320341 Programming in Java

Fall Semester 2015

Lecture 5: Packages

Instructor: Jürgen Schönwälder

Slides: Bendick Mahleko

Objectives

The objective of this lecture is to

- Introduce packages in Java

The Package: The library unit

JACOBS UNIVERSITY

Package (Java keyword package)

- Bundles together components into a cohesive library unit
- Help in organizing your work
- Separate your work from code libraries provided by others
- Packages guarantee the uniqueness of class names

Example

- The standard Java library is distributed over a number of packages
- Ex: `java.lang, java.util, java.net` and so on
- The standard Java library is hierarchically organized
- Standard Java packages are inside the `java` and `javax` package hierarchies

Naming Packages

Use domain name written in reverse

- Ex: `de.jacobs_university`
- The package can then be further subdivided into subpackages
- Ex: `de.jacobs_university.eecs`

From a compiler's viewpoint, nested packages not related

- `de.jacobs_university` and `de.jacobs_university.eecs` packages have nothing to do with each other
- Each has its own independent collection of classes

Class Importation

A class can use all classes from its own package and all *public* classes from other packages

Public classes in other packages can be accessed in two ways:

1. Add full package name in front of every class name

```
java.util.Date today = new java.util.Date();
```

2. Use the `import` statement to refer to classes in the imported package
   - You can import the whole package or specific classes
   - Place the `import` statement at the top of the source file, but below the `package` statement

Example

Import all classes in `java.util` package as follows:

```
import java.util.*;
```

then use:

```
Date today = new Date();
```

without a package prefix

Importing a specific class

```
import java.util.Date;
```

- The `java.util.*` is less tedious
- It has no negative effect on code size

Importation Issues

- The * notation is used only to import a single package
- You can't use `java.*` or `java.*.*` to import all packages with `java` prefix

```
import java.util.*;
import java.sql.*;
...

Date today; // Error message
```

Static Imports

Static *import allows importing of static methods and fields*, not just classes (since JDK 5.0)

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

- Now you can use static methods & fields of the `System` class

```
out.println(„Goodbye, World“); // System.out
exit(0); // System.exit
```

- You can also import a specific method

```
import static java.lang.System.out;
```

- Applications: (1) Mathematical functions (2) Cumbersome constants

sqrt(pow(x, 2)) + pow(y, 2) *clearer than*
Math.sqrt(**Math**.pow(x, 2)) + **Math**.pow(y, 2)

Adding a Class to a Package

Put the name of the package at the top of source file

- Example

```
package com.horstmann.corejava;

// import statements

public class Employee {
  ...
}
```

Declaration Order

JACOBS UNIVERSITY

A Java source code file must have the following order:

1. A package declaration (if any)

2. Import declarations (if any) and

3. Class declarations

- Only one of the class declarations in a particular file can be public
- Non-public classes are in a package to support the reusable classes in the package

Default Package

If no package is declared, then classes in the source file belong to the *default package*

The default package has no package name

- *Place files in a package into subdirectory matching the full package name*

- Files in the package name

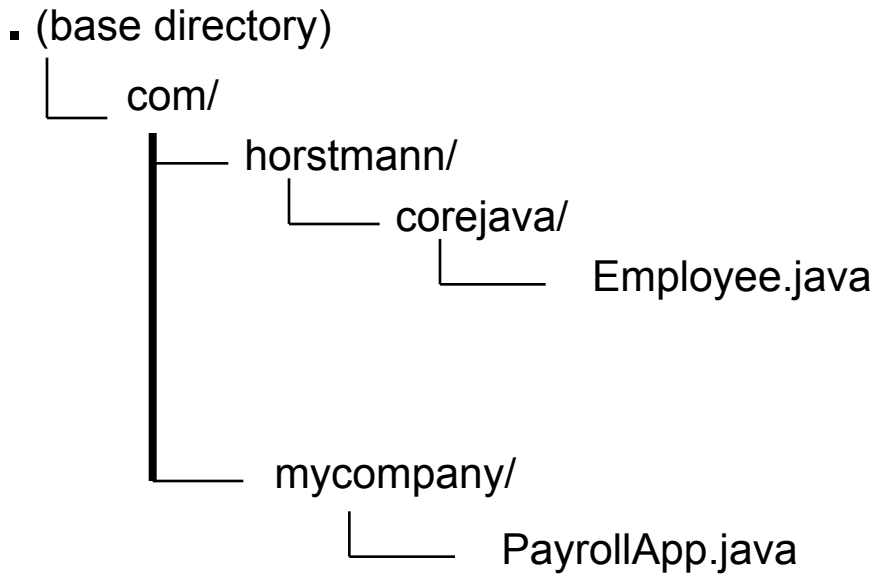
`com.horstmann.corejava`

- should be stored in a directory

`com/horstmann/corejava` in Unix or

`com\horstmann\corejava` in Windows

Compilation and Execution



Compile & run classes from the base directory

- `javac com/mycompany/PayrollApp.java`
- `java com.mycompany.PayrollApp`

Classpaths

JACOBS UNIVERSITY

The search order is as follows:

1. The Java class loader first searches the standard Java classes

2. The class loader then searches optional packages

3. The class loader searches the classpath
   - The classpath contains a list of locations in which classes are stored, each separated by a folder separator (;) on Windows and (:) on Unix/ Linux/ Mac OS X

   - By default the classpath consists only of the current directory

   - You can change the default by providing a **-classpath** option to the javac compiler or

   - Setting the **CLASSPATH** environment variable