Lecture 2: Objects and classes in Java
http://courses.cs.cornell.edu/cs2110
Homework HW1

The answers you handed in at the end of lecture 1 showed mass confusion! Perhaps 80% of you weren’t sure what to write. **This was not graded! It was only to help us and you assess the situation.**

Doing HW1 will eliminate the confusion. Piazza note @30, (it is linked to in the pinned Piazza Recitation/Homework note.)

---------------------------------------------

**Evaluation, Execution, Syntax, Semantics.**

**Presenting an algorithm in English (2.5 minutes).**

**Executing the assignment statement (2.5 minutes).**

Do HW1 and submit on the CMS
CMS VideoNote.com, PPT slides, JavaHyperText.

CMS. Visit course webpage, click “Links”, then “CMS for 2110”.

Videos of lectures from last semester: 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!

Got a Java question? See first if it’s answered on JavaHyperText
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, a blueprint for objects, and how it contains definitions of methods (functions and procedures) that appear in each object of the class.

Third: Talk about keyword null.

Fourth: Introduce Exceptions
Homework

1. Study material of this lecture.
2. Visit JavaHyperText, click on Code Style. 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.
References to JavaHyperText entries

Objects: 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

Fields of an object may be mentioned. We cover these in next lecture

Function: a method that returns a result.
Procedure: method that does not return a result, void method.
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

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

**JFrame@25c7**
- hide()
- show()
- setTitle(String)
- getTitle()
- getX()
- getY()
- setLocation(int, int)
- getWidth()
- getHeight()
- setSize(int, int)
...

**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

\[
\text{new } \text{javax.swing.JFrame}()
\]

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

If evaluation creates this object, value of expression is

\[
\text{JFrame}@25c7
\]

\[
2 + 3 + 4 = 9
\]
A class variable contains the name of an object

Type JFrame: Names of objects of class JFrame

```java
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.
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();

\[
\begin{align*}
x &= y; \\
g &= h; \\
h &= JFrame@25c7
\end{align*}
\]
**Class definition**: Describes format of an object (instance) of the class.

```java
/** 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

/** 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
/** 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!
/** 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();
    }
}

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

Spec, as a comment
Function calls automatically call functions that are in the object
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.
Inside-out rule for finding declaration

```java
/** 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.
/** 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
/** 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.
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
int x = 5;
System.out.println("x is now "+x);
assert x == 6;

When the assert statement is executed and x is not 6, an object of class 
AssertionError is created and “thrown”. It contains info needed to print out a nice message.

java.lang.AssertionError
at A0.main(A0.java:9)
When 5/0 is evaluated, an object of class `ArithmeticException` is created and “thrown”. It contains info needed to print out a nice message.

```java
public static void m() {
    int y = 5/0;
}
```

Exception in thread "main"
java.lang.ArithmeticException: / by zero
at A0.m(A0.java:15)
at A0.main(A0.java:6)
## Intro to Exceptions

You will learn all about exceptions in next week’s recitation!

<table>
<thead>
<tr>
<th>Throwable</th>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>ArithmeticException</td>
</tr>
<tr>
<td></td>
<td>NullPointerException</td>
</tr>
<tr>
<td></td>
<td>IllegalArgumentException</td>
</tr>
</tbody>
</table>

...