CSE 143 Java

Class Packaging

Reading in N&H: Ch. 3, 4 (scattered)

Abstractions for Grouping Classes

- The class is the basic unit of modeling and program construction
- Most applications consist of more than one class
  - From a handful to hundreds of classes
- There should be ways to group related classes
- Three common levels of grouping
  - Within a file
  - Within a package
  - Within a project

Grouping Classes Within a .java file

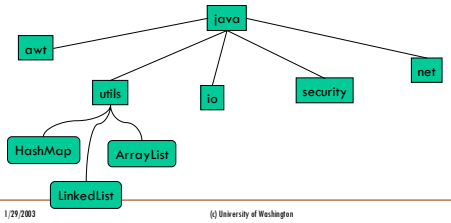
- A single .java file may contain multiple classes
  BUT at most one public class
- If a file contains a public class...
  the file name *must* match the class name (case sensitive)
- Normal practice: put only very tightly related classes in the same file
  When in doubt, use more than one file
- Compiler will still generate a separate .class file for each class in a .java file
- A class can be nested (defined) inside another class
  Called "inner classes" – we will *not* use these in CSE143 at present

Java Packages

- .java files can marked as belonging to a package
  - Use the Java **package** statement
  - package statement must be first non-comment statement in the .java file
  - All .java files in a package must have identical package statements
- Packages vs directories
  - All .java files of a package *must* be in the same directory
  - The directory *must* have the same name as the package (case sensitive)
- Packages are often nested
  - dot notation: package, slash notation: director

Packages and *import* Statement

- What does **import java.util.ArrayList;** mean?
  1. "There is a package named java which contains a package named util which contains a class named ArrayList (and I'm using that class)"
  2. "There is a directory named java which contains a directory named util which contains a class named ArrayList"



Class Paths

- The Java VM must know where to find classes at run-time

Projects

- Many IDE's have a "project" facility
- Files and packages may take part in projects
  - Sets of .java files, .class files, directories, data files, etc.
- Most often used to put together a complete application
- Can a file be part of more than one project?
  - it depends
- Can a package might be used in more than one project?
  - it depends

Packages vs. Projects

- **package** is a Java concept
  - is a Java keyword
    is a concept used in the **import** statement
  - is understood by the compiler and the JVM (Java Virtual Machine)
  - will be the same on all platforms
    at least abstractly
- **project** is *not* a Java concept
  - each development environment has its own notion of what a project is
  - Often, a project is a directory
    May be the same as a package
  - May instead be a non-package directory or something else
  - Some IDEs let you use a file or package in more than one project; some do not
  - Some environments have no project concept