CPSC 2151 Lab 1

Setting up your IDE and Debugger

**Description:**

In this assignment, you will be given an existing project that has some errors in it. While it is very possible to just read the code and fix it without the use of your IDE’s debugger, that defeats the purpose of the lab where we learn to use the debugger.

**Submission:**

There is no submission for this lab. Follow the instructions and add the screenshots / text where the instructions tell you to do so (these are highlighted in yellow). Once you finish the instructions and have added all of the required answers / screenshots, call a TA over and they will confirm everything looks correct. Once they give you the “all clear”, you’re good to go and will receive a 100 for this lab.

1. Download the song\_code.zip from canvas, extract it to your desktop, and open the project in your IDE of choice. Again, since most students prefer IntelliJ, I will use it for this tutorial.
   1. The project should have 3 java files.
      1. Song
      2. songFE (the main)
      3. songManager
2. Go ahead and look at the code. What is it doing?
   1. Briefly explain (1-2 sentences) what this program is meant to do / mimic.

This program is used to create a list of songs with their start and end times. It seems to be mimicking a playlist on Spotify.

1. Now give the code a run. Doesn’t work… We aren’t given an error message meaning the code is running to completion, but the output is wrong… It says it removed the last song from the list, but it’s still there.
   1. Again, given the program’s simplicity, it shouldn’t be hard to just find the error without a debugger. However, real-world programs are hundreds, if not thousands, of lines long and can be a lot harder to “just find the error.”
2. Before we start using the debugger, we need to set **breakpoints.** Breakpoints are where we want the code to stop while it’s running. This allows us to view values of all the local variable at the compilation time of the breakpoint.
   1. It seems like adding the songs don’t create any problems, so we don’t need to include that in our debugger check.
   2. As such, let’s add a breakpoint to line 23, the line where it prints the song list. You can add a breakpoint by clicking slightly to the right of the line number.
3. Let’s open the IDE’s debugger. You can open it via the little green bug icon next to the green triangle in the top right corner (for IntelliJ)



1. Add a screenshot below that shows your breakpoint and your IDE open in debugging mode. You’ll know you’re in debugging mode when the debug tab opens instead of your console and you can see the song variables.

A screenshot of a computer

Description automatically generated

1. Currently our program is “paused” right before it runs line 23. We have a few options for what we want to do.
   1. A screenshot of a computer

      Description automatically generated with low confidence
   2. Look at the blue arrow buttons. In order from left to right they are:
      1. **Step over** -> Runs the current line.
      2. **Step into** -> “steps into” the method that is being called. It will move us from the active line of code to the method. In other words, you’re stepping into the next level of the call stack.
      3. **Step out** -> “steps out” of the method that is being called. Finishes running the method if it is pressed before you get to the last line of the method using step over.
      4. **Run to cursor** -> runs the code until it reaches the line your cursor is on.
2. Our active line is line 23.
   1. Use the step over button to run line 23 and to move the debugger to line 25
   2. Now, use the step into button to step into the .remove() function.
      1. This takes us to line 38 of the songManager file, specifically to the remove method.
   3. Look at this variable window in the bottom right side of the IDE.

A screenshot of a computer

Description automatically generated

* 1. Here we can see the relevant variables and objects.
  2. Clicking on the > next to this will show us the current object, which is our song list. Clicking on the > next to songList will show us the lists of objects sitting within that song list (which happens to be an array).
  3. We can also see the attributes of aSong, the parameterized variable for this method, the length of my song list, and the song list variable itself.

1. We want to better understand the remove() method so that we can figure out why it isn’t working. Let’s use the step over button to take it line-by-line.
   1. You’ll see after hitting the step over button a few times that as new variables get declared, they are added to the variables window. As we do this, we can see what the iterator, int i (from the for loop), is equal to and what is currently at the songList[i]. Let’s keep an eye on these as we continue to step over.
   2. The goal was to remove the final song “now or never (radio edit)” from the song list. We should keep an eye out for any time we see that song appear in our variables list (other than the instance where aSong is equal to the song).
   3. Eventually, after stepping over enough times, we should be within the for loop that starts at line 43 and we can see that songList[i] is equal to the song we want to remove.

(lab continues on the next page)

A screenshot of a computer program

Description automatically generated with medium confidence

* 1. We can see that i is currently equal to 4, which means the program found the song we were looking for in the array, and it found it at the 4th index.
     1. This tells us that our song does exist within the array, and the program is capable of finding it, which means that the issue isn’t locating the song, it’s deleting it from the array.
  2. Let’s move to the part of the code that handles deleting the song. Double click on line 70 so it highlights the whole line and hit the **run to cursor** button so skip the rest of that for loop.

1. Now we’re at the part of the remove() method that handles deletion.
   1. Let’s use the step over button to see if we can find the problem.
   2. Our songIndex is set to 4, so our for loop is going set the i index to the appropriate index. Our songList[i] is set to the song we want to remove, but we can see on line 72 we have an issue where instead of setting the value of the ith index to the value of the ith+1 index (similar to how something like bubblesort works) it just sets the value of the ith index to itself, effectively doing nothing.
   3. Before the next step, go ahead and take a screenshot of your variables window, we should see that the songIndex is 4, i is also 4, and that songList[i] is equal to the “now or never (radio edit)” song.

A screenshot of a computer program

Description automatically generated

* 1. This issue is our problem. Go ahead and change the line to read this instead.



1. Once we fixed our code, go ahead and close the debugger and run the program again. Now, take one final screenshot of the console showing our program giving the correct output where the song is removed the second time it prints the song list.

A screenshot of a computer

Description automatically generated