

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**), Might be new to you
 - a **value** (in the box), and
 - a **type** (limiting what can be put in box)

x 5 **int**

Variable names must start with a letter

area 20.1 **double**

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.

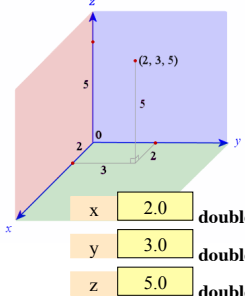
Variable Declaration (p. 26)

- 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)

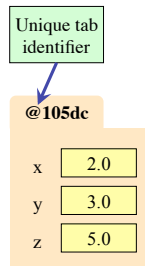
- *Execution of an assignment statement* stores a value in a variable
- To execute the assignment
<var>= <expr>;
evaluate expression <expr> and store its value in variable <var>
- x = x + 1; Evaluate expression x+1 and store its value in variable x.

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
- 

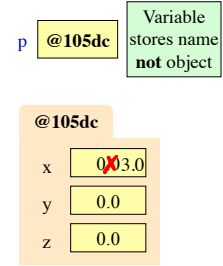
Objects: Organizing Data in Folders

- 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



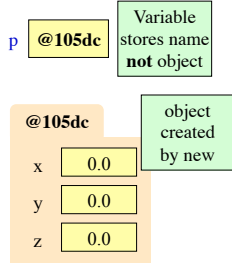
Object Variables

- 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



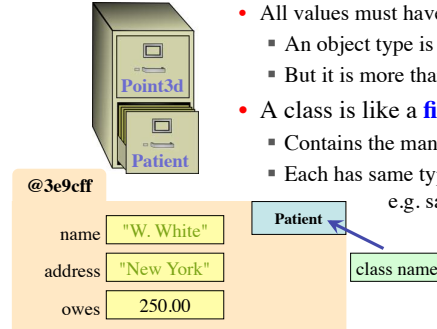
Object Initialization (the new keyword)

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



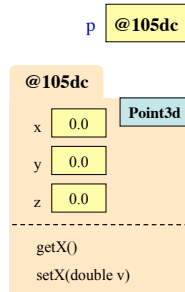
Classes: Types for Objects

- 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



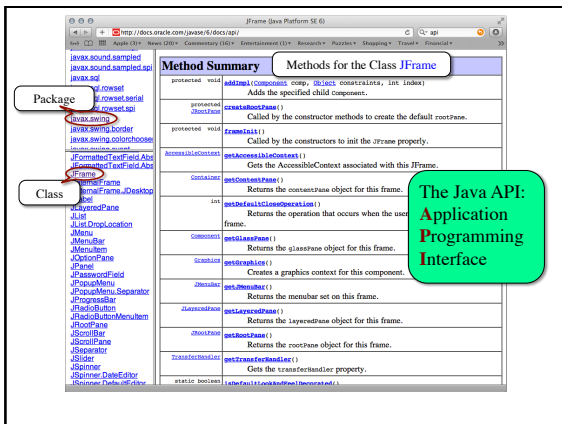
Methods: Operations on Objects

- **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**



String is a Class!

String s = "Hello World";

String Methods

- 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
- **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 position p, ending at e-1