Lecture 1

Types & Objects

Java is a Strongly Typed Language

- **Type: Set of values and operations on them**
  - Examples of operations: +, -, /, *
  - The meaning of these depends on the type
- All values/expressions must have a type
  - Purpose of Wednesday's lab
- **All variables must have a type**
  - Type restricts what can go in the variable
  - Why? It is easier to catch errors this way

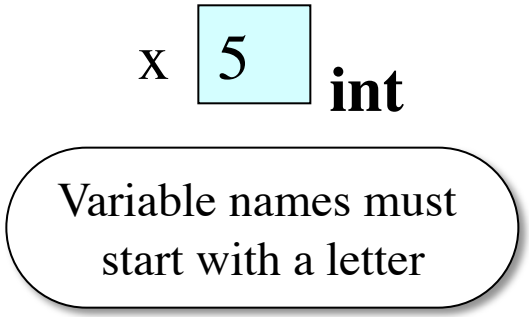
Type: Set of values and the operations on them

- Type **int**:
  - **Values**: integers
  - **Ops**: +, –, *, /, %, *
- Type **double**:
  - **Values**: real numbers
  - **Ops**: +, –, *, /, *
- Type **boolean**:
  - **Values**: **true** and **false**
  - **Ops**: && (and), || (or), ! (not)
- Type **char**:
  - **Values**: single characters
    - Stored in single quotes
    - Example: `'abc'`
  - **Ops**: +, –, *, /, %, *
    - Essentially a number (?!)
- Type **String**:
  - **Values**: string literals
    - **char** list in double quotes
    - Example: `"abc"`
  - **Ops**: + (concatenation)

Variables (p. 26)

- A **variable** is
  - a **named** memory location (**box**),
  - a **value** (in the box), and
  - a **type** (limiting what can be put in box)

Might be new to you



Here is variable **x**, with value 5.
It can contain an **int** value.

Here is variable **area**, with value 20.1.
It can contain a **double** value.

Variable Declaration (p. 26)

- A *declaration of a variable* gives the **name** of the variable and the **type** of value it can contain

int x;

Here is a declaration of x, indicating that it contain an **int** value.

double area;

Here is a declaration of area, indicating that it can contain a **double** value.

Assignment Statement (p. 27)

- *Execution of an assignment statement* stores a value in a variable

To execute the assignment

<var>= <expr>;

evaluate expression <expr> and store its value in variable <var>

x= x + 1;

Evaluate expression x+1 and store its value in variable x.

Initialization: Declaration+Assignment

- Can combine declaration and assignment

int x = 3;

Here is a declaration of x, indicating that it contain an **int** value.
It starts with a value of 3.

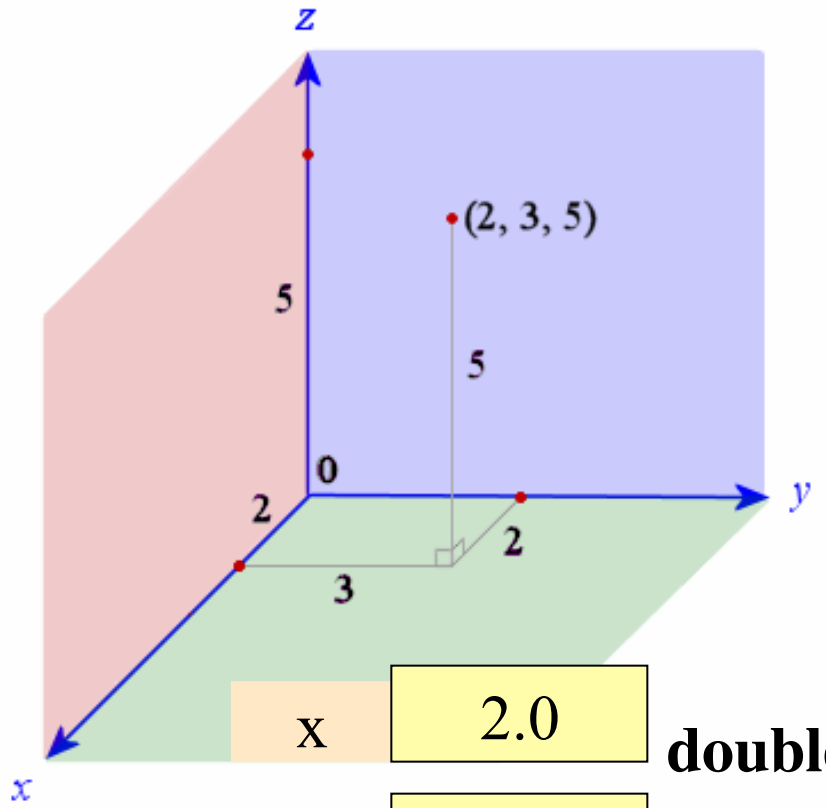
double area = 2.3;

Here is a declaration of area, indicating that it can contain a **double** value.
It starts with a value of 2.3.

- This is called **initializing** the variable.
  - As a rule it is good to initialize all declarations.
  - Will see what happens if you do not, later.

Type: Set of values and the operations on them

- Suppose we want to compute with a 3D point
- We need three variables
  - x, y, z coordinates
  - Each has type double
- What if have a lot of points?
  - Vars x0, y0, z0 for first point
  - Vars x1, y1, z1 for next point
  - …
  - This can get really messy

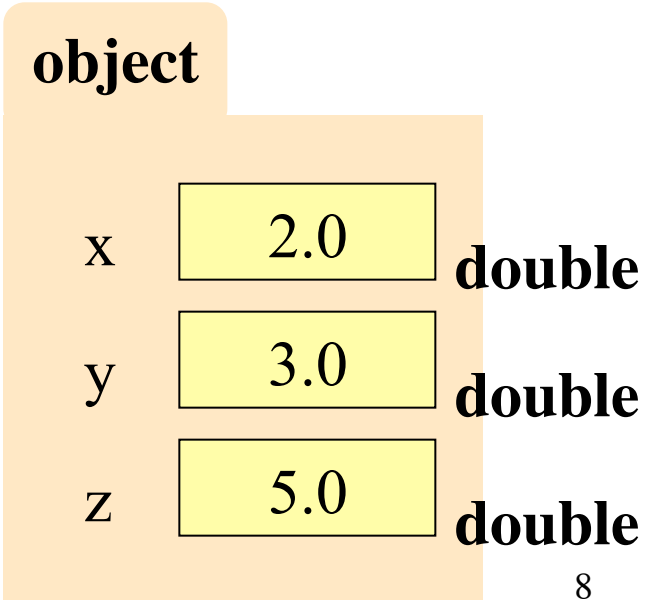


x	2.0	**double**
y	3.0	**double**
z	5.0	**double**

Type: Set of values and the operations on them

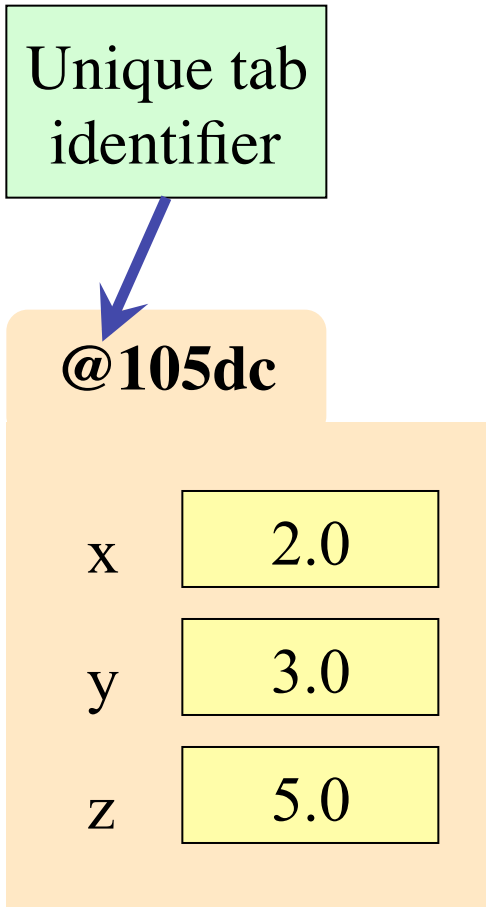
- Suppose we want to compute with a 3D point
- We need three variables
  - x, y, z coordinates
  - Each has type double
- What if have a lot of points?
  - Vars x0, y0, z0 for first point
  - Vars x1, y1, z1 for next point
  - …
  - This can get really messy

- Can we stick them together in a "folder"?
- This is the motivation for **objects**



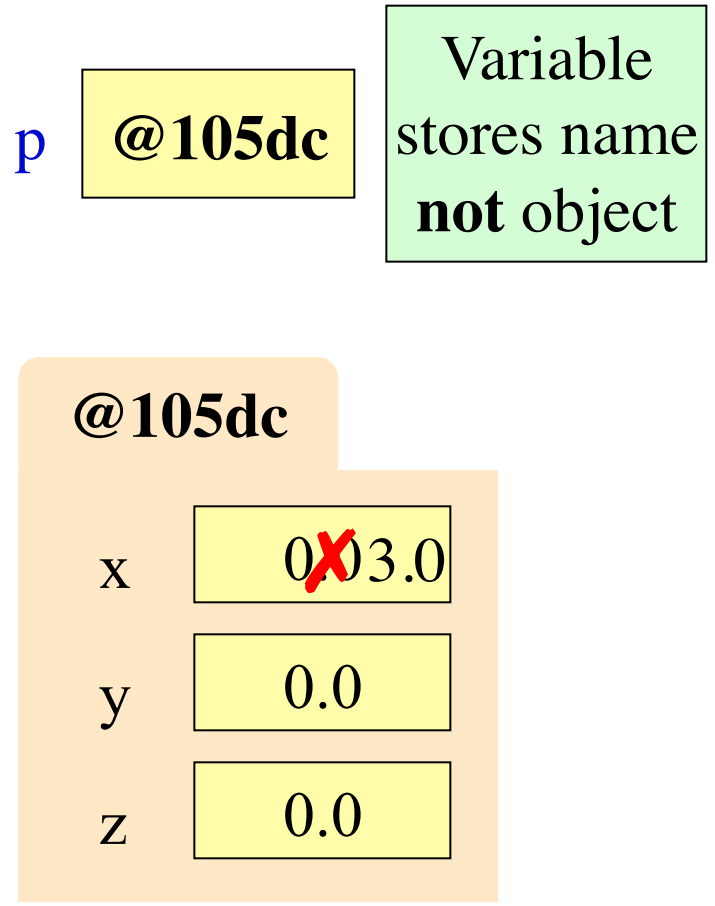
Objects: Organizing Data in Folders

- An object is like a **manila folder**
- It contains other variables
  - These variables are called **fields**
  - Can change their values (with assignments)
- It has a "tab" that identifies it
  - You cannot change this
  - Java assigns it automatically
  - More on this in demo later



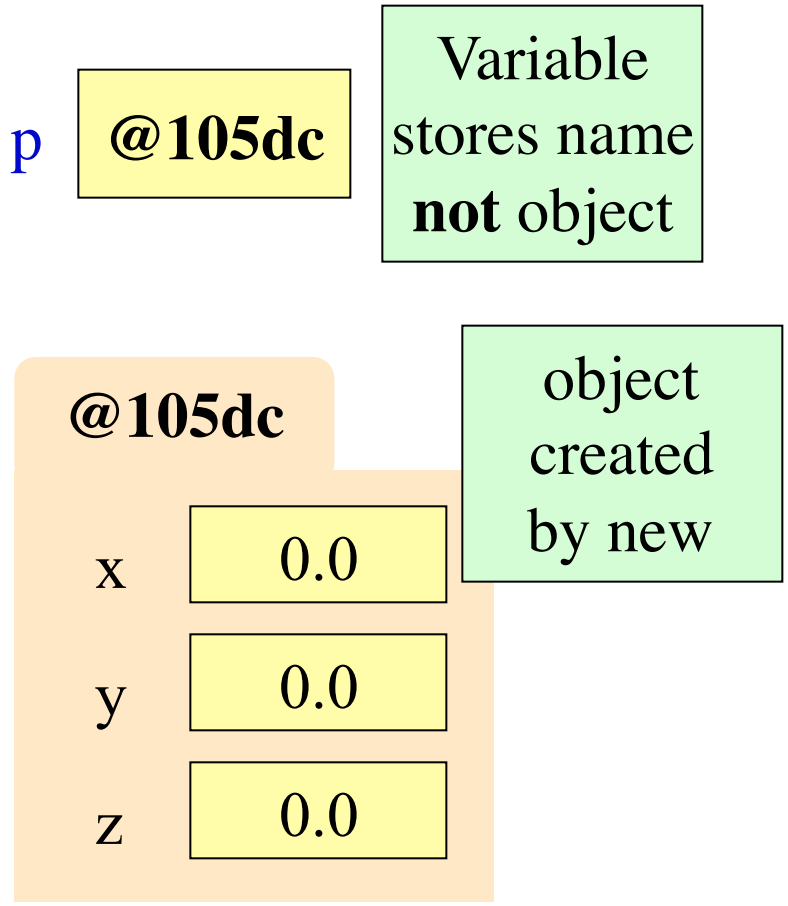
Object Variables

- Variable stores object name
  - **Reference** to the object
  - Reason for folder analogy
- Use "dot" to access folder
  - Use p.x to access to field x
  - **Example**: p.x = 3;
- How do we create objects?
  - Other types have **literals**
  - **Example**: 1, `"abc"`, **true**
  - No such thing for objects



Object Initialization (the new keyword)

- **new** Point3d()
  - An **expression** (produces a value)
  - It creates a object (folder)
  - Value is the "tab name"
- p = **new** Point3d();
  - **Assignment statement**
  - Computes value **new** Point3d()
  - Stores value (tab name) in the variable p



Exercise: Objects and Assignment

- What is the value of q?

  Point3d p = new Point3d();

  Point3d q = p;

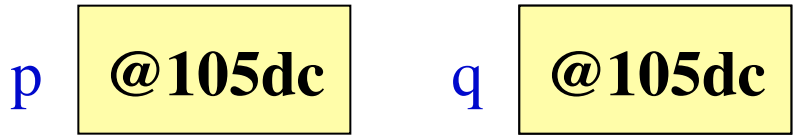
- Execute the commands:

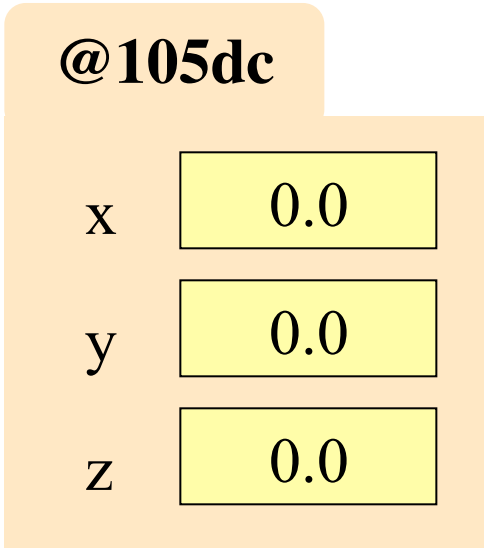
  p.x = 5.6;

  q.x = 7.4;

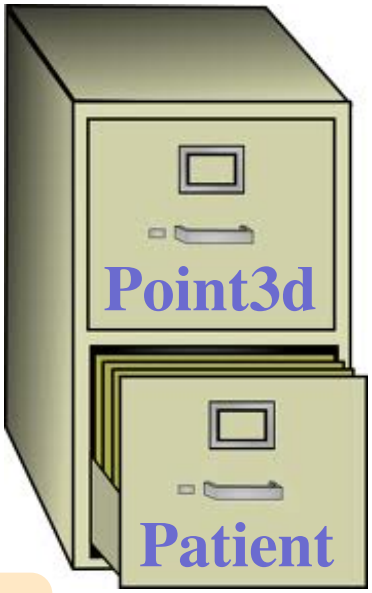
- What is value of p.x?

  A: 5.6
  B: 7.4 **CORRECT**
  C: @105dc
  D: I don't know





Classes: Types for Objects



- All values must have a type
  - An object type is a **class**
  - But it is more than that...
- A class is like a **file drawer**
  - Contains the manila folders
  - Each has same type of info
    e.g. same fields

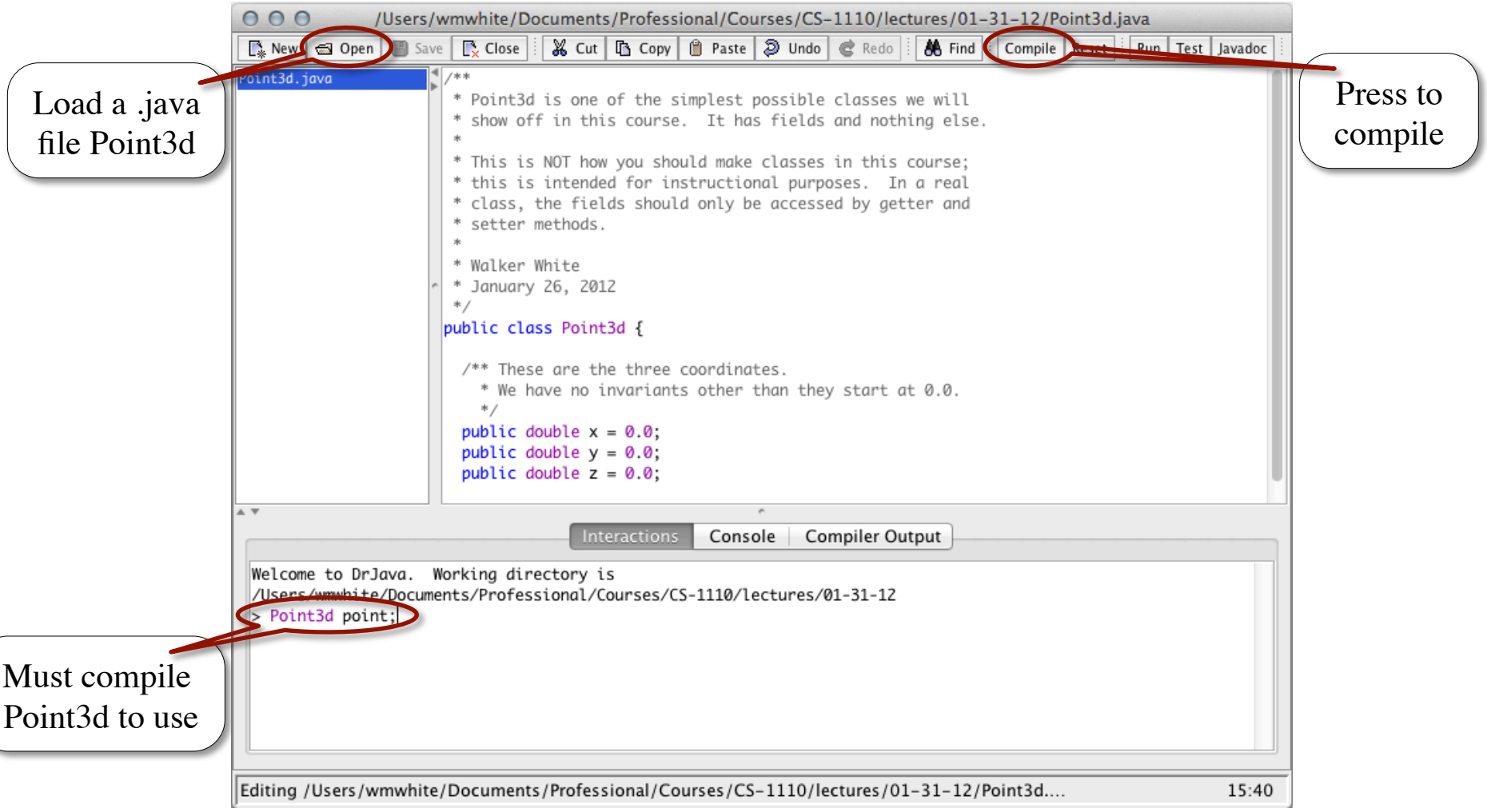
@3e9cff

name	"W. White"
address	"New York"
owes	250.00

Patient

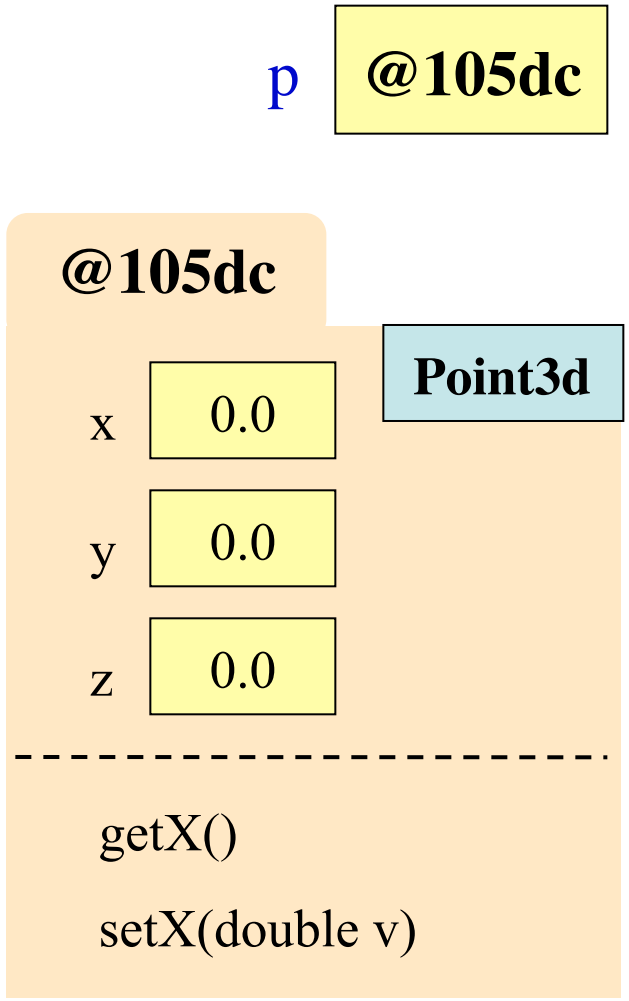
class name

Compiling a Class



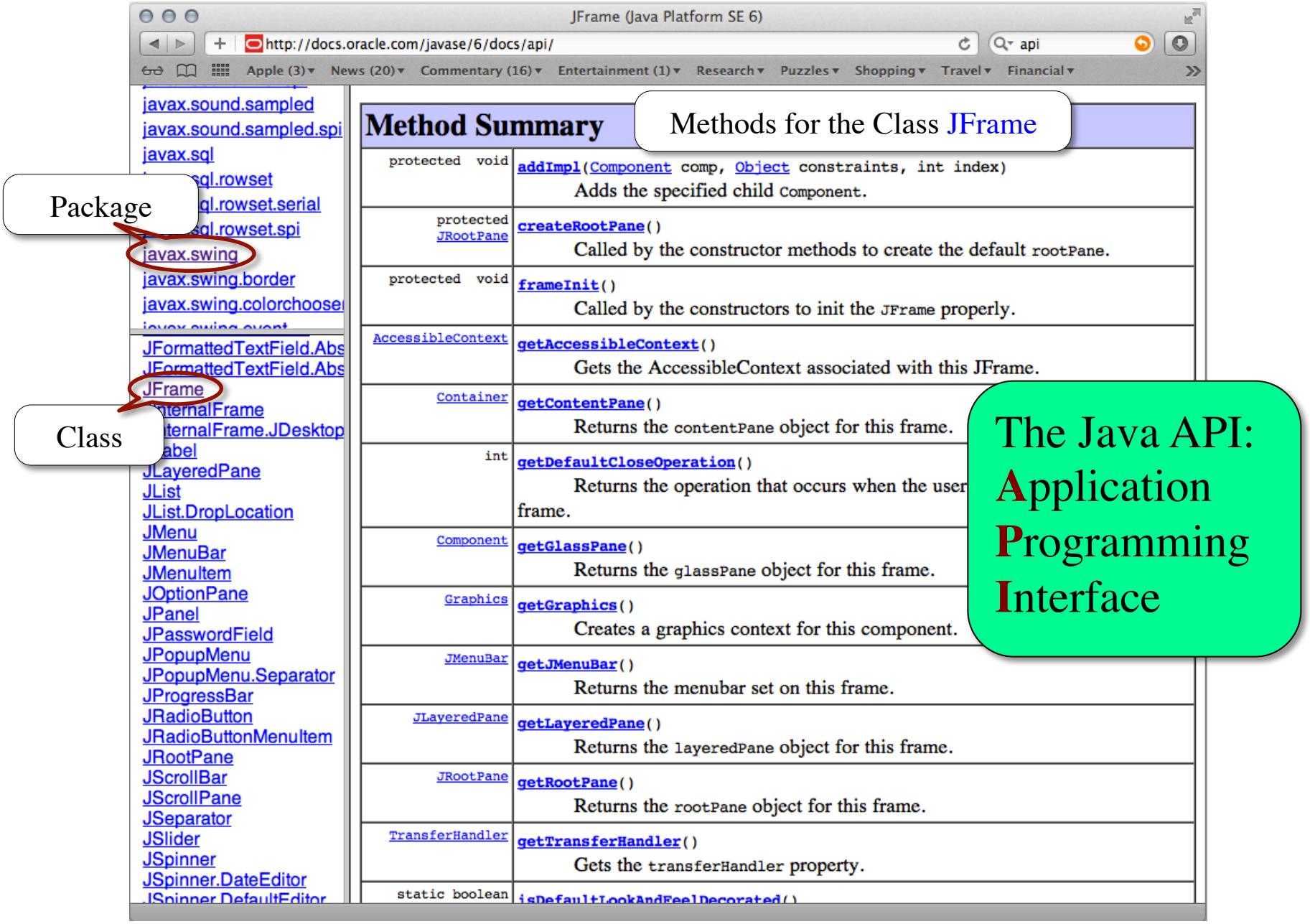
Methods: Operations on Objects

- **Method**: instruction for an object
  - Similar to a function/procedure
  - But attached to an object
  - Can access all of object's fields
- Use of a method is a *method call*
  - <object-variable>.<method-call>
  - Method calls end in parentheses
  - Values in parens are *arguments*
  - **Example**: p.getX()
  - **Example**: p.setX(3.4);



Packages and Built-in Classes

- Java has built-in classes
  - No need to compile them
  - But you have to import them
- Built-in classes are in **packages**
  - Use a command to import
  - **import <package>.<class>;**
  - **import <package>.*;**
    - imports everything in package
- **Example**: JFrame
  - Java class for (empty) Window
  - In package javax.swing


JFrame (Java Platform SE 6)
http://docs.oracle.com/javase/6/docs/api/
api
Apple (3)
News (20)
Commentary (16)
Entertainment (1)
Research
Puzzles
Shopping
Travel
Financial
javax.sound.sampled
javax.sound.sampled.spi
javax.sql
javax.swing
javax.swing.border
javax.swing.colorchoose
JFormattedTextField.Abs
JFormattedTextField.Abs
JFrame
JLayeredPane
JList
JList.DropLocation
JMenu
JMenuBar
JMenuItem
JOptionPane
JPanel
JPasswordField
JPopupMenu
JPopupMenu.Separator
JProgressBar
JRadioButton
JRadioButtonMenuItem
JRootPane
JScrollBar
JScrollPane
JSeparator
JSlider
JSpinner
JSpinner.DateEditor
Method Summary
Methods for the Class JFrame
protected void
addImpl(Component comp, Object constraints, int index)
Adds the specified child Component.
protected JRootPane
createRootPane()
Called by the constructor methods to create the default rootPane.
protected void
frameInit()
Called by the constructors to init the JFrame properly.
AccessibleContext
getAccessibleContext()
Gets the AccessibleContext associated with this JFrame.
Container
getContentPane()
Returns the contentPane object for this frame.
int
getDefaultCloseOperation()
Returns the operation that occurs when the user
frame.
Component
getGlassPane()
Returns the glassPane object for this frame.
Graphics
getGraphics()
Creates a graphics context for this component.
JMenuBar
getJMenuBar()
Returns the menubar set on this frame.
JLayeredPane
getLayeredPane()
Returns the layeredPane object for this frame.
JRootPane
getRootPane()
Returns the rootPane object for this frame.
TransferHandler
getTransferHandler()
Gets the transferHandler property.
static boolean
Package
Class
The Java API:
Application
Programming
Interface

String is a Class!

String s = "Hello World";

- Different from other classes
  - Do **not** create with new
- In package java.lang
  - Imported by default
  - Never need to import
- Great class to "play with"
  - All methods are functions
  - Use in interactions pane

String Methods

- **charAt(int p)**
  Get letter at position p
- **substring(int p)**
  Get suffix starting at position p
- **substring(int p, int e)**
  Get suffix starting at postion p, ending at e-1

Where To From Here?

- OO Programming is about **creating classes**
  - You will learn to make your own classes
  - You will learn what you can do with methods
- Understanding classes and objects is important