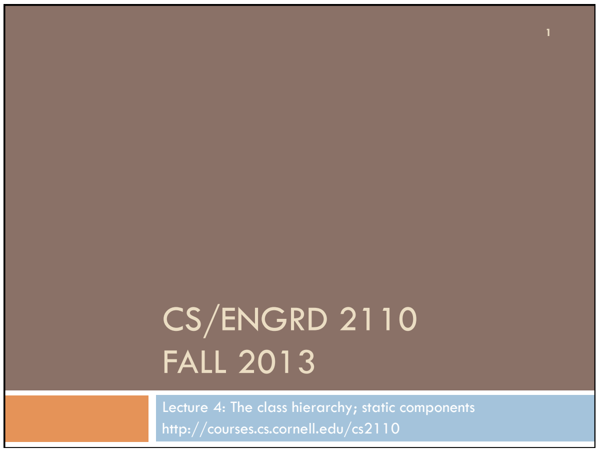
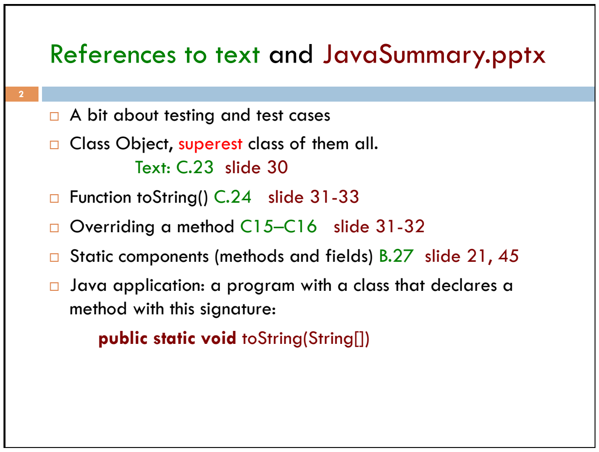


# CS/ENGRD 2110
# FALL 2013

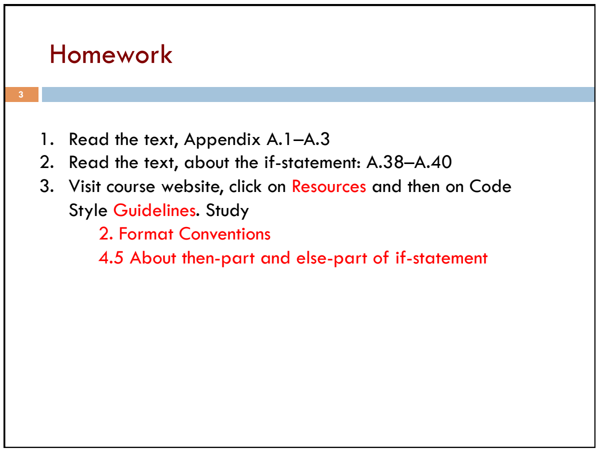
Lecture 4: The class hierarchy; static components
http://courses.cs.cornell.edu/cs2110



## References to text and JavaSummary.pptx

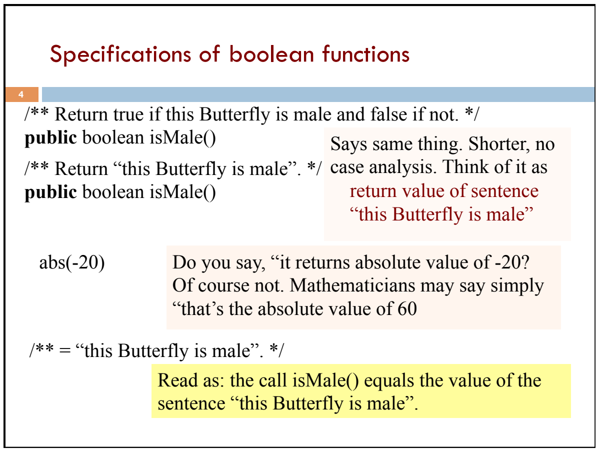
- A bit about testing and test cases
- Class Object, superest class of them all.
  Text: C.23 slide 30
- Function toString() C.24 slide 31-33
- Overriding a method C15–C16 slide 31-32
- Static components (methods and fields) B.27 slide 21, 45
- Java application: a program with a class that declares a method with this signature:

  **public static void** toString(String[])



## Homework

1. Read the text, Appendix A.1–A.3
2. Read the text, about the if-statement: A.38–A.40
3. Visit course website, click on Resources and then on Code Style Guidelines. Study
   2. Format Conventions
   4.5 About then-part and else-part of if-statement



## Specifications of boolean functions

/** Return true if this Butterfly is male and false if not. */
**public** boolean isMale()

/** Return “this Butterfly is male”. */
**public** boolean isMale()

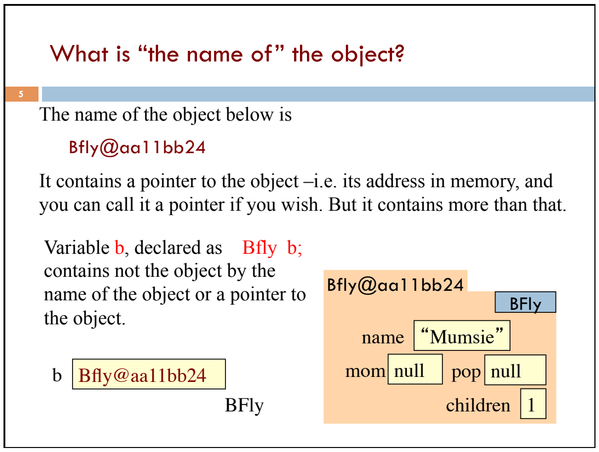
Says same thing. Shorter, no case analysis. Think of it as return value of sentence “this Butterfly is male”

abs(-20)

Do you say, “it returns absolute value of -20? Of course not. Mathematicians may say simply “that’s the absolute value of 60

/** = “this Butterfly is male”. */

Read as: the call isMale() equals the value of the sentence “this Butterfly is male”.



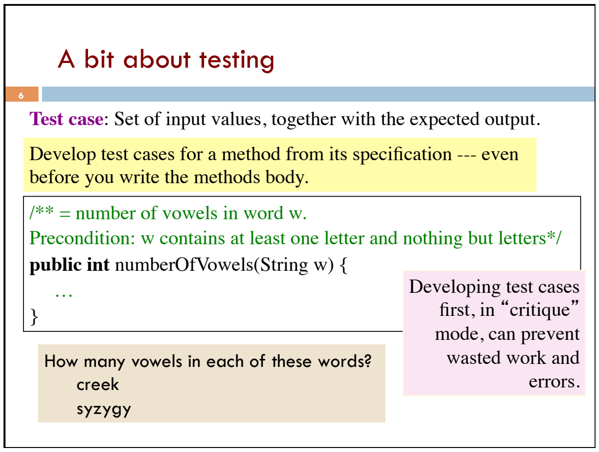
## What is “the name of” the object?

The name of the object below is

Bfly@aa11bb24

It contains a pointer to the object –i.e. its address in memory, and you can call it a pointer if you wish. But it contains more than that.

Variable b, declared as Bfly b; contains not the object by the name of the object or a pointer to the object.



## A bit about testing

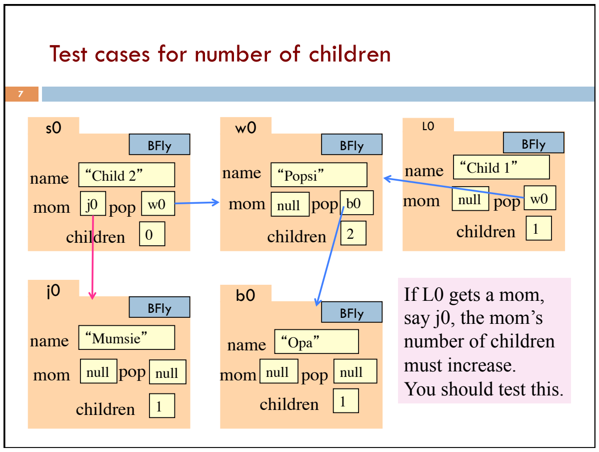
**Test case**: Set of input values, together with the expected output.

Develop test cases for a method from its specification --- even before you write the methods body.

```
/** = number of vowels in word w.
Precondition: w contains at least one letter and nothing but letters*/
public int numberOfVowels(String w) {
    …
}
```

Developing test cases first, in “critique” mode, can prevent wasted work and errors.

How many vowels in each of these words?
creek
syzygy



## Test cases for number of children

s0 BFly
name "Child 2"
mom j0 pop w0
children 0

w0 BFly
name "Popsi"
mom null pop b0
children 2

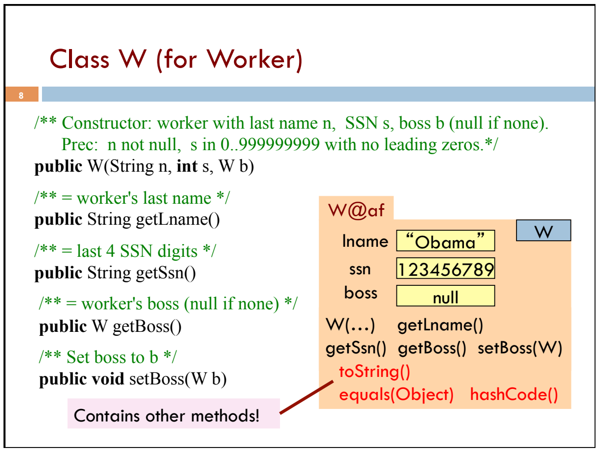
L0 BFly
name "Child 1"
mom null pop w0
children 1

j0 BFly
name "Mumsie"
mom null pop null
children 1

b0 BFly
name "Opa"
mom null pop null
children 1

If L0 gets a mom, say j0, the mom's number of children must increase.
You should test this.

---



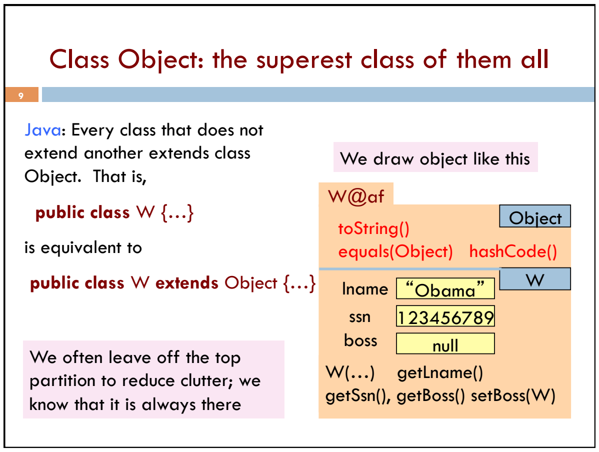
## Class W (for Worker)

```
/** Constructor: worker with last name n,  SSN s, boss b (null if none).
     Prec:  n not null,  s in 0..999999999 with no leading zeros.*/
public W(String n, int s, W b)

/** = worker's last name */
public String getLname()

/** = last 4 SSN digits */
public String getSsn()

 /** = worker's boss (null if none) */
 public W getBoss()

 /** Set boss to b */
 public void setBoss(W b)
```

W@af W
lname "Obama"
ssn 123456789
boss null
W(…) getLname()
getSsn() getBoss() setBoss(W)
toString()
equals(Object) hashCode()

Contains other methods!

---



## Class Object: the superest class of them all

Java: Every class that does not extend another extends class Object. That is,

**public class** W {…}

is equivalent to

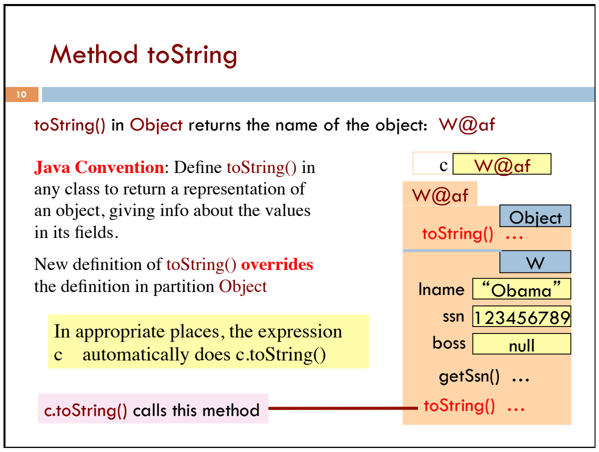
**public class** W **extends** Object {…}

We draw object like this

W@af Object
toString()
equals(Object) hashCode()
W
lname "Obama"
ssn 123456789
boss null
W(…) getLname()
getSsn(), getBoss() setBoss(W)

We often leave off the top partition to reduce clutter; we know that it is always there

---



## Method toString

toString() in Object returns the name of the object: W@af

**Java Convention**: Define toString() in any class to return a representation of an object, giving info about the values in its fields.

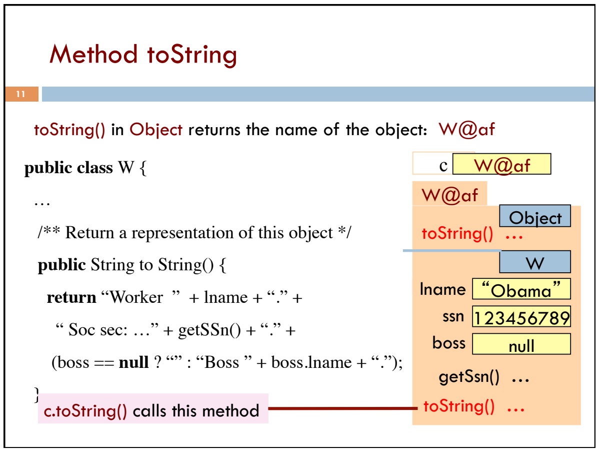
New definition of toString() **overrides** the definition in partition Object

In appropriate places, the expression c automatically does c.toString()

c.toString() calls this method

c W@af
W@af Object
toString() …
W
lname "Obama"
ssn 123456789
boss null
getSsn() …
toString() …

---



## Method toString

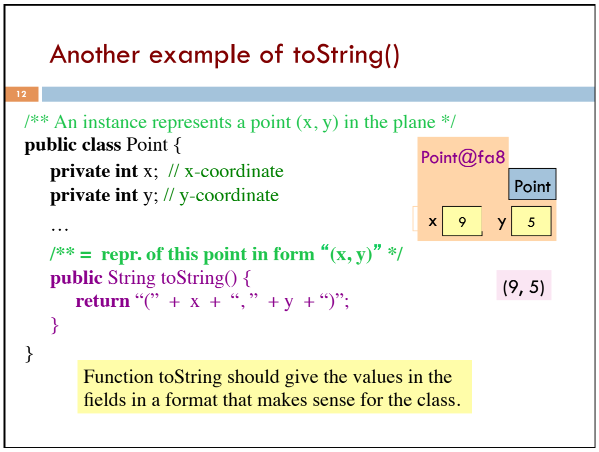
toString() in Object returns the name of the object: W@af

```
public class W {
  …
 /** Return a representation of this object */
 public String to String() {
   return "Worker  "  + lname + "." +
     " Soc sec: …" + getSSn() + "." +
    (boss == null ? "" : "Boss " + boss.lname + ".");
}
```

c.toString() calls this method

c W@af
W@af Object
toString() …
W
lname "Obama"
ssn 123456789
boss null
getSsn() …
toString() …

---



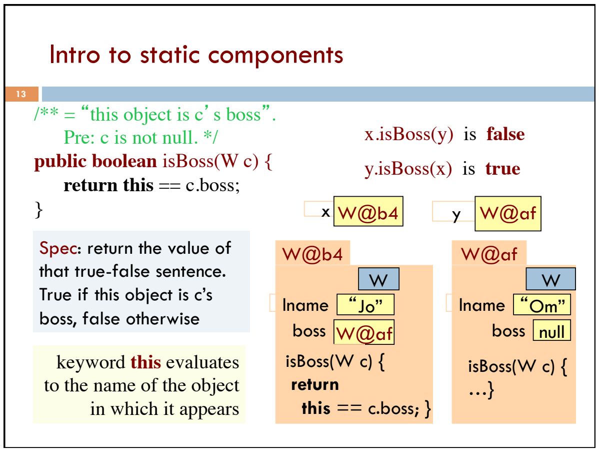
## Another example of toString()

```
/** An instance represents a point (x, y) in the plane */
public class Point {
    private int x;  // x-coordinate
    private int y; // y-coordinate
    …
    /** =  repr. of this point in form "(x, y)" */
    public String toString() {
        return "(" + x + "," + y + ")";
    }
}
```

Point@fa8 Point
x 9 y 5

(9, 5)

Function toString should give the values in the fields in a format that makes sense for the class.



## Intro to static components

```
/** = "this object is c' s boss".
    Pre: c is not null. */
public boolean isBoss(W c) {
    return this == c.boss;
}
```

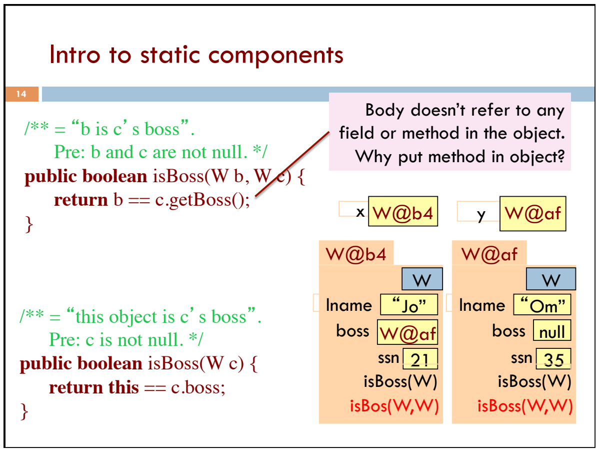
x.isBoss(y) is **false**

y.isBoss(x) is **true**

Spec: return the value of that true-false sentence. True if this object is c's boss, false otherwise

keyword **this** evaluates to the name of the object in which it appears

x W@b4
y W@af
W@b4
W
lname "Jo"
boss W@af
isBoss(W c) {
return
this == c.boss; }
W@af
W
lname "Om"
boss null
isBoss(W c) {
…}



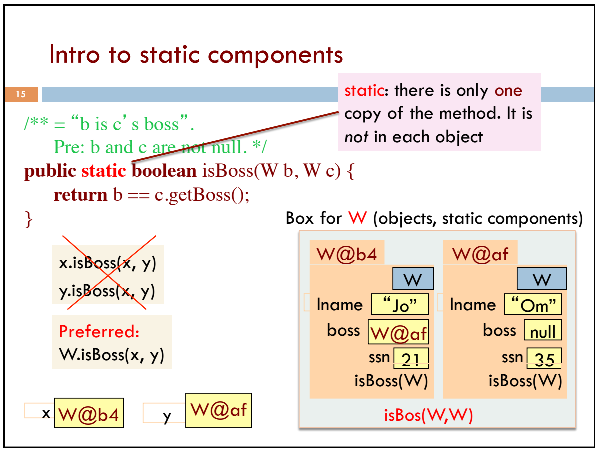
## Intro to static components

Body doesn't refer to any field or method in the object. Why put method in object?

```
/** = "b is c' s boss".
    Pre: b and c are not null. */
public boolean isBoss(W b, W c) {
    return b == c.getBoss();
}
```

```
/** = "this object is c' s boss".
    Pre: c is not null. */
public boolean isBoss(W c) {
    return this == c.boss;
}
```

x W@b4
y W@af
W@b4
W
lname "Jo"
boss W@af
ssn 21
isBoss(W)
isBos(W,W)
W@af
W
lname "Om"
boss null
ssn 35
isBoss(W)
isBoss(W,W)



## Intro to static components

static: there is only one copy of the method. It is *not* in each object

```
/** = "b is c' s boss".
    Pre: b and c are not null. */
public static boolean isBoss(W b, W c) {
    return b == c.getBoss();
}
```

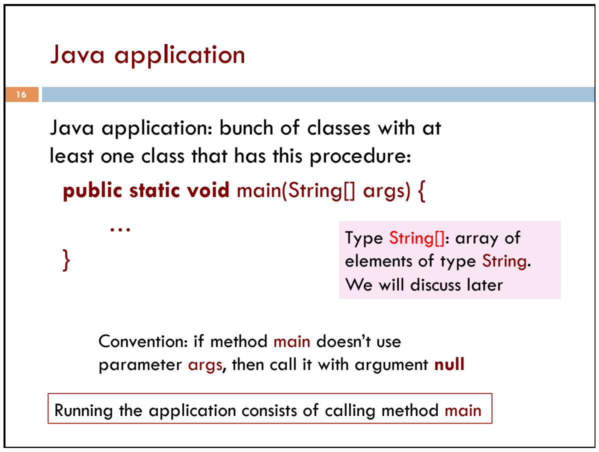
~~x.isBoss(x, y)~~

~~y.isBoss(x, y)~~

Preferred:
W.isBoss(x, y)

Box for W (objects, static components)

x W@b4
y W@af
W@b4
W
lname "Jo"
boss W@af
ssn 21
isBoss(W)
W@af
W
lname "Om"
boss null
ssn 35
isBoss(W)
isBos(W,W)



## Java application

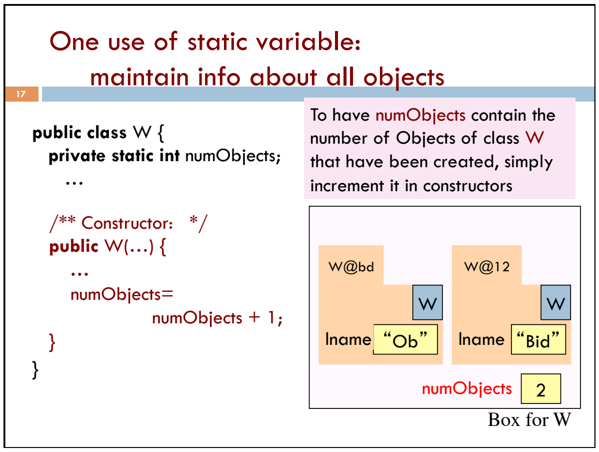
Java application: bunch of classes with at least one class that has this procedure:

```
public static void main(String[] args) {
    …
}
```

Type String[]: array of elements of type String. We will discuss later

Convention: if method main doesn't use parameter args, then call it with argument **null**

Running the application consists of calling method main



## One use of static variable: maintain info about all objects

```
public class W {
  private static int numObjects;
   …

  /** Constructor:   */
  public W(…) {
     …
     numObjects=
               numObjects + 1;
  }
}
```

To have numObjects contain the number of Objects of class W that have been created, simply increment it in constructors

W@bd
W
lname "Ob"
W@12
W
lname "Bid"
numObjects 2
Box for W