CS/ENGRD 2110 SPRING 2014

Lecture 4: The class hierarchy; static components

http://courses.cs.cornell.edu/cs2110

YOU, ROBOT:

MACHINE AUTONOMY IN THE COMPUTER AGE

In honor of the 50th anniversary of the founding of Cornell's Department of Computer Science

The Day the Earth Stood Still (1951)
directed by Robert Wise
introduced by Prof. Charles Van Loan (CS)
Sept 4

2001: A Space Odyssey (1968)
directed by Stanley Kubrick
Sept 12, 14

Robocop (1987)
directed by Paul Verhoeven
introduced by Prof. Walker White (CS) Oct 2
Oct 2, 3

Metropolis (1927) w/live music by the Alloy Orchestra
directed by Fritz Lang
introduced by Prof. Ross Knepper (CS)
Nov 7
$14 general/$12 students & seniors

Metropolis (1927) w/recorded soundtrack
directed by Fritz Lang
Nov 9

Ghost in the Shell (1996)
directed by Mamoru Oshii
Oct 16, 17, 19

Robot & Frank (2012)
directed by Jake Schreier
introduced by Prof. Ashutosh Axena (CS)
Nov 13

Cosponsored with the Department of Computer Science

cinema.cornell.edu
in the historic willard straight theatre

cornell cinema

First show is tonight at 7pm!

Klaatu barada nikto

CUAPPDEV

We are recruiting developers for the Fall semester.

We will also be offering a training program for interested, but inexperienced developers that will teach Swift and iOS8.

Info Sessions:

Sep 4

Sep 8

Olin Hall
Room 155
5:00PM

Website: CUAPPDEV.ORG

Email: INFO@CUAPPDEV.ORG

Announcements

- **A0** will be graded soon —everyone who submitted it gets full credit.
  - It was simple enough that there is no need for us to check anything.
- **A1** is due Saturday night.
  - We will try to get more consultants to be available Saturday
  - Check the schedule on the course website.
  - **Groups:** If you are going to form a group/team of two people, do it BEFORE you submit.
    - Both members must do something: one invites and the other accepts. Thereafter, only ONE member has to submit the files.
- **A2: Practice with Strings**
  - Assignment available now on web + CMS
  - Due on CMS by Friday, 12 September.

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 Bee is male and false if not. */
public boolean isMale()

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

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

abs(-20)

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

/** = “this Bee is male”. */

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

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) {
    …
}
```

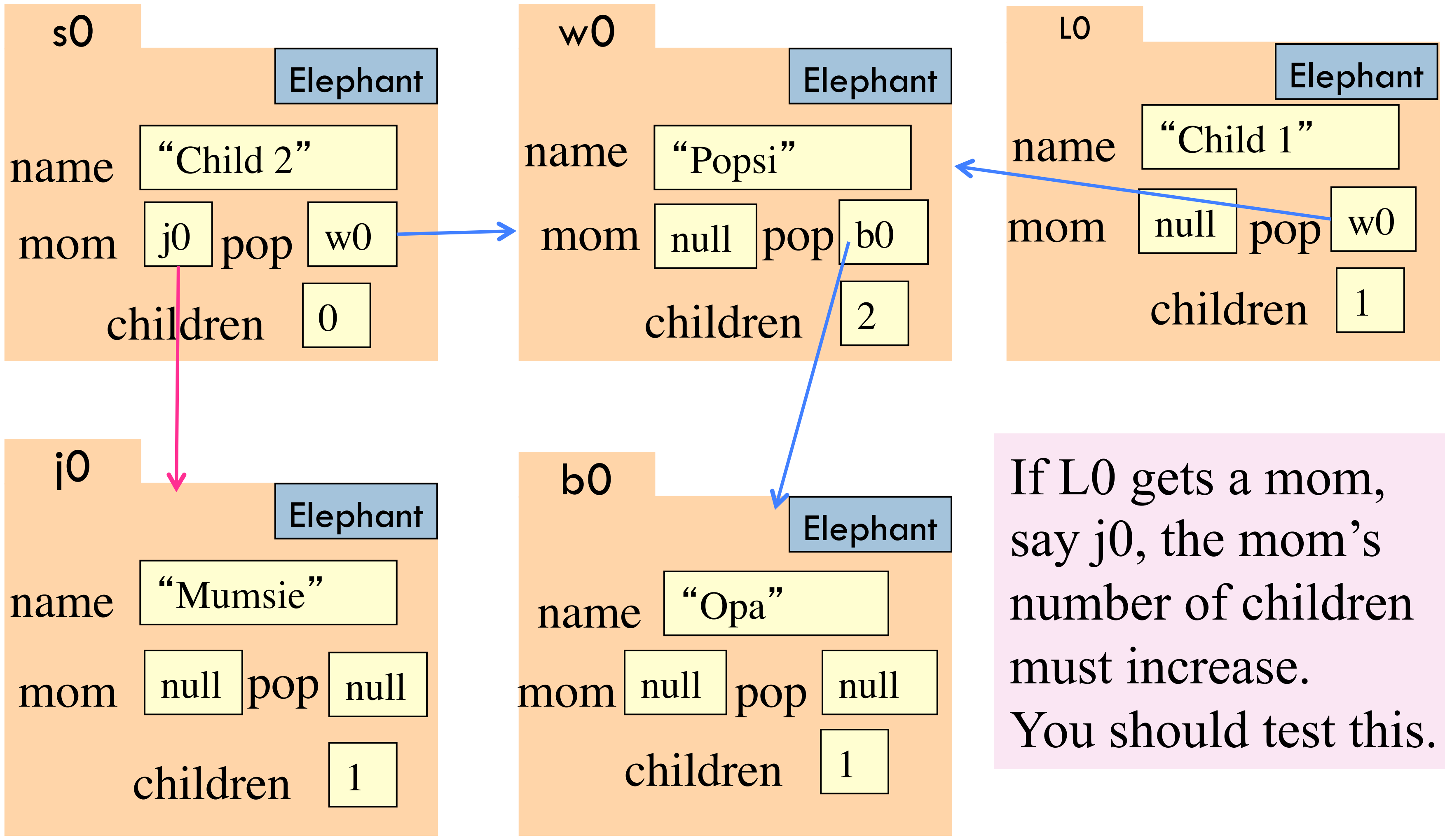
Developing test cases first, in "critique" mode, can prevent wasted work and errors.

How many vowels in each of these words?

creek

syzygy

Test cases for number of children



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

Class W (for Worker)

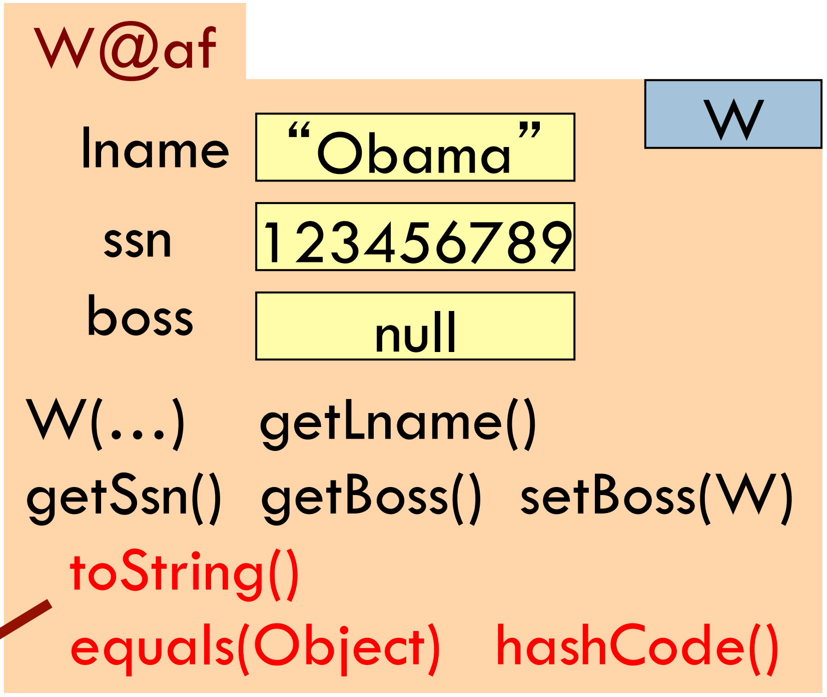
```
/** Constructor: worker with last name n,  SSN s, boss b (null if none).
    Prec:  n not null,  s in 0..999999999 with no leading zeros.*/
public W(String n, int s, W b)

/** = 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)
```



Contains other methods!

Class Object: the superest class of them all

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

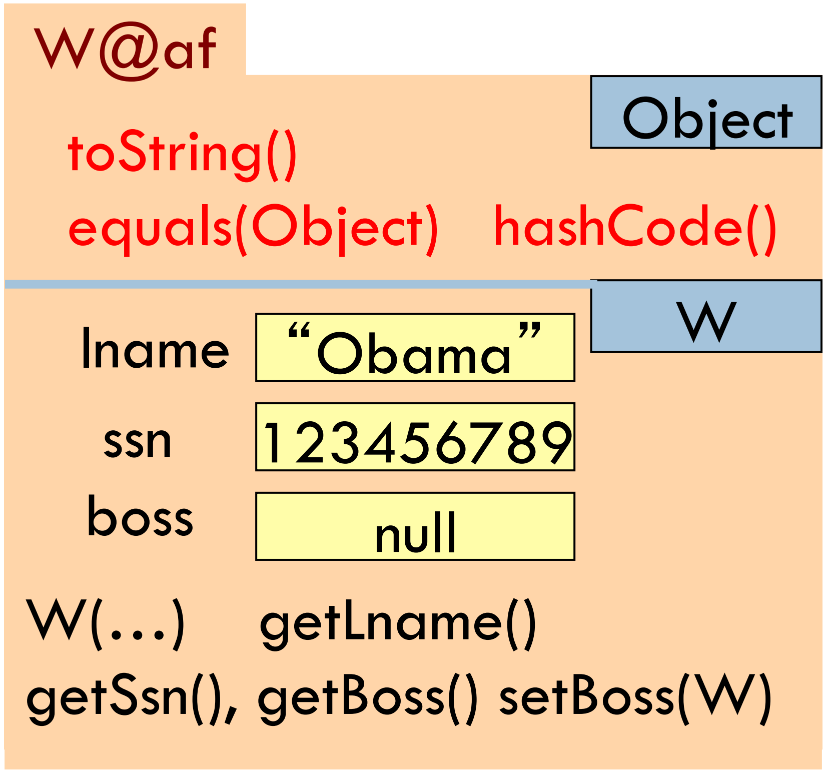
public class W {…}

is equivalent to

public class W **extends** Object {…}

We often leave off this to reduce clutter; we know that it is effectively always there.

We draw object like this



What is "the name of" the object?

The name of the object below is

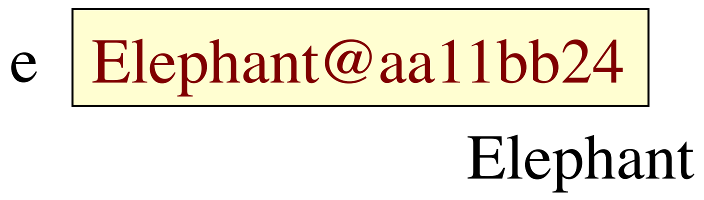
Elephant@aa11bb24

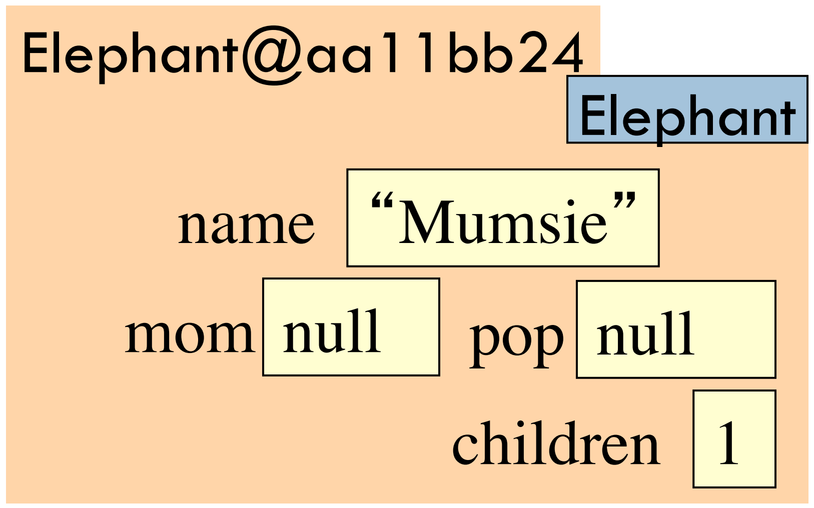
It contains a pointer to the object –i.e. its address in memory, and you can call it a pointer if you wish. But it contains more than that.

Variable e, declared as

Elephant e;

contains not the object but the name of the object (or a pointer to the object).





Method toString

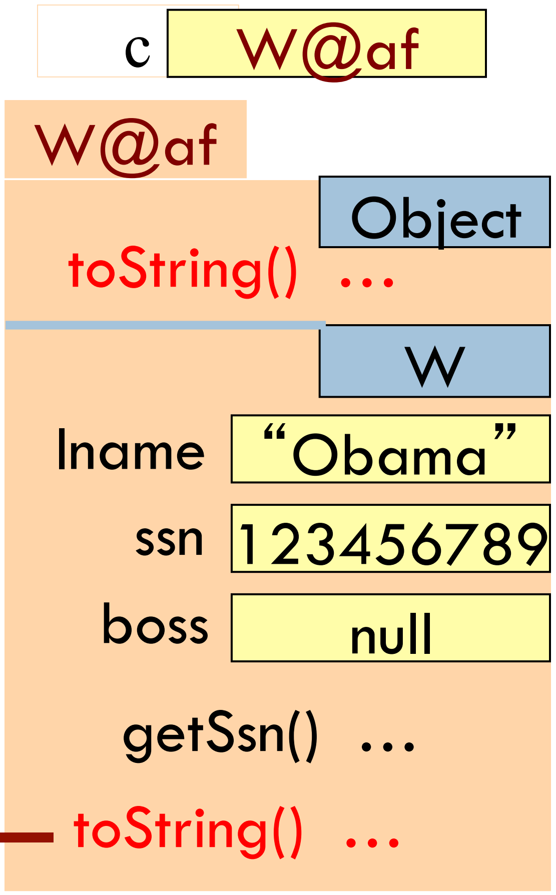
toString() in Object returns the name of the object: W@af

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

New definitions of toString() **override** the definition in Object.toString()

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

c.toString() calls this method

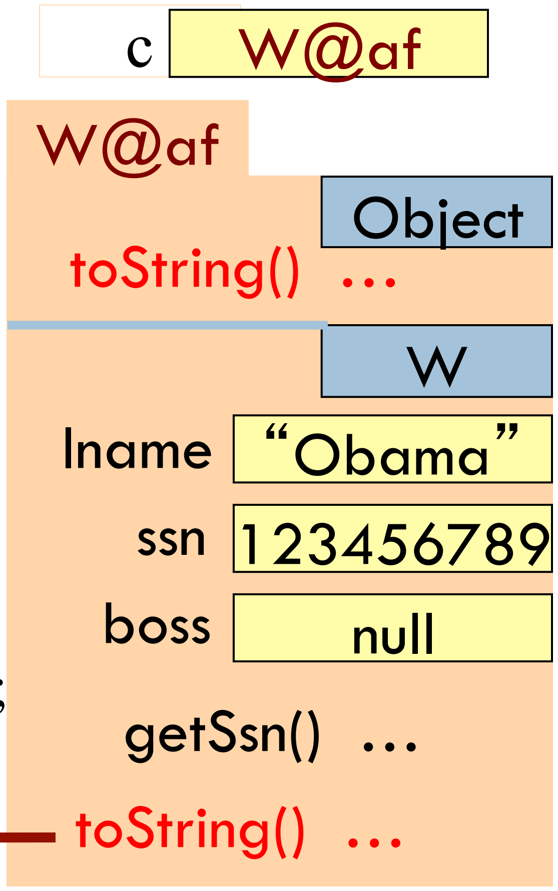


Method toString

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

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

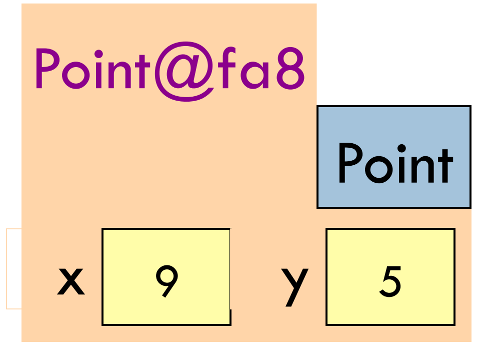
c.toString() calls this method



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  + “)”;
    }
}
```



(9, 5)

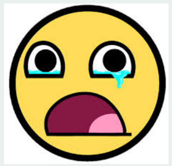
Function toString should give the values in the fields in a format that makes sense for the class.

What about `this`

- **`this`** keyword
- Let's an object instance access its own object reference
- Example: Referencing a shadowed class field

```
public class Point {
    public int x = 0;
    public int y = 0;

    //constructor
    public Point(int x, int y) {
         x = x;
         y = y;
    }
}
```



```
public class Point {
    public int x = 0;
    public int y = 0;

    //constructor
    public Point(int x, int y) {
         this.x = x;
         this.y = y;
    }
}
```

Intro to static components

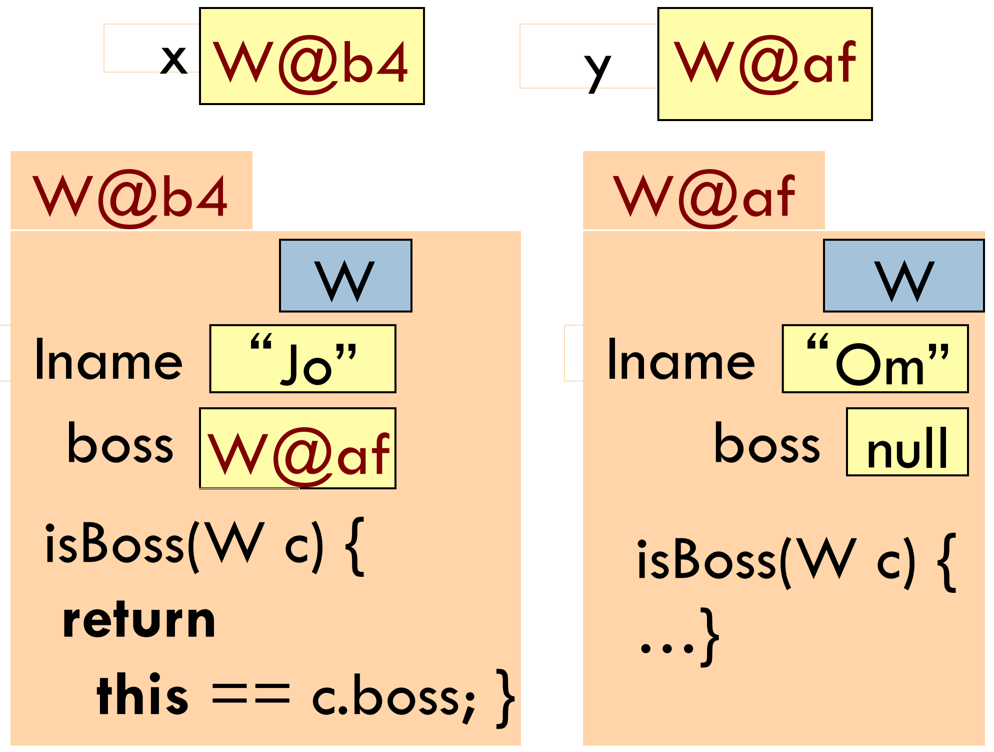
```
/** = “this object is c’s boss”.
    Pre: c is not null. */
public boolean isBoss(W c) {
    return this == c.boss;
}
```

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

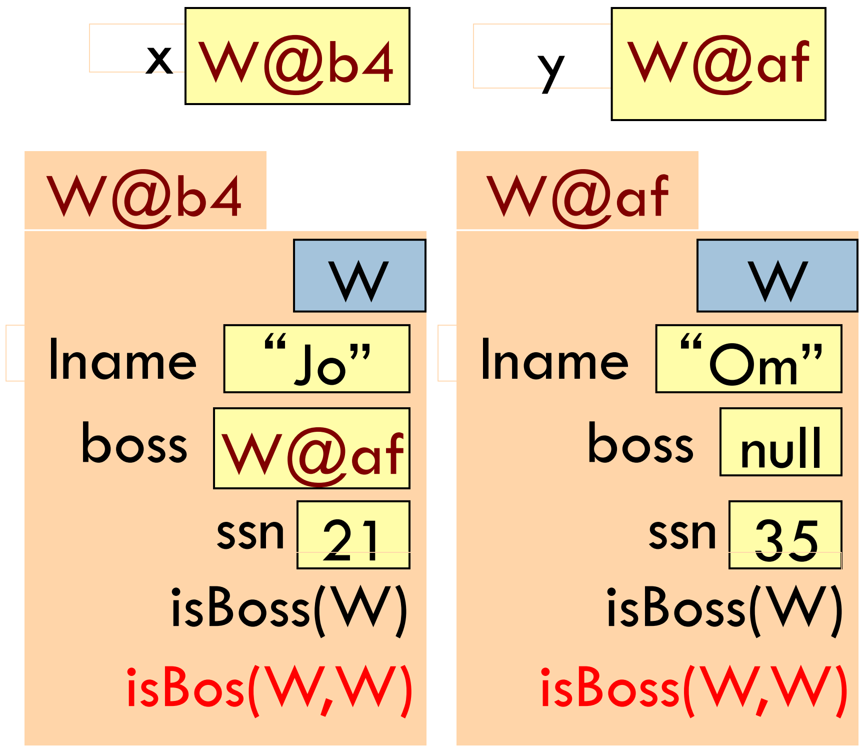
keyword **this** evaluates to the name of the object in which it appears



Intro to static components

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

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



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

Intro to static components

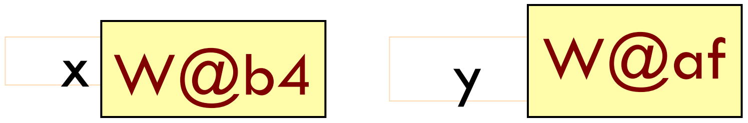
static: there is only one copy of the method. It is *not* in each object

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

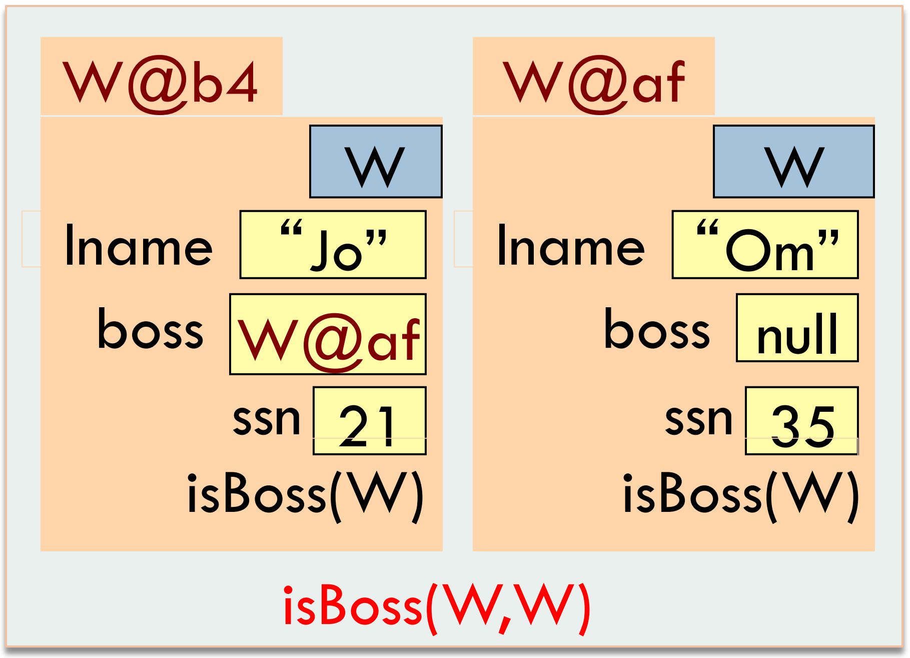
~~x.isBoss(x, y)~~

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

Preferred:
W.isBoss(x, y)



Box for W (objects, static components)



Good example of static methods

- java.lang.Math
  - http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

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

Running the application effectively calls the method main

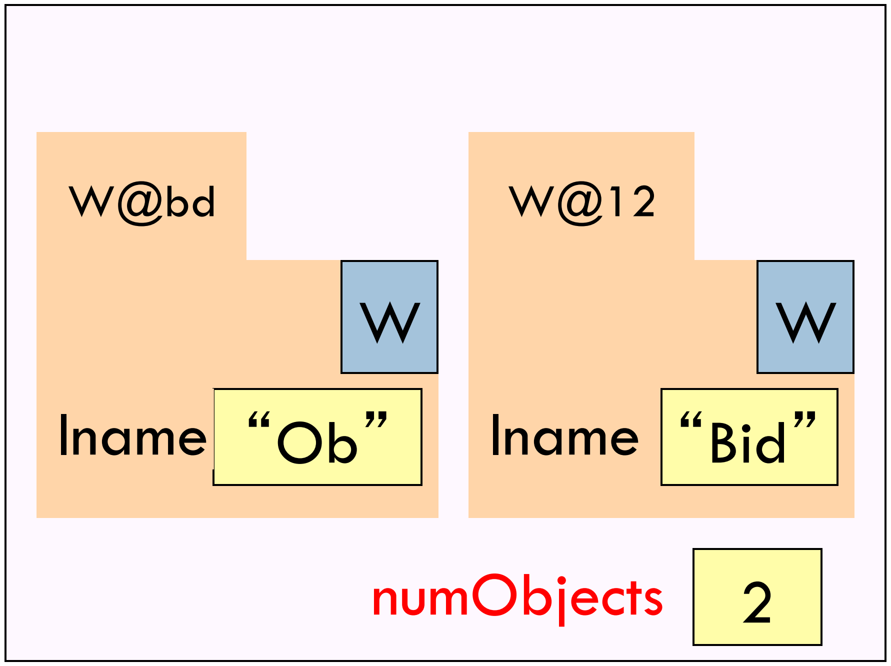
Command line arguments can be entered with args

Uses of static variables: Maintaining info about created 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.



Box for W

Uses of static variables: Implementing the Singleton pattern

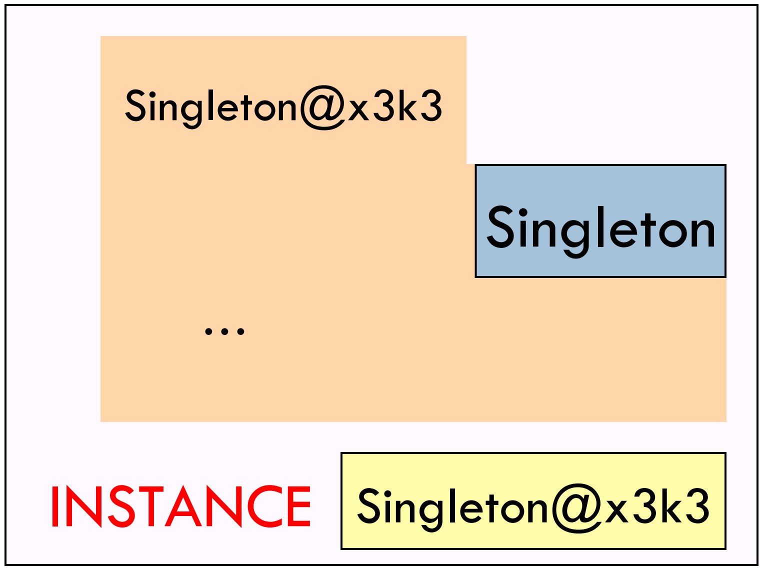
Only one Singleton can ever exist.

```
public class Singleton {
    private static final Singleton INSTANCE = new Singleton();

    private Singleton() { }  // ... constructor

    public static Singleton getInstance() {
        return INSTANCE;
    }

    // ... methods
}
```



Box for Singleton

Example of class hierarchy: Minecraft

- MCP: Minecraft coder pack (http://mcp.ocean-labs.de)
- Warning: Decompiled code with no comments ☹