Understanding Packages

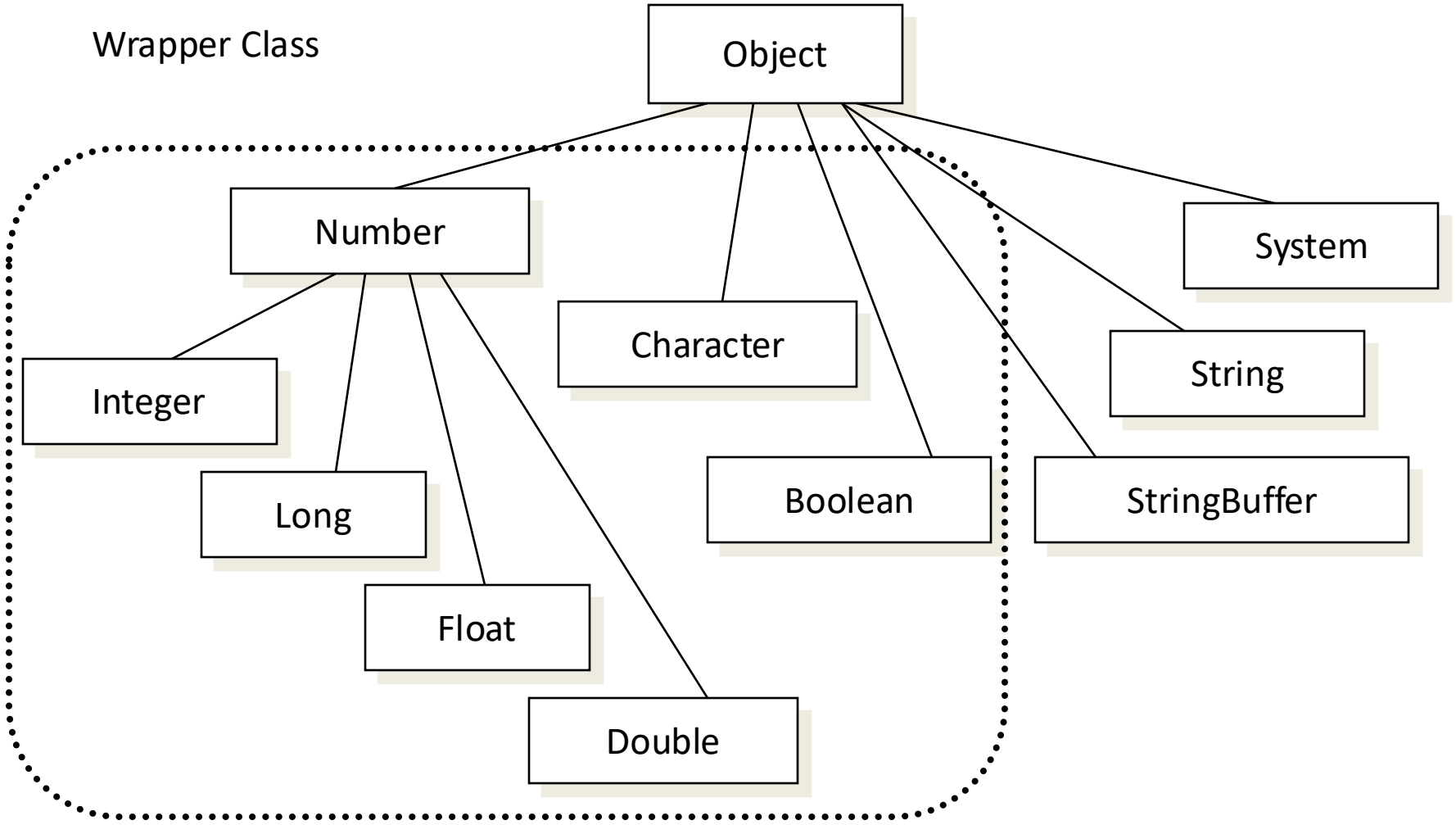
Packages

- A Package is a unique named collection of classes
- The purpose of grouping classes is to make it easy to add any or all of the classes
- Names of the classes within the package should be unique, but, same name can be repeated across the packages – as a class name is qualified with the package name

Package in Java

- A package is a namespace that organizes a set of related classes and interfaces.
- A java package is a group of related classes similar to a class library.
- Keyword ‘import’
- All of the standard classes are contained within the packages.
- If you put multiple types in a single source file, only one can be public, and it must have the same name as the source file.
- *java.lang* is special

Package in Java



Selected packages

- java.lang -- String, wrapper classes, Math
- java.util – Calendar, Date, Vector
- java.applet – Applet
- java.text – DateFormat
- java.awt – Graphics, Button, Label..
- javax.swing – JButton
- java.io – InputStream, OutputStream

Naming Conventions

- Package names are written in all lowercase to avoid conflict with the names of classes or interfaces. Companies use their reversed Internet domain name to begin their package name
  - For example, com.example.orion for a package named orion created by a programmer at example.com.

- Name collisions that occur within a single company need to be handled by convention within that company, perhaps by including the region or the project name after the company name
  - (for example, com.company.region.package).

- Packages in the Java language itself begin with java. or javax.

Naming conventions

- In some cases, the internet domain name may not be a valid package name.
  - This can occur if the domain name contains a hyphen or other special character, if the package name begins with a digit or other character that is illegal to use as the beginning of a Java name, or if the package name contains a reserved Java keyword, such as "int".
  - In this event, the suggested convention is to add an underscore.
  - Ex: clipart-open.org →org.clipart_open

Using Package Members

- The types that comprise a package are known as the *package members*. To use a public package member from outside its package, you must do one of the following:
  - Refer to the member by its fully qualified name
    - *graphics.Rectangle myRect = new graphics.Rectangle();*
  - Import the package member
    - *import graphics.Rectangle;*
      Rectangle myRectangle = new Rectangle();
  - Import the member's entire package
    - *import java.util.*;*
    - *import graphics.*;*
    - *import graphics.A*; //does not work*

Packaging your classes

- Add a package statement as the first statement in the source file containing the class definition
- It must always be the first statement
- Ex: package geometry;


```
//Sphere.java
package geometry;
public class Sphere
{
//details of the class definition
}
```

Packaging your classes

- Directory : geometry
- File: *Sphere.java*

- *Compiling: Ensure that you are out of the directory*

javac /geometry/Sphere.java

To execute

java geometry.Sphere

For a large number of files in a package you can use

javac /geometry/.java*

All interdependencies will be figured out by the compiler

Packages and directory structure

- A package is intimately related to the directory structure in which it is stored
- A class with a *Classname* must be stored in a file called *Classname.java*
- Similarly, all the files for classes within a package, *packageName* must be included in a directory with the name, *packageName.*
- A package need not have a single name. it can be a sequence of names separated by a period

package geometry.shapes3D;

package geometry.shapes2D;

  - shapes3D and shapes2D are sub directories of geometry directory.

Setting classpath

- From the command prompt,

set CLASSPATH = .;c:\MySource; c:\MyPackages

- Unless a class path is set, or set incorrectly, java will not be able to find the classes in any new packages you might create.

Adding classes from a Package to your Program

- Classes with 'public' are accessible to your program by using *'import'* statement.
  - *import geometry.shapes3D.*;* //includes all the classes from this package
  - '*' selects all the classes in the package
  - You can refer any public class in the package
  - If you have to add a specific class, specify explicitly like,

  import geometry.shapes3D.Sphere;
  - '*' can be only used to select all the classes in a package. You cannot use *geometry.** to select all the packages in the directory geometry.

Encapsulation of classes into a package

- Add a class into a package — two steps:
  1. put the name of the package at the top of your source file

```
package com.hostname.corejava;
public class Employee {
        . . .
}
```

  2. put the files in a package into a subdirectory which matches the full package name

stored in the file "Employee.java" which is stored under "somePath/com/hostname/corejava/"

Setting the class path

E.g.

- current classpath:

/home/user/classdir:.:/home/user/archives/archive.jar

- to search for the class file of the com.horstmann.corejava.Employee class

first searchs in the system class files that are stored in archives in the jre/lib and jre/lib/ext directories.

If it can't find the class there it will search in the order:

1) /home/user/classdir/com/horstmann/corejava/Employee.class
2) ./com/horstmann/corejava/Employee.class
3) com/horstmann/corejava/Employee.class inside /home/user/archives/archive.jar

- if found the interpreter stops searching process

Naming conventions

- ***Package names***: start with lowercase letter
  - E.g. java.util, java.net, java.io . . .
- ***Class names***: start with uppercase letter
  - E.g. File, Math . . .
  - avoid name conflicts with packages
  - avoid name conflicts with standard keywords in java system
- ***Variable, field and method names***: start with lowercase letter
  - E.g. x, out, abs . . .

- ***Constant names***: all uppercase letters
  - E.g. PI . . .
- ***Multi-word names***: capitalize the first letter of each word after the first one
  - E.g. HelloWorldApp, getName . . .
- ***Exception class names***: (1) start with uppercase letter (2) end with "Exception" with normal exception and "Error" with fatal exception
  - E.g. OutOfMemoryError, FileNotFoundException