**Here are the rules for a switch statement:**

**1. Only String, byte, char, short, int, (and their wrapper classes Byte, Character, Short, and Integer), and enums can be used as types of a switch variable. (String is allowed only since Java 7).**

**2. The case constants must be assignable to the switch variable. For example, if your switch variable is of class String, your case labels must use Strings as well.**

**3. The switch variable must be big enough to hold all the case constants. For example, if the switch variable is of type char, then none of the case constants can be greater than 65535 because a char's range is from 0 to 65535.**

**4. All case labels should be COMPILE TIME CONSTANTS.**

**5. No two of the case constant expressions associated with a switch statement may have the same value.**

**6. At most one default label may be associated with the same switch statement.**

**Legal Expressions for switch and case**

The general form of the switch statement is

switch (expression) {

case constant1: code block

case constant2: code block

default: code block

}

Expression must be a char,byte,short,int , (and their wrapper classes Byte, Character, Short, and Integer) enum or String or

A case constant

A case constant must evaluate to the same type that the switch expression can

use, with one additional—and big—constraint: the case constant must be a

compile-time constant! Since the case argument has to be resolved at compile time,

you can use only a constant or final variable that is immediately initialized with a

literal value. It is not enough to be final; it must be a compile time *constant*. Here's

an example:

A switch's expression must evaluate to a char, byte, short, int, an enum (as of

Java 5), and a String (as of Java 7). That means if you're not using an enum or a

String, only variables and values that can be automatically promoted (in other

words, implicitly cast) to an int are acceptable.