Chapter 4 - Loops and Files

Chapter 4 introduces the idea of loops in programming, as well as file reading and outputting files. Loops work by repeating a segment of code over and over until a specific condition is met. File reading and output is another important topic, where programmers are able to read data from a file and use it as input, as well as output data into a file.

There are three essential loops introduced in this chapter which are while loops, do-while loops, and for loops. The following are some of the characteristics of each loop.

Loop	Format	Characteristics	When to use

While	while (condition) {
	Statements;
	}
		Repeats the block of code until the condition is false. Is a pretest loop, testing the condition before each iteration.	Use when you do not want the loop to iterate if the starting condition is false.

Do-While	do { Statements; } while (condition);	Repeats the block of code until the condition is false. Is a posttest loop, testing the condition after each iteration.	Use when you want the loop to iterate at least once despite the conditions.

For	for (initialization; test; update) {
	Statements;
	}
		Is a count-controlled loop that requires a control variable, tests the control variable, and updates the control variable after each iteration.	Use when the exact number of iterations you need to execute is known.

For loops require increment and decrement operators to update the control variable. Incrementing is to increase the value by 1, and decrementing is to decrease the value by one. The following are some ways of writing increment and decrement operators.

Increment	Decrement
num = num + 1;	num = num - 1;
num += 1;	num -= 1;
num++;	num--;

*Note that for *num = num + 1* and *num += 1*, the value of 1 can be any arbitrary integer. The same is true for decrement operators. ProgramOneCFour.java shows an example of this.

Loops can also be nested, just like if statements. In this case, a nested loop is a loop that is inside another loop. The logic is that the outer-loop will run for one iteration, and one iteration for the outer-loop is equivalent to the inner-loop running multiple iterations until the condition is met. This allows one outer-loop iteration to end as the outer-loop runs another iteration, repeating the inner-loop iterations. This cycle continues until the outer-loop conditions are met. ProgramTwoCFour.java shows an example of a for loop nested in a while loop.

Java allows programmers to read data from files, as well as store data in files. This is called file input and output. To be able to read data from files, the file needs to be opened, the file is then read (using classes such as Scanner or BufferedReader), and the file needs to be closed. A specific class needs to be imported using *import java.io.*;* to be able to do so. Not only that, a *throws IOException* clause needs to be added to the method header in order for this to work. Storing data in files involves the use of a PrintWriter class that has the ability to write data into a file, and requires the creation of a PrintWriter object. ProgramTwoCFour.java also shows an example of reading and writing files.