# Iteration Lab

**30 Points**

1. Loops provide a mechanism for repeating a block of code called the *loop body*. We begin this lab by experimenting with *while loops*, the most general form of loop code. Many loops are controlled with a single variable which we will refer to as the *loop control variable* or *the loop index*.
2. Consider the code below. What is the output the program produces? Try to determine the output before running the program, then write, compile, and execute it.
3. Comment out the line that increments **i**, then recompile and run the program. Will the program ever stop looping? Answer parts ‘**a**’ and ‘**b**’ in the Submission text box in the lab assignment link in Blackboard.

/\*\*

A simple program that prints a loop control variable.

\*/

public class SimpleLoop

{

public static void main(String[] args)

{

int i = 0;

int limit = 6;

while (i < limit)

{

System.out.println("i = " + i);

i++;

}

}

}

1. Manipulating the loop control variable is a critical skill in learning to write code with loops. Modify the program in part (a) so that it produces the following output and submit your new version of SimpleLoop.java by attaching it to the lab assignment link in Blackboard:

i = 6

i = 8

i = 10

i = 12

i = 14

i = 16

i = 18

…

i = 98

1. There is a famous story about a primary school teacher who wanted to occupy his students’ time by making the children compute the sum of 1 + 2 + 3 + … + 100 by hand. As the story goes, the teacher was astounded when one of the children immediately produced the correct answer: 5050. The student, a child prodigy, was Carl Gauss, who grew up to be one of the most famous mathematicians of the eighteenth century.

Write a simple program called WhileCountLoop that contains a main method with a loop that will compute and print the above sum.

After you have the program working, rewrite it so you can compute 1 + 2 + … + n where n is any positive integer.

1. Java provides three types of loops: *while*, *for*, and *do while*. Theoretically, they are interchangeable - any program you write with one kind of loop could be rewritten using any of the other types of loops. As a practical matter, though, it is often the case that choosing the right kind of loop will make your code easier to produce, debug, and read. It takes time and experience to learn to make the best loop choice, so this is an exercise to give you some of that experience.

Rewrite the program in #2 using a *for* loop, naming the file ForCountLoop. Repeat the exercise again but this time use a *do while* loop and name the file DoCountLoop.

Which form of loop seems to work best? Why?

Submit all three versions (WhileCountLoop, ForCountLoop, and DoCountLoop) by attaching them to the lab assignment link in Blackboard.

1. Some computations require multiple loops where the “inner” loop is code nested inside the body of the “outer” loop code. Suppose we want to produce the following output using a looping control structure:

The sum of positive integers from 1 to 1 is 1

The sum of positive integers from 1 to 2 is 3

The sum of positive integers from 1 to 3 is 6

The sum of positive integers from 1 to 4 is 10

The sum of positive integers from 1 to 5 is 15

…

The sum of positive integers from 1 to 100 is 5050

Since we already have a program that computes 1 + 2 + … + n for any positive integer n, we can imbed that code in the body of a second loop that loops 100 times. The outside loop control variable can be used to control the number of times we loop (the value after the “to” in the output statement) and more importantly, it can be used to control the upper limit of the inner computation.

Write a program called NestedCountLoop containing the code that will produce the output described above and submit your program by attaching it to the lab assignment link in Blackboard.