

Section 1: Let's Shake Off the Rust!

Your section leader should have told you to pair up with another student at one computer. Here's why: Section activities in this class are based on a model known as *pair programming*. One programmer assumes the role of the 'driver,' and controls the keyboard and mouse. The other is the 'navigator' and helps the driver work toward the goal by keeping the driver focused, by pointing out logic errors, etc. The roles will alternate during each lab, so that you'll get to do both each week.

Decide who will be the first driver, and let's get started.

PART I: Setting Up Our Programming Environment

Those of you who took CSc 127A this past spring or summer should already know how to use the DrJava IDE (Integrated Development Environment, a program that helps you write programs). In this part, you will verify that your environment is ready to go, or will get it set up.

1. Log into the driver's account. Don't remember your CS password? Visit the CS Department's On-Line Services Page (<https://www.cs.arizona.edu/computing/services/main.html> ; you'll need to authenticate with your NetID) and choose the "Reset my forgotten Unix password" option. Within an hour, you will receive an email to your UA email account with the new password.
2. Launch a Terminal window. Within it, launch DrJava by typing 'drjava' (no quotes) at the prompt and pressing Enter. A large window labeled "(Untitled)" will appear.

In this class you might find yourself needing to write programs that draw graphics. If so, we'll use a package of code stored in a file named `stdlib.jar`. Before we can use that package, we need to tell DrJava where to find this file.

3. Find the word 'Edit' on the DrJava menu bar and click it, and in the drop-down menu click on "Preferences ...".
4. The Preferences window will appear. On the left side, "Resource Locations" will probably already be highlighted. If not, click on it so that it is.
5. On the right side will be several labels, including one called "Extra Classpath." If you already see `/cs/contrib/linux/stdlib.jar` there, move to the next step. Otherwise, do the following to add it:
 - (a) On the right side, click the "Add" button next to the label "Extra Classpath."
 - (b) A window named Select will appear. Next to the label "File Name:" is a text box. Click on it, then type in this text, just as you see it:

```
/cs/contrib/linux/stdlib.jar
```

When you're sure you've typed it correctly, click "Select".
 - (c) Back in the Preferences window, you should now see that text just above the "Add" button. Click the "Apply" button at the bottom of the window.
6. Close the Preferences window.

(Continued ...)

7. Time to check that `stdlib.jar` is available to DrJava. Use a web browser to visit the class home page (<http://www.cs.arizona.edu/classes/cs127b/fall115/>). Scroll down to the “Assignments, and Section & In-Class Activities” section. In the “Section Activities” subsection is a link to a Java program named `Hello127B.java`. Save that program and load it into DrJava, or just copy and paste it into DrJava (your choice).
8. Save the program (click the “Save” button) and. compile it by clicking the “Compile” button. (If you don’t see this button, resize or reposition the DrJava window so that all of the command buttons are visible.)

If you get errors saying that the `StdDraw` methods aren’t found, double-check that you typed the `stdlib.jar` path correctly in the Edit/Preferences window. If you’re still having trouble, ask your section leader for help.
9. Finally, run the program (click the “Run” button). If everything worked correctly, you should see a graphics window with this content:



10. See it? Great! **Now, log out, switch drivers and repeat steps 1–9!** The new driver (the former navigator) is to repeat the same steps on his or her account. Don’t skip this step! We want you both to be confident that you have DrJava and `stdlib.jar` set up and ready to go. Having done it once with the first driver, those nine steps will go much faster for the second driver.

✓ CHECKPOINT 1 Raise your hand. Your SL will come over and verify that you have DrJava and `stdlib.jar` correctly set up.

(Continued ...)

PART II: Submitting a Program for Grading

So that your section leader can grade your programming assignments, you need a way to submit them to him/her. We will be using a program named 'turnin' on a machine named 'lectura' to do this. The next steps will show you (or remind you!) how to submit `Hello127B.java`.

1. Close the DrJava window.
2. Click on the Terminal window that you used to open DrJava.
3. To run 'turnin', you need to log into a different computer, one named "lectura". In the Terminal window, type the following command (followed by a press of the Enter key):

```
ssh lectura.cs.arizona.edu
```

You might see a warning about connecting to a new computer; ignore it. Enter the driver's password. You should see a command prompt.

4. To submit your `Hello127B.java` program to your SL, you first need to know your section letter. Your SL should have it written on the board or displayed on the projector. Use it in place of the X in the upcoming command. Ready? OK, use the following command to 'turnin' your program:

```
turnin cs127bsXa01 Hello127B.java
```

If you get an error message, check that you typed the command correctly, and that you remembered to replace the X with your section letter.


(And if you're wondering, "What's up with that funny name?", it's just the concatenation of "cs127b" (our class), "sX" (section X), and "a01" (activity 01).)

5. You should see a success message. To be sure that the program was correctly submitted, type the following command in the Terminal window:

```
turnin -ls cs127bsXa01
```

(That's a lower-case 'L' in '-ls'.) You should see the name of the file that you submitted. It's always a good idea to check that your files were correctly received; do this step every time you 'turnin' a file.

6. If you want to repeat this part for your partner, feel free! But, if you both feel that you know how to do this, you can move on.

 **CHECKPOINT 2** Raise your hand. Your SL will come over and verify that you submitted `Hello127B.java` correctly.

7. Close all of your open windows and log out of the driver's account.

(Continued ...)

PART III: Writing Some Instance Methods

Before starting this part, switch roles – the driver becomes the navigator, and the navigator becomes the driver. This is how section activities will usually work: Switch roles when you move to the next part.

By the end of your 127A class (or its equivalent), you should have learned about writing your own classes. For this part, you will refresh your memory of Java and its statements by writing a couple of methods in a class we've already started for you.

1. Log in to the new driver's account.
2. Visit the class web page again, but this time find and load into DrJava the file `Section01.java`.
3. This file has two classes, `Sequence` and `Section01`. `Section01` has the `main()` method, which tests that `Sequence` objects work correctly. At the moment, two of the methods in `Sequence` are incomplete.

Compile and run `Section01`. You should see the output from five test sequences. At the moment, there are several lines that begin with the word "INCORRECT." To complete this part, you and your partner need to correctly complete those two methods.

4. Take a look at the documentation and code we've provided. The documentation is in the form that we expect you to use in the code that you write for this class for the programming assignments. The code is well-structured and easy to read; that's what we expect from your code, too.
5. Locate the `sum()` method. At the moment, `sum()` is just a stub method; that is, it is a syntactically-correct method that doesn't yet do anything useful. Perhaps surprisingly, it already has documentation. Your job: Complete the `sum()` method so that, when the program is compiled and run, all of the test cases for `sum()` display "Correct."

Not sure how to start? Take a look at the code of the provided methods, `numTerms()` and `isNonDecreasing()`. Study them until you and your partner understand how they work. Then tackle `sum()`. All `sum()` needs to do is compute the sum of the values in the array. You probably wrote that kind of loop several times in 127A; if you finished that class successfully, you can do this!


6. Time to complete the `isArithmetic()` method. Locate it. You've probably already noticed that it has a block comment, but no useful content. You're going to complete the method's block comment *before* you write the method! **This is how you should write all of your code.** (Why? Because "Thinking Before Doing" is far better than "Doing Before Thinking!") Use the information in the next paragraph to fill-in the sections of the block comment.

Use the information in the next paragraph to fill-in the sections of the block comment. A key detail: Write the documentation in your own words. Do not just copy-and-paste the following paragraph; that's called *plagiarism*, and is a clear violation of the UA Code of Conduct and Academic Integrity policies.

A numeric sequence is ARITHMETIC if all adjacent pairs of elements differ by exactly the same amount (known as the 'common difference'). For example, the sequence 4, 7, 10, 13, 16 is arithmetic because $7 - 4$, $10 - 7$, $13 - 10$, and $16 - 13$ all equal 3. If the `Sequence` object's sequence has a common difference, the method `isArithmetic()` returns the value `true`. Otherwise, it returns `false`.

7. Now (finally!) you can write the code for `isArithmetic()`. You'll know you're done when you compile and run the program and all of the test cases display "Correct."

Feel as though you need a little help? Look at `isNonDecreasing()` again. `isArithmetic()` will be similar in construction.

 **CHECKPOINT 3** Raise your hand. Your SL will come over and ask you a couple of questions about these methods.

(Continued ...)

PART IV: Clean Up!

The GS 930 lab is hosts sections from several of our programming classes, and is available the rest of the time for any CS student to use to work on their programming assignments. To help you get in the habit of leaving your work space neat and clean for those who will use it next, we'll end each section activity with this checkpoint, which also serves as confirmation that you attended section – even if you get nothing else done, you can clean up before you go, thereby earning one checkpoint and proving you attended.

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 verify that your workspace is tidy.

4. You're free to go! But, if you have time, we recommend that you ...
 - ...follow the instructions in the “Using CS Lab PCs in GS228 & GS930” web page (<https://www.cs.arizona.edu/computing/help/CS-Labs.html>) to change your password, especially if you had to reset it because you'd forgotten your old one.
 - ...visit the lab in GS 228 and see how running DrJava there is different from running it in the GS 930 lab. Remember that the “Getting Oriented on a Macintosh” handout is available to help.