## CS1110 3 February 2009

In 1968, the Defense Department hired Bolt Beranek and Newman (BBN) of Boston to help develop the ARPANET, which later became the internet. In 1971, Ray Tomlinson of BBN was given the task of figuring out how to send files from one person to another. He created email with file attachments. He selected @ as the separator between an email name and location. Names for @ in other languages:

| | | |
|---|---|---|
| Italian: chiocciolina | = | little snail |
| French: petit escargot | = | little snail |
| German: klammeraffe | = | spider monkey |
| Dutch: api | = | short for apestaart (monkey's tail) |
| Finnish: miau | = | cat tail |
| Israeli: strudel | = | a pastry |
| Danish: snabel | = | an "A" with a trunk |
| Spanish: un arroba | = | a unit of about 25 pounds |
| Norwegian: kanel-bolle | = | spiral-shaped cinnamon cake |

For more info: **http://www.mailmsg.com/history.htm**

## CS1110 3 February 2009

TODAY:
- A bit more on testing.
- Object: the superest class of them all. pp. 153-154.
- Function toString. pp. 112-113.
- Static variables and methods. Sec. 1.5 (p. 47).

NEXT LECTURE: Discussion of Methods
- Executing method calls.
- If-statements.
- The return statement in a function.
- Local variables.

Reading for the next two lectures: Section 2, but NOT 2.3.8!!!!

**Testing** using Junit. Pages 385-388 (through Sec. 14.1.1).

**Bug**: Error in a program.
**Testing**: Process of analyzing, running program, looking for bugs.
**Test case**: A set of input values, together with the expected output.
**Debugging**: Process of finding a bug and removing it.

- In DrJava, use **File** item **new Junit test case…** to create a testing framework; it does some set-up for you.
- **assertEquals(X,Y)** is your friend.

Get in the habit of writing test cases for a method based on its specification even before you write the method.

Use test cases to clarify a specification (i.e., your ideas) first.

```
/** = number of vowels in word w.
Precondition: w contains at least one letter and no blanks or
punctuation. */
public int numberOfVowels(String w) {
 // #@ nothing here yet
}
```

Note: we sometimes notice more when in “critique mode”.

### Class Object: The superest class of them all

**See 1/2-page section 4.3.1 on page 154.**

**The reason for this material will become clear later.**

**You need this information to do assignment A2.**

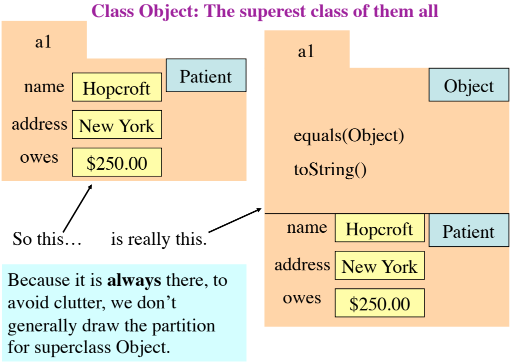
Every class that does not extend another one automatically extends class Object (“the mother of all classes”). That is,

**public class** C { … }

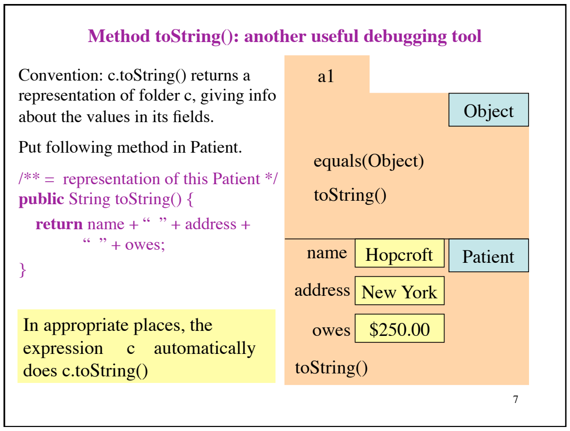
is equivalent to

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

### Class Object: The superest class of them all



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



## Method toString(): another useful debugging tool

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

Put following method in Patient.

```
/** = representation of this Patient */
public String toString() {
    return name + "  " + address +
        "  " + owes;
}
```

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

a1

Object

equals(Object)

toString()

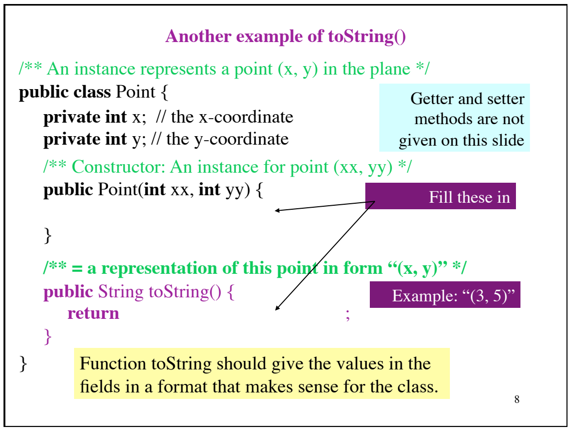
name Hopcroft

Patient

address New York

owes $250.00

toString()



## Another example of toString()

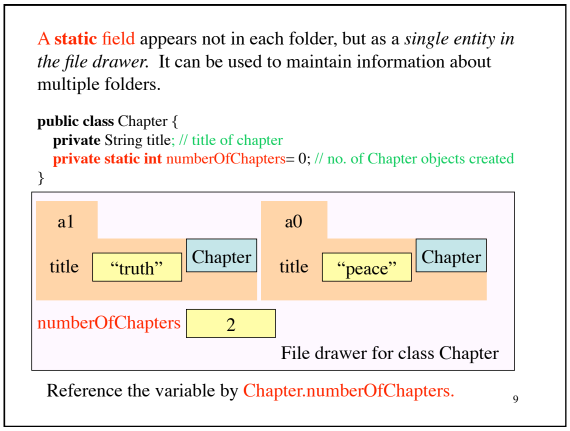
```
/** An instance represents a point (x, y) in the plane */
public class Point {
    private int x;  // the x-coordinate
    private int y; // the y-coordinate
    /** Constructor: An instance for point (xx, yy) */
    public Point(int xx, int yy) {

    }
    /** = a representation of this point in form "(x, y)" */
    public String toString() {
        return                          ;
    }
}
```

Getter and setter methods are not given on this slide

Fill these in

Example: "(3, 5)"

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



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.

```
public class Chapter {
   private String title; // title of chapter
   private static int numberOfChapters= 0; // no. of Chapter objects created
}
```

a1

title "truth"

Chapter

a0

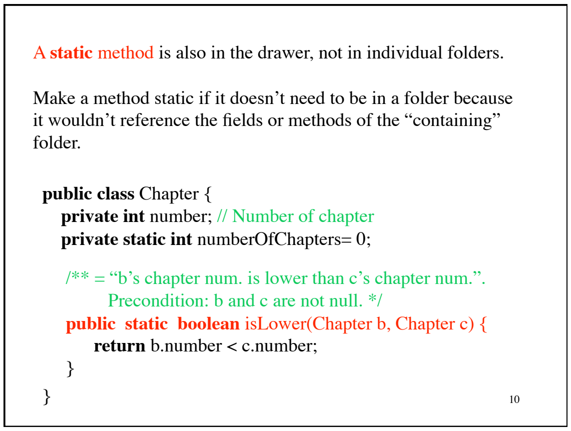
title "peace"

Chapter

numberOfChapters 2

File drawer for class Chapter

Reference the variable by Chapter.numberOfChapters.



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 fields or methods of the "containing" folder.

```
public class Chapter {
    private int number; // Number of chapter
    private static int numberOfChapters= 0;

    /** = "b's chapter num. is lower than c's chapter num.".
            Precondition: b and c are not null. */
    public  static  boolean isLower(Chapter b, Chapter c) {
        return b.number < c.number;
    }
}
```