13. Packages

Java

Fall 2009

Instructor: Dr. Masoud Yaghini

Outline

Introduction

Introduction

- **Packages** are used to group classes.
- Reasons for using packages
  - **To avoid naming conflicts**: When you develop reusable classes to be shared by other programmers, naming conflicts often occur. To prevent this, put your classes into packages so that they can be referenced through package names.
  - **To distribute software conveniently**: Packages group related classes so that they can be easily distributed.
  - **To protect classes**: Packages provide protection so that the protected members of the classes are accessible to the classes in the same package, but not to the external classes.

Package Naming & Directories

Package Naming

- Packages are hierarchical, and you can have packages within packages.
- For example, java.lang.Math indicates that:
  - Math is a class in the package lang and that
  - lang is a package in the package java.
- Levels of nesting can be used to ensure the uniqueness of package names.

Package Naming

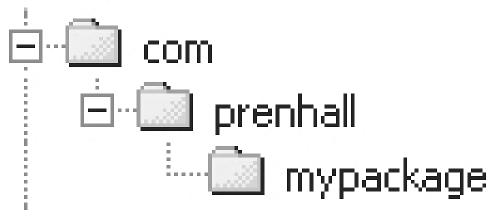
- Choosing a unique name is important because your package may be used on the Internet by other programs.
- Java designers recommend that you use your **Internet domain name** in reverse order as a package prefix.
- Since Internet domain names are unique, this prevents naming conflicts.
- Suppose you want to create a package named mypackage on a host machine with the Internet domain name prenhall.com.
- To follow the naming convention, you would name the entire package com.prenhall.mypackage.

Naming a Package

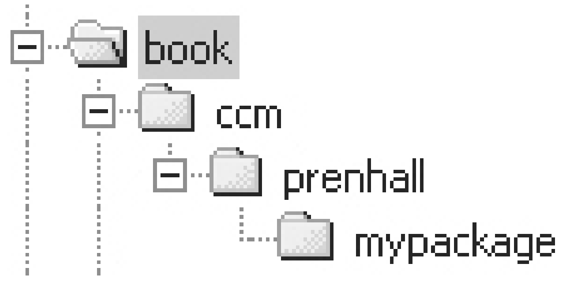
- Package names are written in all lowercase to avoid conflict with the names of classes or interfaces.
- Packages in the Java language itself begin with java. or javax.

Package Directories

- Java expects one-to-one mapping of the package name and the file system directory structure.
- For the package named com.prenhall.mypackage, you must create a directory, as shown below:



(a)



(b)

Putting Classes into Packages

Putting Classes into Packages

- Every class in Java belongs to a package.
- All the classes that you have used **so far** were placed in the current directory (a **default package**) when the Java source programs were compiled.
- To put a class in a specific package, you need to add the following line as the first noncomment and nonblank statement in the program:

  ```
  package packagename;
  ```

Putting Classes into Packages

- Let us create a class named Format and place it in the package com.prenhall.mypackage.
- The Format class contains the format(number, numberOfDecimalDigits) method
- It returns a new number with the specified number of digits after the decimal point.
- For example, format(10.3422345, 2) returns 10.34, and format (-0.343434, 3) returns -0.343.
- Java program:
  - Format.java

Putting Classes into Packages

- A class must be defined as public in order to be accessed by other programs.
- If you want to put several public classes into the package, you have to create separate source files for them, because each file can have only **one public class**.

Source& Class file directory in IntelliJ IDEA

- **NetBeans** uses the <projectname>\src directory path to store source files
- For example, if the project name is MyProject , then the source code file is automatically stored in \MyProject\src\com\prenhall\mypackage\
- **NetBeans** uses the <projectname>\build\classes\ directory path to store class files
- If the project name is MyProject , then the class files is automatically stored in \MyProject\build\classes\com\prenhall\mypackage\

Using Classes from Packages

Using Classes from Packages

- Example: Creating two classes
  - **Format**: It contains the format(number, numberOfDecimalDigits) method and returns a new number with the specified number of digits after the decimal point.
  - **TestFormatClass**: to invoke Format class and test it.

Option 1

- Creating the both classes in the default package and in the same Java file.
- Program:
  - TestFormatClass.java

Option 2

- Creating both classes in the com.prenhall.mypackage and in the same Java file.
- Program:
  - TestFormatClass.java

Option 3

- Creating both classes in the com.prenhall.mypackage and in the different Java file.
- If you create TestFormatClass class in the same package with Format, you can invoke the format method using ClassName.methodName (e.g., Format.format).
- Program:
  - TestFormatClass.java
  - Format.java

Calling a method from another package

- If you want to call a method from another package, you can invoke that method in two ways.
  - One way is to use the fully qualified name of the class (option 4) ,
    - i.e.: packagename.ClassName.methodName
    - This is convenient if the class is used only a few times in the program.
  - The other way is to use the import statement (option 5).

Option 4

- Creating the TestFormatClass in the default package and Format class in the com.prenhall.mypackage package and call it by using the fully qualified name of the Format class.
- Program:
  - TestFormatClass.java
  - Format.java (same as option 3)

Option 5

- Creating the TestFormatClass in the default package and Format class in the com.prenhall.mypackage package and call it by using the import statement.
  - TestFormatClass.java
  - Format.java (same as option 3)

Using Classes from Packages

- The program uses an import statement to get the class Format.
- You can import entire classes by:

  import com.prenhall.mypackage.*;
- You cannot import entire packages, such as:

  import com.prenhall.*.*;
- Only one asterisk (*) can be used in an import statement.

References

References

- Y. Daniel Liang, **Introduction to Java Programming**, Sixth Edition, Pearson Education, 2007. (Chapter 5)

The End