Kotlin Standard Input/Output

Kotlin standard input output operations are performed to flow byte stream from input device (**keyboard**) to main memory and from main memory to output device (**screen**).

Kotlin Output

Kotlin output operation is performed using the standard methods print() and println().

Let's see an example:

```
fun main(args: Array<String>) {
    println("Hello World!")
    print("Welcome to  Tutorials of Kotlin")
}
```

Difference between print() and println() methods:

- print(): print() method is used to print values provided inside the method "()".
- println(): println() method is used to print values provided inside the method "()" and moves cursor to the beginning of next line.

Kotlin Input

Kotlin has standard library function **readLine()** which is used for reads line of string input from standard input stream.
It returns the line read or null.
Let's see an example:

```
fun main(args: Array<String>) {
    println("Enter your name")
    val name = readLine()
    println("Enter your age")
    var age: Int =Integer.valueOf(readLine())
    println("Your name is $name and your age is $age")
}
```

Example Getting Integer Input

```
import java.util.Scanner
fun main(args: Array<String>) {
    val read = Scanner(System.`in`)
    println("Enter your age")
    var age = read.nextInt()
    println("Your input age is "+age)
}
```

Kotlin Control Flow

Kotlin if Expression

In Kotlin, if is an expression is which returns a value. It is used for control the flow of program structure. There is various type of if expression in Kotlin.

- if-else expression
- if-else if-else ladder expression
- nested if expression

Kotlin if-else Expression Example

```
fun main(args: Array<String>) {
        val num1 = 10
        val num2 =20
        val result = if (num1 > num2) {
            "$num1 is greater than $num2"
        } else {
            "$num1 is smaller than $num2"
        }
        println(result)
}
```

We can remove the curly braces of if-else body by writing if expression in only one statement.

```
fun main(args: Array<String>) {
      val num1 = 10     val num2 =20
      val result = if (num1 > num2) "$num1 is greater than $num2"
else "$num1 is smaller than $num2"
     println(result)
}
```

Kotlin if-else if-else Ladder Expression

```
fun main(args: Array<String>){
    val num = 10
    val result = if (num > 0){
        "$numispositive"
    }elseif(num<0){
        "$numisnegative"
    }else{
        "$numiszero"
    }
    println(result)
}
```

Kotlin Nested if Expression

```
fun main(args: Array<String>){
    val  num1=25
    val  num2=20
    val  num3=30
    val  result=if(num1>num2){
        val  max=if(num1>num3){
                num1
            }else{
                num3
            }
            "bodyofif"+max
        }elseif(num2>num3){
            "bodyofelseif"+num2
        }else{
            "bodyofelse"+num3
        }

        println("$result")
}
```

Kotlin when Expression

Kotlin, when expression is a conditional expression which returns the value. Kotlin, when expression is replacement of switch statement.
Kotlin, when expression works as a switch statement of other language (Java, C++, C).

```
fun  main(args: Array<String>){
     var  number=4
     var  numberProvided=when(number){
          1->"One"
          2->"Two"
          3->"Three"
          4->"Four"
          5->"Five"
          else->"invalidnumber"
     }
     println("You provide $numberProvided")
}
```

Using when Without Expression

```
fun main(args:Array<String>){
     var number=4
     when(number){
          1->println("One")
          2->println("Two")
          3->println("Three")
          4->println("Four")
          5->println("Five")
          else->println("invalidnumber")
     }

}
```

Multiple Statement of when Using Braces

```
fun main(args:Array<String>){
      var number=1
      when(number){
            1->{
                  println("Monday")
                  println("First day of the week")
            }
            7->println("Sunday")
            else->println("Other days")
      }
}
```

Multiple branches of when

```
      fun main(args: Array<String>){
            var number=8
            when(number){
                  3,4,5,6->
                        println("It is summer season")
                  7,8,9->
                        println("It is rainy season")
                  10,11->
                        println("It is autumn season")
                  12,1,2->
                        println("It is winter season")
                  else->println("invalid input")
            }
      }
```

Using when in the range

```
fun main(args:Array<String>){
      var number=7
      when(number){
            in 1..5->println("Input is provided in the range 1 to 5")
            in 6..10->println("Input is provided in the range 6 to 10")
            else->println("none of the above")
      }
}
```

Kotlin for Loop

- Kotlin for loop is used to iterate a part of program several times.
- It iterates through arrays, ranges, collections, or anything that provides for iterate.
- Kotlin for loop is equivalent to the **foreach** loop in languages like C#.

Syntax of for loop in Kotlin:

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

Iterate through array

```
fun main(args: Array<String>){
     val marks=arrayOf(80,85,60,90,70)
     for(item in marks){
          println(item)
     }
}
```

The elements of an array are iterated on the basis of indices (index) of array. For example:

```
fun main(args: Array<String>){
     val marks=arrayOf(80,85,60,90,70)
     for(item in marks.indices)
          println("marks[$item]:"+marks[item])
}
```

Iterate through range

```
fun main(args: Array<String>){
      print("for(iin1..5)print(i)=")
      for(i in 1..5) print(i)

      println()
      print("for(i in 5..1) print(i)=")
      for(i in 5..1) print(i)//prints nothing

      println()
      print("for(i in 5 downTo 1) print(i)=")
      for(i in 5 downTo 1) print(i)

      println()
      print("for(i in 5 downTo 2) print(i)=")
      for(i in 5 downTo 2) print(i)
      println()

      print("for(i in 1..5 step 2) print(i)=")
      for(i in 1..5 step 2) print(i)
      println()

      print("for(i in 5 downTo 1 step 2)print(i)=")
      for(i in 5 downTo 1 step 2) print(i)
}
```

Output:

```
for (i in 1..5) print(i) = 12345
for (i in 5..1) print(i) =
for (i in 5 downTo 1) print(i) = 54321
for (i in 5 downTo 2) print(i) = 5432
for (i in 1..5 step 2) print(i) = 135
for (i in 5 downTo 1 step 2) print(i) = 531
```

Kotlin while Loop

Syntax:

```
while(condition){
        //body of loop
}
```

Example:

```
fun main(args: Array<String>){
    var i=1
    while(i<=5){
        println(i)
        i++
    }
}
```

Kotlin do-while Loop

Syntax:

```
do{
//body of do block
}while(condition);
```

Example:

```
fun main(args:Array<String>){
    var i=1
    do{
        println(i)
        i++
    }while(i<=5);
}
```

Kotlin Return and Jump

There are three jump expressions in Kotlin.
These jump expressions are used for control the flow of program execution.
These jump structures are:

- break
- continue
- return

Break Expression:

A break expression is used for terminate the nearest enclosing loop.
It is almost used with if-else condition.

For example:

```
for(..){
     //bodyoffor
     if(checkCondition){
          break;
     }
}
```

In the above example, for loop terminates its loop when if condition execute break expression.

Kotlin break example:

```
fun main(args: Array<String>){

     for(i in 1..5){
          if(i==3){
               break
          }
          println(i)
     }
}
```

Kotlin Labeled break Expression

Kotlin labeled break example

```
fun main(args: Array<String>){
    loop@ for(iin1..3){
            for(j in 1..3){
                println("i=$iandj=$j")
                if(i==2)
                    break @loop
            }
    }
}
```

Kotlin continue Jump Structure

Kotlin, continue statement is used to repeat the loop. It continues the current flow of the program and skips the remaining code at specified condition.

The continue statement within a nested loop only affects the inner loop.

For example

```
for(..){
    //body of for above if
    if(checkCondition){
        continue
    }
    //body of for below if
}
```

In the above example, for loop repeat its loop when if condition execute continue. The continue statement makes repetition of loop without executing the below code of if condition.

Kotlin continue example

```
fun main(args: Array<String>){
    for(i in 1..3){
        println("i=$i")
        if(j==2){
            continue
        }
        println("this is below if")
    }
}
```

Kotlin Labeled continue Expression

Labeled is the form of identifier followed by the @ sign, for example abc@, test@. To make an expression as label, we just put a label in front of expression.

Kotlin, labeled continue expression is used for repetition of specific loop (labeled loop). This is done by using continue expression with @ sign followed by label name (continue@labelname).

Kotlin labeled continue example

```
fun main(args: Array<String>){
    labelname@ for(i in 1..3){
        for(j in 1..3){
            println("i=$i and j=$j")
            if(i==2){
                continue@labelname
            }
            println("this is below if")
        }
    }
}
```

Refer:https://kotlinlang.org/docs/reference/coding-conventions.html

https://kotlinlang.org/docs/reference/basic-syntax.html

https://codelabs.developers.google.com/codelabs/build-your-first-android-app-kotlin/index.html#0