# Transfer statements

Java allows us to use break and continue statements to transfer control from a loop or switch (break only) constructs.

## Objectives

Upon completion of this topic, we will be able to:

- Understand and use the 'break' statement
- Understand and use the 'continue' statement

# Break Statement

By using **break**, you can force the immediate termination of a loop, bypassing the conditional expression and any remaining code in the body of the loop. When a break statement is encountered inside a loop, the loop is terminated and program control resumes at the next statement following the loop. Here is a simple example:

```
// Using break to exit a loop

class BreakLoop {

        public static void main(String args[ ]) {

                for(int i=0; i<100; i++) {

                        if(i == 10) break; // terminate loop if i is 10

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

                }

                System.out.println("Loop complete.");

        }

}
```

This program generates the following output:

```
i: 0

i: 1

i: 2

i: 3
```

**i: 4**

**i: 5**

**i: 6**

**i: 7**

**i: 8**

**i: 9**

**Loop complete.**

As you can see, although the for loop is designed to run from 0 to 99, the break statement causes it to terminate early, when i equals to 10.

# Continue Statement

Sometimes it is useful to force an early iteration of a loop. That is, you might want to continue running the loop, but stop processing the remainder of the code in its body for this particular iteration. The continue statement performs such an action. In while and do-while loops, a continue statement causes control to be transferred directly to the conditional expression that controls the loop. In a for loop, control goes first to the iteration portion of the for statement and then to the conditional expression. For all the three loops, any intermediate code is bypassed.

Here is an example program that uses **continue** to cause two numbers to be printed on each line:

```
// Demonstrate continue.

class Continue {

        public static void main (String args[ ]) {

                for (int i=0; i<10; i++) {

                        System.out.print (i + " ");

                        if (i%2 == 0) continue;

                        System.out.println ("");

                }

        }

}
```

This code uses the % operator to check if 'I' is even. If that is the case, then the loop continues without printing a newline. Here is the output from this program:

**0 1**

**2 3**

**4 5**

**6 7**

**8 9**

As with the **break** statement, **continue** may specify a label to describe which enclosing loops to continue.

# Summary

Here are the key takeaways:

- The break statement can take the control away from any loop or switch-case constructs.
- The continue statement is used for skipping the remaining parts of the loop and to start a fresh iteration.