Packages: Making Development Easier

Sara Sprenkle

June 13, 2006

Packages

- Java groups classes into collections called *packages*.
- The standard Java class libraries come in a number of packages, for example:
  - `java.lang`
  - `java.util`
  - `java.net`

Package Hierarchy

- The standard class libraries are hierarchical
  - similar to subdirectories on a file system
- All standard Java packages are in the `java` and `javax` package hierarchies.
- How do we use classes that are part of packages?

Package Name Specification

- A class can use all classes contained in its own package, as well as all public classes in other packages.
- You can fully specify the package name in front of the class to be used:

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

  - Tedious to write out every time the class is used

Importation

- Use the `import` keyword
  - `import` statements appear at the very beginning of your class' source code file.
  - A single class or an entire package can be imported
    - import java.util.Date; // single class
    - import java.util.*; // entire package
  - Now, can use the class as

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


Programmer-defined Packages

- You can create your own packages
- Declare a class to be part of a package by using the `package` keyword
  - beginning of the class source code file, before the class definition

Programmer-defined Packages

- If we want Chicken in a package farm.animals

```
package farm.animals;
public class Chicken {
…
}
```

- All of the classes in farm.animals are in a directory called animals that is contained in a directory called farm
- Another example of the hierarchy, packages
  - Java class libraries

Using Packages

- When making your own collection of classes that are all related and work together (hence, a package), put them all in the same directory that has the same name as the package
  - This is a good idea for all classes in a package, not just the public ones.
    - Then, you know where all of the classes in the package are defined.

Importing User-Defined Packages

- Use `import` keyword

```
package farm;
import farm.animals.*;
public class Farm {
      Chicken hank = …
      …
}
```


CLASSPATH

- javac, java need to find the classes
- The import statement tells the compiler to look in the farm directory for an animals directory and then to look in there for the class files it needs
- So how does the compiler find the farm directory?
- farm's parent directory must be present in the CLASSPATH environmental variable.

CLASSPATH

- If all of your package directories are in /home/username/java_files, that directory needs to be in the CLASSPATH environmental variable.
- Add the line:

```
setenv CLASSPATH /home/username/java_files
```

  to your .cshrc file (with a C-type Unix shell)
  - For multiple directories, separate with ":"
- CLASSPATH variable is checked during compilation and the running of the JVM
  - directories in the CLASSPATH are where the compiler looks for the directories indicated in import statements

Importing a User-defined Package

- The Java compiler/JVM will first look in the default path for Java system classes to find a package you want to import
  - How it found java.lang package for Strings, etc.
- If the compiler/JVM cannot find it in the default path, it looks inside each directory (":"-delimited) specified in the CLASSPATH environment variable
- If a class attempts to import the farm.animals package, it would look in

  /home/username/java_files/farm/animals

  to find the classes that compose the package.

Alternative to Defining CLASSPATH

- If you don't want some packages to permanently be in your CLASSPATH
  - Use -cp option to java or javac

```
java -cp /home/username/java_files *.java
```


Using classes from command line

- For the Farm class in the farm package
  - `java farm.Farm`

Package Scope and Access Modifiers

- Instance fields and method of a class have access modifiers
  - Public: an object of any class can access it.
  - Private: only objects of the same class can access it
  - Protected: objects of same class or subclass can access
  - And, I promised more …
    - If no access modifier is provided, it's the default
    - package scope: The feature can be accessed by an object of any class in the same package

Package Scope and Access Modifiers

- Same for class definitions
  - Public: any other class can access it.
  - Private: only objects of that class can access it.
  - No access modifier-->package scope: only other classes in the same package can access it

Conventions

- Packages are named in all lower case
- Package names include where the code is coming from, similar to DNS names
  - To guarantee uniqueness of package (avoid conflicts)
  - com.ibm.eclipse
  - org.apache.log4j
- For some recent packages that I released
  - edu.udel.cis.hiper.framework.replay
  - edu.udel.cis.hiper.framework.parser
- In this class, you'll start to use packages with your last name, e.g., sprenkle.assign2