Java supports three jump statements **break, continue**, and **return**.

break

The **break** statement has three uses. First, it terminates a statement sequence in a **switch** statement. Second, it can be used to exit a loop. Third, it can be used as a structured form of **goto**.

By using **break**, you can force 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.

When used inside a set of nested loops, the **break** statement will only break out of the innermost loop.

Here are two important points to remember about **break**. First, more than one **break** statement may appear in a loop. Be careful. Too many **break** statements can destructure your code. Second, the **break** that terminates a **switch** statement affects only that **switch** statement and not any enclosing loops.

Using break as a Form of Goto

The **break** statement can also be used as a structured form of the **goto** statement. Java does not have a goto statement because it provides a way to branch in an arbitrary and unstructured way. There are places where the goto is a valuable and legitimate construcut for flow control. The goto can be useful when you are exiting from a deeply nested set of loops. To handle such situations, Java defines an expanded form of the **break** statement. By using **break** you can break out of one or more blocks. These blocks don't need to be part of a loop or a **switch**. They can be any block. You can specify precisely where execution will resume, because this form of **break** works with a label.

Here is the general form of the labeled **break** statement:

`break` *label*;

Most often, *label* is the name of a label that identifies a block of code. Than be a standalone block of code but it can also be a block that is the target of another statement, When this form of **break** executes, control is transferred out of the named block. The labeled block must enclose the **break** statement, but it does not need to be the immediately enclosing block. This means, for example, that you can use a labeled **break** statement to exit from a set of nested blocks. But you cannot use **break** to transfer control out of a block that does not enclose the **break** statement.

To name a block, put a label at the start of it. A *label* is any valid Java identifier followed by a colon. Once you have labeled a block, you can then use this label as the target of a **break** statement. Doing so causes execution to resume at the *end* of the labeled block.