

Section 3: Test Cases and File Objects

As before, you are to pair up with a new student, someone you have not paired with before. Decide who will be the first driver, and let's get started.

PART I: Writing Test Cases

*As part of Program #2, you need to write your own testing class for objects created by your `BinaryNumber` class. The purpose of this part of this week's section is to show you how to choose test cases. You may be surprised to learn that test cases have something in common with program documentation: Both should be created **before** code is written.*

1. Get out a sheet of paper, or open a word processor (on a lab machine, click on Applications in the lower left of your screen, then Office, and finally LibreOffice Writer); your choice. This part will ask you several questions, and you'll want to write/type your answers.
2. Imagine that we need a class named `OrderedStrings`. Each `OrderedStrings` object stores and manipulates a sorted list (small to large, alphabetically) of `String` objects. Here is the set of public methods (including the constructor) of this class:
 - `OrderedStrings()` – creates an `OrderedStrings` object containing no strings but with a maximum capacity of 5 strings.
 - `size()` – returns the quantity of strings in this `OrderedStrings` object's list.
 - `location(String)` – returns the position of the given `String` object in the list (first position is 1, second is 2, etc.). If the string isn't in the `OrderedStrings()` object, `location()` returns 0.
 - `insert(String)` – added the given `String` object to this `OrderedStrings` object's list, in the correct location. If the string already exists in the list, or the list is full, `insert()` returns `false`, otherwise it returns `true`.

Let's assume that `size()` works correctly. Under that assumption, how can you use `size()` to test that the constructor is working correctly? (Make note of your answer; your SL will want to know.)

3. Hopefully, that one was easy; there's just one correct state of a new `OrderedStrings()` object. Now let's think about testing `location()`. Is there anything about the correct operation of `location()` that we can test without using `insert()`? Again, make note of your answer.
4. As your answer to the last question indicates, in order to fully test `location()` we need to use (and therefore start testing) `insert()`, too. This co-dependence of methods is common in testing, and a pain when debugging — a test that gives an incorrect result could be caused by either method, or by an incorrect test!


For the moment, assume that `insert()` works as it should. How can we use `insert()` to more completely test `location()`?

5. Finally, we can think about `insert()`. Because calling this method can change the order of the values (remember, the list needs to be sorted alphabetically), testing it will be more involved than the testing of the other methods.

Take several minutes to discuss (and write down!) different insertion situations `insert()` must handle correctly, **and** how you can use `size()` and `location()` to check that that each were actually performed correctly.

(Continued ...)

6. One last question: Did you need to know anything other than the documentation for these methods to plan their testing? (That is, did you need to know how the list is going to be stored? What the instance variables will be? Whether or not `size()` re-counts the elements in the list on each call or remembers the size between calls?)

 **CHECKPOINT 1** Raise your hand. Your SL will come over and ask you about your answers to some of those questions.

PART II: Implementing `OrderedStrings`


Remember: New part, new driver!

1. Log into the driver's account, start a Terminal window, and launch DrJava.
2. Launch a web browser and type in the URL your section leader has written on the board (or typed on the screen). What you will see is a testing program for `OrderedStrings`. (We didn't post it to the class web page because we wanted you to think about the test cases on you own without sneaking a peek at ours. Please don't tell your classmates in other sections about it; we want them to think about it, too!)

Copy/paste (or load directly) that program into Dr. Java.

3. Take a couple of minutes to look at the test cases in that program. How many are situations you thought about in Part I, and how many are new? Did you think of any situations in Part I that this program isn't testing?
4. Visit the class web page, find the `OrderedStrings.java` file, and load it into DrJava, too.
5. Take a good look at `OrderedStrings.java`. In particular, notice that there are instance variables for the list (an array of strings) and for the last element of the array that holds a string, and a constant for the list's capacity (which is quite small, to facilitate testing). Also notice that some code is missing ...
6. Compile the testing program. There should be no syntax errors, but the output shows that most of the test cases aren't producing the expected output.
7. To complete this part of the section activity, you need to write, complete, and/or fix the following pieces of code:
 - (a) Replace the stub of the `size()` method.
 - (b) Debug the `location()` method. (Remember, you have the testing program; use it to help you find the bugs.)
 - (c) Complete the `insert()` method. The code for the first two situations (inserting twice and inserting into an empty list) are already done. You need to:
 - i. Write the code to handle trying to insert into a full list.
 - ii. Write the code to handle inserting at the end of the list (which needs to be done when the new string is larger than the list's current largest string – remember, the list is in order).
 - iii. Debug the code that's supposed to handle inserting ahead of some or all of the list content. (Hint: There are two bugs.)

You'll know that you're done when the testing program doesn't report any errors.

 **CHECKPOINT 2** Raise your hand. Your SL will come over and verify that you're done.


(Continued ...)

PART III: Learning File Characteristics

In class on Wednesday we'll start talking about how to use Java to read from and write to files. Java also allows us to ask a file about itself.


(Did you remember to switch drivers again?)

1. Using a web browser, to the the Java API and find the `File` class. A `File` object is Java's in-memory representation of a file residing on secondary storage (e.g., a hard disk or an SSD).
2. For each of the following questions a programmer might wish to have answered about a file, determine the method that provides the answer.
 - (a) Has the file already been created?
 - (b) How many bytes of data does the file contain?
 - (c) Is the file really a directory?
 - (d) Can the file's content be read?
 - (e) What are the names of the contents of a directory?
3. What kind of things are directories, given that `File` objects are used to represent both files and directories?
4. On the class web page is a shell of a program named `Dir.java`. Its job: Get the name of a directory from the user; verify that the directory exists, is actually a directory, and is readable; and, if all of those conditions are met, lists the names of the files and directories contained within that directory, one name per line.
5. Your job: Complete that program!

 **CHECKPOINT 3** Raise your hand. Your SL will come over and give you a pathname on which to test your program.

PART IV: Clean Up!

1. Log out of your computer.
2. Pick up your papers, writing implements, cell phones, trash, etc.
3. Push in your chair(s).

 **CHECKPOINT 4** Raise your hand. Your SL will come over and tell you that your dorm room / apartment / house should be this clean.

You're free to go! But, if you have time, we recommend that you use it to work some more on the programming assignment, if you're not already done.

In fact, if you and your partner are agreeable, you could spend the remaining time thinking about test cases for Program #2. Remember, it's OK to discuss test case situations, but you can't share the code that uses those test cases; you have to put the test cases into a Java program yourself.