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

Java OO (Object Orientation)
Note: Assignment A0 and VideoNote available

- 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
- Methods may have parameters
- Method calls may have arguments

---

**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
  - 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 is a statement to do something

- Evaluation of new-expression creates an object
  - Evaluation of `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
  ```

---

**Evaluation of**

```
new javax.swing.JFrame()
```
A class variable contains the name of an object

Type `JFrame`: Names of objects of class `JFrame`

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

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

Consequence: a class variable contains not an object but the name of an object. Objects are referenced indirectly.

```java
JFrame@25c7f37d
```

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

```java
h.hide();
h.show();
```

```java
setTitle(String)
getTitle()
getX()
getY()
setLocation(int, int)
getWidth()
getHeight()
setSize(int, int)
```

```java
JFrame@25c7f37d
```

**Class definition**

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

**First class definition**

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

Then, execution of

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

creates object shown to right and stores its name in `k`

**Class extends (is a subclass of) JFrame**

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

```java
C@6667f34e
```

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

Easy re-use of program part!

**Class definition with a function definition**

```java
/**
 * 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();
    }
}
```

```java
C@6667f34e
```

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

Function calls automatically call functions that are in the object

Spec, as a comment
Inside-out rule for finding declaration

```java
/** 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.

Class definition with a procedure definition

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

    public void setSize(int, int) {
        setSize(getWidth(), getHeight());
    }
}
```

Using an object of class Date

```java
/** An instance is a JFrame with more methods */
public class C extends javax.swing.JFrame {

    public void setTitleToDate() {
        setTitle(new java.util.Date().toString());
    }
}
```

About null

```java
v1 = C@16;
null
```

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!

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

    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