JAVA SWITCH STATEMENT

Description

A programming language uses control statements to cause the flow of execution to advance and branch based on changes to the state of a program. Java supports two flow control statements: if and switch. These statements allow you to control the flow of your program's execution based upon conditions known only during run time. We have discussed if statement in logical operator tutorial.

Switch Statement

The switch statement is Java's multiway branch statement. It provides an easy way to dispatch execution to different parts of your code based on the value of an expression. Here is the general form of a switch statement:

```
switch (expression) {

case value1:

// statement sequence

break;

case value2:

// statement sequence
```

```
break;

….

case valueN:

// statement sequence

break;

default:

// default statement sequence

}
```

The expression must be of type byte, short, int, or char; each of the values specified in the case statements must be of a type compatible with the expression. An enumeration value can also be used to control a switch statement. From Java 7 onward String is also allowed as case expression. Each case value must be a unique literal (that is, it must be a constant, not a variable). Duplicate case values are not allowed.

How switch statement works

The value of the expression is compared with each of the literal values in the case statements. If a match is found, the code sequence following that case statement is executed. If none of the constants matches the value of the expression, then the

default statement is executed. However, the default statement is optional. If no case matches and no default is present, then no further action is taken.

The break statement is used inside the switch to terminate a statement sequence. When a break statement is encountered, execution branches to the first line of code that follows the entire switch statement.

Nested switch Statements

You can use a switch as part of the statement sequence of an outer switch. This is called a nested switch. Since a switch statement defines its own block, no conflicts arise between the case constants in the inner switch and those in the outer switch.

Important Points related to Switch-Case statements:

- The switch can only check for equality. This means that the other relational operators such as greater than are rendered unusable in a case.
- Case constants are evaluated from the top down, and the first case constant that matches the switch's expression is the execution entry point. If no break statement used, all the case after entry point will be executed.
- No two case constants in the same switch can have identical values. Of course, a switch statement and an enclosing outer switch can have case constants in common.

- The default case can be located at the end, middle, or top. Generally default appears at end of all cases.

Break Statements

The break statement included with each case section determines when to stop executing statements in response to a matching case. Without a break statement in a case section, after a match is made, the statements for that match and all the statements further down the switch are executed until a break or the end of the switch is found. In some situations, this might be exactly what you want to do. Otherwise, you should include break statements to ensure that only the right code is executed. It is sometimes desirable to have multiple cases without break statements between them.

Summary

- The switch statement is Java's multi-way branch statement.
- The switch can only check for equality. This means that the other relational operators such as greater than are rendered unusable in a case.
- Break statement is used to stop current iteration of loop or end Switch-case block.

Source: http://www.w3resource.com/java-tutorial/java-switch-break-continue-statements.php