SLIIT

Discover Your Future

IT2010 – Mobile Application Development

Lecture 9 – Introduction to Kotlin

What is this Kotlin

- It is a Russian Island close to St. Petersburg
- Initially belongs to Sweden but after Russia annexed it and…
- Inspired the name for a new programming language running in the JVM
- Created by JetBrains, creators of PhpStorm, WebStorm and PyCharm etc.

What is this Kotlin

- Cross-platform
- Statically typed | Compiled
- General purpose programming language with Type inference
- Designed to interoperate fully with Java, and hence the JVM
- File extension | **.kt**

Primitive Data Types

- Integer types

Type	Bits	Min value	Max value
Long	64	-9223372036854775808	9223372036854775807
Int	32	-2147483648	2147483647
Short	16	-32768	32767
Byte	8	-128	127

Primitive Data Types

- Floating point and other types

Type	Bits	Notes
Double	64	16-17 significant digits (same as float in Python)
Float	32	6-7 significant digits
Char	16	UTF-16 code unit (see the section on strings - in most cases, this is one Unicode character, but it might be just one half of a Unicode character)
Boolean	8	true or false

Primitive Data Types

- Arrays
- Kotlin also has specialized classes to represent arrays of primitive types
- Array of integers of size 5 with values [0, 0, 0, 0, 0]

```
val arr = IntArray(5)
```

- Array of integers of size 5 with values [42, 42, 42, 42, 42]

```
val arr = IntArray(5) { 42 }
```

Declaring Variables

- Every variable **must** be declared
- Local variables are typically declared and initialized at the same time | Type Inference

```
var number = 42
var message = "Hello"
```

- Type of the variable is ***inferred*** to be the type of the expression

Declaring Read-only Variables

- Frequently, you'll find that during the lifetime of your variable, it only ever needs to refer to one object
- Then, you can declare it with ***val***

```
val message = "Hello"
val number = 42
```

- *The terminology is that **var** declares a mutable variable, and that **val** declares a read-only or assign-once variable*

Declaring Constants

- If you have a value that is truly constant or properties which are immutable in nature **AND**
- The value of these properties must be known at the compile-time THEN you can declare as:

```
const val x = 2
```

- Must be initialized with a String type or primitive type

Decision Making

- Conditional statements in Kotlin can be used to take decisions based on certain conditions…

1. If Then
2. If Then Else
3. If Then Else If Else
4. Nested If Then
5. When statement

```kotlin
fun main(args: Array<String>) {
    var x: Int = 1
    when(x)
    {
        1 -> print("x is one")
        2 -> print("x is two")
        else ->
        {
            print("x is not good")
        }
    }
}
```

Output:
x is one

Looping…

- Loops: in cases where you need to repeat over and over until a certain condition is met…

1	For Loop
2	While Loop
3	Do While Loop

```
1  fun main(args: Array<String>) {
2      for( i in 1..5)
3      {
4          print(i)
5      }
6  }
```

Output:

12345

Ranges…

- You can create a range in Kotlin via .. Operator

```
1 | i..j
```

- It will create a range i to j including both i and j
- What if we want to exclude the last value in a range?

```
1 | fun main(args: Array<String>)
2 | {
3 |     for (i in 1 until 5)
4 |     {
5 |         print(i)
6 |     }
7 | }
```

Output:
1234

Ranges…

- Use ***downTo*** function when you want to go reverse in a range…

```
1  fun main(args: Array<String>)
2  {
3      for(i in 5 downTo 1)
4      {
5          print(i)
6      }
7  }
```

Output:

54321

Ranges…

- Use ***step*** function to increase the step count in a range...

```
1  fun main(args: Array<String>)
2  {
3      for (i in 1..10 step 2)
4      {
5          print(i)
6      }
7  }
```

Above example prints odd numbers:

Output:

13579

Functions

- Function in Kotlin is declared using ***fun*** keyword
- Function name, arguments and return type after that

Example function to add two integers:

```
1  fun add(i: Int, j:Int): Int
2  {
3      return i+j
4  }
```

Example of inferred function:

```
1  fun add(i: Int, j: Int) = i + j
```

Functions -> Default Arguments

- Assign default value to some arguments in Kotlin

```
1  fun add(i: Int, j: Int=3): Int
2  {
3      return i + j
4  }
```

```
1  fun main(args: Array<String>) {
2      val res = add(2)
3      print(res)
4  }
```

Output:

5

- We have assigned default value 3 to second argument j
- Then how to call the ***add*** function?