POO-IG

Programmation Orientée Objet et Interfaces Graphiques

Cristina Sirangelo
IRIF, Université Paris Diderot
cristina@irif.fr

Exemples et materiel empruntés :

* Core Java - C.Horstmann - Prentice Hall Ed.
* POO in Java - L.Nigro & C.Nigro - Pitagora Ed.

Packages

Packages

- Un package est un regroupement de classes
- Des milliers disponibles dans la bibliothèque de java :
  - java.lang, java.util, java.net, …

- On peut créer ses propres packages
  - structuration du code en bibliothèques de classes réutilisables
  - fondamental pour la gestion de gros projets

- On regroupe dans un package des classes aux fonctionnalités proches et/ou liées

Définition d'un package

- Pour déclarer l'appartenance d'une classe ou plusieurs classes à un package :

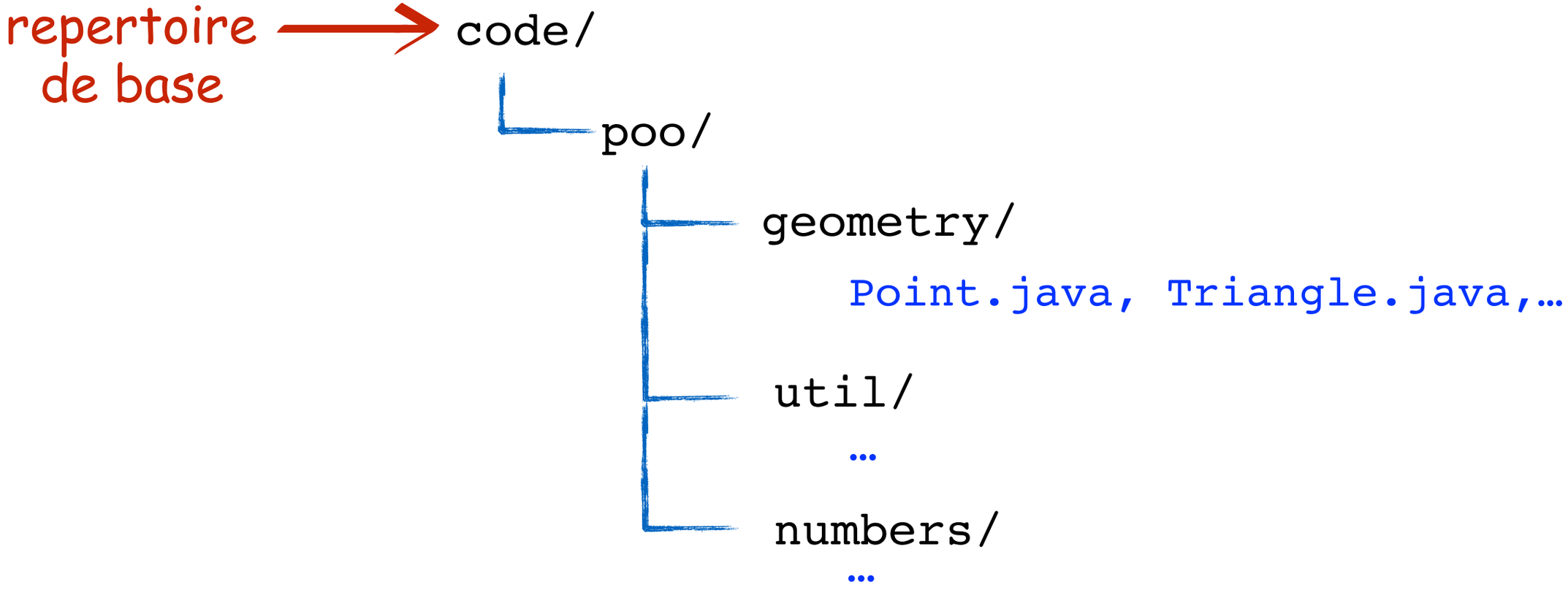
```
//fichier Point.java
package poo.geometry; //premiere ligne non-commentaire
                      //du fichier
public class Point() {…}
```

```
//fichier Triangle.java
package poo.geometry;
public class Triangle() {…}
```

- `poo.geometry` est le nom du package
  - contient les classes `Point` et `Triangle`
  - Des éventuelles classes `Polygone`, `Cercle` etc, pourraient aller dans ce même package

Définition d'un package

- Le nom du package doit correspondre à la structure des fichiers.
  - Les classes dans le package poo.geometry doivent toutes se trouver dans un répertoire poo/geometry



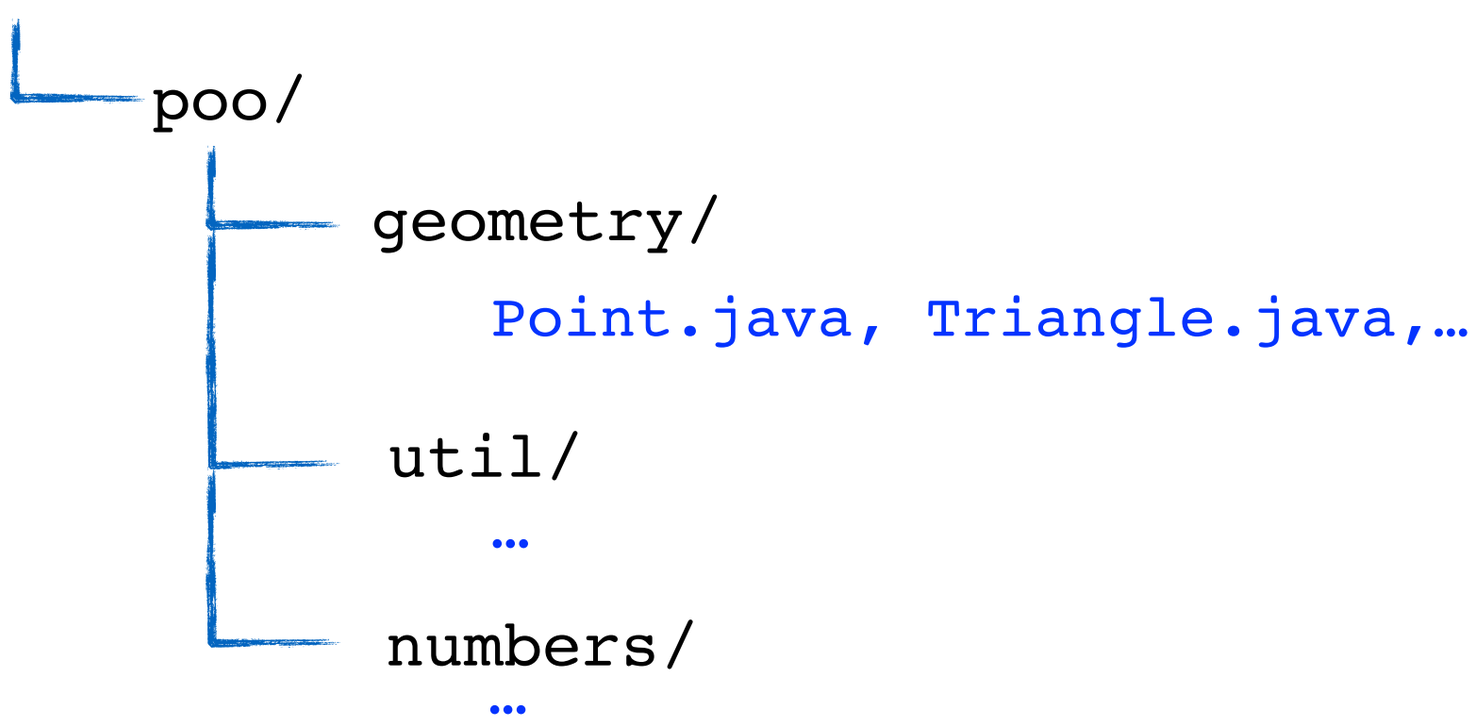
- Plusieurs packages peuvent partager le meme préfixe
  - e.g. poo.util, poo.numbers

Packages : compilation et execution

- Pour compiler et executer les classes d'un package on se place dans le repertoire de base :

```
code > javac poo/geometry/Point.java
code > java poo.geometry.Point
```

repertoire de base → code/

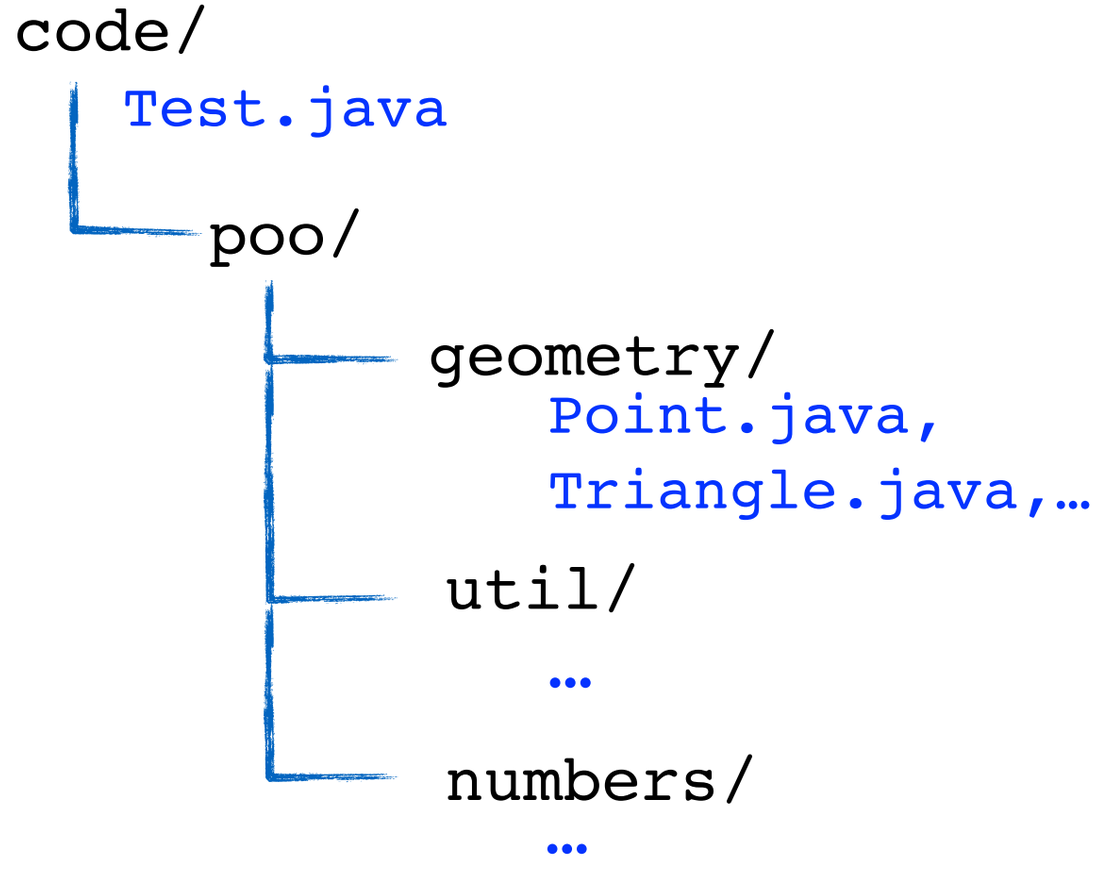


Importation de packages

```
//Test.java
import poo.geometry.*
class Test {
    Point p; …
}
```

```
//Test.java
import poo.geometry.Point
class Test {
    Point p; …
}
```

```
//Test.java
class Test {
    poo.geometry.Point p; …
}
```



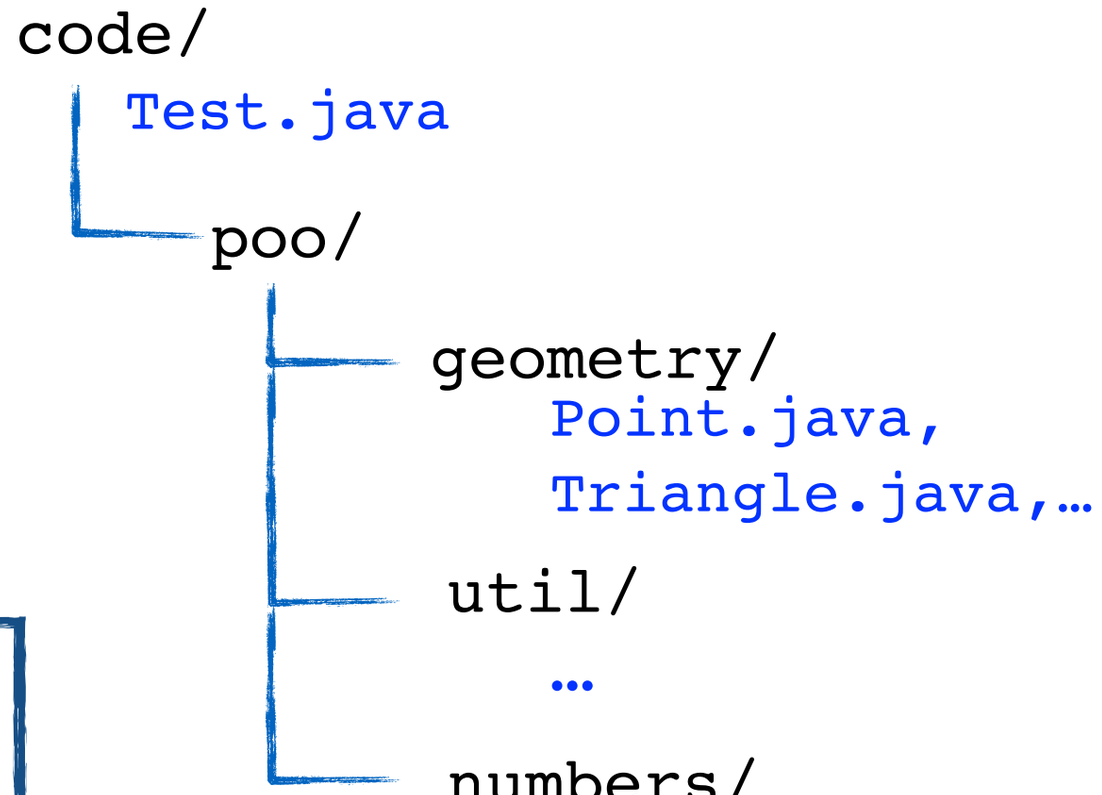
- Remarque : `javac Test.java` recompile automatiquement les classes importées si nécessaire

Importation de packages : remarque

- pas besoin d'importation pour utiliser les classes (visibles) du même package

```
//fichier Point.java
package poo.geometry;
public class Point() {…}
```

```
//fichier Triangle.java
package poo.geometry;
public class Triangle() {
     private Point p1,p2,p3;
}
```



Importation de packages

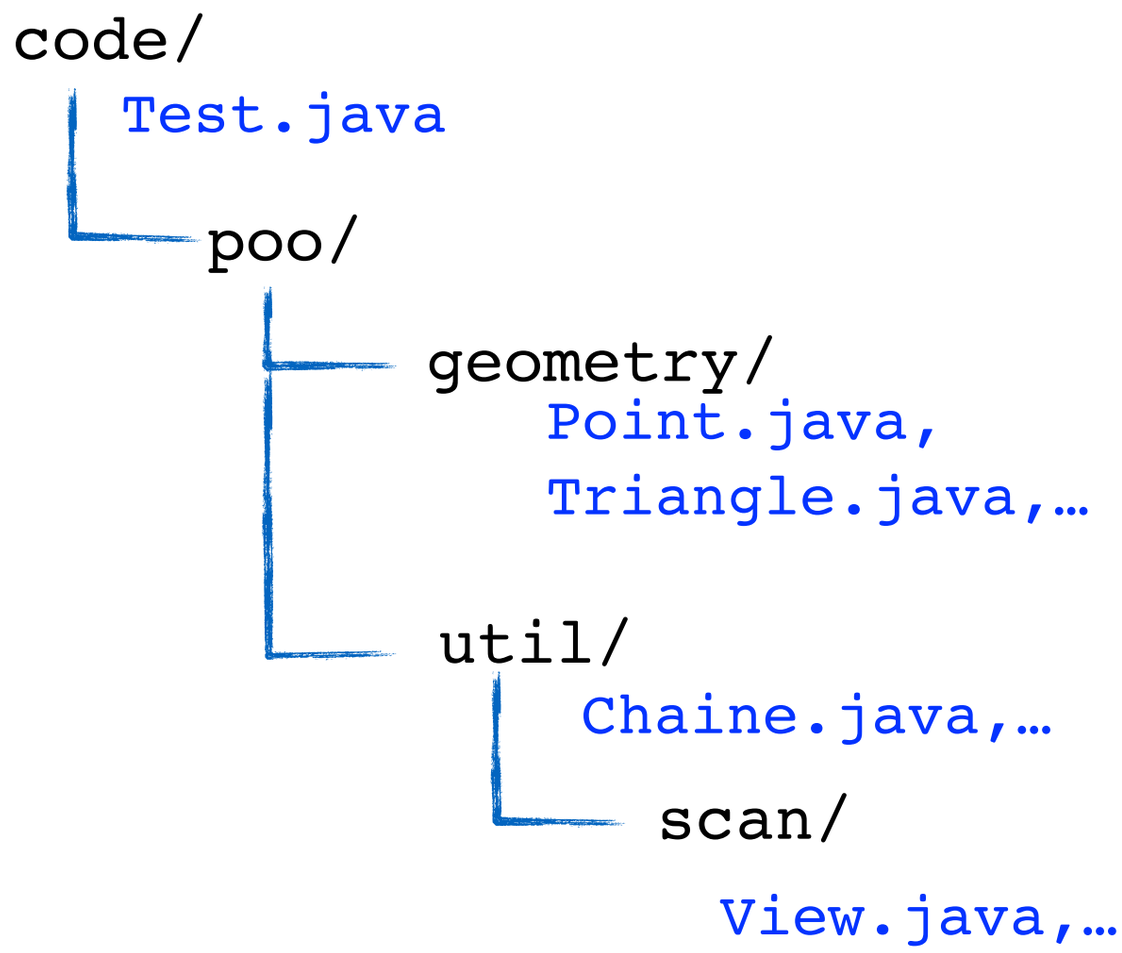
- Pour pouvoir importer un package (qui n'appartient pas à la bibliothèque standard) son répertoire de base doit être listé dans le classpath

- Classpath : liste de repertoires de base séparée par ':'

- Le classpath contient par default '.'
  - => On peut importer tous les packages dont le répertoire de base est le repertoire de travail

Importation de packages et classpath

```
//Test.java
import poo.geometry.*
import poo.util.*
import poo.util.scan.*
class Test {
    Point p;
    Chaine c;
    View v; …
}
```



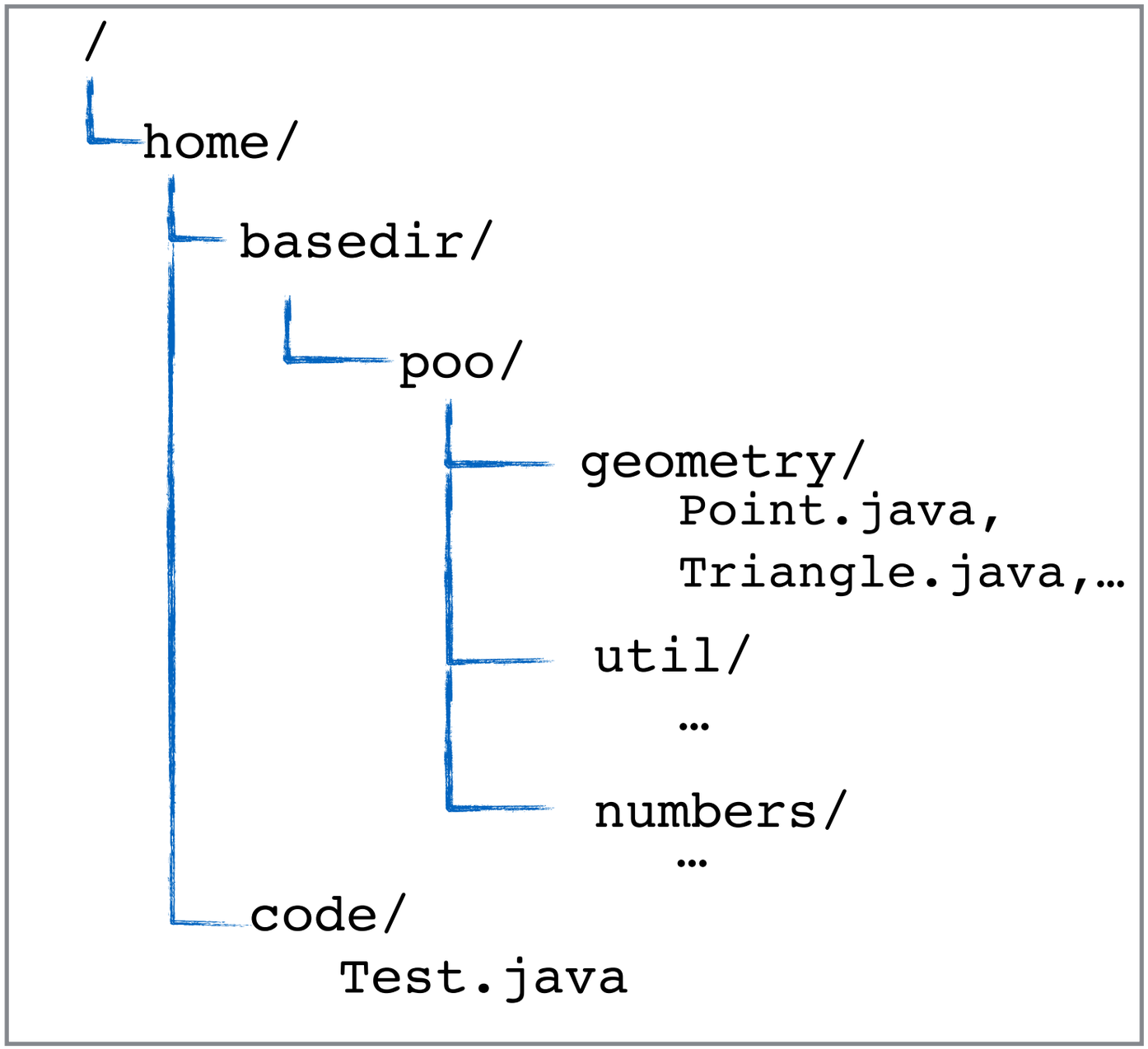
- Attention : import poo.util.* importe les classes du package poo.util, et non pas les sous-packages
  - poo.util.scan doit être importé explicitement

Importation de packages et classpath

- Pour importer des packages externes, il faut modifier le classpath (qui contient par default '.')

```
>export CLASSPATH=/home/basesdir:.:/home/archives/archive.jar
>javac Test.java
>java Test
```

```
//Test.java
import poo.geometry.*
class Test {
    Point p; …
}
```



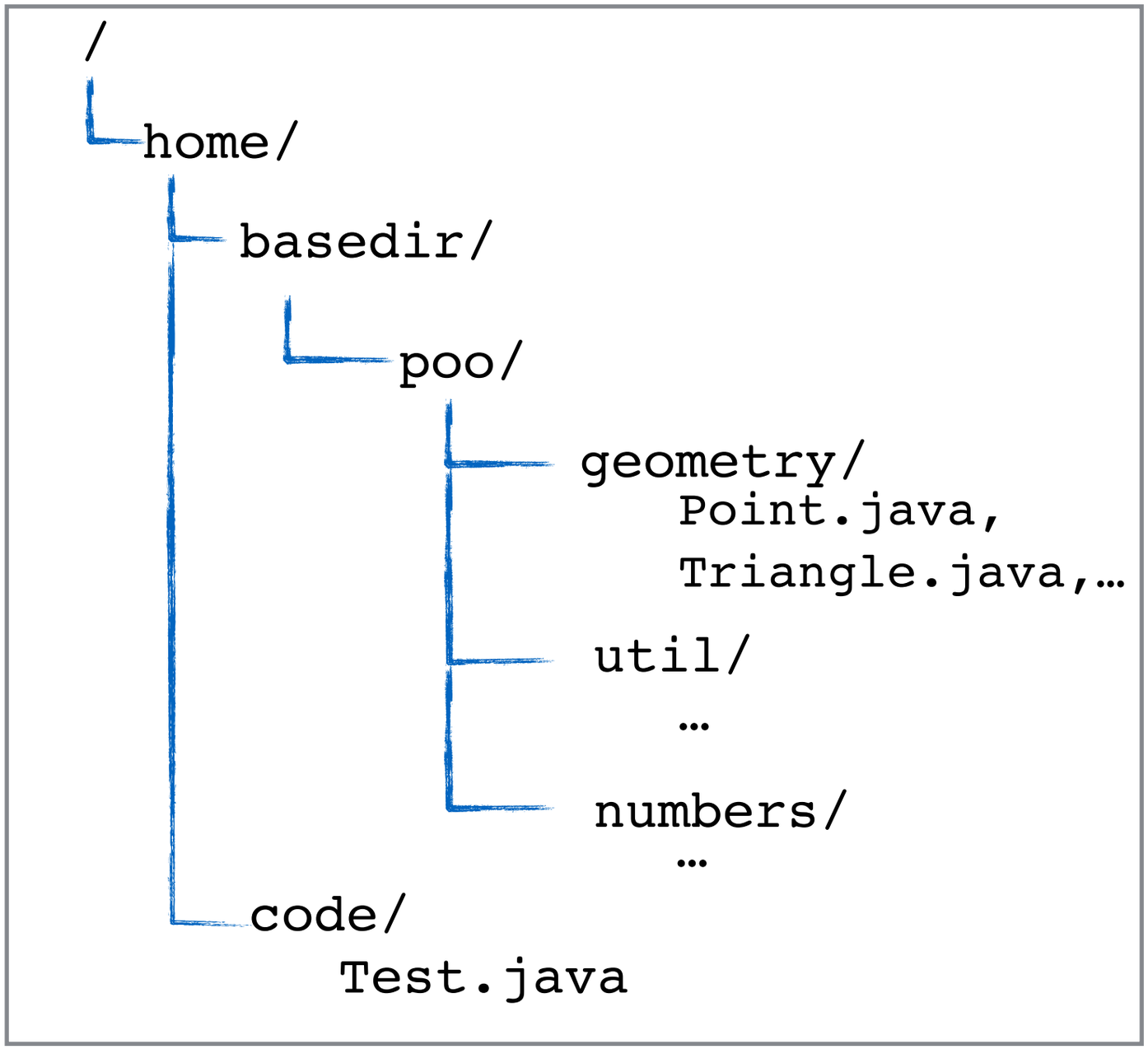
Importation de packages et classpath

- Ou bien pour ajouter au classpath, sans le redéfinir

Si dans ~/.bash_profile permanent

```
>export CLASSPATH=$CLASSPATH:/home/basesdir
>javac Test.java
>java Test
```

```
//Test.java
import poo.geometry.*
class Test {
    Point p; …
}
```



Importation de packages et classpath

- Alternative : modifier le classpath au niveau de chaque commande :

```
> javac -classpath /home/basedir:…  Test.java
> java -classpath /home/basedir:…   Test
```

- Ou pour ajouter au classpath courant :

```
> javac -classpath $classpath:/home/basedir  Test.java
> java -classpath $classpath:/home/basedir   Test
```

Static import

- Depuis Java 5.0 une variante de `import` permet d'importer directement toutes les méthodes et champs static d'une classe

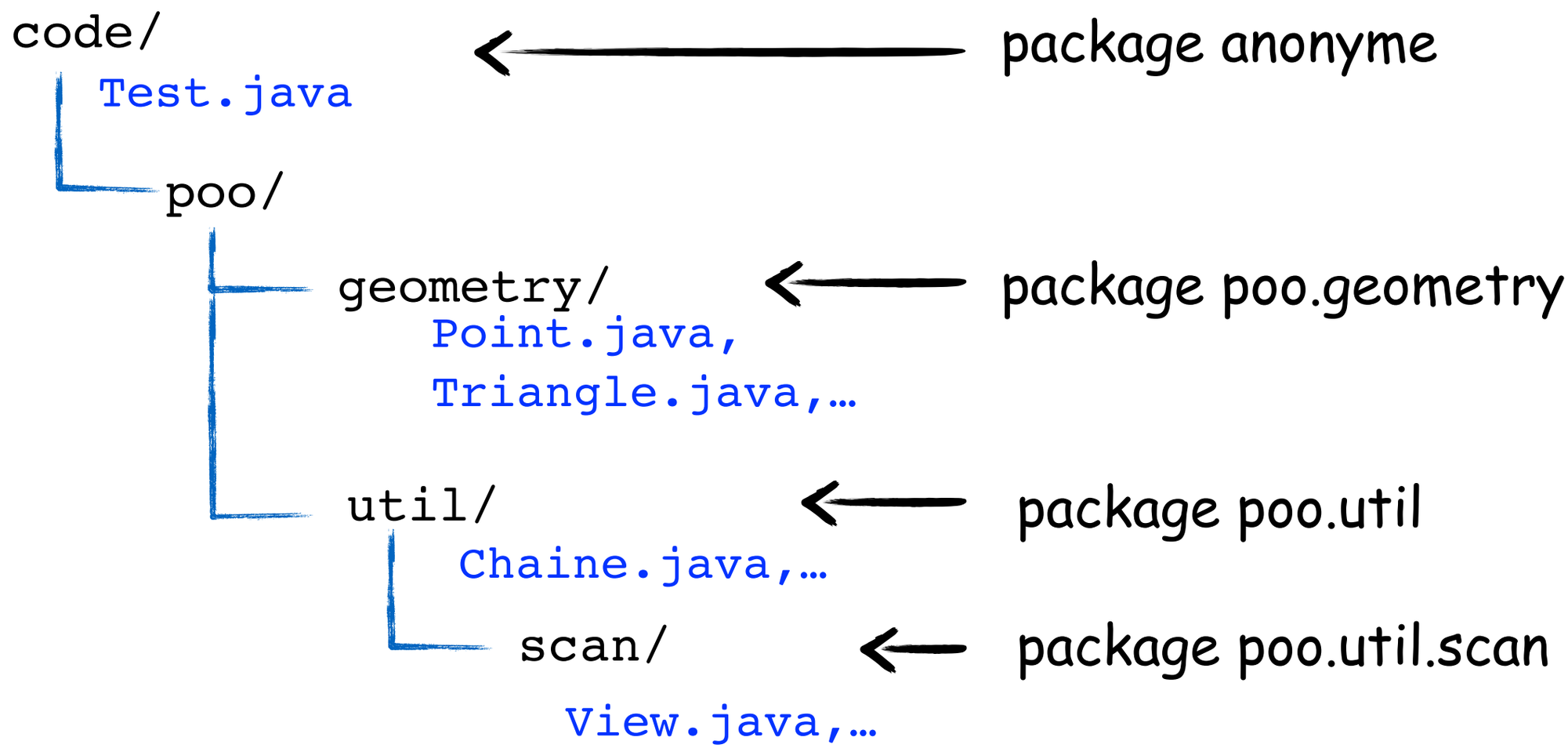
```
//import static nomPackage.nomClasse.*
```

```
// import java.lang.* implicite
class Test {
 public static void main (String[] args) {
   … Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2)); …
 }
}
```

```
import static java.lang.Math.*;
class Test {
 public static void main (String[] args) {
   … sqrt(pow(x, 2) + pow(y, 2)); …
 }
}
```

Package par default

- Toute classe est dans un package
- Un package par default anonyme est créé dans le répertoire de travail si aucun package n'est spécifié



Visibilité package

- Pour les classes et membres d'une classe la visibilité par default (pas de modificateur d'accès) est package
  - toutes les classes du package y ont accès

```
//poo/util/B.java
package poo.util;
class A {
   …
}
public class B {
   private int i;
   int j;
   …
}
```

```
//poo/util/C.java
package poo.util;
public class C {
   A a = new A();
   B b = new B();
   public C () {
      B.i = 0; //ERREUR
      B.j = 0; //OK
      …
   }
}
```

Visibilité package

- Pour les classes et membres d'une classe la visibilité par default (pas de modificateur d'accès) est package
  - toutes les classes du package y ont accès

```
//poo/util/B.java
package poo.util;
class A {
   …
}
public classe B {
   private int i;
   int j;
   …
}
```

```
//poo/numbers/D.java
package poo.numbers;
import poo.util.*;
public classe D {
   A a = new A(); //ERREUR
   B b = new B();
   public D () {
      B.i = 0; //ERREUR
      B.j = 0; //ERREUR
       …
   }
}
```

Java Archive Tool (JAR)

- Les package peuvent être archivés dans le format d'archivage de java (JAR):
- Pour cela on utilise le "Java Archive Tool" fourni par JDK

- Il est invoqué par la commande :

```
jar cf MyJar.jar MyPackage/*.class
```

  - On mets uniquement les .class dans le jar, la seule chose qui sert au compilateur
  - Cela crée un .jar dans le répertoire ou on a lancé la commande

- Pour importer les classes dans un Jar meme mécanisme que pour les package non archivés mais il faut donner le chemin exact du jar

```
export CLASSPATH=$CLASSPATH:chemin/MyJar.jar
```