Lab 1a: Review and Infix Calculator
CSSSKL 162: Programming Methodology

Purpose
This lab serves as a refresher for programming constructs and structures in Java. We will consider Selection Control Structures (if/switch), Repetition Control Structures (for/while), arrays, and File IO. The first handful of programs is simple enough to be done all in one main (thus one file), while the last three programs require their own file and main driver.

Diving into Java Programming, V2.0
Let's review some Java code that was core to CSS 161. We'll start with some control structures and move on to data, arrays, algorithms and a bit of file IO. Start by picking a java IDE of your choice (frequently BlueJ or Eclipse), make a project, and make a main(). Do the following exercises in this main function.

1. Control Structures (For, While, If, Switch)
- Write a loop to print out the first 10 digits.
    - Use a for() loop to accomplish this
    - for(int a = 0; a < 10; a++) {
- Write a loop to print out the first 10 even digits
    - Use a while() loop to accomplish this
    - Use an if(a % 2 == 0) to determine even or odd
    - while(<condition>) {
- Write a loop to sum the first 5 integer quiz scores entered by the console
    - Use Scanner keys = new Scanner(System.in) for console input
    - Use a sum variable
- Average the integers entered in the previous problem
    - Produce a double result, as in 96.5
- Use an if statement to produce a letter grade {A,B,C,D,F}
    - if(grade >= 90) {//print A
    - else if(grade>=84) { //print B
- Rewrite the if statement above using a switch statement
    - Are there any problems with this?

2. Arrays

- Write a new main and declare an array of integers
  - int[] data = {3,1,-8,4,-5,2};
- Write a separate function to sum the contents of an array
  - public static int sum(int[] data)
- Find the average of the function above
- Write a function to find the index of a specified element
  - public int indexOf(int[] data, int target)
- Write a function to sum up only positive integers in an array
  - public int positiveSum(int[] data)
- Write a function to populate an int array with values obtained from the console
  - public void populateArray(int[] data)

3. File IO

- Write the contents of the array above to a file, all on one line separated by a comma
  - PrintWriter pw = new PrintWriter(new File("data.txt"));
- Write the sum of the array to the file on the next line
- Write the average of the array to the third line
- Make a new main that reads the contents of this file ("data.txt") and prints it to the console
  - Scanner fin = new Scanner(new File("data.txt");

4. Standalone Programs using Loop Structures – Part I

In this section, your job will be to write a new program/class that will use scanner to ask the user for a number. Once you've got the number, your job is to calculate the factorial corresponding to the number. We'll use a loop (a Repetition Control Structure) inside our main to build the final output. Here is an example execution from the console (although you may use JOptionPane if you wish) – note the user input to the program is in red.

Enter the number: 5

1 * 2 * 3 * 4 * 5

120

Steps

(1) Declare a new program or class (if this is eclipse, make a new project first)
(2) Define the main function
(3) Inside main, declare a scanner (don't forget your import!)
(4) Define a variable called product. Should this start at 0?
(5) Prompt the user as the output indicates above for a number n
(6) Write a for loop that loops from 1 to n, printing out the current number each time
(7) In the loop, multiply your loop control variable by product.
(8) Print out your final product.

5. A Second Java Program (Selection Control Structures)

Create a second java program and define its main function. In this program, we'll also use a scanner to ask the user for a number n. Once we have this number, we'd like to find its GCD (greatest common divisor) by using a simple algorithm described below. Notice that, to determine the GCD, we'll need to make use of an if statement (a Selection Control Structure) to determine if the number n is evenly divisible by some divisor d. The code to determine if a number n is evenly divisible by a divisor d is written here:

" if n divided by d produces a remainder of zero, then d evenly divides n"
Or in Java…

```
if( n % d == 0 )  {  //if true, d is a divisor
```

Notice we will also need an if statement for special cases; what if the user enters a 0 or a 1? Handle these special cases by wrapping your logic in an if statement that handles the case where n is 0, 1, or greater than 1. Here is a sample execution…

```
Enter a number:  12
Divisors are 6 4 3 2 1
```

Steps

(1) Make a new program and define its main().
(2) Define a scanner and ask the user for a number n.
(3) Build an if statement to handle cases such as 0, 1, or greater than 1.
(4) In the greater than one case, build a loop starting at n (or n/2?) and ending at 1.
(5) Using the if statement above, print out your loop variable if it evenly divides n

6. File IO using PrintWriter

Revisit your first two programs, and instead of outputting the answers to the console, use PrintWriter to write your data to a file (say "data.txt").

Steps

(1) Declare a PrintWriter as follows : "PrintWriter pw = new PrintWriter(new File("data.txt"));"
(2) Next, catch or declare your problematic code, now that you are attempting File IO
(3) Instead of printing your output to System.out.println(), use your PrintWriter (pw.println())
(4) Close your printwriter.

7. File IO using Scanner

Build one final program/class and define a main(). This program will open the text files from the first two programs (say, "data1.txt" and "data2.txt") and output the results of the file to the console. Consider a sample execution below:

Which file to view? data1.txt

1 * 2 * 3 * 4 * 5

120

We will use Scanner to accomplish the file reading, but instead of making a Scanner which points to System.in, you must define a Scanner as follows:

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

Steps

(1) Define a program and define its main()
(2) Ask the user for a filename to view
(3) Declare your scanner to read from the file as demonstrated above
(4) While your file has more lines, print out each line
(5) Close your scanner (foo).

8. The Infix Calculator Program [individual assignment]

Build a simple calculator that ignores order of operations. This "infix" calculator will read in a String from the user and calculate the results of that String from left to right. Consider the following left-to-right calculations:

"4 + 4 / 2"
"Answer is 4" //not 6, since the addition occurs first when reading from left to right

"1 * -3 + 6 / 3"
"Answer is 1" //and not -1
Start by copying the driver code below. Read this program and notice the comments before you proceed – your assignments will require you to comment accordingly, or they will drop a complete letter grade! Begin tracing the program in main, and your code will go in the calculate() function, so start writing code there.

Hints

- Build the starter code using a snippet below.
- Assume spaces between all numbers and operators
- If a number is negative (like -3), there will be no space between the sign and the number
- Use Scanner and nextInt() to read each integer
    - Use Scanner and what operation to read a single character? (our operator?)
- You can also use StringTokenizer to do these tasks as well
- Using the comments below, can you write the regular expression that would recognize this grammar?

Once you have completed the infix calculator and tested it, then augment your program with File IO. Read each input string from a file ("inputData.txt") and write each integer result to a file ("output.txt"). Move the input Strings from main below to inputData.txt to test your software.

```
import ?;
/*
 * InFixCalc, V0.0 (concept borrowed from Carol Zander's Infix Calculator)
 * Exercise author: Rob Nash
 * Complete the calculate() function below to build a simple, infix
 * calculator that evaluates a compound expression from left to right,
 * regardless of operator precedence
 *
 * Example: " 1 * -3 + 6 / 3"
 * Execution: calculate 1 * -3 first, then -3 + 6 next, then 3 / 3
 * last to obtain the value 1
 *
 * Solution by <your name goes here>
```

```
 */

public class InFixCalc {
	//example pattern: "3 + 5"
	//general pattern: <lhs='3'> <operation='+'> <rhs='5'>
	//extended pattern: <int> <op> <int> <op> <int>...<op> <int>
	//special case: <int>
	//other special cases?

	public static void main(String[] args) {
		//String input = "4 + 4";
		//String input = "4 + 4 / 2";
		//String input ="1 * -3";
		String input ="1 * -3 + 6 / 3";
		//String input ="5";
		//String input ="-5";

		int answer = calculate(input);
		System.out.println("Answer is " + answer);
	}

	//preconditions: all binary operations are separated via a space
	//postconditions: returns the result of the processed string
	public static int calculate(String input){
		int lhs,rhs;  //short for left-hand & right-hand side
		char operation;

		/*todo: your name and code goes here*/

		/*You need a Scanner(or StringTokenizer) to get tokens
		 *Then you need a loop, and switch inside that loop*/


		return lhs;
	}
}
```