.class
monapi/
pkg1/
A.class
pkg2/
B.class

Exécution :

On est dans le répertoire bin
java –classpath "." apitest.testPP.MainPP

L'accès se fait par le chemin d'accès du package.

3. Exemple : Exemple19 et 99

Dans cette exemple, nous voulons gérer la médiathèque sous la forme d'un package qui utilise d'autres packages se trouvant ailleurs.

Nous avons l'arborescence suivante :

```
Site
   Exemple19_MediathequeRecherche
      exemple19
         Exemple19.java
         FormAjoutMediatheque.java        → dedans: import cnam.ihm.*;
         FormRechercheMediatheque.java
         Mediatheque.java
         Media.java
         Livre.java
         LivreEtendu.java
         DVD.java
   Exemple99_UtilitairesPkg
      cnam
         ihm
            Formulaire.java
            Formulaireint.java
            FormulaireException.java
bin
   Exemple19_compil.bat
   Exemple19_run.bat
```

Tous les fichiers du répertoire de exemple19 ont en en tête

```
package exemple19;
```

Tous les fichiers du répertoire de ihm ont en en tête

```
package cnam.ihm;
```

Cela signifie que nous avons 2 packages :
exemple19
cnam.ihm

De plus nous voulons mettre les .class dans le répertoire **bin**

On a donc le fichier de compilation suivant :

Fichier Exemple19_compil.bat :
On est dans le répertoire bin.

```
del exemple19\*.class
javac -d "." -classpath "." ..\Site\Exemple19_MediathequeRecherche\
                                                    exemple19\*.java
pause
```

Dans le répertoire **bin**, on obtient alors l'arborescence suivante :

```
bin
   exemple19
      Exemple19.class
      FormAjoutMediatheque.class
      FormRechercheMediatheque.class
      Mediatheque.class
      Media.class
      Livre.class
      LivreEtendu.class
      DVD.class

   cnam
      ihm
         Formulaire$FormulaireWindowListener.class
         Formulaire$SubmitListener.class
         Formulaire.class
         FormulaireException.class
         FormulaireInt.class
         SaisieString.class
         TesterFormulaire.class
```

Le programme principal (main) est dans la classe Exemple19.class.
Pour exécuter
Fichier Exemple19_run.bat

```
java -classpath "." exemple19.Exemple19
```

---

4. Les environnements de compilation

ant
maven
eclipse

netbean