# CSC 205 Lab 1 : Back in the Saddle Again!

*Due In Your Account By 11:59 PM, Tuesday, September 5th*

## Goals

After completing this lab, you should be able to:

* Log onto the Cobra computer.
* Do simple *Linux* file manipulations, including creating and using a subdirectory, copying files from a class directory, creating scripts, and printing files.
* Create files.
* Compile and run Java programs.
* Work with methods, arrays, and String objects in Java.

## Lab Preparation

Now, let's make a directory named Labs by typing mkdir Labs. Now, change into this directory by typing cd Labs. Next, make and change into a Lab1 directory within this one.

**Basic Linux**

Copy the Lab1 files from the class directory. The command:

cp /pub/digh/CSC205/Lab1/**\* .**

will copy the files to have the same name in your working directory. Note : after the asterisk, be sure you type a space, followed by a period, and then the enter key.

Compile the program in the file Prog1.java. Recall, you compile programs using the command :

javac Prog1.java

It should compile without any errors, producing a file named Prog1.class.

This program will read integers from the input until the number 0 is read and print the product of the numbers (excluding a 0). Run the program interactively by typing :

java Prog1

and entering a few integers on separate lines. Be sure to end your input with a 0. Run the file in batch mode, using the data in the file data1.dat. To run the program using the data in the file, use redirection:

java Prog1 < data1.dat

You should get as output :

The product is -3628800

Display the file by typing cat Prog1.java, and make sure you understand all of the source code.

Create a script of the listing, compilation, and run of the program. To create a script, first turn on the recorder using the command:

script

Then give the commands you want to record:

cat Prog1.java

javac Prog1.java

java Prog1 < data1.dat

Finally, turn off the recorder:

exit

**Entering a program**

Enter the following program. To enter the vim editor to create a file called Prog2.java, use the command: vim Prog2.java

// This program will read a string and give its length

// Written by : <your name>

import java.util.\*;

class Prog2

{

public static void main(String[] args)

{

String data;

System.out.print("Enter a string: ");

Scanner stdin = new Scanner(System.in);

data = stdin.nextLine();

System.out.println("The string has " + data.length() +

" characters.");

}

}

Compile and run the program.

**Completing a Program Using Class Methods, Arrays, and Strings**

Type in the program which follows and complete the body of the method FindMin so that it takes as input an array of size numItems of strings and returns to the calling function the minimum string. That is, the string that would appear first alphabetically.

For example, consider the following sample input and output.

**Sample Input (from a user-created external text file States.dat)**

Georgia

Florida

North Carolina

Tennessee

Alabama

**Sample Output (to screen)**

Alabama

Method FindMin is a value-returning class method so you'll need a return statement at the end of your method. Be sure and notice how this method is invoked in main. Also, notice how both methods in main actually send the array list off as a parameter, and then notice how this array parameter is defined in the headers for the two methods it is being sent to.

In writing FindMin, remember that when you compare two string objects you must use the compareTo method rather than a < sign. **Do not sort the array**. Also, make sure your loop reaches a maximum value of numItems (not list.length).

import java.io.\*;

import java.util.\*;

public class MinString

{

private static final int SIZE = 10;

public static void main(String[] args)

{

String[] list = new String[SIZE];

int numItems;

numItems = Initialize (list);

System.out.println(FindMin (list, numItems));

}

private static int Initialize (String[] list)

{

//post : List is initialized with all strings from file.

String filename, stateInput;

int i = 0, numItems = 0;

try {

System.out.print("Input File : ");

Scanner stdin = new Scanner(System.in);

filename = stdin.nextLine();

stdin = new Scanner(new File(filename));

while ((stdin.hasNext()) && (i < list.length))

{

stateInput = stdin.nextLine();

System.out.println("S = " + stateInput);

list[i] = stateInput;

i++;

}

numItems = i;

}

catch (IOException e) {

System.out.println(e.getMessage());

}

return numItems;

}

// Method FindMin goes here

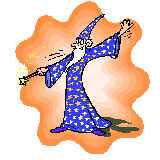
}

**Lab Extra Credit**

**Writing Programs that Review CSC 204 Concepts**

You can receive lab bonus points by attempting the two programs which follow. They’re a little challenging, but fun! They both review a lot of the different concepts you learned in CSC 204. Let me know if I can help you in any way in completing them.

**Program #1 : Checking for Magic Squares**



Write a program that reads in data for a 3 by 3 matrix of integers from an external file, and than checks that matrix to see if it is a *magic square*. A magic square is a square matrix

(a matrix with an even number of rows and columns) where the sum of the rows, columns, and diagonals of are all equal.

For example, the following is a magic square since all the rows, columns, and diagonals sum up to 15 :

**Sample Program Input** (from infile.dat)



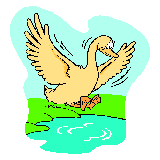
8 1 6

3 5 7

4 9 2

**Sample Program Output** (to screen)

Yes, It’s A Magic Square!



**Program #2 :**

**Calling a Duck a Duck!**

*When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.*

Write a program to recognize a *duck* using the above criteria. More precisely, write a program to read lines of up to 80 characters and tell whether that line contains a duck. But ducks don’t like to come out into the open, so you’ll have to check if the line has the words “Waddle”, “Swim”, and “Quack.” The words can be in any order, in any case, and may be embedded in other words. There may not be characters embedded within them.

As a special case, a line may have a *duckling*, if all occurrences of the words “Waddle”, “Swim” and “Quack” are in lower case. HINT: Use nextLine to input each line into a string variable, and then use your indexOf operation to check for substrings.

**Sample Program Input** (from infile.txt)

I saw it waddling, swimming, and quacking.

It waddled, swimmed, and quacked.

My, SWIM, SWIM, SWIM, Wadd, lead on, quackers.

Swim, swim, little moquackette, waddle home to mother.

Aren’t ducks silly?

**Sample Program Output**  (to screen)

Line 1 is not a duck

Line 2 is a duckling

Line 3 is not a duck

Line 4 is a duck

Line 5 is not a duck