Class path, packages, archives

Olivier Cailloux

LAMSADE, Université Paris-Dauphine

Version du 19 janvier 2018

LAMSADE
UMR CNRS 7243
laboratoire d'analyse et modélisation de systèmes pour l'aide à la décision

DAUPHINE
UNIVERSITÉ PARIS

PSL
RESEARCH
UNIVERSITY

Packages : principe

- Chaque classe principale Java : dans son fichier
- Classe `MyClass` dans fichier ?
- Compilée dans fichier ?
- Chaque fichier déclaré dans un *package* Sauf si package par défaut, non recommandé.
- Packages structurés hiérarchiquement, comme un arbre
- Structure indiquée par des points
- Exemples dans l'API Java : `java.nio.file`, `java.util.logging`
- Quand utilisation d'une classe, il faut l'importer en indiquant son package

Packages : principe

- Chaque classe principale Java : dans son fichier
- Classe `MyClass` dans fichier ? `MyClass.java`
- Compilée dans fichier ?
- Chaque fichier déclaré dans un *package* Sauf si package par défaut, non recommandé.
- Packages structurés hiérarchiquement, comme un arbre
- Structure indiquée par des points
- Exemples dans l'API Java : `java.nio.file`, `java.util.logging`
- Quand utilisation d'une classe, il faut l'importer en indiquant son package

Packages : principe

- Chaque classe principale Java : dans son fichier
- Classe `MyClass` dans fichier ? `MyClass.java`
- Compilée dans fichier ? `MyClass.class`
- Chaque fichier déclaré dans un *package* Sauf si package par défaut, non recommandé.
- Packages structurés hiérarchiquement, comme un arbre
- Structure indiquée par des points
- Exemples dans l'API Java : `java.nio.file`, `java.util.logging`
- Quand utilisation d'une classe, il faut l'importer en indiquant son package

Utilité des packages

- Plusieurs classes de même nom peuvent exister (`MathUtils`)
- Nom complet d'une classe doit être unique
- Nom complet = nom du package point nom de la classe
- Organiser par thème

Packages et répertoires

- Fichier dans un package `nom1.nom2.nom3`
- Se trouve dans sous-répertoire `nom1/nom2/nom3`
- Utilité de cette séparation en répertoires ?

- Chemin considéré à partir du répertoire source
- Dans Eclipse par défaut : `MyProject/src/nom1/nom2/nom3`

Packages et répertoires

- Fichier dans un package `nom1.nom2.nom3`
- Se trouve dans sous-répertoire `nom1/nom2/nom3`
- Utilité de cette séparation en répertoires ? Organisation ; Unicité de nom
- Chemin considéré à partir du répertoire source
- Dans Eclipse par défaut : `MyProject/src/nom1/nom2/nom3`

Compilation

- Classes compilées : où ?
- Structure des sources conservée !
- Package ?

Compilation

- Classes compilées : où ?
- Souvent : dans répertoire `bin` ou `target`
- Structure des sources conservée !
- Exemple : classe dans `src/myutils/draft/MathUtils.java` compilée dans `bin/myutils/draft/MathUtils.class`
- Package ?

Compilation

- Classes compilées : où ?
- Souvent : dans répertoire `bin` ou `target`
- Structure des sources conservée !
- Exemple : classe dans `src/myutils/draft/MathUtils.java` compilée dans `bin/myutils/draft/MathUtils.class`
- Package ? `myutils.draft`

Archives

- Pour distribuer vos classes
- Ou partager entre vos propres projets
- JAR ?
- Fichier compressé
- Contenu ?

Archives

- Pour distribuer vos classes
- Ou partager entre vos propres projets
- JAR ? Java ARchive
- Fichier compressé
- Contenu ?

Archives

- Pour distribuer vos classes
- Ou partager entre vos propres projets
- JAR ? Java ARchive
- Fichier compressé
- Contenu ? Collection de classes compilées (`.class`) et leur package
- Facultatif : code source
- Facultatif : `Main-Class` dans META-INF/MANIFEST.MF
- Et autres méta-données facultatives

Unicité de nommage

- Objectif du système de packages et de classes : assurer l'unicité de nom
- Pour éviter conflits lors utilisation d'un jar tiers (par exemple)
- Problème ?
- Convention : base = votre domaine internet inversé
- Exemple : `com.google.common.math.Quantiles`, `org.apache.crunch.lib.Quantiles`

Unicité de nommage

- Objectif du système de packages et de classes : assurer l'unicité de nom
- Pour éviter conflits lors utilisation d'un jar tiers (par exemple)
- Problème ? Il faut packages uniques !
- Convention : base = votre domaine internet inversé
- Exemple : `com.google.common.math.Quantiles`, `org.apache.crunch.lib.Quantiles`

Class path

- Comment lancer une application ?
- Il faut indiquer où aller chercher toutes les classes requises
- Dans de multiples répertoires ou fichiers JAR
- Class path : les chemins que la JVM utilise pendant son exécution pour charger les classes (sauf celles de l'API Java)
- Class path fourni à la JVM au démarrage, exemple `java -cp path1:path2:path3 name1.name2.MyClass`
- NB il doit contenir des classes compilées
- Classes dans l'API Java : rien à préciser
- Class path par défaut ?
- Dans Eclipse : voir Java Build Path

Class path

- Comment lancer une application ?
- Il faut indiquer où aller chercher toutes les classes requises
- Dans de multiples répertoires ou fichiers JAR
- Class path : les chemins que la JVM utilise pendant son exécution pour charger les classes (sauf celles de l'API Java)
- Class path fourni à la JVM au démarrage, exemple `java -cp path1:path2:path3 name1.name2.MyClass`
- NB il doit contenir des classes compilées
- Classes dans l'API Java : rien à préciser
- Class path par défaut ? "."
- Dans Eclipse : voir Java Build Path

Licence