Announcements for This Lecture

Readings

- Section 1.6, 4.1 (today)
- Section 4.2 (Thursday)

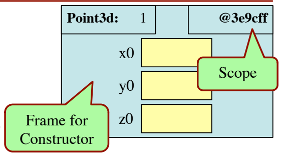
- **Prelim, March 8th 7:30-9:30**
  - Material up to next Tuesday
  - Sample prelims from past years on course web page
- **Conflict with Prelim time?**
  - Submit to Prelim 1 Conflict assignment on CMS
  - Do not submit if no conflict

Announcements

- Assignment 1 Resubmissions
  - Want "final version" tonight
  - But keep doing until get a 10
- Assignment 2 at end of class
- Assignment 3 is now posted
  - Due next Tuesday to CMS
  - Even if still working on A1
  - Keep A1, A3 in separate folders
- It calms down after this…

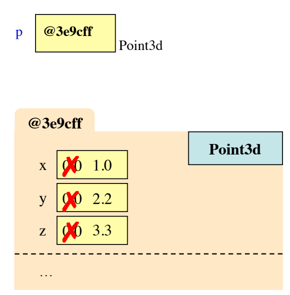
Constructors are Instance Methods

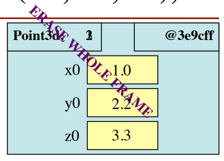
1. Make a new object (folder)
   - Java gives the folder a name
   - All fields are default (0 or null)
2. Draw a frame for the call
3. Assign the argument value to the parameter (in frame)
4. Execute the method body
   - Look for variables in the frame
   - Execute statements to initialize the fields to non-default values
   - Give the folder name as the result
5. Erase the frame for the call



```
public Point3d(  double x0,
         double y0,
         double z0) {
   x = x0;
   y = y0;
   z = z0;
}
```

Example: p = new Point3d(1.0, 2.2, 3.3);

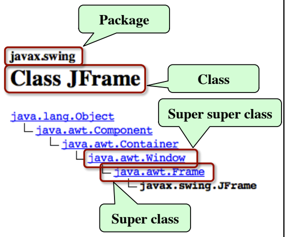




```
public Point3d(  double x0,
         double y0,
         double z0) {
   x = x0;
   y = y0;
   z = z0;
}
```

Subclasses in the Java API

- Subclassing creates a hierarchy of classes
  - Subclass has a super class or "parent" class
  - That parent may have a super class as well
- Explicit in the Java API
  - API does not respecify **inherited** methods
  - Often have to go to super class for specification



```
java.lang.Object
  └ java.awt.Component
      └ java.awt.Container
          └ java.awt.Window
              └ java.awt.Frame
                  └ javax.swing.JFrame
```

Class Definition REVISITED

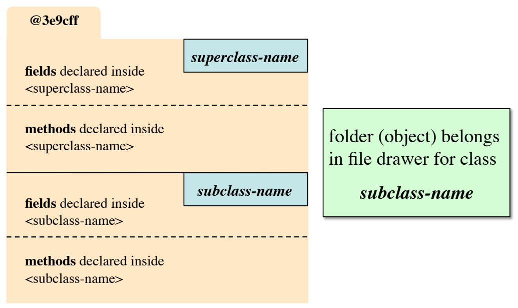
- Describes the format of a folder (instance, object) of the class.

```
/**
 * Description of what the class is for
 */
public class <class-name> extends <super-class> {

     declarations of fields and methods (in any order)
}
```

- Class <class-name> has all methods and fields of its parent
  - We say that it **inherits** them
- Also has any new fields or methods declared inside of it

Folder Analogy and Subclasses


@3e9cff
superclass-name
fields declared inside <superclass-name>
methods declared inside <superclass-name>
subclass-name
fields declared inside <subclass-name>
methods declared inside <subclass-name>
folder (object) belongs in file drawer for class
subclass-name

Object: The Superest Class of All

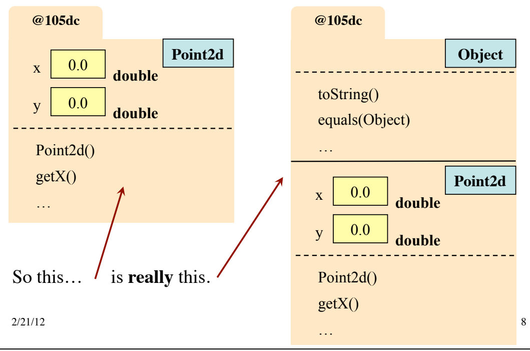
- How does toString() work?
  - All classes have a toString() by default
  - Default string is the folder name
  - Defining toString() in subclass **overrides** this method
- **Java Feature**: Every class that does not extend another class automatically extends class Object.

public class C { … }

‖

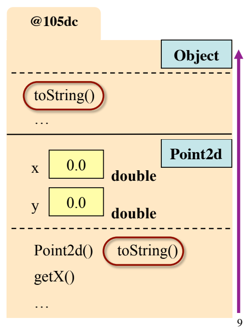
public class C **extends** Object { …}

Object: The Superest Class of All



The Bottom-Up Rule

- Which toString() is called?
  - Work the way up from the bottom of the folder.
  - Find the first method header that matches
  - Use the definition from the .java file for that class
- New method definitions **override** those of super class



Keywords `this` and `super`

`this`

- Refers to the object name in scope box of the method call
- this.<field> is field in object
  - Example: this.x
- this.<method-call> calls a method in this object
  - Example: this.getX()
- this(<parameters>) calls a constructor
  - Example: this(0.0,0.0,0.0)

`super`

- Functions mostly the same as this (refers to object in scope)
- super.<method-call> calls a method in the superclass or even higher up!
- super(<parameters>) calls constructor of super class
  - Useful for initialization
  - Necessary if fields private

Using `this` as a Constructor

- Usage: `this(<params>)`
  - Looks for constructor with parameters of that type
  - Calls that constructor as a helper method
  - Can only do this inside another constructor
- This is why object name must be in the scope box
  - Else what is `this`?
  - `this` = name in scope box

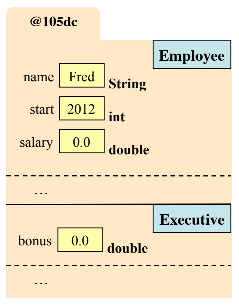
```
public Point3d(double x0,
                double y0,
                double z0) {
    x = x0;
    y = y0;
    z = z0;
}

public Point3d() {
    // Uses other constructor.
    this(0.0,0.0,0.0)
}
```

Using `super` in a Constructor

```
public Employee(String n, int d) {
   name= n;
   start= d;
   salary= 50000;
}
```

```
public Executive(String n, int d,
                 double b) {
   super(n,d);
   bonus = b;
}
```



@105dc
Employee
name Fred String
start 2012 int
salary 0.0 double
…
Executive
bonus 0.0 double
…