What’s the Point?

One of the most common types that you’ll use in a Java program (other than int) is String - a variable which represents a series of letters and symbols. String is used for reading keyboard input, typing values out to the screen, and storing text data in files. (Files that use binary values need a more advanced type.)

What We’ll Be Doing

Here are the key methods and techniques we’re going to practice:

- Declaring variables of type String
- Setting, modifying, concatenating, and printing Strings
- Using the length() method, which returns the number of characters in the String
- Looping over the characters in a String
- Using the charAt() method, which gets one character out of a String
- Using the indexOf() method, which searches a String for a specific character
- Using the substring() method, which slices out a piece of a String
- Using the equals() and compareTo() methods.

Pair up!

Remember, we’ll be doing pair programming this semester. So choose a partner (somebody you’ve never worked with in section), and find a computer.

Part 1: Pseudocode

I heard good things about the group pseudocode assignment last week! So let’s try doing it again. For our first checkpoint, the entire Section will work together to write pseudocode for the first program.

Your program that you write will have three inputs, named a,b,c. You can assume, in the pseudocode, that those three variables all have values (Strings) before the pseudocode begins.
The program should perform the following steps (your SL may choose to shorten this list if necessary):

1. Print out the value of all three inputs, along with their lengths. (Should you put them all on the same line, or on different lines? The Section can decide.)

2. Concatenate the three inputs together - with no spaces between them, and store the result into a new variable. Then do the same - but with spaces - and store it into another variable.

3. Print out both of the concatenated variables.

4. Iterate over all of the characters in one of the inputs - to do this, count from 0 to length()-1. For each character, use charAt() to read that character from the String, and print it out.

5. Use the indexOf() method to see if the second input includes the character 'X'. If it does, then print out the value returned by indexOf(). If not, then print out an error message.

   Can you do this without calling indexOf() a second time?

6. Use the substring() method to extract the second, third, and fourth characters from the third input, and print them out.

7. Create a new String variable, and initialize it to the empty string. Then write a for() loop which will replicate the substring() call - it should extract the second, third, and fourth characters from the third input. Store them into the variable - and then print out its contents at the end.

This section ends when your Section Leader says so; you may finish all of the points above, or your Section Leader may end it early.

✅ CHECKPOINT 1

Everybody who has been taking part should get this checkpoint for free!

Part 2: Convert it to Java

Log onto your lab computer. Open up a text editor, and copy down all of the pseudocode that the Section has written (unless your SL comes up with a faster way, that lets you download it).

Next, open up the class homepage: http://www.cs.arizona.edu/classes/cs127a/spring16/

Once you're there, open up two things:

- This section activity
• Slide deck 02 ("Intro to Java")
• Slide deck 04 ("Loops")
• Slide deck A ("String")

Now we’re ready to get started.

Java, Step 1: The Class, and the Inputs
Open up DrJava, and create a new class. Create the shell of the program (remember, you can look at Slide Deck 02, slides 42-44).

In our previous programs, we’ve read integer inputs, using the parseInt() method; in this one, we’re using String inputs. But since the args[] array is made up of Strings, it’s even simpler:

```java
String a = args[0];
```

Java, Step 2: Convert from Pseudocode
Now, convert the pseudocode to Java, a few lines at a time. It’s wise to run your program every few lines (or at least compile it), so that you can find errors early.

When you have converted the pseudocode to Java, test your code to make sure it works. When it does, you’re done with this checkpoint!

✓ CHECKPOINT 2
Raise your hand. Your SL will come over and verify that you completed this step correctly.

Time to switch!
Log out, and switch seats.

Program 3: equals() and compareTo()
In this program, you will compare some Strings using equals() and compareTo(). Create a new program, and have it read three Strings from the command line - just like before.

This time, your program simply needs to compare the Strings. I want you to compare them in three ways: with the == operator, with equals(), and with compareTo().

```java
==
```

For the first comparison, I’ll give you the code. Compare the first and second arguments like this:

```java
==
```
if (a == b)
{
    System.out.println("a and b are the same.");
}
else
{
    System.out.println("a and b are different.");
}

Go ahead an try it out. You'll find out that even if the first two inputs are exactly the same, the == operator still returns false! I'll explain why this is, later in the course - but for now, just notice that it happens.

equals()
Now it's time to experiment with the equals() method. Write another if() block, but this time, compare a and be using equals(). You'll find that now, it works exactly as you would expect - if you give the same input twice, your program will notice it!

Once you get the if() block working, cut-and-paste it twice; make another if() block which compares a to c, and a third which compares b to c.

cmpareTo()
Finally, let's recreate the “find the minimum” program (Slide Deck 01). You’re welcome to use the same basic layout - compare a to b, and a to c, and so on - but this time, compare the String inputs - and Strings require compareTo().

The compareTo() method returns 0 if the two Strings are identical, negative if the one on the left is “less”, and positive if the one on the left is “more.”

So take the pseudocode from the slide deck, and modify it to work with your three Strings. Print out the minimum string, along with a note about which one it was. (If you have extra time, print out all three of them, in order.)

ASCII
You will find that compareTo() gives some strange results. For instance, numbers come before letters, and capital letters before lowercase - meaning that the String "ZZZ" comes before "a".

If you're curious why this is, you can read up about ASCII, the standard way to convert (English) letters into numbers:
http://www.asciitable.com/
https://en.wikipedia.org/wiki/ASCII

✓ CHECKPOINT 3
Raise your hand. Your SL will come over and verify that you completed this step correctly.

**The End: Clean Up**

The G/S 930 lab is host to 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, 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.

- Log out of your computer.
- Pick up your papers, writing implements, cell phones, etc.
- Push in your chairs.

 ✓ **CHECKPOINT 4**

Raise your hand. Your SL will come over and verify that you completed this step correctly.

As with so much in this course, thanks to Dr. McCann for some of the ideas - and even some of the exact text - for this Section Activity!