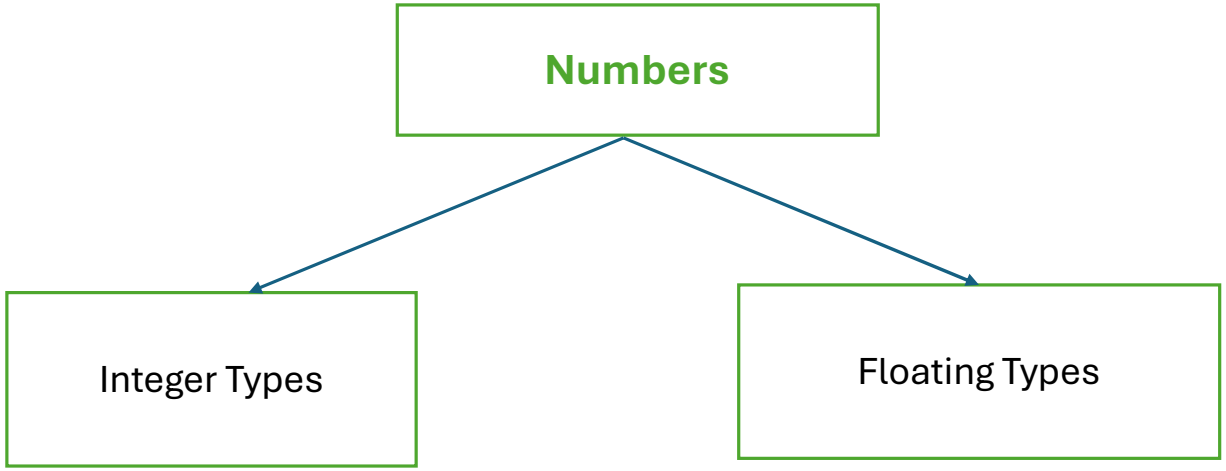
Mobile Application Development Lab

Topic: Data Types

Ashwin

Date:15-7-2024

Numbers



- **Integer Types:**

  Byte (8 bits)

  Short (16 bits)

  Int (32 bits)

  Long (64 bits

Floating Types:

Float (32 bits)

Double (64 bits)

Integer

- **Byte:**

  Size: 8-bit

  Range: -128 to 127

- **Short:**

  Size: 16-bit

  Range: -32768 to 32767

- **Int:**

  Size: 32-bit

  Range: -2^31 to 2^31 - 1 (-2147483648 to 2147483647)

- **Long:**

  Size: 64-bit

  Range: -2^63 to 2^63 – 1

Booleans

- Boolean type represents logical values.
- It can have only two possible values : true and false

```
fun main() {
 val isKotlinFun: Boolean = true
 val isCoffeeCold: Boolean = false

 val result = 5 > 3
 println(result)

 val age = 15
 val isAdult = if (age >= 18) true else false
 println(isAdult)
}
```

Characters

- Characters in Kotlin are represented using char type.
- They can be created using single quotes.

```kotlin
fun main() {
 val letter: Char = 'A'
 val digit: Char = '7'
 val symbol: Char = '$'
 val newLine: Char = '\n'

 println(letter)
 println(digit)
 println(symbol)
 println(newLine)
}
```

Strings

- Strings in Kotlin are represented by String type.
- They are an immutable sequence of characters.

```kotlin
fun main() {
 val simple: String = "Hello, Everyone!"

 val escaped_sequence = "This is a \"quoted\" string

 var multiline="""
  This
  is an
  Example for MultiLine String.
 """.trimIndent()

 val result = "The sum of 2 and 3 is ${2 + 3}"

 println(escaped_sequence)
 println(multiline)
 println(result)

}
```

5. Arrays

- Arrays in Kotlin are represented by the Array class.

```kotlin
fun main() {
 val numbers: IntArray = intArrayOf(1, 2, 3, 4, 5)

 val fruits: Array<String> = arrayOf("Apple", "Banana", "Cherry")

 val chars = charArrayOf('K', 'o', 't', 'l', 'i', 'n')

 println(numbers.contentToString())
 println(fruits.contentToString())
 println(chars)
}
```

6.Collections

- Kotlin Language provides several collection types called List, Set and Map.
- These can be Mutable and Immutable.
- **List**: It is an ordered collection of elements.
- Kotlin provides two types of lists.
- Mutable List: It can be modified after it is created. We can add, remove or update elements in a list.
- It can be created using **mutableListOf()** function.

```kotlin
fun main() {
 val courses = mutableListOf("C", "Unix", "C++")
 courses.add("Kotlin")
 println(courses)
}
```

- **Immutable List:** An immutable list is a read-only list that cannot be modified after it is created.
- We cannot add, remove, or update elements in an immutable list.
- Immutable list is created using **listof()** function.

```
fun main() {
 val branches = listOf("CSE", "ECE", "EEE","AIML")
 println("The List of Courses are:$branches")

}
```

- **Set:**
- **Immutable Set**: Read-only and cannot contain duplicate elements.
- It is created using setOf().

```
fun main() {
 val uniqueNumbers = setOf(1, 2, 3, 4)
 println(uniqueNumbers)

}
```

- **Mutable set:** It can be modified and cannot contain duplicate elements.
- It is created using mutableSetOf()

```
fun main() {
 val uniqueNumbers = mutableSetOf(1, 2, 3, 4)
 uniqueNumbers.add(5)
 println(uniqueNumbers)
}
```

- **Map:**
  - **Immutable Map:** It is Read-only and consists of Key-Value pairs.
  - It is created using **mapOf( )** function.

```
fun main() {
 val numberMap = mapOf("Sno" to "10", "Branch" to "CSE
 println(numberMap)

}
```

- **Mutable Map:** It can be modified and consists of Key-Value Pairs.
- It is created using "**mutableMapOf()**"

```
fun main() {
 val numberMap = mutableMapOf(1 to "one", 2 to "two")
 numberMap[3] = "three"
 println(numberMap)
}
```

```
fun main() {
 val oneToTen = 1 ≤ .. ≤ 10
 val alphabetLowercase = 'a' ≤ .. ≤ 'z'
 val reverseRange = 10 ≥  downTo  ≥ 1
 val stepRange = 1 ≤ .. ≤ 10 step 2

 // Print all numbers in the range oneToTen
for (el in oneToTen){
 println(el)
}
 println()
}
```

7.Ranges

- Ranges represent sequence of values within a given interval.
- They are created using the “..” operator.

8.Nullable Types

- Nullability allows to represent the absence of a value.
- Append a '?' to declare a variable that can hold a null value.

```
fun main() {
 var s:String? = null
 println(s)

}
```

- **Non-nullable Types:** By default Kotlin variables are non-nullable.
- They cannot hold a ‘null’ value.

```kotlin
fun main() {
 var Course:String = "Kotlin"
 println(Course)

}
```

Tasks List

- 1. **Develop a Finance application that calculates interest rates:**

Declare variables to store different interest rates using Float and Double, and print their values.

- 2. **Developing a simple text editor:** Declare variables to store a single character and a sentence. Print the character and the sentence. Concatenate the character at the beginning and the end of the sentence, and print the result.
- 3. **Build a simple weather app that stores temperatures for a week:**

Declare an array of integers to store daily temperatures, print each temperature, and modify one of the temperatures. Print the updated array.