

CS1110 15 Sept 2009

Testing; the class Object; toString; static fields and methods

For today's quiz: Keep out a blank sheet of paper with your name and netID on it. Put your last name in ALL CAPS but your first name with regular capitalization, like these examples:

"David GRIES, djg17"

"Simon PEYTON JONES, sep26" (a last name with a space)

Reading for this lecture: Testing with JUnit (Appendix I.2.4 & pp. 385–388), the class Object (pp. 153–154), function toString (pp. 112–113), static variables and methods (Sec. 1.5, p. 47).

Reading for next two lectures: Executing method calls, if-statements, the return statement in a function, local variables. Section 2 except 2.3.8.

Note: this will some clarify some concepts, such as method parameters, that we've had to gloss over so far.

Assignment 1 due Friday on CMS.

Help available at office and consulting hours; see "Staff" webpage.

Organizing and streamlining the testing process

Testing: Process of analyzing, running program, looking for bugs (errors).

Test case: A set of input values, together with the expected output.

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

```

/** = number of vowels in word w.
Precondition: w contains >0 letters and no blanks or punctuation. */
public int numberOfVowels(String w) {
    // (nothing here yet!)
}

```

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

```

/** An instance is a worker in a certain organization */
public class Worker {

    private String name; /* Last name (null if unknown/none,
                        * o.w. at least one character) */
    private int ssn; /* Social security #: in 0..999999999
    private Worker boss; /* Immediate boss (null if none)

    /** Constructor: a worker with
     * last name n (null if unknown/none), SSN s, and boss b (null if none).
     * Precondition: if n is not null, it has at least one character.
     * Precondition: s in 0..999999999 with no leading zeros.
     * so SSN 012:34:5678 should be given as 12345678.*/
    public Worker(String n, int s, Worker b) {
        name= n;
        ssn= s;
        boss= b;
    }
    [etc.]
}

```

A testMethod for getBoss()

= the boss of this Worker (null if none)

File->new JUnit test case ... [save in same directory as Worker.java] imports junit.framework.TestCase, with key methods.

/** Test getBoss [we assume the constructor's been tested]*/

public void testConstructor() {

Worker w1= new Worker("Obama", 123456789, null);

assertEquals(null, w1.getBoss());

Worker w2= new Worker("Biden", 2, w1);

assertEquals(w1, w2.getBoss());

}

assertEquals(x,y): test whether x equals y ; print an error message and stop the method if they are not equal.

x: expected value,

y: actual value.

See page 488 for some other methods.

Every time you click button **Test in DrJava**, this method (and all other testX methods) will be called.

Class Object: The superest class of them all

A minor mystery: since Worker doesn't extend anything, it seems that it should only have the methods we wrote for it. But it has some other methods, too.

Solution: Every class that does not extend another one automatically extends class Object. That is,

```
public class C { ... }
```

is equivalent to

```
public class C extends Object { ... }
```

Class Object: The superest class of them all

So this... is really this.

Because it is **always** there, to avoid clutter, we don't generally draw the partition for superclass Object.

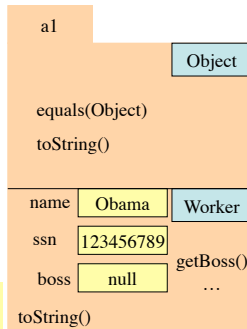
Method toString()

Convention: c.toString() returns a representation of folder c, giving info about the values in its fields.

Put following method in Worker.

```
/** = representation of this Worker
 * [etc., see full program] */
public String toString() {
    return name + ", XXX-XX-" +
        ssn4 + ", Boss:" + boss;
}
```

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



7

Another example of toString()

/** An instance represents a point (x, y) in the plane */

public class Point {

private int x; // the x-coordinate

private int y; // the y-coordinate

/** Constructor: An instance for point (xx, yy) */

public Point(int xx, int yy) {

}

/** = a representation of this point in form "(x, y)" */

public String toString() {

return ;

}

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

Getter and setter methods are not given on this slide

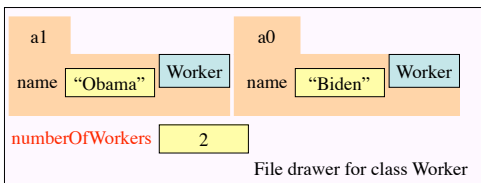
Fill these in

Example: "(3, 5)"

8

A **static field** appears not in each folder, but as a *single entity in the file drawer*. It can be used to maintain information about multiple folders.

```
...
private String name; //last name of this Worker (null if unknown/none)
private static int numberOfWorkers=0; // no. of Worker objects created
...
```



Reference the variable by Worker.numberOfWorkers.

9

A **static method** is also in the drawer, not in individual folders.

Make a method static if it doesn't need to be in a folder because it wouldn't reference the contents of the "containing" folder.

/** = "b is the boss of c".

Precondition: b and c are not null. */

public static boolean isBoss(Worker b, Worker c) {

return b == c.getBoss();

}

10