1

# CS/ENGRD 2110
# FALL 2017

Lecture 2: Objects and classes in Java
http://courses.cs.cornell.edu/cs2110

## CMS VideoNote.com, PPT slides, DrJava

2

CMS. Visit course webpage, click "Links", then "CMS for 2110".

Videos of our lectures: Look at

http://cornell.videonote.com/channels/1027/videos

Download ppt slides the evening before each lecture, have them available in class. Please don't ask questions on the piazza about that material the day before the lecture!

Download DrJava (the jar file, not the app). It may require downloading an old version of Java.

Got a Java question? See first if it's answered on JavaHyperText

## Next week's recitation/discussion: Exception handling

3

Visit course webpage, click on "Lecture notes"

For the row for recitation 2, click on "tutorial".

Your job: BEFORE your recitation, look at all videos in the tutorial. There are about 25-30 minutes of tutorial! This is the longest tutorial you will have to do.

Then, in recitation/discussion, you will have a problem set to do. Can do it with 1-2 other people (up to groups of 3). TA will walk around, helping, answering questions, giving pointers, etc.

The problem set is due on the CMS no later than one week after the recitation (always on Wednesday night). But we encourage you to finish during the recitation and turn it in immediately.

Anything to be submitted is always on the course assignments page!

## Assignment A2

4

Get assignment A2 from course website later today.

Objective:

- Get practice with Java functions.
- Learn about and use JUnit testing

Given to you *before* ***A1***, but due *after* A1. Provide flexibility, allow you to get ahead and learn Java early.

Groups. You can do A2 with 1 other person. FORM YOUR GROUP EARLY! Use pinned Piazza Note @5 to search for partner!

## Java OO (Object Orientation)

5

Python and Matlab have objects and classes.

Strong-typing nature of Java changes how OO is done and how useful it is. Put aside your previous experience with OO (if any).

This lecture:

**First**: describe objects, demoing their creation and use.

**Second**: Show you a class definition and how it contains definitions of functions and procedures that appear in each object of the class.

**Third**: Talk about keyword **null**.

**Fourth (if there is time)**. Show you a Java application, a class with a "static" procedure with a certain parameter.

## Homework

6

1. Study material of this lecture.
2. Visit course website, click on Resources and then on Code Style Guidelines. Study
   - 3. Documentation
     - 3.1 Kinds of comments
     - 3.2 Don't over-comment
     - 3.4 Method specifications
       - 3.4.1 Precondition and postcondition
3. Spend a few minutes perusing slides for lecture 3; bring them to lecture 3.

## Java OO

7

References to JavaHyperText entries

- Objects: B.1 object
- Calling methods: method call
- Class definition: class
- **public, private:** public private
- method
- Parameter vs argument: parameter, argument
- Inside-out rule

Methods may have parameters
Method calls may have arguments

Text mentions fields of an object. We cover these in next lecture

Text uses value-producing method for function and void method for procedure. Get used to terminology: function and procedure

## Drawing an object of class javax.swing.JFrame

8

Object is associated with a window on your computer monitor

Name of object, giving class name and its memory location (hexadecimal). Java creates name when it creates object

JFrame@25c7f37d

JFrame

hide() show()
setTitle(String) getTitle()
getX() getY() setLocation(int, int)
getWidth() getHeight() setSize(int,int)
…

Object contains methods (functions and procedures), which can be called to operate on the object

Function: returns a value; call on it is an expression
Procedure: does not return a value; call on it is a statement

## Evaluation of new-expression creates an object

9

Evaluation of **new** javax.swing.JFrame() JFrame@25c7f37d

creates an object and gives as its value the name of the object

If evaluation creates this object, value of expression is

JFrame@25c7f37d

9

2 + 3 + 4

JFrame@25c7f37d

JFrame

hide() show()
setTitle(String) getTitle()
getX() getY() setLocation(int, int)
getWidth() getHeight() setSize(int,int)
…

## A class variable contains the name of an object

10

Type JFrame: Names of objects of class JFrame

```
javax.swing.JFrame h;
h= new javax.swing.JFrame();
```

If evaluation of new-exp creates the object shown, name of object is stored in h

Consequence: a class variable contains not an object but name of an object, pointer to it. Objects are referenced indirectly.

h JFrame@25c7f37d
JFrame

JFrame@25c7f37d

JFrame

hide() show()
setTitle(String) getTitle()
getX() getY() setLocation(int, int)
getWidth() getHeight() setSize(int,int)
…

## A class variable contains the name of an object

11

If variable h contains the name of an object, you can call methods of the object using dot-notation:

Procedure calls: h.show(); h.setTitle("this is a title");

Function calls: h.getX() h.getX() + h.getWidth()

```
x= y;
g= h;
```

h JFrame@25c7f37d
JFrame

JFrame@25c7f37d

JFrame

hide() show()
setTitle(String) getTitle()
getX() getY() setLocation(int, int)
getWidth() getHeight() setSize(int,int)
…

## Class definition: a blueprint for objects of the class

12

**Class definition: Describes format of an object (instance) of the class.**

```
/** description of what the class is for */
public class C {
    declarations of methods (in any order)
}
```

This is a comment

Access modifier **public** means C can be used anywhere

Class definition C goes in its own file named
C.java

On your hard drive, have separate directory for each Java project you write; put all class definitions for program in that directory. You'll see this when we demo.

## First class definition

13

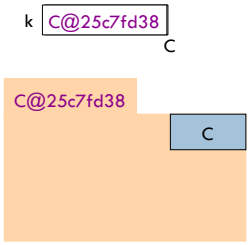
```
/** An instance (object of the class) has (almost) no methods */
public class C {

}
```

Then, execution of

```
C k;
k= new C();
```

creates object shown to right and stores its name in k



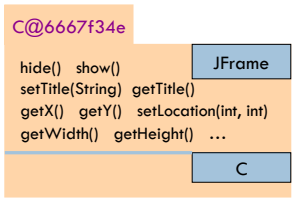

## Class extends (is a subclass of) JFrame

14

```
/** An instance is a subclass of JFrame */
public class C extends javax.swing.JFrame {

}
```

C: subclass of JFrame
JFrame: superclass of C
C *inherits* all methods that are in a JFrame




Object has 2 partitions: one for JFrame methods, one for C methods

Easy re-use of program part!

## Class definition with a function definition

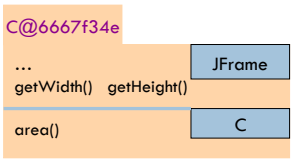
15

```
/** An instance is a subclass of JFrame with a function area */
public class C extends javax.swing.JFrame {
   /** Return area of window */
   public int area() {
      return getWidth() * getHeight();
   }
}
```

Spec, as a comment

Function calls automatically call functions that are in the object

You know it is a function because it has a return type



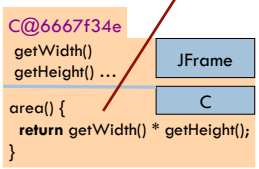

## Inside-out rule for finding declaration

16

```
/** An instance … */
public class C extends javax.swing.JFrame {
   /** Return area of window */
   public int area() {
      return getWidth() * getHeight();
   }
}
```

The whole method is in the object

To what declaration does a name refer? Use **inside-out rule**: Look first in method body, starting from name and moving out; then look at parameters; then look outside method in the object.

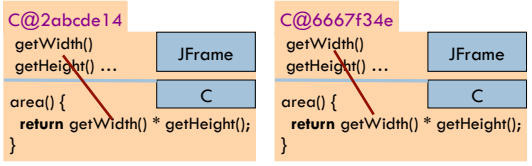



## Inside-out rule for finding declaration

17

```
/** An instance … */
public class C extends …JFrame {
   /** Return area of window */
   public int area() {
      return getWidth() * getHeight();
   }
}
```

Function area: in each object. getWidth() calls function getWidth in the object in which it appears.




## Class definition with a procedure definition

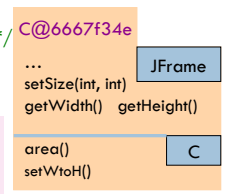
18

```
/** An instance is a JFrame with more methods */
public class C extends javax.swing.JFrame {
   public int area() {
      return getWidth() * getHeight();
   }

   /** Set width of window to its height */
   public void setWtoH() {
      setSize(getHeight(), getHeight());
   }
}
```

Call on procedure setSize

It is a procedure because it has **void** instead of return type



## Using an object of class Date

19

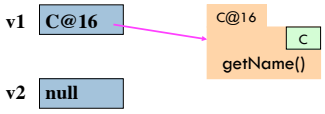
```
/** An instance is a JFrame with more methods */
public class C extends javax.swing.JFrame {
  …
  /** Put the date and time in the title */
  public void setTitleToDate() {
      setTitle((new java.util.Date()).toString());
  }
}
```

An object of class java.util.Date contains the date and time at which it was created.
It has a function toString(), which yields the data as a String.

C@6667f34e
… JFrame
setSize(int, int)
setTitle(String)
area() { } C
setWtoH() setTitleToDate

## About null

20




**null** denotes the absence of a name.

**v2**.getName() is a mistake! Program stops with a NullPointerException

You can write assignments like: v1= **null**;

and expressions like: v1 == **null**

## Hello World!

21

```
/** A simple program that prints Hello, world! */
public class myClass {

    /** Called to start program. */
    public static void main(String[ ] args) {
        System.out.println("Hello, world!");
    }
}
```

args is an array of String elements

We explain **static** next week.
**Briefly**: there is only one copy of procedure main, and it is not in any object