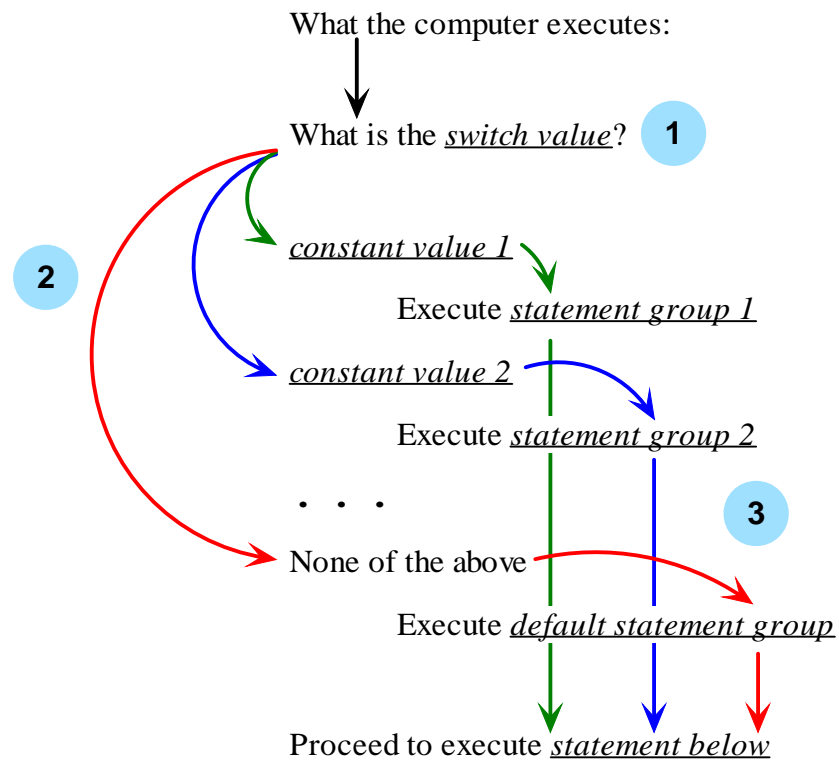# THE JAVA SWITCH STATEMENT

Like the **if-else** statement, the **switch** statement introduces a "fork in the road" for the program's execution path. Unlike the **if-else** (1) the switch statement's fork has many tines and (2) these execution paths are not necessarily separate because the flow of execution can end up moving through several of them.

Java Switch Statement Syntax	

switch (_switch value_ **)**
 {
    case _constant value 1_ **:**
       statement group 1
    case _constant value 2_ **:**
       statement group 2
    . . .
 }
 statement below
	switch (_switch value_ **)**
	 {
	    case _constant value 1_ **:**
	       statement group 1
	    case _constant value 2_ **:**
	       statement group 2
	    . . .
	    default :
	       default statement group
	 }
	 statement below

switch value is a variable or expression that is evaluated at run-time. It must be of the data type **byte**, **short**, **int**, **long, char** or, as of Java 7, **String**.

Each _constant value_ must be determinable at compile time.[1] Also, each must be of the same data type as the _switch value_ and no two of them can have the same value. It can be a literal, a previously declared constant identifier or an expression involving literals and constant identifiers.

---

[1] See Programming 102 – Coding and Testing   Program Readability, p. 3.

Java Switch Statement Semantics

What the computer executes:

What is the *switch value*? **1**

2

constant value 1

Execute *statement group 1*

constant value 2

Execute *statement group 2*

. . .

3

None of the above

Execute *default statement group*

Proceed to execute *statement below*

As illustrated in the picture, the computer begins execution of the **switch** statement by (1) evaluating the *switch value*. The computer (2) branches to the *statement group* immediately following the *constant value* that matches the *switch value* and (3) continues executing each subsequent *statement group*. If no *constant value* matches the *switch value* then execution branches to the *default statement group*, if there is one.

This code illustrates the semantics of the **switch** statement without a **default** part. The **switch** statement takes an integer representing a month (1 for January, 2 for February, etc.) and calculates the number of days in all the months prior to and including it.

```
 1  int totalDays = 0;
 2  switch ( month )
 3  {
 4     case 12:
 5        totalDays += 31;
 6     case 11:
 7        totalDays += 30;
 8     case 10:
 9        totalDays += 31;
10     case 9:
11        totalDays += 30;
12     case 8:
13        totalDays += 31;
14     case 7:
15        totalDays += 31;
16     case 6:
17        totalDays += 30;
18     case 5:
19        totalDays += 31;
20     case 4:
21        totalDays += 30;
22     case 3:
23        totalDays += 31;
24     case 2:
25        totalDays += 28;
26     case 1:
27        totalDays += 31;
28  }
29  System.out.println( "days = " + totalDays );
```

If **month** is 4, the **switch** executes lines 21-27. The output is:	If **month** is 12, the **switch** executes lines 5-27. The output is:	If **month** is 13, the **switch** skips everything. The output is:
`days = 120`	`days = 365`	`days = 0`

This code illustrates the use of the **break** statement to insure that each *statement group* is executed at most once. The **switch** statement takes a **String** object representing a student's grade and prints his or her award.

```
 1  switch ( grade )
 2  {
 3     case "A+" :
 4        System.out.println( "Highest honors" );
 5        break;
 6     case "A" :
 7     case "A-":
 8        System.out.println( "Honors" );
 9        break;
10     case "B+":
11     case "B" :
12        System.out.println(  "Favorable Mention" );
13        break;
14  }
15  . . .
```

If **grade** is **A+**, the **switch** executes lines 4 and 5. Line 5 breaks to line 15. The output is:	If **grade** is **A-**, the **switch** executes lines 8 and 9. Line 9 breaks to line 15. The output is:	If **grade** is **B+**, the **switch** executes lines 12 and 13. Line 13 breaks to line 15. The output is:
`Highest honors`	`Honors`	`Favorable Mention`

In the example above, the **break** at line 13 is not necessary since (without it) line 15 is the next to execute. A seasoned Java programmer includes it, anticipating that he or she may later add additional cases to the **switch** statement.

This code illustrates the semantics of the **switch** statement with a **default** part. The switch statement takes an integer representing a month (1 for January, 2 for February, etc.) and sets the variable **lastDate** to its last date, following the old rhyme:

> *Thirty days hath September, April, June and November.*
> *All the rest have 31, save February alone,*
> *Which has 28 and, in leap year, 29.*

```java
 1  switch ( month )
 2  {
 3     case  9:
 4     case  4:
 5     case  6:
 6     case 11:
 7        lastDate = 30;
 8        break;
 9     case 2:
10        if ( new GregorianCalendar( ).isLeapYear( year ) )
11           lastDate = 29;
12        else
13           lastDate = 28;
14        break;
15     default:
16        lastDate = 31;
17        break;
18  }
19  System.out.println( "last date = " + lastDate );
```

If **month** is 9, the code executes lines 7, 8 and 19, outputting:	If **month** is 2, the code executes lines 10, 11, 14 and 19, outputting, if **year** is a leap year:	If **month** is 12, the code executes lines 16, 17 and 19, outputting:
`last date = 30`	`last date = 29`	`last date = 31`

Beginner Errors on Switch Statements

A common beginner error is to forget to break out of a case; thus, executing all the subsequent cases.

Example

The code segment below (incorrectly) attempts to simulate the toss of a coin by choosing 0 or 1 at random and setting **coinToss** to *HEADS* or *TAILS*, respectively. **coinToss** always comes out *TAILS*. If 0 is chosen, lines 4 and 6 are executed. If 1 is chosen, line 6 is executed.

```
1    switch ( (int)(Math.random( )*2) )
2    {
3       case 0:
4          coinToss = "HEADS";
5       case 1:
6          coinToss = "TAILS";
7    }
```

Another common error is to try to **switch** on a variable that is not of the correct data type.

Example

For the code fragment below, the compiler issues the diagnostic:

```
MyApp.java:30: error: incompatible types
    switch (success)
            ^
      required: int
      found:    boolean
```

```
10  boolean success;
~     . . .
30  switch ( success )
31  {
~       . . .
44  }
```

Exercises

For each code fragment below, give the output if the **char** variable **letter** has the value **a**. If **letter** has the value **b**. If **letter** is **c**. If **letter** is **d**. If **letter** is **e**.

1.
```java
char letter;
. . .
int  r = 0;
switch ( letter )
{
   case 'a':
      r += 1;
   case 'b':
   case 'c':
      r += 2;
   case 'd':
      r += 3;
}
System.out.println( "r = " + r );
```

2.
```java
char letter;
. . .
int  r = 0;
switch ( letter )
{
   case 'a':
      r += 1;
      break;
   case 'b':
   case 'c':
      r += 2;
      break;
   case 'd':
      r += 3;
      break;
}
System.out.println( "r = " + r );
```

For each code fragment below, give the output if the **char** variable **letter** has the value **a**. If **letter** has the value **b**. If **letter** is **c**. If **letter** is **d**. If **letter** is **e**.

3.
```
char letter;
.  .  .
int r = 0;
switch ( letter )
{
   case 'a':
      r += 1;
      break;
   case 'b':
   case 'c':
      r += 2;
      break;
   case 'd':
      r += 3;
      break;
   default:
      r += 4;
      break;
}
System.out.println( "r = " + r );
```

For each **switch** statement below, circle what's wrong and explain. None of them is correct.

4.
```
int code;
.  .  .
Switch ( code ) ;
{
   Case 0:
      msg = "System operating normally";
      break;
   Case 1:
      msg = "System startup error";
      break;
}
```

For each **switch** statement below, circle what's wrong and explain. None of them is correct.

5.
```java
short code;
. . .
switch ( code )
   case 0
      msg = "System operating normally";
      break;
   case 1
      msg = "System startup error";
      break;
```

6.
```java
final byte ON = 0;
final byte OFF = 1;
byte code;
. . .
switch code {
   case ON:
      msg = "System operating normally";
      break;
   case OFF:
      msg = "System startup error";
      break;
}
```

7.
```java
double code;
. . .
switch ( code ) {
   case 0: msg = "System operating normally";
      break;
   case 1: msg = "System startup error";
      break;
}
```

For each **switch** statement below, circle what's wrong and explain. None of them is correct.

8.
```java
int code, on, off;
. . .
switch ( code )
{
    case on:
        msg = "System operating normally";
        break;
    case off:
        msg = "System startup error";
        break;
}
```

9. Explain the logic error in this **switch** statement.

```java
char code;
. . .
code = '1';
. . .
switch ( code )
{
    case 0:
        msg = "System operating normally";
        break;
    case 1:
        msg = "System startup error";
        break;
}
```

For each of the following write the Java code fragment that uses a **switch** statement to accomplish the result.

10.	Given an **int** variable **day** equal to 1 through 7, set a string variable to the name of the weekday Sunday, Monday, etc.
11.	Given the **String** variable **day** holding the name of the weekday Sunday, Monday, etc., set an **int** variable to 1, 2, etc.

12.	Randomly select an integer 0, 1, 2 or 3 and set a string variable to a card suit (Hearts, Clubs, Spades or Diamonds) based on the random selection.
13.	Given the **String** variable **suit** holding the name of a card suit (Hearts, Clubs, Spades or Diamonds) set an **int** variable to 0, 1, 2 or 3 based on the selection.
14.	Given the **char** variable **letter**, the **int** variable **vowelCount** and the **int** variable **consonantCount**, increment **vowelCount** if **letter** contains a vowel and **consonantCount** if letter contains a consonant.
15.	Given an **int** variable **month** equal to 1, …, 12, set a string variable to the name of the month January, February, etc.