CS/ENGRD 2110
SPRING 2017

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

CMS VideoNote.com, PPT slides, DrJava, Book

CMS available. Visit course webpage, click "Links", then "CMS for 2110". Not enrolled? Ask Jenna Edwards jls478@cornell.edu to enroll you (needs your netid).

Look at http://cornell.videonote.com/channels/583
to see a previous lecture from fall 2015.

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 also requires downloading an old version of Java.

Assignment A2

Get handout from course Piazza, the pinned note A2 FAQs.

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!

Homework HW1

The two short videos and HW1 are working beyond our expectations.

Very few people understood what we wanted when we asked you to tell us how the assignment statement was executed at the end of the last lecture.

By Monday morning at 9:30, over 1/3 of the class had submitted HW1. A sampling of about 25 of them showed that the lesson had been learned and that you were by and large grateful for it. Gries has been trying for years to get this message across; this is the first real success.

Java OO (Object Orientation)

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

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

References to course text and JavaSummary.pptx

Objects: B.1 slide 10-16

Calling methods: B.2-B.3 slide 18

Class definition: B.5 slide 11

public, private: B.5 slide 11, 12

Indirect reference, aliasing: B.6 slide 17

Method declarations: B.7

Parameter vs argument: B.12-B.14
     slide 14

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

Methods may have parameters
Method calls may have arguments

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

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

Evaluation of   JFrame@25c7f37d

   new javax.swing.JFrame()

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

hide()   show()
setTitle(String) getTitle()

getX()   getY()   setLocation(int, int)

getWidth()   getHeight()   setSize(int,int)

…

JFrame

A class variable contains the name of an object

Type JFrame: Names of objects of class JFrame

h is the pointer to the new object

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.

JFrame@25c7f37d

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

JFrame

h JFrame@25c7f37d

JFrame

A class variable contains the name of an object

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

hide()   show()
setTitle(String) getTitle()

getX()   getY()   setLocation(int, int)
getWidth()   getHeight()   setSize(int,int)
…

JFrame

Class definition

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

/** description of what the class is for */

This is a comment

public class C {

    declarations of methods (in any order)

}

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 projectyou write; put all class definitions for program in that directory. You'll see this when we demo.

First class definition

/** 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

k | C@25c7fd38

C

C@25c7fd38

C

Class extends (is a subclass of) JFrame

/** An instance is a subclass of JFrame */   can only extends one class
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

C@6667f34e

JFrame

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

C

Easy re-use of program part!

Class definition with a function definition

```
/** 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

C@6667f34e

...

getWidth()   getHeight()

area()

JFrame

C

Inside-out rule for finding declaration

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

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.

The whole method is in the object

C@6667f34e

getWidth()
getHeight() …

JFrame

C

```
area() {
    return getWidth() * getHeight();
}
```

Inside-out rule for finding declaration

```
/** 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.

C@2abcde14

getWidth()
getHeight() …

JFrame

area() {
  return getWidth() * getHeight();
}

C

C@6667f34e

getWidth()
getHeight() …

JFrame

area() {
  return getWidth() * getHeight();
}

C

Class definition with a procedure definition

/** 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

C@6667f34e

…
setSize(int, int)
getWidth()   getHeight()

JFrame

area()
setWtoH()

C

Using an object of class Date

```
/** 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.
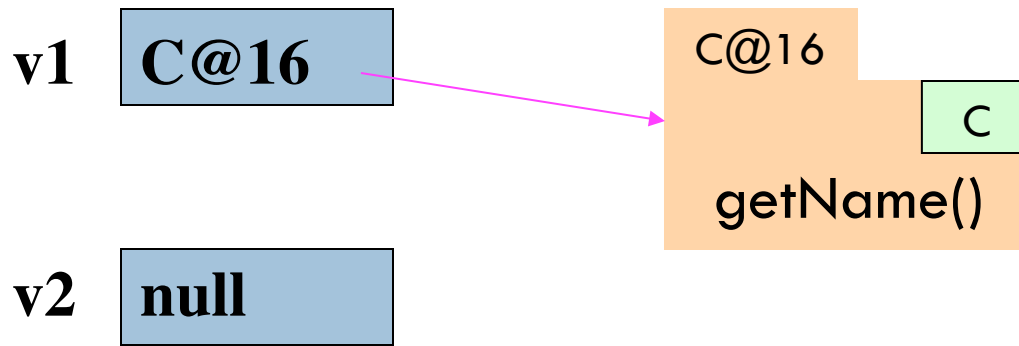
C@6667f34e

…
setSize(int, int)
setTitle(String)

JFrame

area() {   }   C

setWtoH()   setTitleToDate

About null

v1 | **C@16**

C@16

C

getName()

v2 | **null**

null is a key word. So if we type "g = null", then g no
longer contain the pointer, but just a name

CC g = new CC();
CC h;
h = g; // they are pointing to the same object

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!

```
/** 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