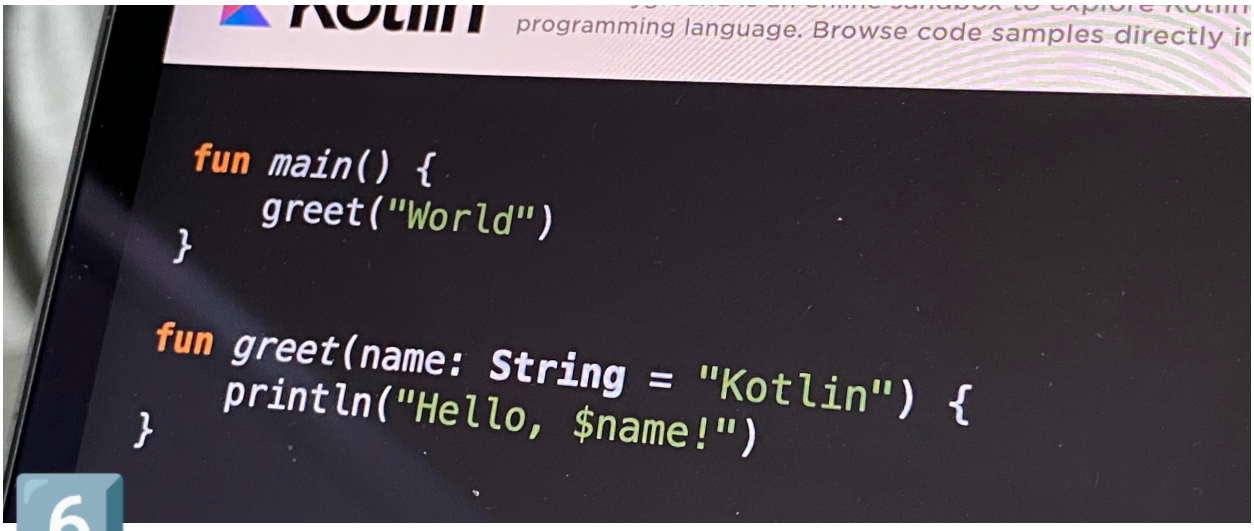

Class 6

Kotlin Type Conversion

In Kotlin, a numeric value of one type is not automatically converted to another type even when the other type is larger. This is different from how Java handles numeric conversions. For example;

In Java,

```
int number1 = 55;
long number2 = number1;    // Valid code
```

Here, value of `number1` of type `int` is automatically converted to type `long`, and assigned to variable number2.

In Kotlin,

```
val number1: Int = 55
val number2: Long = number1   // Error: type mismatch.
```

Though the size of Long is larger than `Int`, Kotlin doesn't automatically convert Int to Long.

Instead, you need to use `toLong()` explicitly (to convert to type Long). Kotlin does it for type safety to avoid surprises.

```
val number1: Int = 55
val number2: Long = number1.toLong()
```

Here's a list of functions in Kotlin used for type conversion:

- `toByte()`
- `toShort()`
- `toInt()`
- `toLong()`
- `toFloat()`
- `toDouble()`
- `toChar()`

Note, there is no conversion for `Boolean` types.

Conversion from Larger to Smaller Type

The functions mentioned above can be used in both directions (conversion from larger to smaller type and conversion from smaller to larger type).

However, conversion from larger to smaller type may truncate the value. For example,

```
fun main(args : Array<String>) {
    val number1: Int = 545344
    val number2: Byte = number1.toByte()
    println("number1 = $number1")
    println("number2 = $number2")
}
```

When you run the program, the output will be:

```
number1 = 545344
```

```
number2 = 64
```

Also check out these articles related to type conversion:

- *String to Int, and Int to String Conversion*
- *Long to Int, and Int to Long Conversion*
- *Double to Int, and Int to Double Conversion*
- *Long to Double, and Double to Long Conversion*
- *Char to Int, and Int to Char*
- *String to Long, and Long to String Conversion*
- *String to Array, and Array to String Conversion*
- *String to Boolean, and Boolean to String Conversion*
- *String to Byte, and Byte to String Conversion*
- *Int to Byte, and Byte to Int Conversion*

Kotlin Basic Input/Output

Koltin Output

You can use `println()` and `print()` functions to send output to the standard output (screen). Let's take an example:

```
fun main(args : Array<String>) {
    println("Kotlin is interesting.")
}
```

When you run the program, the output will be:

```
Kotlin is interesting.
```

Here, `println()` outputs the string (inside quotes).

Difference Between println() and print()

- `print()` - prints string inside the quotes.
- `println()` - prints string inside the quotes similar like `print()` function. Then the cursor moves to the beginning of the next line.

When you use `println()` function, it calls `System.out.println()` function internally. (`System.out.println()` is used to print output to the screen in Java).

If you are using IntelliJ IDEA, put your mouse cursor next to `println` and go to `Navigate` > `Declaration` (Shortcut: **Ctrl + B** . For Mac: **Cmd + B**), this will open `Console.kt` (declaration file). You can see that `println()` function is internally calling `System.out.println()`.

Similarly, when you use `print()` function, it calls `System.out.print()` function.

Example 1: print() and println()

```
fun main(args : Array<String>) {
    println("1. println ");
    println("2. println ");

    print("1. print ");
    print("2. print");
}
```

When you run the program, the output will be:

```
1. println
2. println
1. print 2. print
```

Example 2: Print Variables and Literals

```
fun main(args : Array<String>) {
    val score = 12.3

    println("score")
```

```
    println("$score")
    println("score = $score")
    println("${score + score}")
    println(12.3)
}
```

When you run the program, the output will be:

```
score
12.3
score = 12.3
24.6
12.3
```

Kotlin Input

In this section, you will learn to take input from the user..

To read a line of string in Kotlin, you can use `readline()` function.

Example 3: Print String Entered By the User

```
fun main(args: Array<String>) {
    print("Enter text: ")

    val stringInput = readLine()!!
    println("You entered: $stringInput")
}
```

When you run the program, the output will be:

```
Enter text: Hmm, interesting!
You entered: Hmm, interesting!
```

It's possible to take input as a string using `readLine()` function, and convert it to values of other data type (like `Int`) explicitly.

If you want input of other data types, you can use `Scanner` object.

For that, you need to import `Scanner` class from Java standard library using:

```
import java.util.Scanner
```

Then, you need to create `Scanner` object from this class.

```
val reader = Scanner(System.`in`)
```

Now, the reader object is used to take input from the user.

Example 4: Getting Integer Input from the User

```
import java.util.Scanner

fun main(args: Array<String>) {

    // Creates an instance which takes input from standard input (keyboard)
    val reader = Scanner(System.`in`)
    print("Enter a number: ")

    // nextInt() reads the next integer from the keyboard
    var integer:Int = reader.nextInt()

    println("You entered: $integer")
}
```

When you run the program, the output will be:

```
Enter a number: -12
You entered: -12
```

Here, `reader` object of `Scanner` class is created. Then, the `nextInt()` method is called which takes integer input from the user which is stored in variable integer.

To get `Long`, `Float`, `double` and `Boolean` input from the user, you can use `nextLong()`, `nextFloat()`, `nextDouble()` and `nextBoolean()` methods respectively.

Class Example:

```
fun main(args: Array<String>) {
//    var name:String = readLine()!!
    var num1:Int = readLine()!!.toInt()
    var num2:Int = readLine()!!.toInt()

    var sum:Int = num1+num2
    println("The sum is: ${sum}")
}
```

```
import java.util.Scanner
fun main(args: Array<String>) {
    var reader = Scanner(System.`in`)

    var num1:Int = reader.nextInt()
    var num2:Int = reader.nextInt()

    var sum = num1+num2
    println("The Sum is: ${sum}")
}
```

Kotlin if Expression

Traditional Usage of if...else

The syntax of if...else is:

```
if (testExpression) {
   // codes to run if testExpression is true
}
else {
  // codes to run if testExpression is false
}
```

`if` executes a certain section of code if the testExpression is evaluated to `true`. It can have optional else clause. Codes inside `else` clause are executed if the `testExpression` is false.

Example: Traditional Usage of if...else

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

    val number = -10

    if (number > 0) {
        print("Positive number")
    } else {
        print("Negative number")
    }
}
```

When you run the program, the output will be:

```
Negative number
```

Kotlin if expression

Unlike Java (and other many programming languages), `if` can be used an expression in Kotlin; it returns a value. **Recommended Reading:** Kotlin expression

---

Here is an example:

Example: Kotin if expression

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

    val number = -10

    val result = if (number > 0) {
        "Positive number"
    } else {
        "Negative number"
    }

    println(result)
}
```

When you run the program, the output will be:

```
Negative number
```

The `else` branch is mandatory when using `if` as an expression.

The curly braces are optional if the body of `if` has only one statement. For example,

```
fun main(args: Array<String>) {
    val number = -10
    val result = if (number > 0) "Positive number" else "Negative number"
    println(result)
}
```

This is similar to ternary operator in Java. Hence, there is no ternary operator in Kotlin.

Example: if block With Multiple Expressions

If the block of `if` branch contains more than one expression, the last expression is returned as the value of the block.

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

    val a = -9
    val b = -11

    val max = if (a > b) {
        println("$a is larger than $b.")
        println("max variable holds value of a.")
        a
    } else {
        println("$b is larger than $a.")
        println("max variable holds value of b.")
        b
    }
    println("max = $max")
}
```

When you run the program, the output will be:

```
-9 is larger than -11.
max variable holds value of a.
```

```
max = -9
```

Recommended Reading: *Kotlin when Statement*

Kotlin if..else..if Ladder

You can return a block of code among many blocks in Kotlin using `if..else...if` ladder.

Example: if...else...if Ladder

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

    val number = 0

    val result = if (number > 0)
        "positive number"
    else if (number < 0)
        "negative number"
    else
        "zero"

    println("number is $result")
}
```

This program checks whether number is positive number, negative number, or zero.

Kotlin Nested if Expression

An if expression can be inside the block of another if expression known as nested if expression.

Example: Nested if Expression

This program computes the largest number among three numbers.

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

    val n1 = 3
    val n2 = 5
```

```
    val n3 = -2

    val max = if (n1 > n2) {
        if (n1 > n3)
            n1
        else
            n3
    } else {
        if (n2 > n3)
            n2
        else
            n3
    }

    println("max = $max")
}
```

When you run the program, the output will be:

```
max = 5
```

Class Example:

```
fun main(args: Array<String>) {
    var a:Int = 30
    var b:Int = 20

    if(a>b)
    {
        println("${a} is greater")
    }
    else if(b>a)
    {
        println("${b} is greater")
    }
    else{
        println("Both are equal")
    }

}
```

```
import java.util.Scanner
fun main(args: Array<String>) {
    var sc = Scanner(System.`in`)
```

```
    var a:Int = sc.nextInt()
    var b:Int = sc.nextInt()
    var c:Int = sc.nextInt()

    if(a>b && a>c)
    {
        println("A is greater")
    }
    else if(b>a && b>c){
        println("B is greater")
    }
    else if(c>a && c>b){
        println("C is greater")
    }
    else if(a==b && a>c){
        println("A and B are greater")
    }
    else if(a==c && a>b){
        println("A and C are greater")
    }
    else if(b==c && b>a){
        println("B and C are greater")
    }
    else{
        println("They are equal")
    }
}
```