CS/ENGRD 2110 SPRING 2019

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 70% 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 @22, (find a link to it 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

PPT slides, JavaHyperText.

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

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

Java OO

References to JavaHyperText entries

Objects: object

Calling methods: method call

Class definition: class def

public, private: public private

method

Parameter vs argument:

parameter, argument

Inside-out rule

Methods may have parameters
Method calls may have arguments

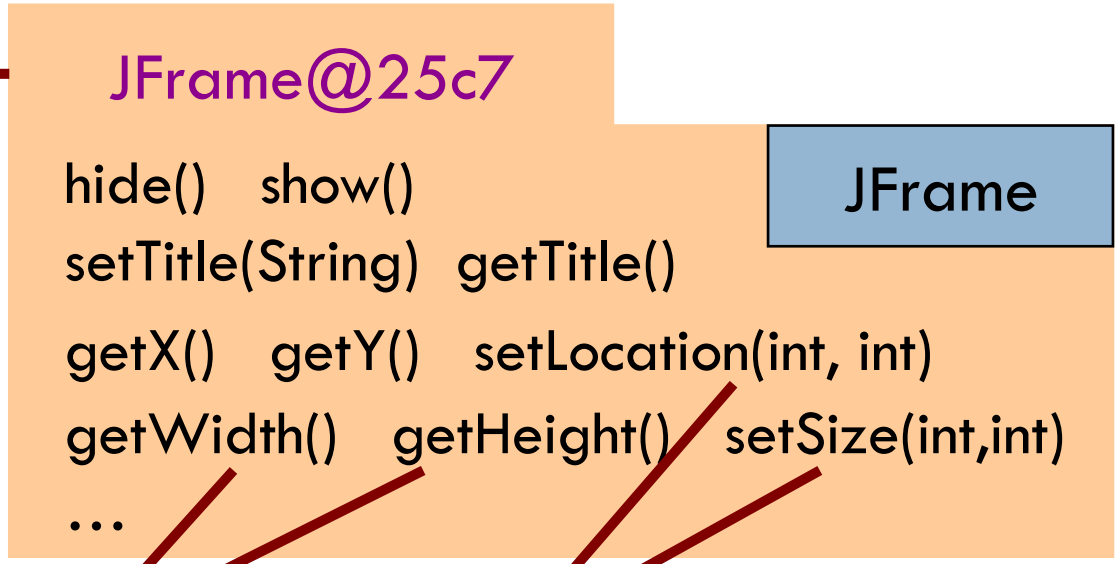
Fields of an object may be mentioned. We cover these in next lecture

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

Drawing an object of class javax.swing.JFrame

This object is associated with a window on your computer monitor

Name of object, giving class name and its memory location (hexadecimal 25c7). 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 on it is a statement

Evaluation of new-expression creates an object

Evaluation of **new** javax.swing.JFrame() JFrame@25c7

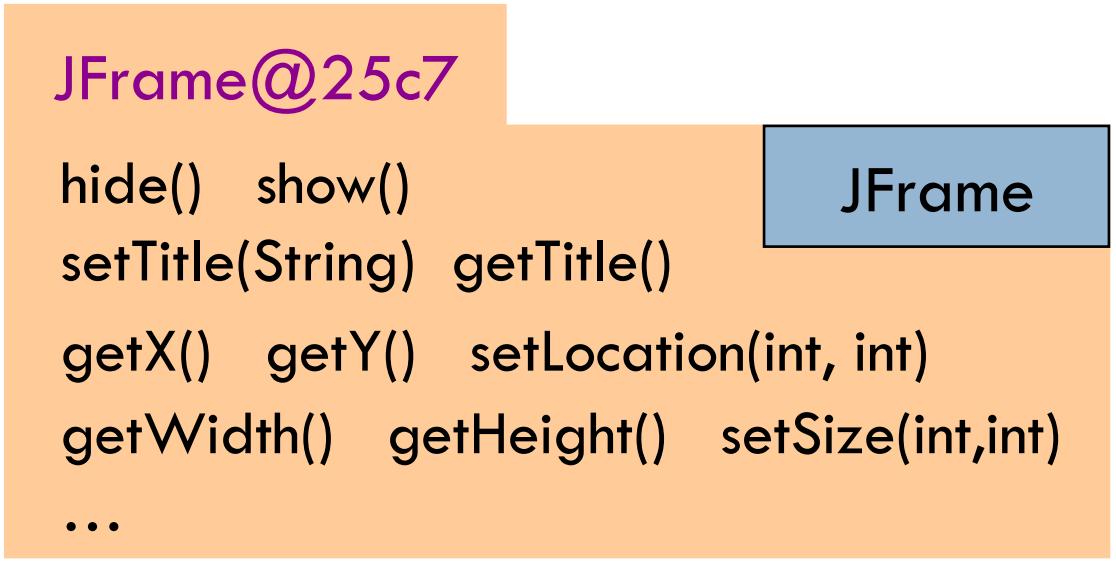
creates an object and gives as its value the name of the object

If evaluation creates this object, value of expression is

JFrame@25c7

9

2 + 3 + 4



A class variable contains the name of an object

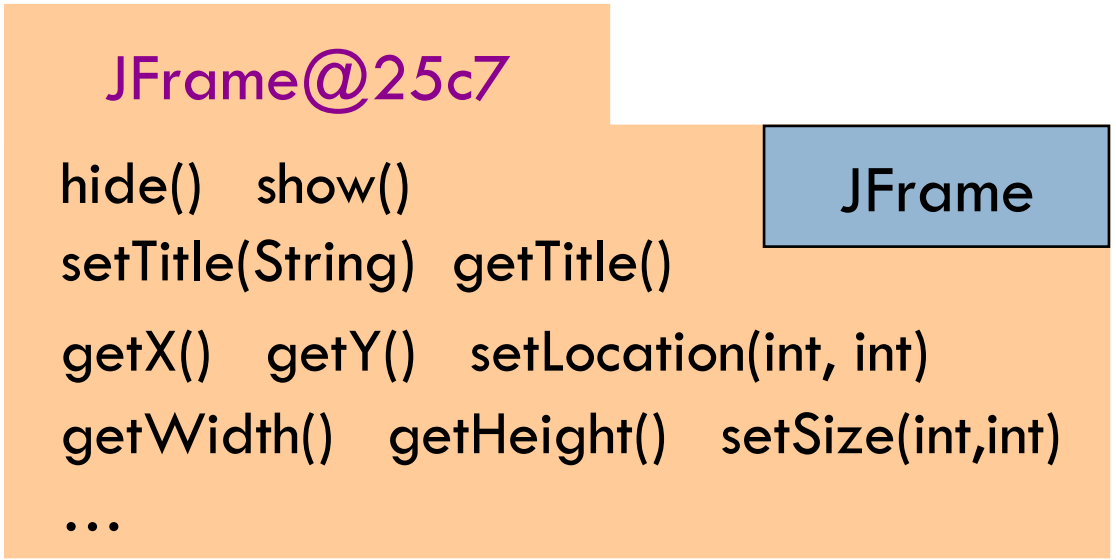
Type JFrame: Names of objects of class JFrame

```
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.

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");

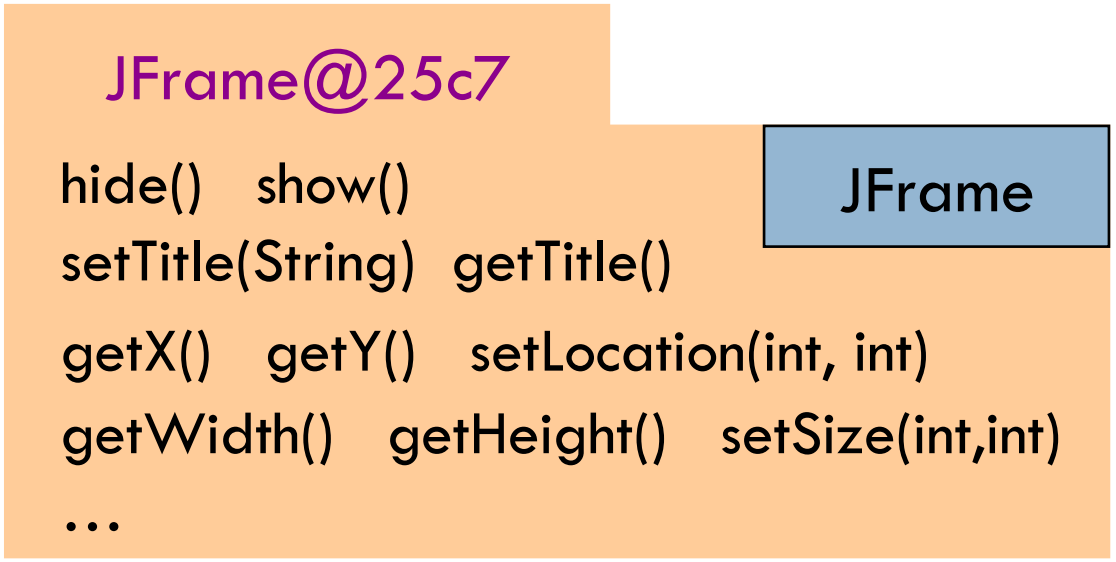
Function calls: h.getX() h.getX() + h.getWidth()

x= y;
g= h;

h | JFrame@25c7 | |

JFrame

DEMO TIME



Class definition: a blueprint for objects of the class

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)
}
```

DEMO TIME

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

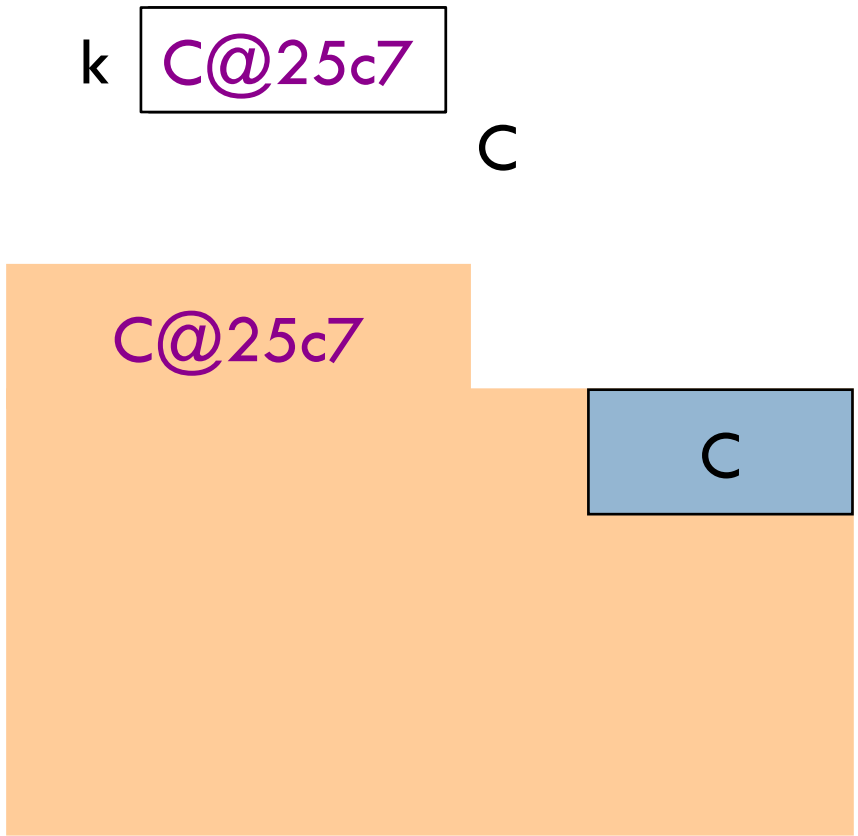

```
/** An instance (object of the class) has (almost) no methods */
public class C {

}
```

Then, execution of

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

creates object shown to right and stores its name in k

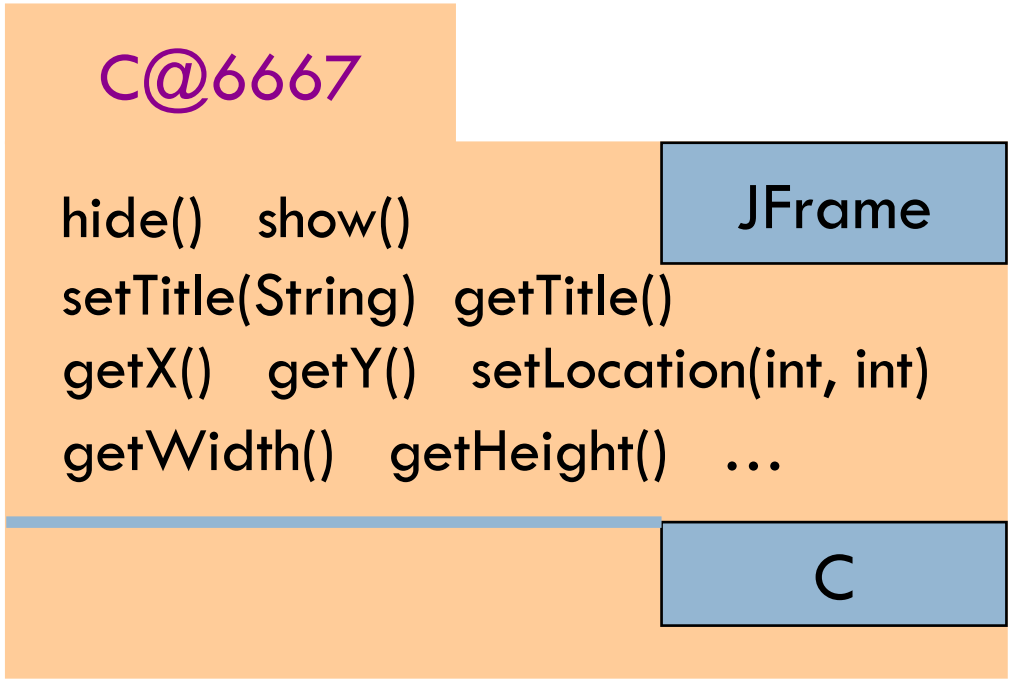


Class extends (is a subclass of) JFrame


```
/** 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!

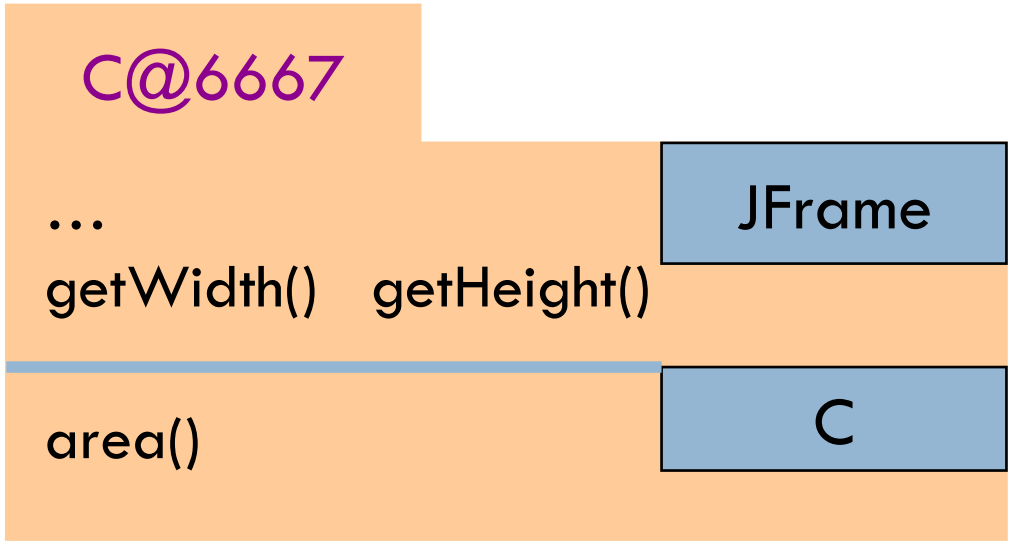
Class definition with a function definition


```
/** 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

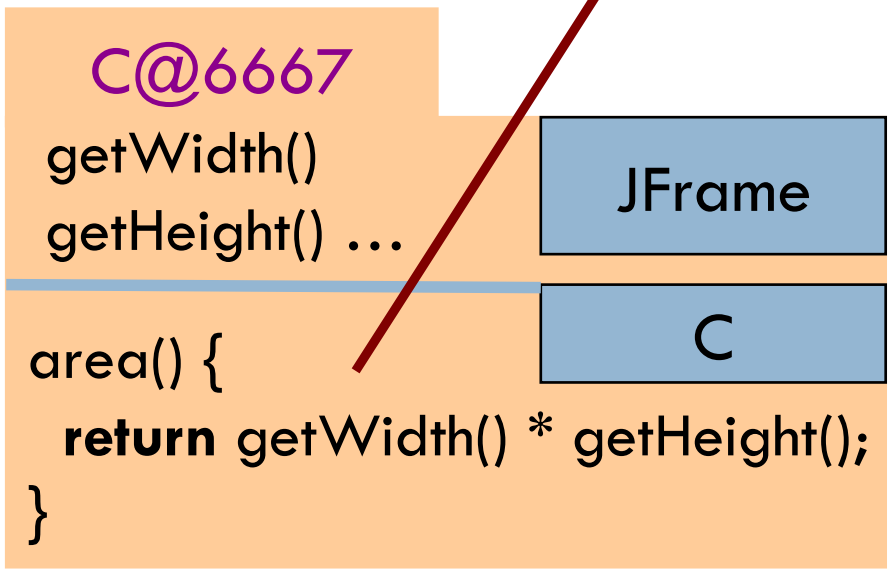


Inside-out rule for finding declaration


```
/** 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.

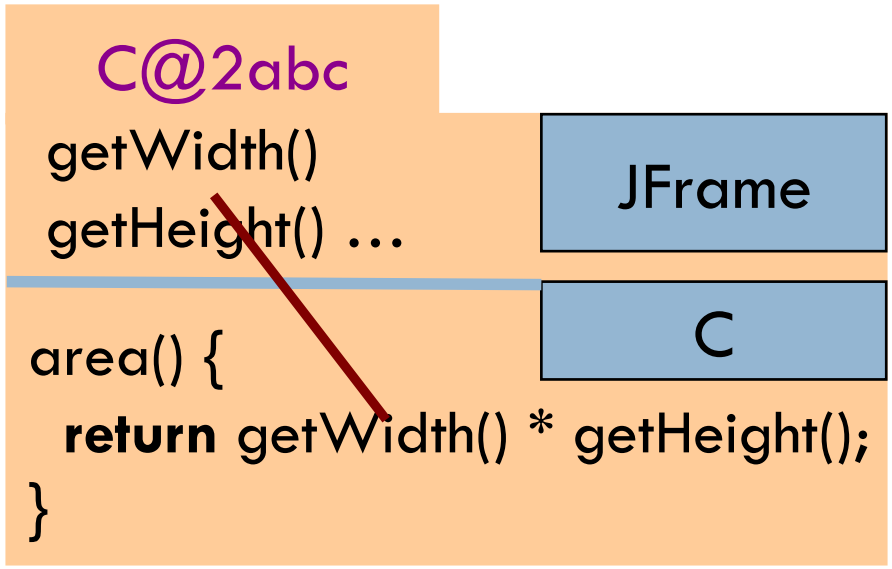
The whole method is in the object

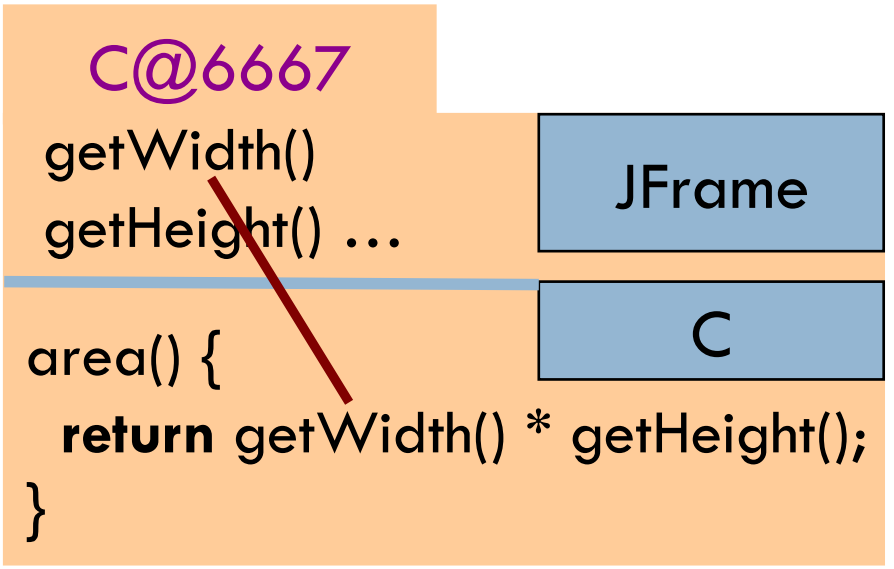


Inside-out rule for finding declaration


```
/** 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.





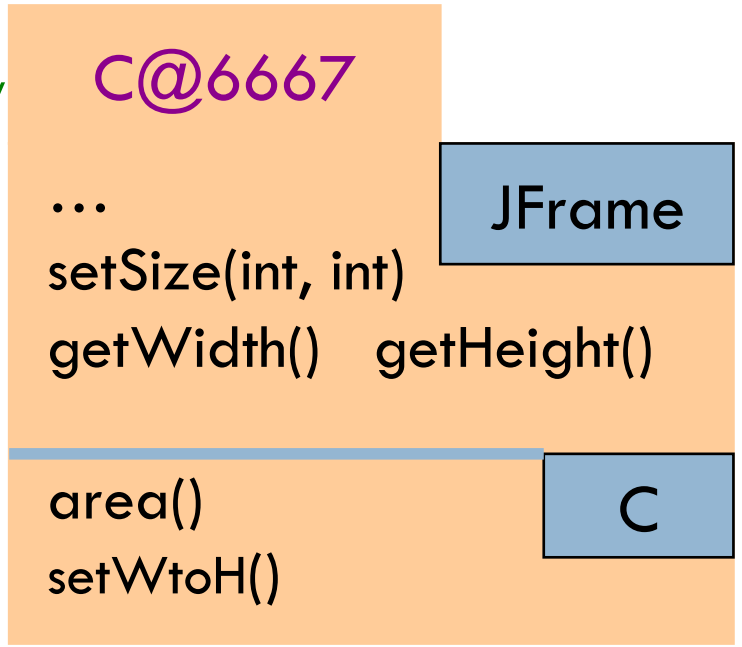
Class definition with a procedure definition


```
/** 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

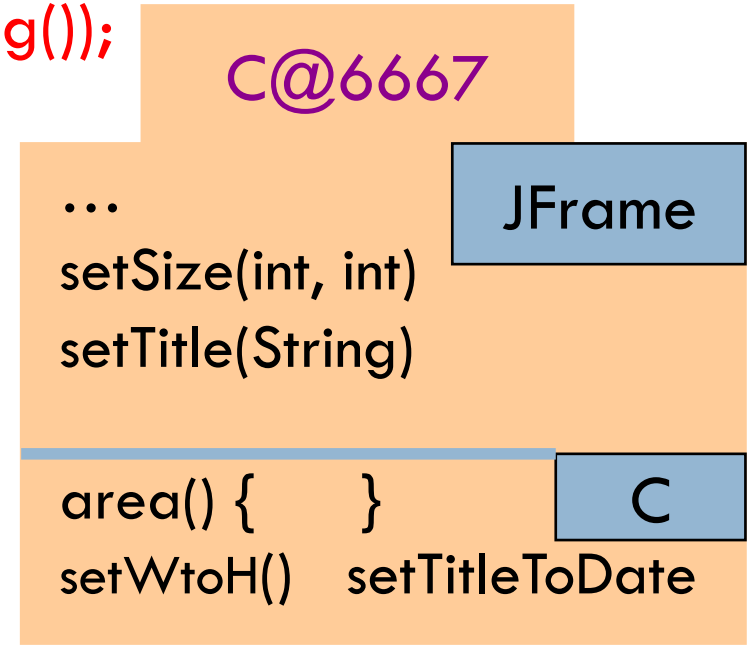
It is a procedure because it has **void** instead of return type



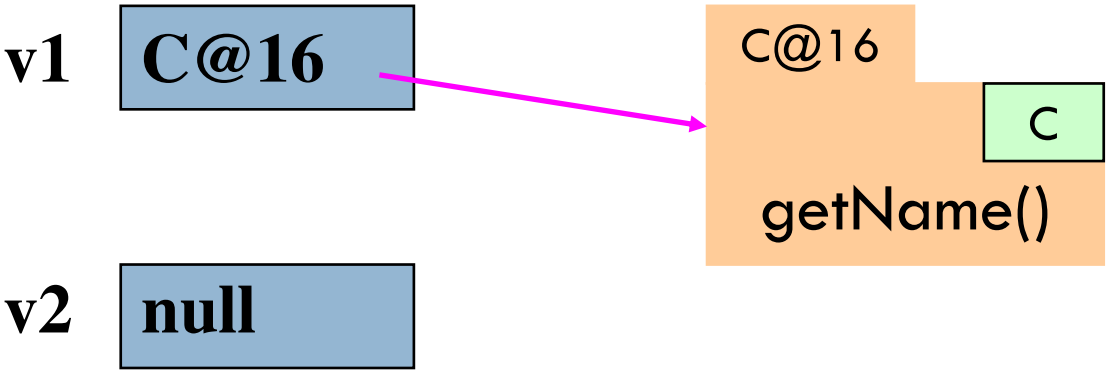
Using an object of class Date


```
/** 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.



About null



null denotes the absence of a name.

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

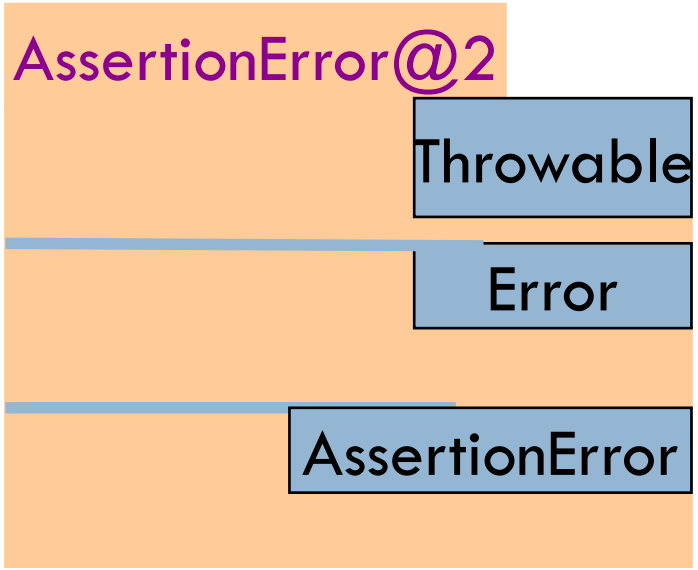
You can write assignments like: v1= **null**;

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

Intro to Exceptions


```
7  int x= 5;
8  System.out.println("x is now "+x);
9  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.



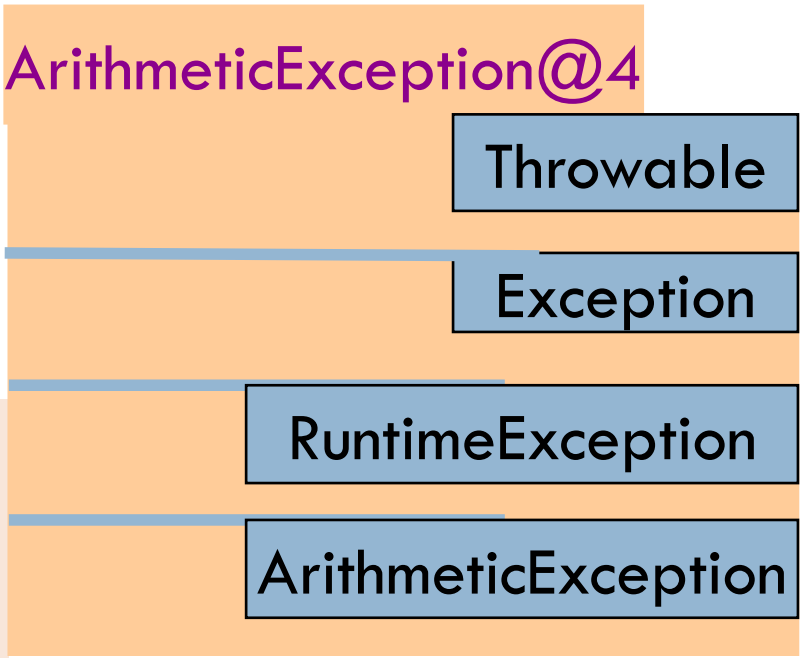

```
java.lang.AssertionError
at A0.main(A0.java:9)
```

Intro to Exceptions


```
06    m();
```

```
14    public static void m() {
15    int y= 5/0;
16    }
```

When 5/0 is evaluated, an object of class ArithmeticException is created and "thrown". It contains info needed to print out a nice message.




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

where it occurred

where m was called

Intro to Exceptions

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

```
Throwable
    Error
        AssertionError
        …
    Exception
        IOException
        RuntimeException
            ArithmeticException
            NullPointerException
            IllegalArgumentException
            …
```