break expression in Kotlin

In Kotlin, `break`, `continue` and `return` are called structural jump expression. In this tutorial, you will learn break statement along with its working. Also, you will learn labeled break.

Sometimes, it is required to terminate a loop on the basis of the certain condition. In that situation, break keyword is used. It terminates the nearest enclosing loop and works similar to break statement in Java.

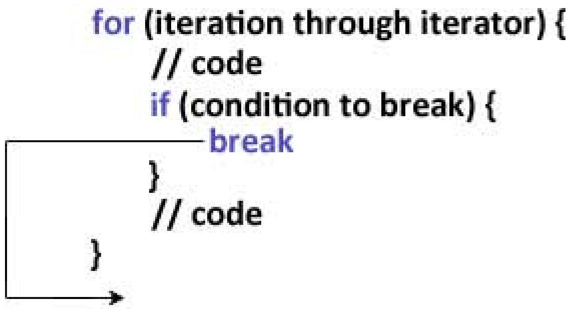
Working of break statement

It is mostly used with if statement.

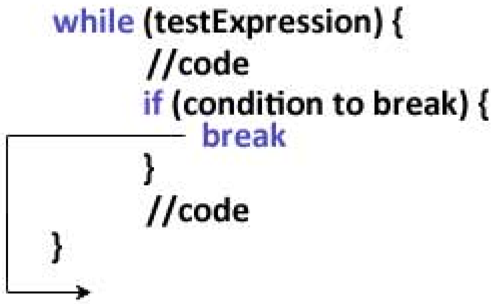
```
while(condition){
   //code to be executed
   if(breakCondition){
      break
   }
   //code to be executed
}
```

When while loop executes and breakCondition present inside it evaluates to true, break statement is executed and while loop is terminated.

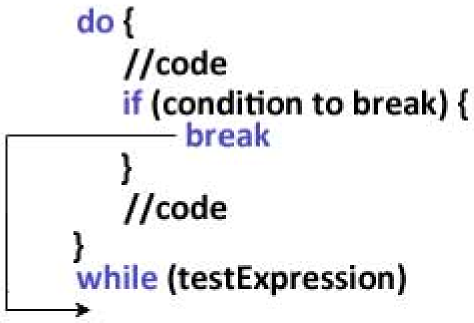
Similarly, you can use break with for and do-while loop.



```
for (iteration through iterator) {
    // code
    if (condition to break) {
        break
    }
    // code
}
```



```
while (testExpression) {
        //code
        if (condition to break) {
            break
        }
        //code
}
```



```
do {
    //code
    if (condition to break) {
        break
    }
    //code
}
while (testExpression)
```

Example of break expression

```
fun main(arr: Array<String>){
   for(i in 1..10){
      if(i==6){
         break
      }
      println(3*i)
   }
}
```

On running the above program, you will get the below output-

```
3
6
9
12
15
```

When the value of `i` is equal to 6, the condition of `if` statement is true, and break statement is executed. And then `for` loop is terminated.

Labeled break in Kotlin

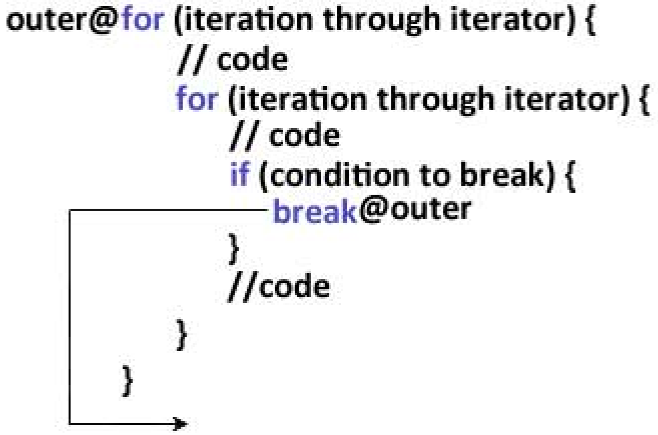
Have you ever thought rather than terminating the nearest enclosing loop, you want to terminate a different loop? If yes, then you need to study labeled break.

An identifier followed by the @ sign is called a label. For example- outer@, inner@, etc. are some valid labels in Kotlin. You can apply a label to any expression in Kotlin. To label an expression, all you need to do is write label in front of it. Let's see how labeled break works with the help of examples-

Syntax of labeled break

```
break@label
```

Working of labeled break



```
outer@for (iteration through iterator) {
            // code
            for (iteration through iterator) {
                // code
                if (condition to break) {
                    break@outer
                }
                //code
            }
        }
```

As you can see in the above image, when condition inside while loop evaluates to true, then `breakouter@`) is executed and while loop marked with label `outer@` is terminated.

Example of labeled break

```
fun main(arr: Array<String>){
   outer@ for(i in 1..10){
      inner@ for(j in 1..5){
         println("$i $j")
         if(i==3){
                break@outer
         }
      }
   }
}
```

Output

```
1 1
1 2
1 3
1 4
1 5
2 1
2 2
2 3
2 4
2 5
3 1
```