

Shortcuts: switch

Multiple tests on the same expression [supp]

Now we'll look at the second method of choosing among different options, the `switch` statement

switch

Syntax:

```
switch ( testvalue ) {  
    case value1 : statement1 [ break; ]  
    case value2 : statement2 [ break; ]  
    ...  
    [ default : statement ]  
}
```

The *switch* statement is an ideal way to choose among many options. Given the *testvalue*, in the *case* that it takes a given *value*_{*i*}, execute *statement*_{*i*}.

inside switch

switch: The reserved word to say we're using a switch statement

case: A given case, corresponding to a given value, matched exactly. Any number of cases are permitted, and they are tested in order. The same value shouldn't occur in multiple cases (it's a compile-error in fact).

break: Optional word at the end of each case statement, instructing the program to leave the *switch* statement. If absent, then continue with the next case statement.

default: Optional, covers all cases not previously found.

switch example

Here's a simple *switch* statement to show you how they work:

```
1  public class LetterSwitch {
2      public static void main(String [] args) {
3          int score = 0;
4          char [] ch = new char[] { 'a', 'b', 'c', 'd', 'e' };
5          for (int i = 0; i < ch.length; ++i) {
6              switch (ch[i]) {
7                  case 'a': score = 1; break; // skip the other cases
8                  case 'b': score = 2; break;
9                  case 'c': score = 3; break;
10                 case 'd': break;
11                 default: return; // if ch isn't a,b,c,d then return 0
12             }
13             System.out.println(score);
14         }
15     }
16 }
```

...which produces:

```
~> javac LetterSwitch.java
~> java LetterSwitch
1
2
3
3
```

switch — case

- The value put after the case reserved word must be a primitive type like int, short, byte. As of Java 7 you can also use String, but *don't do it for your assessments!*
- If the *testvalue* is equal to the case value then that statement is executed.
- Cases don't have to be in any particular order, but they are *tested* in order.

switch — break

The break keyword is required if you want to skip the rest of the switch statement.

If you don't have the break there, then the next statement will be executed, like this:

```
1 public class ScrabbleSwitch {
2     public static void main(String[] args) {
3         if (args.length < 0) {
4             return; // this finishes the main method, so ends the progr
5         }
6         char ch = args[0].charAt(0); // note the [] and ()
7         int score = 0;
8         switch (ch) {
9             case 'a': case 'e': case 'i':
10             case 'l': case 'n': case 'o':
11             case 'r': case 's': case 't':
12             case 'u': // all cases a,e,i,l,n,o,r,s,t,u
13                 score = 1;
14                 // break; // the break is commented out
15             case 'd': case 'g':
16                 score = 2;
```


switch — break (cont.)

```
17         break;
18     default:
19         score = 0;
20     }
21     System.out.println("score = " + score);
22 }
23 }
```

switch — break (cont.)

```
~> javac ScrabbleSwitch.java
~> java ScrabbleSwitch Hello
score = 0
~> java ScrabbleSwitch hello
score = 0
~> java ScrabbleSwitch ello
score = 2
```

switch — default

The last case to be executed will be the default case. You don't need to give it a value for comparison, just the keyword default.

default is *optional*:

```
1  public class SwitchBrief {
2      public static void main(String[] args) {
3          if (args.length < 0) {
4              return;
5          }
6          char ch = args[0].charAt(0);
7          switch (ch) {
8              case 'a':
9                  System.out.println("Eh?");
10             case 'y':
11                 System.out.println("Why?");
12             default:
13                 System.out.println("Because.");
14         }
15     }
16 }
```

```
~> java SwitchBrief Hello!
```

Because.

```
~> java SwitchBrief Why?
```

Because.

```
~> java SwitchBrief yes
```

Why?

Because.

```
~> java SwitchBrief also
```

Eh?

Why?

Because.

You can miss out quite a bit from the switch statement with no compilation errors – the following is fine:

```
1 public class SwitchEmpty {  
2     public static void main(String [] args) {  
3         int i = 5;  
4         switch (i) {  
5             }  
6     }  
7 }
```

Doesn't do anything interesting though.

common switch error

Don't forget the break!




Kit-Kat is a trademark of Nestlé, who are not sponsoring this Unit.



If you miss the break then execution will “fall through” to the next case. This may be what you want, but it may not. *Begin* by putting the break in, and then remove it only if you’re really really sure.

switch or if/else/if ?

Sometimes it's not clear which to use: `switch` or some `ifs` and `elses`. Neither is necessarily “correct” in such cases, so here are some guidelines that might help you choose:

- cases in the *switch* cannot check a range of values, only equality (unlike for example `if (x < 5)`);
- `switch` can be very compact (see previous examples)
- A control flow statement (`switch`, `if/else/if` etc.) that hides the logical structure is a poor choice
-  A structure that is not very general, such as using `Strings` in the `switch` when you can't guarantee the version of Java, is a *very bad idea*.