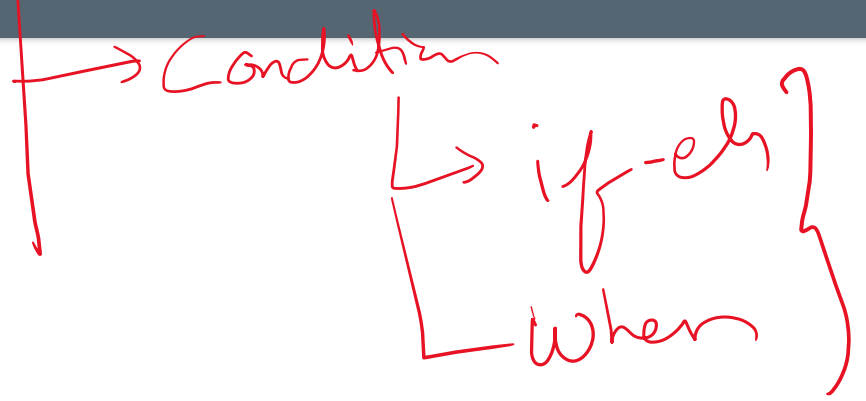
Week 1 : Session 3

Typecasting, Operations and Control flow Statements



Agenda

- What Kotlin Type Conversion
  - Explicit Type Casting.
- Kotlin Operators
  - Arithmetic
  - Assignment
  - Unary prefix and Increment / Decrement Operators
  - Comparison and Equality Operators
  - Logical Operators
  - **in** Operator
  - **as** and **as?** Type Casting Operator
- Kotlin Control Flow
  - **if - else** Expression
  - **when** Expression

Kotlin Type Conversion

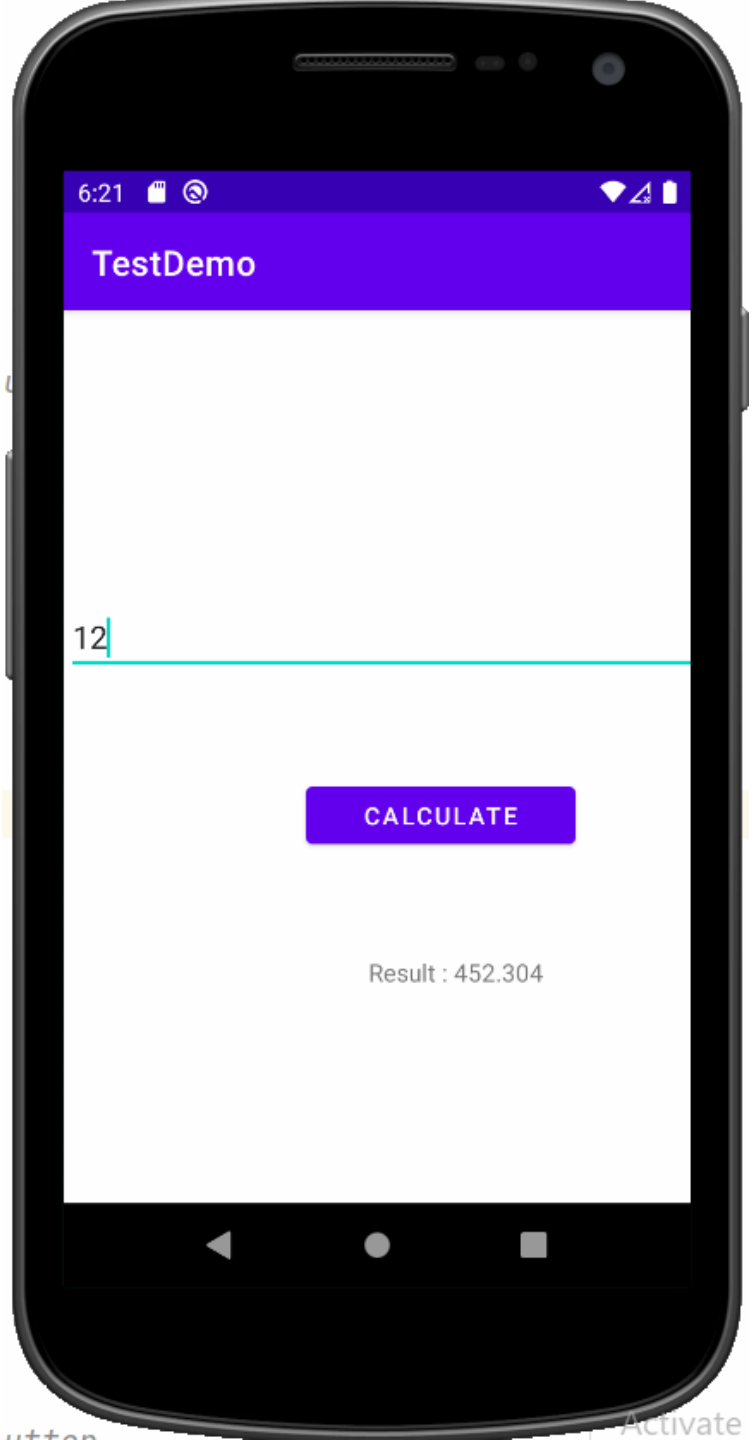
- Type conversion is a process in which one data type variable is converted into another data type.

```
1   package com.example.testdemo
2
3   fun main(){
4       var value1 = 10
5       val value2: Long = value1  //Compile error, type mismatch
6       print("$value2")
7   }
```

- The list of helper functions used for numeric conversion in Kotlin are:
  - toByte()
  - toShort()
  - toInt()
  - toLong()
  - toFloat()
  - toDouble()
  - toChar()

```
1   package com.example.testdemo
2
3   fun main(){
4       var value1 = 10
5       val value2: Long = value1.toLong()
6       print("$value2")
7   }
```

Exercise - 1



Kotlin Operators

- Set of operators to perform arithmetic, assignment, comparison operators and more.
- Arithmetic Operator

Expression	Function name	Translates to
a + b	plus	a.plus(b)
a - b	minus	a.minus(b)
a * b	times	a.times(b)
a / b	div	a.div(b)
a % b	mod	a.mod(b)

```kotlin
package com.example.testdemo

fun main(){
    val number1 = 12.5
    val number2 = 3.5
    var result: Double

    result = number1 + number2
    println("number1 + number2 = $result")

    result = number1 - number2
    println("number1 - number2 = $result")

    result = number1 * number2
    println("number1 * number2 = $result")

    result = number1 / number2
    println("number1 / number2 = $result")

    result = number1 % number2
    println("number1 % number2 = $result")
}
```

Kotlin Operators

- Assignment Operators

Expression	Equivalent to	Translates to
a +=b	a = a + b	a.plusAssign(b)
a -= b	a = a - b	a.minusAssign(b)
a *= b	a = a * b	a.timesAssign(b)
a /= b	a = a / b	a.divAssign(b)
a %= b	a = a % b	a.modAssign(b)

```
1   package com.example.testdemo
2
3   fun main(){
4       var number = 12
5
6       number *= 5   // number = number*5
7       println("number  = $number")
8   }
```

Kotlin Operators

- Unary prefix and Increment / Decrement Operators

Operator	Meaning	Expression	Translates to
+	Unary plus	+a	a.unaryPlus()
-	Unary minus (inverts sign)	-a	a.unaryMinus()
!	not (inverts value)	!a	a.not()
++	Increment: increases value by1	++a	a.inc()
--	Decrement: decreases value by 1	--a	a.dec()

```kotlin
package com.example.testdemo

fun main(){
    val a = 1
    val b = true
    var c = 1

    var result: Int
    var booleanResult: Boolean

    result = -a
    println("-a = $result")

    booleanResult = !b
    println("!b = $booleanResult")

    --c
    println("--c = $c")
}
```

Kotlin Operators

- Comparison and Equality Operators

Operator	Meaning	Expression	Translates to
>	greater than	a > b	a.compareTo(b) > 0
<	less than	a < b	a.compareTo(b) < 0
>=	greater than or equals to	a >= b	a.compareTo(b) >= 0
<=	less than or equals to	a < = b	a.compareTo(b) <= 0
==	is equal to	a == b	a?.equals(b) ?: (b === null)
!=	not equal to	a != b	!(a?.equals(b) ?: (b === null))

```
1   package com.example.testdemo
2
3   fun main(){
4       val a = -12
5       val b = 12
6       println(a<b)
7
8       // use of greater than operator
9       val max = if (a > b) {
10          println("a is larger than b.")
11          a
12      } else {
13          println("b is larger than a.")
14          b
15      }
16
17      println("max = $max")
18  }
```

Kotlin Operators

- Logical Operators
  - There are two logical operators in Kotlin: || and &&

Operator	Description	Expression	Corresponding Function
||	`true` if either of the Boolean expression is `true`	`(a>b)||(a<c)`	`(a>b).or(a<c)`
&&	true if all Boolean expressions are `true`	`(a>b)&&(a<c)`	`(a>b).and(a<c)`

```
1   package com.example.testdemo
2
3   fun main(){
4       val a = 10
5       val b = 9
6       val c = -1
7       var result: Boolean
8
9       // result is true is a is largest
10      result = (a<b) && (a>c) // result = (a>b) and (a>c)
11      println(result)
12      result = (a<b) || (a>c) // result = (a>b) or (a>c)
13      println(result)
14  }
```

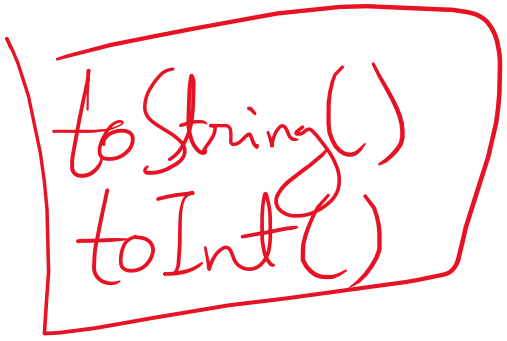
Kotlin Operators

- **in** Operator
  - The **in** operator is used to check whether an object belongs to a collection.

Operator	Expression	Translates to
in	a in b	b.contains(a)
!in	a !in b	!b.contains(a)

```
1  package com.example.testdemo
2
3  fun main(){
4      val numbers = arrayOf<Int>(1, 4, 42, -3)
5
6      if (4 in numbers) {
7          println("numbers array contains 4.")
8      }
9  }
```

Kotlin Operators

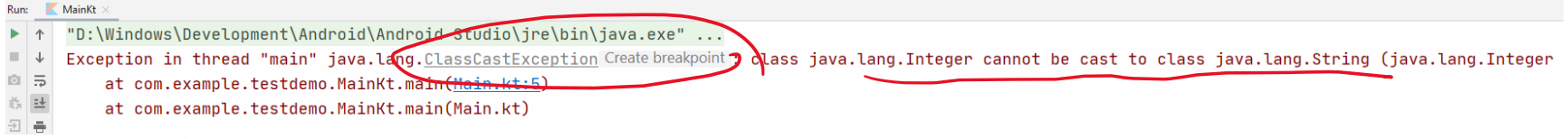


- Type Casting Operator
- Explicit type casting can be done using :
  - Unsafe cast operator as
    - Manually, we use type cast operator as to cast a variable to target type.

```
1  package com.example.testdemo
2
3  fun main(){
4      val str1: String = "It works fine"
5      val str2: String = str1 as String      // Works
6      println(str1)
7  }
```

    - There might be possibility that we can not cast variable to target type and it throws an exception at runtime, thats why it is called as unsafe casting.

```
1  package com.example.testdemo
2
3  fun main(){
4      val str1: Any = 11
5      val str2: String = str1 as String      // throw exception
6      println(str1)
7  }
```



  - Safe cast operator: as?
    - If casting is not possible it returns null instead of throwing an ClassCastException exception.

```
1  package com.example.testdemo
2
3  fun main(){
4      val str1: Any = 11
5      val str2: String? = str1 as? String      // prints null
6      println(str2)
7  }
```

Kotlin Control Flow – if - else Expression

- **If** is an expression is which returns a value.
- Various type of if expression
  - if-else expression
  - if-else if-else ladder expression
  - nested if expression
- The syntax of if...else is

```
1   package com.example.testdemo
2
3   fun main(){
4       val number = -10
5
6       val result = if (number > 0) {
7           "Positive number"
8       } else {
9           "Negative number"
10      }
11
12      println(result)
13  }
```

- The syntax of if...else...if ladder is

```
1   package com.example.testdemo
2
3   fun main(){
4       val number = 0
5
6       val result = if (number > 0)
7           "positive number"
8       else if (number < 0)
9           "negative number"
10      else
11          "zero"
12
13      println("number is $result")
14  }
```

- The syntax of nested if is

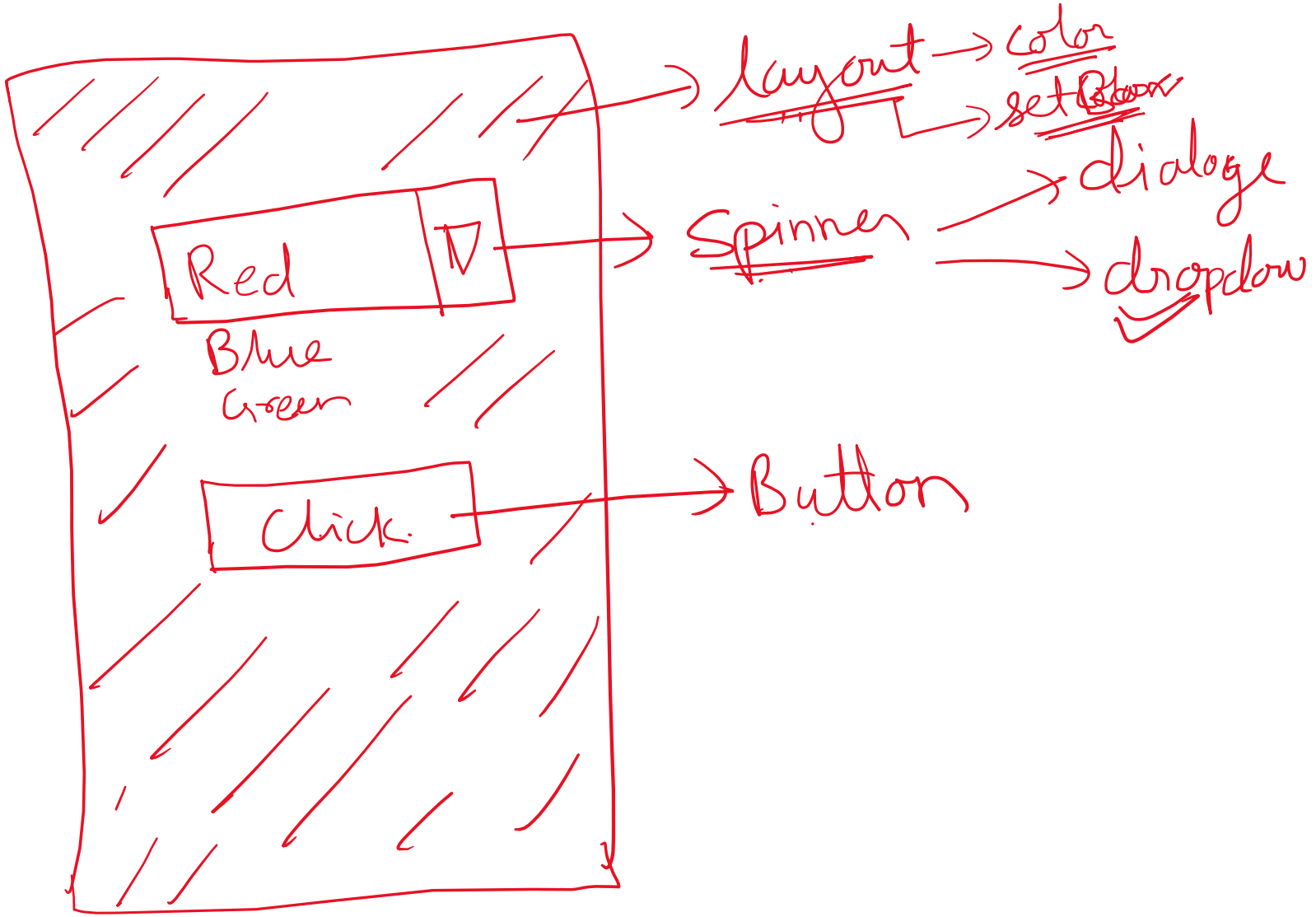
```
1   package com.example.testdemo
2
3   fun main(){
4       val n1 = 3
5       val n2 = 5
6       val n3 = -2
7
8       val max = if (n1 > n2) {
9           if (n1 > n3)
10              n1
11          else
12              n3
13      } else {
14          if (n2 > n3)
15              n2
16          else
17              n3
18      }
19
20      println("max = $max")
21  }
```

Kotlin Control Flow – **when** Expression

- The **when** in Kotlin is a replacement for switch Statement.
- It evaluates a section of code among many alternatives
- The syntax of when is

```kotlin
package com.example.testdemo

fun main(){
    val a = 12
    val b = 5

    val operator = "*"

    val result = when (operator) {
        "+" -> a + b
        "-" -> a - b
        "*" -> a * b
        "/" -> a / b
        else -> "$operator operator is invalid operator."
    }

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

```kotlin
package com.example.testdemo

fun main(){
    val n = 11

    when (n) {
        1, 2, 3 -> println("n is a positive integer less than 4.")
        0 -> println("n is zero")
        -1, -2 -> println("n is a negative integer greater than 3.")
        in 1..10 -> println("A positive number less than 11.")
        "11".toInt() -> println("Its 11")
    }
}
```


Red
Blue
Green
Click
layout
color
setBackground
Spinner
dialoge
dropdow
Button