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

```java
public static void toString(String[])
```
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”.
What is “the name of” the object?

The name of the object below is

\[
\text{Bee@aa11bb24}
\]

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 \( \text{Bee \ b} \); contains not the object but 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.

```java
/** = 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!
Java: Every class that does not extend another extends class Object. That is,

```java
public class W {...}
```

is equivalent to

```java
public class W extends Object {...}
```

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
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 \(\text{(}x, y\text{)}\) */
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
}

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;
}

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
/** = “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;
}
/** = “b is c’s boss”. Pre: b and c are not null. */

public static boolean isBoss(W b, W c) {
    return b == c.getBoss();
}

Box for W (objects, static components)

Preferred: W.isBoss(x, y)
Java application

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

```java
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

```java
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`