
Packages

Notion de package

- un package est un groupe de classes associées à une fonctionnalité et/ou qui coopèrent
- exemples de packages
 - *`java.lang`* : rassemble les classes de base JAVA, *Object*, *String*, *System*...
 - *`java.util`* : classes pour les collection
 - *`java.awt`* : classes pour interfaces utilisateurs (Abstract Window Toolkit)
 - *`java.awt.images`* : classes pour manipulation d'images bitmap
 -

Package : espace de nommage

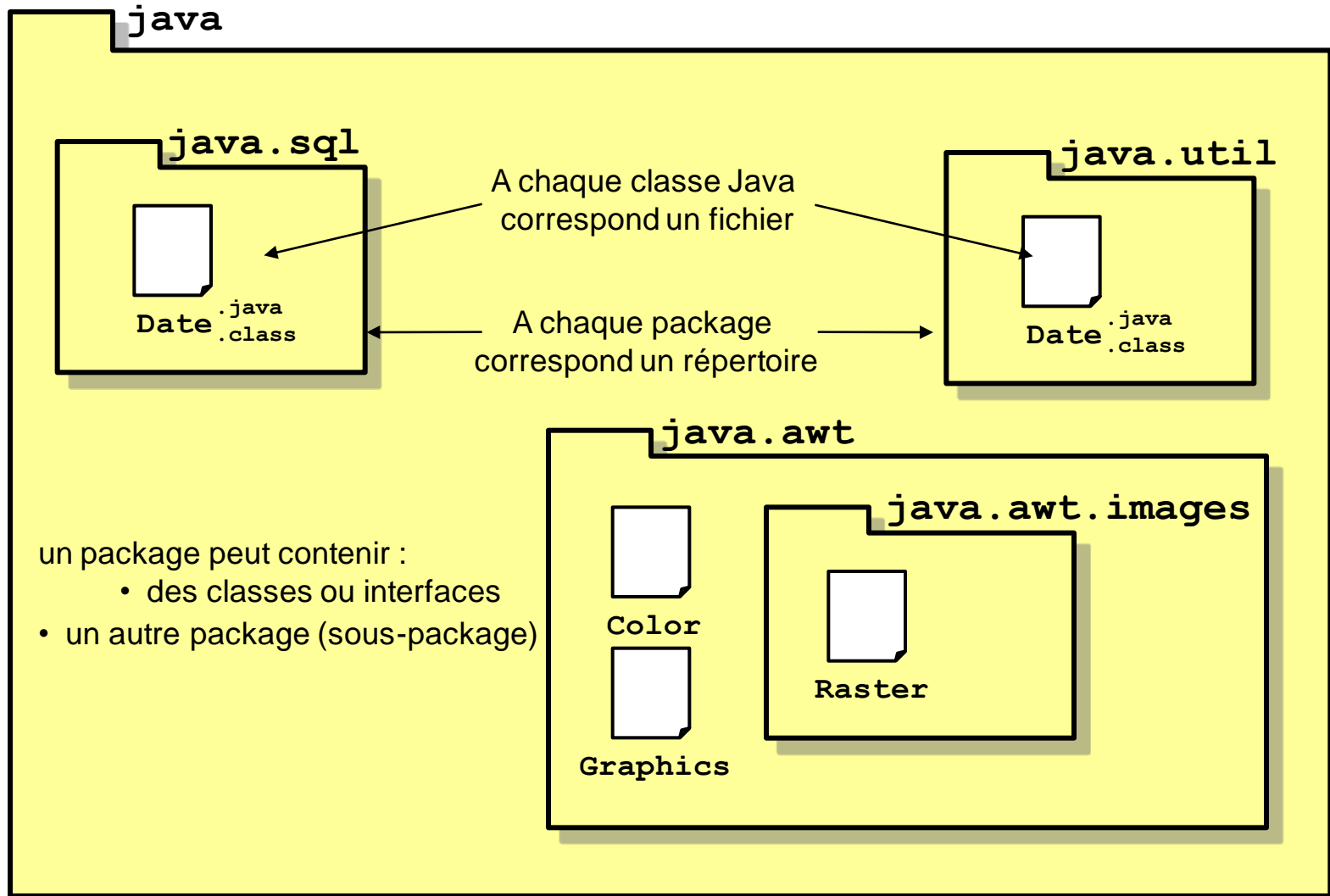
- Le regroupement des classes dans des packages permet d'organiser les bibliothèques de classes Java et d'éviter d'éventuels conflits de noms.
- Exemple :
Dans les bibliothèques standards du JDK deux classes `Date`...

The screenshot shows the Java Platform SE 7 API documentation. In the left sidebar, the 'Date' class is highlighted under the 'java.util' package. In the main content area, the 'Date' class is shown with its package 'java.util' and its superclass 'java.lang.Object'. Below this, the 'All Implemented Interfaces' section lists 'Serializable', 'Cloneable', and 'Comparable<Date>'. Another 'Date' class is shown in the 'java.sql' package, also implementing 'Serializable', 'Cloneable', and 'Comparable<Date>'. Red circles and arrows highlight the package names and the class names to illustrate the concept of fully qualified names.

Mais définies dans des packages différents

nom complet de la classe (Fully Qualified Name) :
nomDupackage.nomDeLaClasse

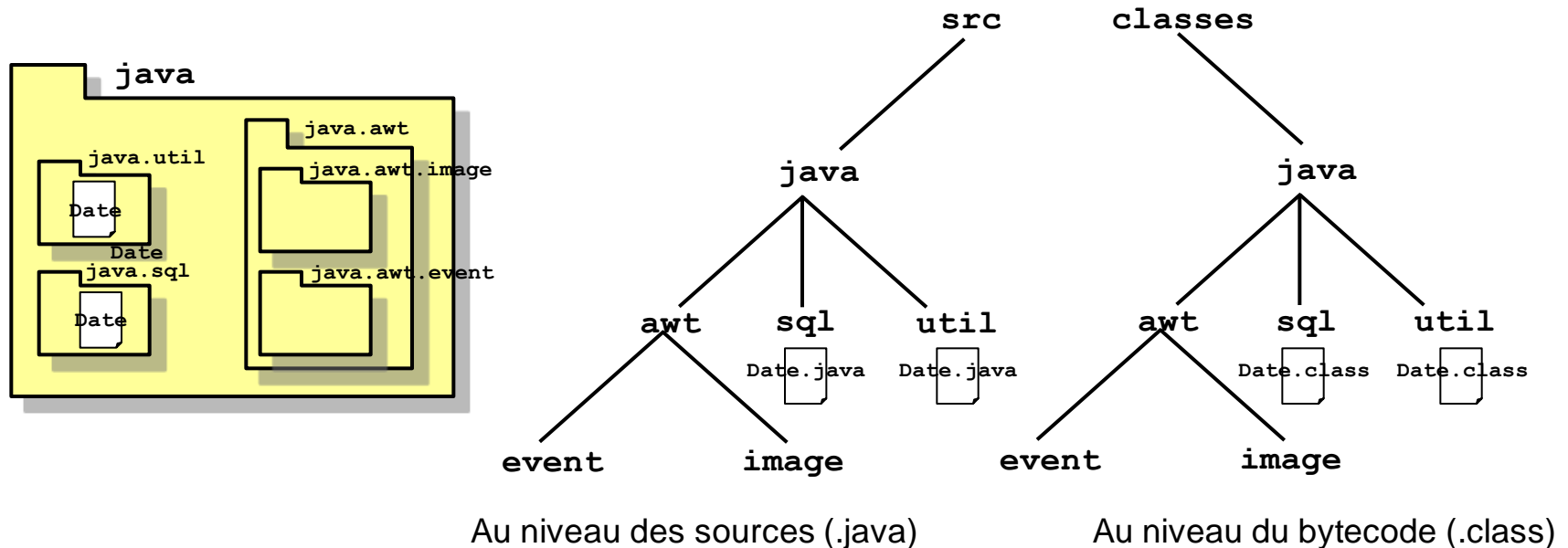
Packages et système de fichiers



La hiérarchie des packages correspond à une hiérarchie de répertoires

Packages et système de fichiers

- A une hiérarchie de packages correspond une hiérarchie de répertoires dont les noms coïncident avec les composants des noms de package



Dans les fichiers jar



rt.jar

```
jar tvf rt.jar
 0 Sat Feb 07 16:12:14 CET 2004 META-INF/
68 Sat Feb 07 16:12:14 CET 2004 META-INF/MANIFEST.MF
 0 Sat Feb 07 16:06:28 CET 2004 java/
 0 Sat Feb 07 16:11:48 CET 2004 java/awt/
62 Sat Feb 07 16:11:40 CET 2004 java/awt/Color.class
85 Sat Feb 07 16:11:58 CET 2004 java/awt/Graphics.class
 0 Sat Feb 07 16:06:40 CET 2004 java/awt/image/
...
```

Instruction package

- Le package d'appartenance d'une classe est défini par l'instruction

package *nomDuPackage*;

Optionnelle

Si elle est présente elle doit être la première instruction du fichier, elle définit le package auquel est rattaché le code contenu dans le fichier,

```
package courspoo;
```

```
public class UneClasse {
```

```
} package courspoo.banque;
```

```
public class Compte {
```

```
package courspoo.banque;
```

```
public class Client {
```

```
package courspoo.geom;
```

```
public class Point {
```

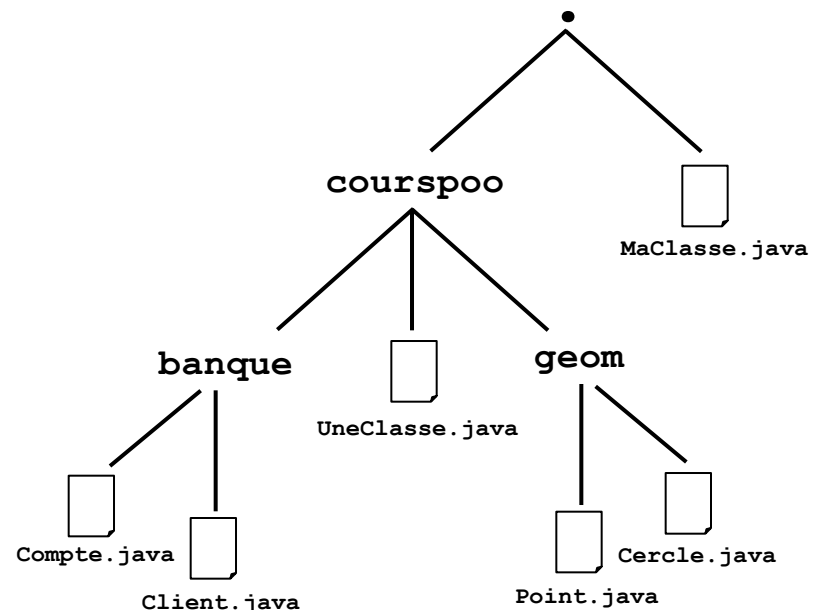
```
package courspoo.geom;
```

```
public class Cercle {
```

```
public class MaClasse { entre;
```

```
...
```

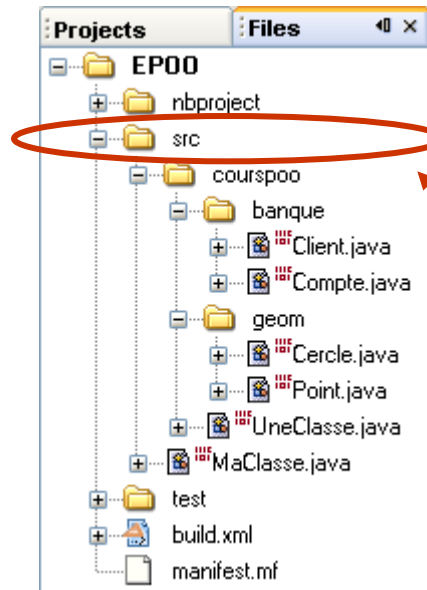
```
}
```



Si pas d'instruction package : le code défini dans le fichier est associé à un package par défaut, sans nom et correspondant au répertoire courant.

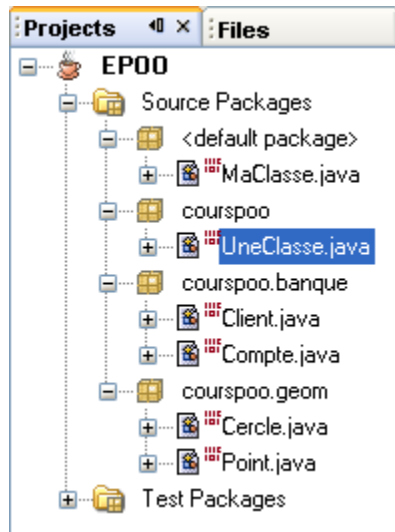
Packages dans NetBeans

La fenêtre **Files** donne une vision physique des fichiers de l'application

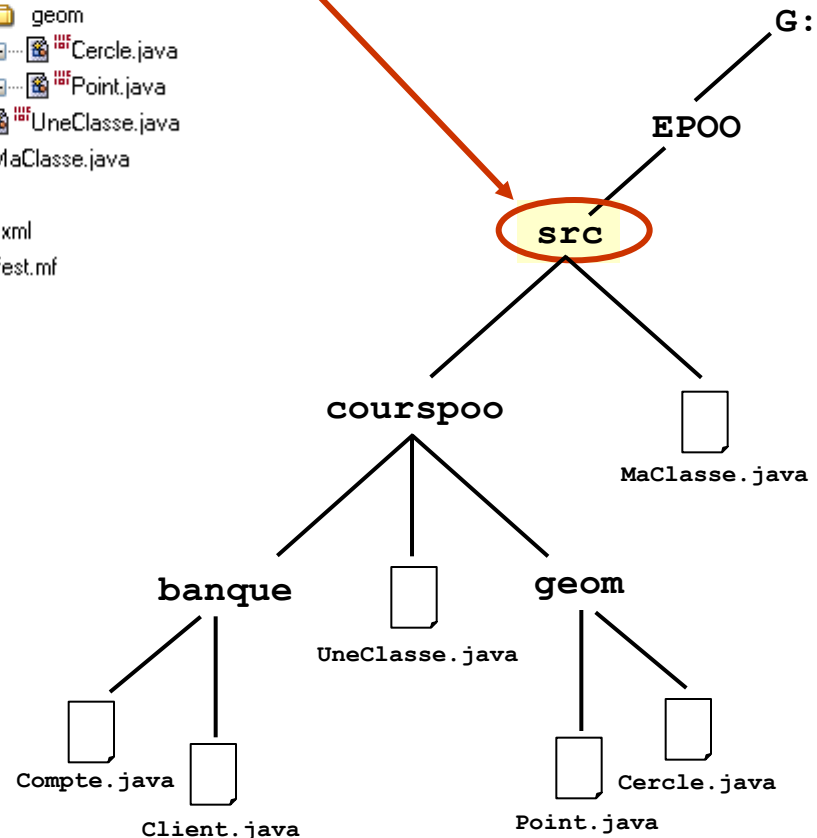


Le répertoire **src** est la racine des hiérarchies de packages d'un projet

Options `-classpath` et `-sourcepath` automatiquement placées sur ce répertoire



La fenêtre **Projects** donne une vision logique des packages de l'application



accès aux éléments d'un package

- Pour pouvoir utiliser une classe issue d'un autre package il faut signifier son origine au compilateur

```
package courspoo.banque;

import java.util.*;
import java.sql.Connection;
public class Compte {

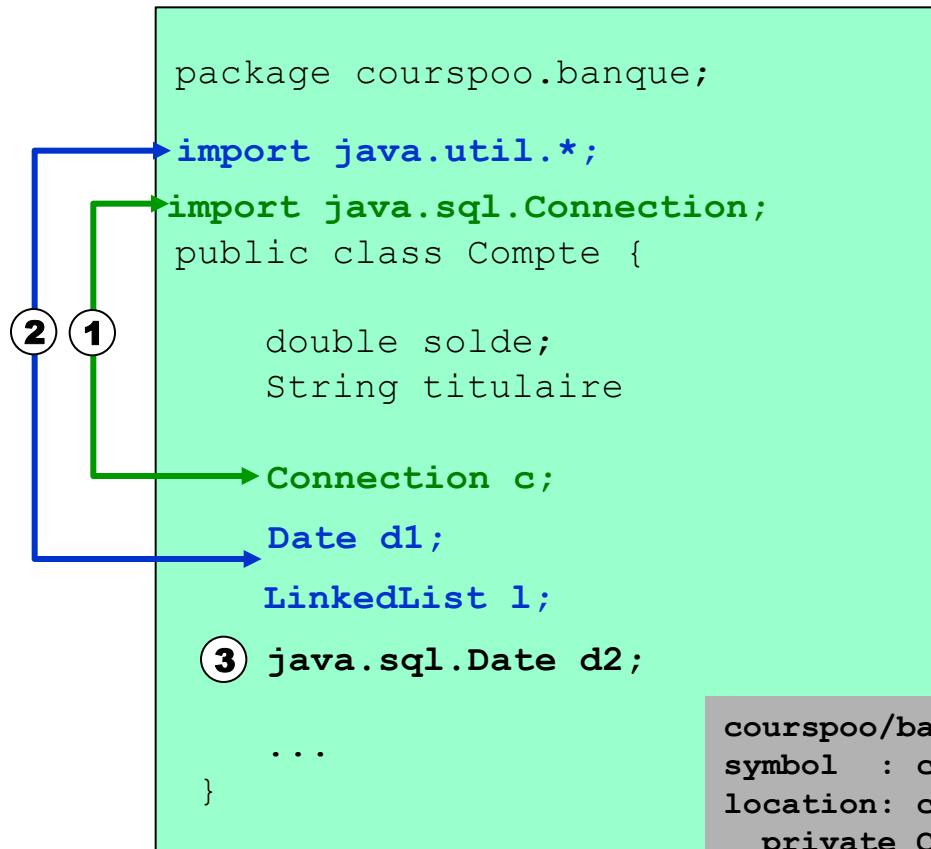
    double solde;
    String titulaire

    Connection c;

    Date d1;
    LinkedList l;

    ③ java.sql.Date d2;

    ...
}
```

A diagram with three numbered circles (1, 2, 3) and arrows pointing to specific lines in the code. Circle 1 points to 'import java.util.*;', circle 2 points to 'import java.sql.Connection;', and circle 3 points to 'java.sql.Date d2;'. The code is enclosed in a light green box.

- ① en important la classe
- ② en important tout le package où est définie la classe
- ③ en indiquant le nom complet de la classe (Fully Qualified Name).

import n'insère pas du code dans le code de la classe (comme le fait #include du C). Il s'agit simplement d'une indication pour le compilateur. En interne seuls les FQN sont utilisés.

```
courspoo/banque/Compte.java [32:1] cannot resolve symbol
symbol  : class Connection
location: class courspoo.compte.Compte
    private Connection s;
        ^
1 error
Errors compiling Compte.
```


accès aux éléments d'un package

- L'instruction `import nomPackage.*` ne concerne que les classes du package indiqué. Elle ne s'applique pas aux classes des sous packages.

```
package courspoo.geom;

import java.awt.image.*;
import java.awt.*;
public class Star {

    double x,
    double y;

    Color c;

    public void draw(Graphics g) {
        ...
        Raster r;

    }

    ...
}
```

Classe **Star** utilise les classes **Color** et **Graphics** du package **java.awt** et la classe **Raster** du package **java.awt.image**

Pas besoin d'import
pour les classes du
package
java.lang

Imports statiques

- Jusqu'à la version 1.4 de Java, pour utiliser un membre statique d'une classe, il faut obligatoirement préfixer ce membre par le nom de la classe qui le contient.

```
public class version1_4 {  
    public static void main(String[] args) {  
        System.out.println(Math.PI);  
        System.out.println(Math.cos(0));  
    }  
}
```

- Java 1.5 propose une solution pour réduire le code à écrire concernant les membres statiques en proposant une nouvelle fonctionnalité concernant l'importation de package : **l'import statique** (static import).
 - *permet d'appliquer les mêmes règles aux membres statiques qu'aux classes et interfaces pour l'importation classique*

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

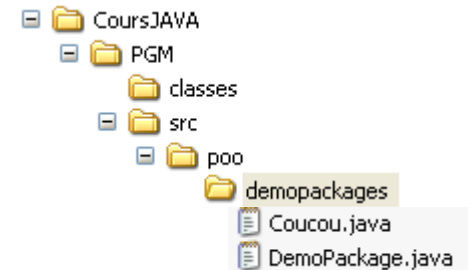
```
public class version1_5{  
    public static void main(String[] args) {  
        System.out.println(PI);  
        System.out.println(cos(0));  
    }  
}
```

importation statique
s'applique à tous les
membres statiques :
constantes et méthodes
statiques de l'élément
importé

Compiler et exécuter

- Package **poo.demopackages**

```
package poo.demopackages;  
  
public class DemoPackage {  
  
    public static void main(String[] args) {  
        System.out.println("Demo packages");  
        Coucou.hello();  
    }  
}
```



- Compiler

- Se placer dans le répertoire racine des packages (**src**)
- Spécifier tout le chemin pour désigner la classe

```
...\PGM\src>javac -d ../classes poo/demopackages/DemoPackage.java
```

- La hiérarchie des packages est reconstruite dans le répertoire de destination

- Exécuter

- Se placer dans le répertoire racine des packages (**classes**)
- Spécifier la classe en utilisant un nom complet (Fully qualified name)

```
...\PGM\classes>java poo.demopackages.DemoPackage
```

