static Variables and Methods - Methods

- Sometimes a class defines methods that are not invoked on an object
  - Called a **static method**
- Example: sqrt method of Math class
  - if x is a number, then the call x.sqrt() is not legal
  - Math class provides a static method: invoked as Math.sqrt(x)
  - No object of the Math class is constructed.
  - The Math qualifier simply tells the compiler where to find the sqrt method.

static Variables and Methods

- You can define your own static methods:

```
public class Financial
{
   /** Computes a percentage of an amount.
       @param percentage the percentage to apply
       @param amount the amount to which the percentage is applied
       @return the requested percentage of the amount
   */
   public static double percentOf(double percentage, double amount)
   {
      return (percentage / 100) * amount;
   }
}
```

- When calling such a method, supply the name of the class containing it:

```
double tax = Financial.percentOf(taxRate, total);
```

- The main method is always static.
  - When the program starts, there aren't any objects.
  - Therefore, the first method of a program must be a static method.
- Programming Tip: Minimize the Use of static Methods

Packages

- **Package:** Set of related classes
- Important packages in the Java library:

Package	Purpose	Sample Class
java.lang	Language support	Math
java.util	Utilities	Random
java.io	Input and output	PrintStream
java.awt	Abstract Windowing Toolkit	Color
java.applet	Applets	Applet
java.net	Networking	Socket
java.sql	Database Access	ResultSet
javax.swing	Swing user interface	JButton
omg.w3c.dom	Document Object Model for XML documents	Document

Organizing Related Classes into Packages

- To put classes in a package, you must place a line
  ```
  package packageName;
  ```
  as the first instruction in the source file containing the classes.
- Package name consists of one or more identifiers separated by periods.
- To put the `Financial` class into a package named `com.horstmann.bigjava`, the `Financial.java` file must start as follows:

```
package com.horstmann.bigjava;
public class Financial
{
   . . .
}
```

Organizing Related Classes into Packages

- A special package: default package
  - Has no name
  - No `package` statement
  - If you did not include any `package` statement at the top of your source file
    - its classes are placed in the default package.

Importing Packages

- Can use a class without importing: refer to it by its full name (package name plus class name):

```
java.util.Scanner in = new java.util.Scanner(System.in);
```

- Inconvenient
- `import` directive lets you refer to a class of a package by its class name, without the package prefix:

```
import java.util.Scanner;
```

- Now you can refer to the class as `Scanner` without the package prefix.
- Can import all classes in a package:

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

- Never need to import `java.lang`.
- You don't need to import other classes in the same package .

Package Names

- Use packages to avoid name clashes:

```
java.util.Timer
```

vs.

```
javax.swing.Timer
```

- Package names should be unique.
- To get a package name: turn the domain name around:

```
com.horstmann.bigjava
```

- Or write your email address backwards:

```
edu.sjsu.cs.walters
```

Packages and Source Files

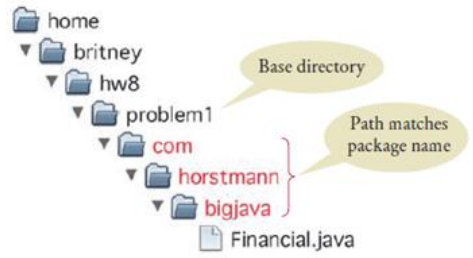

Figure 6 Base Directories and Subdirectories for Packages