Using an IDE Debugger

CS 221 Homework Assignment

*"As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."*– Maurice Wilkes, 1949

*"To err is human. To really foul things up requires a computer.”*– Anonymous

# Objectives

The purpose of this assignment is to become familiar with the debugging facilities in Eclipse or a similar IDE - to observe the internal workings of a program *while it is running*.

This exercise reflects the same kind of testing situation you will be facing as we continue through the rest of the semester.

# Files

The SetTester test class contains 111 tests to confirm that ArraySet is a valid implementation of the SimpleSet ADT. Unfortunately, ArraySet contains some bugs. In this assignment, you will use the Eclipse debugger to find and fix all bugs in ArraySet, so that all 111 SetTester tests pass. (See sample final console output, below.)

[Handouts (zip)](https://cs.boisestate.edu/~mvail/221/assignments/homeworks/DebugSimpleSet/files/Debug.zip). Copy all of the handout files into an Eclipse project.

* [ArraySet.java](https://cs.boisestate.edu/~mvail/221/assignments/homeworks/DebugSimpleSet/files/ArraySet.java) - an array-based implementation of SimpleSet - this is the file you are trying to fix
* [SimpleSet.java](https://cs.boisestate.edu/~mvail/221/assignments/homeworks/DebugSimpleSet/files/SimpleSet.java) - an interface defining a simple Set ADT - DO NOT MODIFY
* [SetTester.java](https://cs.boisestate.edu/~mvail/221/assignments/homeworks/DebugSimpleSet/files/SetTester.java) - a test class for SimpleSet implementations - DO NOT MODIFY

[README](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/DebugSimpleSet/files/README) - an example README file, if this had been a project

# Tasks

Before anything else, read through the javadocs of the SimpleSet interface to understand what each method is expected to do. Look through ArraySet and understand the relationship between the array and rear variable. Also skim through SetTester to become generally familiar with the approach to testing being used.

Document your debugging process, as you work, in a plain text file named Debug.txt. For each bug you are tracking down, name the failed test where you are beginning your hunt, where you placed breakpoints, where the bug was found, and what you did to fix it.

It would also be helpful to describe how you recognized the bug. For example, you might list the contents of the list before and after a particular method call, and explain how that result revealed the problem.

You may place debugger breakpoints wherever you need to in SetTester, but NO MODIFICATIONS may be made to the code - do not add so much as a print statement. Use the debugger to determine the state of the program as it is running. Likewise, NO MODIFICATIONS are allowed in SimpleSet. ArraySet.java is the **only** file you may edit.

Begin debugging:

* Run SetTester to get current test results. Tests are ordered from more basic to more advanced. If basic tests are failing, more advanced tests that build on basic scenarios are meaningless.
* Start with the first test that is failing. Find that test in SetTester and place a breakpoint at the line where it is called.
* Run SetTester in debug mode. When it pauses at your breakpoint, use a combination of step-over and step-into to step through the code and observe the state of the ArraySet as it is tested. Be aware, the test that revealed the bug may not be testing the method where the bug occurs. The problem may actually be in a method call that *preceded* the failed call.
* When you have located a bug, stop the debugger and attempt to fix it.
* Rerun SetTester (go back to the top of these steps). Continue this process until all 111 tests in SetTester pass.

Note: To ease the challenge, somewhat, you can be confident that the toString() method is bug-free, so the Strings showing Set contents while debugging are reliable.

# Debugging Tips & Tricks

* When looking at a FAILed test, think about the whole sequence of steps that led up to the failed call. The last call is often not where the problem actually occurred. For example, if there is an error in the add() or remove() method, then it is reasonable to assume that size() might FAIL, even if the size() method is correctly written.
* After placing a breakpoint, plan to make multiple run throughs with the debugger. Your first run through may be fairly quick - simply trying to get a feel for the sequence of steps leading up to the FAIL result. Subsequent run throughs may be slower - observing changes in the data through the sequence of calls, and trying to identify where something unexpected occurs. As you rule out early steps, your run throughs may quickly advance to steps where you suspect the bug is occurring and then slow down greatly.
* You will find your own rhythm of how to step-into and step-over. When in doubt, step-into. This will give you the opportunity to read more code and hunt down what is happening. You’re learning a new tool, so take your time to try your best to understand what is going on.
* Once you have corrected all of the bugs and passed all 111 tests, try breaking code with [off by one errors](http://en.wikipedia.org/wiki/Off-by-one_error) and other accidental bugs you think would be easy to make in the ArraySet code. See what happens to the test results as a result of each bug you introduced. (Just be sure to fix them, again, before turning in the assignment!)
* In addition to helping you fix buggy code, the debugger is also a useful tool for getting familiar with working, bug-free code!

# Grading

Points will be awarded according to the following breakdown:

| Tasks | Points |
| --- | --- |
| All bugs found and fixed correctly | 10 |
| Debug process well documented in *Debug.txt* | 10 |

# Submission

Submit this assignment (a plain-text file named Debug.txt and all .java source files) electronically using the command given on your course's web page.

* Debug.txt
* *All .java source files* including debugged ArraySet

# Sample Console Output

Sample Output (when all 111 tests PASS).

NEW EMPTY SET

toString() output: []

emptySet\_testToString PASS

emptySet\_testIsEmpty PASS

emptySet\_testSize PASS

emptySet\_testContainsA PASS

emptySet\_testAddA PASS

emptySet\_testRemoveA PASS

=======================================================

SCENARIO: [ ] -> add(A) -> [A]

toString() output: [1]

emptySet\_addA\_A\_testToString PASS

emptySet\_addA\_A\_testIsEmpty PASS

emptySet\_addA\_A\_testSize PASS

emptySet\_addA\_A\_testContainsA PASS

emptySet\_addA\_A\_testContainsB PASS

emptySet\_addA\_A\_testAddA PASS

emptySet\_addA\_A\_testRemoveA PASS

emptySet\_addA\_A\_testRemoveB PASS

=======================================================

SCENARIO: [A] -> remove(A) -> [ ]

toString() output: []

A\_removeA\_emptySet\_testToString PASS

A\_removeA\_emptySet\_testIsEmpty PASS

A\_removeA\_emptySet\_testSize PASS

A\_removeA\_emptySet\_testContainsA PASS

A\_removeA\_emptySet\_testAddA PASS

A\_removeA\_emptySet\_testRemoveA PASS

=======================================================

SCENARIO: [A] -> add(A) -> [A]

toString() output: [1]

A\_addA\_A\_testToString PASS

A\_addA\_A\_testIsEmpty PASS

A\_addA\_A\_testSize PASS

A\_addA\_A\_testContainsA PASS

A\_addA\_A\_testContainsB PASS

A\_addA\_A\_testAddA PASS

A\_addA\_A\_testAddB PASS

A\_addA\_A\_testRemoveA PASS

A\_addA\_A\_testRemoveB PASS

=======================================================

SCENARIO: [A] -> add(B) -> [A,B]

toString() output: [1, 2]

A\_addB\_AB\_testToString PASS

A\_addB\_AB\_testIsEmpty PASS

A\_addB\_AB\_testSize PASS

A\_addB\_AB\_testContainsA PASS

A\_addB\_AB\_testContainsB PASS

A\_addB\_AB\_testContainsC PASS

A\_addB\_AB\_testAddA PASS

A\_addB\_AB\_testAddB PASS

A\_addB\_AB\_testAddC PASS

A\_addB\_AB\_testRemoveA PASS

A\_addB\_AB\_testRemoveB PASS

A\_addB\_AB\_testRemoveC PASS

=======================================================

SCENARIO: [A,B] -> remove(A) -> [B]

toString() output: [2]

AB\_removeA\_B\_testToString PASS

AB\_removeA\_B\_testIsEmpty PASS

AB\_removeA\_B\_testSize PASS

AB\_removeA\_B\_testContainsA PASS

AB\_removeA\_B\_testContainsB PASS

AB\_removeA\_B\_testAddA PASS

AB\_removeA\_B\_testAddB PASS

AB\_removeA\_B\_testRemoveA PASS

AB\_removeA\_B\_testRemoveB PASS

=======================================================

SCENARIO: [A,B] -> remove(B) -> [A]

toString() output: [1]

AB\_removeB\_A\_testToString PASS

AB\_removeB\_A\_testSize PASS

AB\_removeB\_A\_testIsEmpty PASS

AB\_removeB\_A\_testContainsA PASS

AB\_removeB\_A\_testContainsB PASS

AB\_removeB\_A\_testAddA PASS

AB\_removeB\_A\_testAddB PASS

AB\_removeB\_A\_testRemoveA PASS

AB\_removeB\_A\_testRemoveB PASS

=======================================================

SCENARIO: [A,B] -> add(C) -> [A,B,C]

toString() output: [1, 2, 3]

AB\_addC\_ABC\_testToString PASS

AB\_addC\_ABC\_testIsEmpty PASS

AB\_addC\_ABC\_testSize PASS

AB\_addC\_ABC\_testContainsA PASS

AB\_addC\_ABC\_testContainsB PASS

AB\_addC\_ABC\_testContainsC PASS

AB\_addC\_ABC\_testContainsD PASS

AB\_addC\_ABC\_testAddA PASS

AB\_addC\_ABC\_testAddB PASS

AB\_addC\_ABC\_testAddC PASS

AB\_addC\_ABC\_testAddD PASS

AB\_addC\_ABC\_testRemoveA PASS

AB\_addC\_ABC\_testRemoveB PASS

AB\_addC\_ABC\_testRemoveC PASS

AB\_addC\_ABC\_testRemoveD PASS

=======================================================

SCENARIO: [A,B,C] -> remove(A) -> [B,C]

toString() output: [2, 3]

ABC\_removeA\_BC\_testToString PASS

ABC\_removeA\_BC\_testIsEmpty PASS

ABC\_removeA\_BC\_testSize PASS

ABC\_removeA\_BC\_testContainsA PASS

ABC\_removeA\_BC\_testContainsB PASS

ABC\_removeA\_BC\_testContainsC PASS

ABC\_removeA\_BC\_testAddA PASS

ABC\_removeA\_BC\_testAddB PASS

ABC\_removeA\_BC\_testAddC PASS

ABC\_removeA\_BC\_testRemoveA PASS

ABC\_removeA\_BC\_testRemoveB PASS

ABC\_removeA\_BC\_testRemoveC PASS

=======================================================

SCENARIO: [A,B,C] -> remove(B) -> [A,C]

toString() output: [1, 3]

ABC\_removeB\_AC\_testToString PASS

ABC\_removeB\_AC\_testIsEmpty PASS

ABC\_removeB\_AC\_testSize PASS

ABC\_removeB\_AC\_testContainsA PASS

ABC\_removeB\_AC\_testContainsB PASS

ABC\_removeB\_AC\_testContainsC PASS

ABC\_removeB\_AC\_testAddA PASS

ABC\_removeB\_AC\_testAddB PASS

ABC\_removeB\_AC\_testAddC PASS

ABC\_removeB\_AC\_testRemoveA PASS

ABC\_removeB\_AC\_testRemoveB PASS

ABC\_removeB\_AC\_testRemoveC PASS

=======================================================

SCENARIO: [A,B,C] -> remove(C) -> [A,B]

toString() output: [1, 2]

ABC\_removeC\_AB\_testToString PASS

ABC\_removeC\_AB\_testIsEmpty PASS

ABC\_removeC\_AB\_testSize PASS

ABC\_removeC\_AB\_testContainsA PASS

ABC\_removeC\_AB\_testContainsB PASS

ABC\_removeC\_AB\_testContainsC PASS

ABC\_removeC\_AB\_testAddA PASS

ABC\_removeC\_AB\_testAddB PASS

ABC\_removeC\_AB\_testAddC PASS

ABC\_removeC\_AB\_testRemoveA PASS

ABC\_removeC\_AB\_testRemoveB PASS

ABC\_removeC\_AB\_testRemoveC PASS

=======================================================

SCENARIO: [] -> add 1000 elements

empty\_addManyElements\_bigSet PASS

Total Tests: 111, Passed: 111, Failed: 0