BREAK AND CONTINUE

The syntax of the `while` and `do..while` loops allows you to test the continuation condition at either the beginning of a loop or at the end. Sometimes, it is more natural to have the test in the middle of the loop, or to have several tests at different places in the same loop. Java provides a general method for breaking out of the middle of any loop. It's called the `break` statement, which takes the form

```
break;
```

When the computer executes a `break` statement in a loop, it will immediately jump out of the loop. It then continues on to whatever follows the loop in the program. Consider for example:

```
while (true) {  // looks like it will run forever!
   TextIO.put("Enter a positive number: ");
   N = TextIO.getlnInt();
   if (N > 0)   // input is OK; jump out of loop
      break;
   TextIO.putln("Your answer must be > 0.");
}
// continue here after break
```

If the number entered by the user is greater than zero, the `break` statement will be executed and the computer will jump out of the loop. Otherwise, the computer will print out "Your answer must be > 0." and will jump back to the start of the loop to read another input value.

The first line of this loop, "`while (true)`" might look a bit strange, but it's perfectly legitimate. The condition in a `while` loop can be any boolean-valued expression. The computer evaluates this expression and checks whether the value is `true` or `false`. The boolean literal "`true`" is just a boolean expression that

always evaluates to true. So "`while (true)`" can be used to write an infinite loop, or one that will be terminated by a `break` statement.

A `break` statement terminates the loop that immediately encloses the `break` statement. It is possible to have nested loops, where one loop statement is contained inside another. If you use a `break` statement inside a nested loop, it will only break out of that loop, not out of the loop that contains the nested loop. There is something called a labeled break statement that allows you to specify which loop you want to break. This is not very common, so I will go over it quickly. Labels work like this: You can put a label in front of any loop. A label consists of a simple identifier followed by a colon. For example, a `while` with a label might look like "`mainloop: while...`". Inside this loop you can use the labeled break statement "`break mainloop;`" to break out of the labeled loop. For example, here is a code segment that checks whether two strings, `s1` and `s2`, have a character in common. If a common character is found, the value of the flag variable `nothingInCommon` is set to `false`, and a labeled break is used to end the processing at that point:

```
boolean nothingInCommon;
nothingInCommon = true;  // Assume s1 and s2 have no chars in
common.
int i,j;  // Variables for iterating through the chars in s1
and s2.

i = 0;
bigloop: while (i < s1.length()) {
   j = 0;
   while (j < s2.length()) {
      if (s1.charAt(i) == s2.charAt(j)) { // s1 and s2 have a
common char.
          nothingInCommon = false;
          break bigloop;  // break out of BOTH loops
      }
```

```
            j++;  // Go on to the next char in s2.
        }
        i++;  //Go on to the next char in s1.
    }
```

The `continue` statement is related to `break`, but less commonly used. A `continue` statement tells the computer to skip the rest of the current iteration of the loop. However, instead of jumping out of the loop altogether, it jumps back to the beginning of the loop and continues with the next iteration (including evaluating the loop's continuation condition to see whether any further iterations are required). As with `break`, when a `continue` is in a nested loop, it will continue the loop that directly contains it; a "labeled continue" can be used to continue the containing loop instead.

`break` and `continue` can be used in `while` loops and `do..while` loops. They can also be used in `for` loops, which are covered in the next section. In Section 3.6, we'll see that `break` can also be used to break out of a `switch` statement. A `break` can occur inside an `if` statement, but in that case, it does **not** mean to break out of the `if`. Instead, it breaks out of the loop or `switch` statement that contains the `if`statement. If the `if` statement is not contained inside a loop or `switch`, then the `if` statement cannot legally contain a `break`. A similar consideration applies to `continue` statements inside `if`s.

Source : http://math.hws.edu/javanotes/c3/s3.html