Kotlin – Control Statements

Tushar B. Kute,
http://tusharkute.com

Loop Statements

- Imagine a situation when you need to print a sentence 20 times on your screen. You can do it by using print statement 20 times.
- What about if you need to print the same sentence one thousand times?
- This is where we need to use loops to simplify the programming job. Actually, Loops are used in programming to *repeat* a specific block of code until certain condition is met.

for Loop

- Kotlin for loop iterates through anything that provides an iterator ie. that contains a countable number of values, for example arrays, ranges, maps or any other collection available in Kotlin.
- Kotlin for loop is equivalent to the foreach loop in languages like C#.
  - Kotlin does not provide a conventional for loop which is available in C, C++ and Java etc.

for Loop

- The syntax of the Kotlin for loop is as follows:

```
for (item in collection) {
    // body of loop
}
```

for Loop

- Iterate Through a Array
  - Array is another data type which provides iterator, so we can use for loop to iterate through a Kotlin array in the similar way as we did it for the ranges.

While loop

- while loop executes its body continuously as long as the specified condition is true.
- Kotlin while loop is similar to Java while loop.
  - Syntax
  -
```
while (condition) {
    // body of the loop
}
```

do-while loop

- The do..while is similar to the while loop with a difference that the this loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.
- The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested.

Break statement

- Kotlin break statement is used to come out of a loop once a certain condition is met.
- This loop could be a for, while or do...while loop.

Break statement

```
// Using break in for loop
for (...) {
   if(test){
      break
   }
}
// Using break in while loop
while (condition) {
   if(test){
      break
   }
}
// Using break in do...while loop
do {
   if(test){
      break
   }
}while(condition)
```

Labeled Break statement

- Kotlin label is the form of identifier followed by the @ sign, for example test@ or outer@.
- To make any Kotlin Expression as labeled one, we just need to put a label in front of the expression.

```
outerLoop@ for (i in 1..100) {
    // ...
}
```

- Kotlin labeled break statement is used to terminate the specific loop. This is done by using break expression with @ sign followed by label name (break@LabelName).

continue Statement

- The Kotlin continue statement breaks the loop iteration in between (skips the part next to the continue statement till end of the loop) and continues with the next iteration in the loop.

continue Statement

```
// Using continue in for loop
for (...) {
   if(test){
      continue
   }
}
// Using continue in while loop
while (condition) {
   if(test){
      continue
   }
}
// Using continue in do...while loop
do {
   if(test){
      continue
   }
}while(condition)
```

Labeled continue Statement

- Kotlin labeled continue statement is used to skip the part of a specific loop.
- This is done by using continue expression with @ sign followed by label name (continue@LabelName).

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com