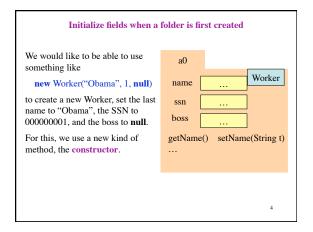


Getter and setter methods In the definition of Worker a0 (we post our code on the website): Worker /** = worker's last name*/ name public String getName() { ssn return name: /** Set worker's last name to n */ getName() setName(String t) public void setName(String n) { name= n; Getter methods (functions) get or retrieve values from a folder. /** = last 4 SSN digits, as an int*/ Setter methods (procedures) set (Try writing it yourself. or change fields of a folder Should there also be a setter? What about for boss?)



Purpose of a constructor: To initialize (some) fields of a newly created object In the class definition of Worker: /** Constructor: an instance with last name n. SSN s (an int in 0..999999999. Worker and boss b (null if none) */ name public Worker(String n, int s, ssn Worker b) { name=\n; boss ssn= s boss= b; getName() ... Worker(String n, int s, Worker b) The name of a constructor: the name of the class. Do not put a type or void here

New description of evaluation of a new-expression new Worker("Obama", 1, null) 1. Create a new folder of class Worker Worker, with fields initialized to name default values (e.g. 0 for int) -of course, put the folder in the file ssn drawer. boss 2. Execute the constructor call getName() setName(String Worker("Obama", 1, null) 3. Use the name of the new object Worker(String t, as the value of the newint i, Worker c) ... expression. Memorize this new definition! Today! Now!

Testing -using JUnit

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.

Get in the habit of writing test cases for a method from the method's specification --- even before you write the method's

A feature called Junit in DrJava helps us develop test cases and use them. You have to use this feature in assignment A1. 1. w1= **new** Worker("Obama", 1, **null**); Name should be: "Obama"; SSN: 1; boss: null.

Here are two test cases 2. w2= **new** Worker("Biden", 2, w1); Name should be: "Biden"; SSN: 2; boss: w1.

Need a way to run these test cases, to see whether the fields are set correctly. We could use the interactions pane, but then repeating the test is time-consuming.

To create a testing framework: select menu File item new Junit test case.... At prompt, put in class name WorkerTester. This creates a new class with that name. Save it in same directory as class Worker.

The class imports junit.framework.TestCase, which provides some methods for testing.

```
/** A JUnit test case class.
* Every method starting with "test" will be called when running
* the test with JUnit. */
public class WorkerTester extends TestCase {
   /** A test method.
    * (Replace "X" with a name describing the test. Write as
    * many "testSomething" methods in this class as you wish,
    * and each one will be called when testing.) */
   public void testX() {
  One method you can use in testX is
                         assertEquals(x,y)
   which tests whether expected value x equals y
```

A testMethod to test constructor and getter methods /** Test first constructor and getter methods getName, getSSN4, and getBoss*/ public void testConstructor() { Worker w1= new Worker("Obama", 123456789, null); first assertEquals("Obama", w1.getName(),); test assertEquals(6789, w1.getSSN4()); assertEquals(x,y): case assertEquals(null, w1.getBoss()) test whether x equals y; Worker w2= **new** Worker("Biden", 2, w1); print an error message second assertEquals("Biden", w2.getName()); assertEquals(2, w2.getSSN4()); and stop the method if test they are not equal. case assertEquals(w1, w2.getBoss()); x: expected value, y: actual value. Every time you click button Test in A few other methods that DrJava, this method (and all other can be used are listed on testX methods) will be called. page 488.

2