

 免費電子書

學習

# Kotlin

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#kotlin

.....	1
<b>1: Kotlin</b> .....	<b>2</b>
.....	2
Kotlin.....	2
.....	2
Examples.....	2
.....	3
Hello World.....	3
Hello WorldCompanion.....	3
varargs.....	4
Kotlin.....	4
.....	4
<b>2: DSL</b> .....	<b>6</b>
.....	6
Examples.....	6
DSLInfix.....	6
DSL.....	6
lambda.....	6
lambdas.....	6
<b>3: Java 8 Stream</b> .....	<b>8</b>
.....	8
.....	8
.....	8
.....	8
.....	8
.....	8
.....	9
Examples.....	9
.....	9
.....	9
.....	9
.....	9
.....	9

.....	10
.....	10
.....	10
.....	10
.....	10
2 - .....	11
3 - .....	11
4 - .....	11
5 - Int.....	11
6 - Ints.....	12
7 - IntString.....	12
.....	12
- .....	12
1 - .....	13
5 - .....	13
6 - .....	14
7a - .....	15
7b - SummarizingInt.....	15
<b>4: JUnit.....</b>	<b>17</b>
Examples.....	17
.....	17
<b>5: Kotlin Android.....</b>	<b>18</b>
.....	18
Examples.....	18
.....	18
.....	18
.....	19
Kotlin.....	19
<b>6: Kotlin for Java Developers.....</b>	<b>21</b>
.....	21
Examples.....	21
.....	.....

.....	21
.....	21
IFTRY.....	22
<b>7: Kotlin.....</b>	<b>23</b>
.....	23
.....	23
Examples.....	23
.....	23
<b>8: Kotlin.....</b>	<b>25</b>
Examples.....	25
toString.....	25
<b>9:.....</b>	<b>26</b>
Examples.....	26
try-catch-finally.....	26
<b>10: Vararg.....</b>	<b>27</b>
.....	27
Examples.....	27
vararg.....	27
Spread Operatorvararg.....	27
<b>11:.....</b>	<b>29</b>
.....	29
.....	29
Examples.....	29
.....	29
Lambda.....	29
.....	30
.....	31
.....	32
.....	32
.....	32
<b>12:.....</b>	<b>33</b>

.....	33
Examples.....	33
1.....	33
<b>13:</b> .....	<b>34</b>
.....	34
.....	34
Examples.....	34
.....	34
.....	34
Java.....	34
.....	34
.....	35
<b>14:</b> .....	<b>37</b>
.....	37
.....	37
Examples.....	37
.....	37
<b>15:</b> .....	<b>38</b>
.....	38
Examples.....	38
java/.....	38
.....	38
<b>16: KotlinRecyclerView</b> .....	<b>40</b>
.....	40
Examples.....	40
.....	40
<b>17: Lambdas</b> .....	<b>42</b>
.....	42
.....	42
Examples.....	43
Lambda.....	43

Lambda.....	43
Lambda.....	43
<b>18:</b> .....	<b>44</b>
.....	44
Examples.....	44
.....	44
.....	44
.....	44
.....	44
.....	44
<b>19:</b> .....	<b>46</b>
Examples.....	46
.....	46
.....	46
.....	46
.....	47
<b>20:</b> .....	<b>48</b>
Examples.....	48
DTPPOJO / POCO.....	48
.....	48
.....	48
KotlinSerializableserialVersionUID.....	48
Kotlin.....	49
let.....	49
apply.....	50
<b>21:</b> .....	<b>51</b>
.....	51
Examples.....	51
.....	51
.....	51
.....	51
.....	51

.....	52
.....	53
<b>22:</b> .....	<b>54</b>
.....	54
.....	54
Examples.....	54
.....	54
.....	54
.....	54
Java 7+ Path.....	55
.....	55
Java 8 TemporalISO.....	55
Companion.....	56
.....	56
.....	56
.....	56
.....	57
<b>23:</b> .....	<b>58</b>
Examples.....	58
.....	58
.....	58
.....	58
.....	59
.....	59
.....	59
.....	59
<b>24:</b> .....	<b>60</b>
.....	60
Examples.....	60
.....	60
.....	60

.....	60
<b>25:</b> .....	<b>62</b>
.....	62
Examples.....	62
if.....	62
if.....	62
whenif-else-if.....	62
when.....	63
.....	63
.....	64
<b>26:</b> .....	<b>65</b>
Examples.....	65
.....	65
.....	65
.....	65
.....	65
Kotlin.....	66
<b>Regex</b> .....	<b>66</b>
.....	66
.....	66
<b>findCharSequencestartIndexIntMatchResult</b> .....	<b>66</b>
<b>findAllinputCharSequencestartIndexIntSequence</b> .....	<b>67</b>
<b>matchEntireCharSequenceMatchResult</b> .....	<b>67</b>
<b>matchesCharSequenceBoolean</b> .....	<b>67</b>
<b>containsMatchIninputCharSequenceBoolean</b> .....	<b>67</b>
<b>splitCharSequencelimitIntList</b> .....	<b>67</b>
<b>replaceCharSequencereplacementStringString</b> .....	<b>68</b>
<b>27:</b> .....	<b>69</b>
.....	69
.....	69



.....	69
.....	69
.....	<b>69</b>
Examples.....	69
- .....	69
.....	69
<b>28:</b> .....	<b>71</b>
Examples.....	71
NullableNon Nullable.....	71
.....	71
null.....	71
.....	71
Iterable.....	72
Null Coalescing / Elvis.....	72
.....	72
:). .....	72
<b>29:</b> .....	<b>74</b>
.....	74
Examples.....	74
.....	74
<b>30: kotlin</b> .....	<b>75</b>
.....	75
Examples.....	75
kotlin.logging.....	75
<b>31:</b> .....	<b>76</b>
.....	76
Examples.....	76
x.....	76
iterables.....	76
.....	76
.....	77
kotlin.....	77

.....	77
.....	77
<b>32:</b> .....	<b>79</b>
.....	79
Examples.....	79
.....	79
downTo.....	79
step.....	79
.....	79
<b>33:</b> .....	<b>80</b>
Examples.....	80
.....	80
.....	80
<b>34:</b> .....	<b>81</b>
.....	81
.....	81
.....	81
Examples.....	81
.....	81
.....	81
<b>35: Kotlin</b> .....	<b>82</b>
Examples.....	82
Gradle.....	82
JVM.....	82
Android.....	82
JS.....	82
Android Studio.....	83
.....	83
.....	83
Java.....	83
GroovyGradleKotlin.....	84
<b>36:</b> .....	<b>86</b>

.....	86
.....	86
Examples.....	86
.....	86
.....	86
set.....	86
<b>37:</b> .....	<b>87</b>
.....	87
.....	87
Kotlin.....	87
Examples.....	87
.....	87
<b>38:</b> .....	<b>89</b>
.....	89
.....	89
.....	89
Examples.....	89
'open'.....	89
.....	89
.....	89
.....	89
.....	90
.....	90
.....	90
.....	90
Person.....	90
.....	90
.....	90
.....	90
.....	92

---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [kotlin](#)

It is an unofficial and free Kotlin ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Kotlin.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

# 1: Kotlin

[KotlinJetBrainsJVM](#)。 [KotlinJava 100](#)。 [KotlinJava](#)。 [KotlinJVMJavaJavaScript](#)。

## Kotlin

[KotlinEclipseIntelliJIDE](#)。 [KotlinMaven](#) [Ant](#) [Gradle](#)。

```
$ kotlinc Main.ktjavaMainKt.class Kt $ java MainKt$ java MainKt
```

```
Exception in thread "main" java.lang.NoClassDefFoundError: kotlin/jvm/internal/Intrinsics
    at MainKt.main(Main.kt)
Caused by: java.lang.ClassNotFoundException: kotlin.jvm.internal.Intrinsics
    at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:335)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    ... 1 more
```

JavaKotlin runt-time jar。

```
java -cp ../path/to/kotlin/runtime/jar/kotlin-runtime.jar MainKt
```

1.0.0	2016215
1.0.1	2016316
1.0.2	2016513
1.0.3	2016630
1.0.4	2016922
1.0.5	2016118
1.0.6	20161227
1.1.0	201731
1.1.1	2017314
1.1.2	2017425
1.1.3	2017623

## Examples

## Kotlinmain◦ Kotlin“Hello World”

```
package my.program

fun main(args: Array<String>) {
    println("Hello, world!")
}
```

Main.kt

JVM◦ my.program.MainKt ◦

package

```
@file:JvmName("MyApp")
```

my.program.MyApp ◦

- @JvmName◦
- 

## Hello World

Kotlin ◦

```
package my.program

object App {
    @JvmStatic fun main(args: Array<String>) {
        println("Hello World")
    }
}
```

my.program.App ◦

App ◦ App◦

- @JvmStatic

## Hello WorldCompanion

CompanionKotlinmain◦

```
package my.program

class App {
    companion object {
        @JvmStatic fun main(args: Array<String>) {
            println("Hello World")
        }
    }
}
```

my.program.App ◦

App ◦ Object Declaration◦

“hello”

```
class App {
    companion object {
        @JvmStatic fun main(args: Array<String>) {
            App().run()
        }
    }

    fun run() {
        println("Hello World")
    }
}
```

- @JvmStatic

**varargs**

varargs

```
package my.program

fun main(vararg args: String) {
    println("Hello, world!")
}
```

**Kotlin**

javaJava◦ Kotlin◦

javacjava◦ javajava◦

kotlincotlinkotlinkotlin◦

Kotlin◦ N1 2 3◦

Kotlin◦

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

    println("Enter Two number")
    var (a, b) = readLine()!!.split(' ') // !! this operator use for
    NPE(NullPointerException).

    println("Max number is : ${maxNum(a.toInt(), b.toInt())}")
}
```

```
fun maxNum(a: Int, b: Int): Int {  
  
    var max = if (a > b) {  
        println("The value of a is $a");  
        a  
    } else {  
        println("The value of b is $b")  
        b  
    }  
  
    return max;  
  
}
```

◦

```
Enter Two number  
71 89 // Enter two number from command line  
  
The value of b is 89  
Max number is: 89
```

**!!Null Safety** ◦

**IntelliJ** ◦

**Kotlin** <https://riptutorial.com/zh-TW/kotlin/topic/490/kotlin>



## 2: DSL

KotlinDSL ◦

### Examples

#### DSLInfix

```
infix fun <T> T?.shouldBe(expected: T?) = assertEquals(expected, this)
```

#### DSL

```
@Test
fun test() {
    100.plusOne() shouldBe 101
}
```

#### DSL

```
class MyExample(val i: Int) {
    operator fun <R> invoke(block: MyExample.() -> R) = block()
    fun Int.bigger() = this > i
}
```

#### DSL

```
fun main2(args: Array<String>) {
    val ex = MyExample(233)
    ex {
        // bigger is defined in the context of `ex`
        // you can only call this method inside this context
        if (777.bigger()) kotlin.io.println("why")
    }
}
```

#### lambda

```
val r = Random(233)
infix inline operator fun Int.rem(block: () -> Unit) {
    if (r.nextInt(100) < this) block()
}
```

#### DSL

```
20 % { println("The possibility you see this message is 20%") }
```

#### lambdas

```
operator fun <R> String.invoke(block: () -> R) = {
    try { block.invoke() }
    catch (e: AssertionError) { System.err.println("$this\n${e.message}") }
}
```

## DSL

```
"it should return 2" {
    parse("1 + 1").buildAST().evaluate() shouldBe 2
}
```

shouldBeInfix approach to build DSLInfix approach to build DSL ◦

DSL <https://riptutorial.com/zh-TW/kotlin/topic/10042/dsl>

# 3: Java 8 Stream

Kotlin ◦ Sequence ◦

asSequence() Sequence ◦ Sequence ◦ toList() toSet() toMap() Sequence ◦

```
// switch to and from lazy
val someList = items.asSequence().filter { ... }.take(10).map { ... }.toList()

// switch to lazy, but sorted() brings us out again at the end
val someList = items.asSequence().filter { ... }.take(10).map { ... }.sorted()
```

Kotlin ◦ Kotlin ◦ JavaNPE ◦ Kotlin

```
val someList = people.filter { it.age <= 30 }.map { it.name }
```

```
val someList: List<String> = people.filter { it.age <= 30 }.map { it.name }
```

Kotlin people people.age Int Int people.name String map List<String> String List ◦

people null List<People>?

```
val someList = people?.filter { it.age <= 30 }?.map { it.name }
```

List<String>? *Kotlin Kotlin Kotlin*

Kotlin ◦ Sequence ""start ◦ Java 8 Kotlin

```
// Java:
Stream<String> stream =
Stream.of("d2", "a2", "b1", "b3", "c").filter(s -> s.startsWith("b"));

stream.anyMatch(s -> true); // ok
stream.noneMatch(s -> true); // exception
```

```
// Kotlin:
val stream = listOf("d2", "a2", "b1", "b3", "c").asSequence().filter { it.startsWith('b') }

stream.forEach(::println) // b1, b2

println("Any B ${stream.any { it.startsWith('b') }}") // Any B true
println("Any C ${stream.any { it.startsWith('c') }}") // Any C false

stream.forEach(::println) // b1, b2
```

Java

```
// Java:
Supplier<Stream<String>> streamSupplier =
```

```

() -> Stream.of("d2", "a2", "b1", "b3", "c")
    .filter(s -> s.startsWith("a"));

streamSupplier.get().anyMatch(s -> true); // ok
streamSupplier.get().noneMatch(s -> true); // ok

```

**Kotlin**◦ SequenceSequenceconstrainOnce()

```

val stream = listOf("d2", "a2", "b1", "b3", "c").asSequence().filter { it.startsWith('b' ) }
    .constrainOnce()

stream.forEach(::println) // b1, b2
stream.forEach(::println) // Error:java.lang.IllegalStateException: This sequence can be
consumed only once.

```

- [Iterable API](#)
- [Array API](#)
- [List API](#)
- [Map API](#)

## Examples

```

// Java:
List<String> list = people.stream().map(Person::getName).collect(Collectors.toList());

```

```

// Kotlin:
val list = people.map { it.name } // toList() not needed

```

```

// Java:
String joined = things.stream()
    .map(Object::toString)
    .collect(Collectors.joining(", "));

```

```

// Kotlin:
val joined = things.joinToString() // ", " is used as separator, by default

```

```

// Java:
int total = employees.stream()
    .collect(Collectors.summingInt(Employee::getSalary));

```

```

// Kotlin:
val total = employees.sumBy { it.salary }

```

```

// Java:
Map<Department, List<Employee>> byDept
    = employees.stream()
    .collect(Collectors.groupingBy(Employee::getDepartment));

```

```

// Kotlin:

```

```
val byDept = employees.groupBy { it.department }
```

```
// Java:  
Map<Department, Integer> totalByDept  
    = employees.stream()  
        .collect(Collectors.groupingBy(Employee::getDepartment,  
            Collectors.summingInt(Employee::getSalary)));
```

```
// Kotlin:  
val totalByDept = employees.groupBy { it.dept }.mapValues { it.value.sumBy { it.salary } }
```

```
// Java:  
Map<Boolean, List<Student>> passingFailing =  
    students.stream()  
        .collect(Collectors.partitioningBy(s -> s.getGrade() >= PASS_THRESHOLD));
```

```
// Kotlin:  
val passingFailing = students.partition { it.grade >= PASS_THRESHOLD }
```

```
// Java:  
List<String> namesOfMaleMembersCollect = roster  
    .stream()  
    .filter(p -> p.getGender() == Person.Sex.MALE)  
    .map(p -> p.getName())  
    .collect(Collectors.toList());
```

```
// Kotlin:  
val namesOfMaleMembers = roster.filter { it.gender == Person.Sex.MALE }.map { it.name }
```

```
// Java:  
Map<Person.Sex, List<String>> namesByGender =  
    roster.stream().collect(  
        Collectors.groupingBy(  
            Person::getGender,  
            Collectors.mapping(  
                Person::getName,  
                Collectors.toList())));
```

```
// Kotlin:  
val namesByGender = roster.groupBy { it.gender }.mapValues { it.value.map { it.name } }
```

```
// Java:  
List<String> filtered = items.stream()  
    .filter( item -> item.startsWith("o") )  
    .collect(Collectors.toList());
```

```
// Kotlin:  
val filtered = items.filter { item.startsWith('o') }
```

```
// Java:  
String shortest = items.stream()
```

```
.min(Comparator.comparing(item -> item.length()))
.get();
```

```
// Kotlin:
val shortest = items.minBy { it.length }
```

## 2 -

```
// Java:
Stream.of("a1", "a2", "a3")
    .findFirst()
    .ifPresent(System.out::println);
```

```
// Kotlin:
sequenceOf("a1", "a2", "a3").firstOrNull()?.apply(::println)
```

## 3 -

```
// Java:
IntStream.range(1, 4).forEach(System.out::println);
```

```
// Kotlin: (inclusive range)
(1..3).forEach(::println)
```

## 4 -

```
// Java:
Arrays.stream(new int[] {1, 2, 3})
    .map(n -> 2 * n + 1)
    .average()
    .ifPresent(System.out::println); // 5.0
```

```
// Kotlin:
arrayOf(1,2,3).map { 2 * it + 1}.average().apply(::println)
```

## 5 - Int

```
// Java:
Stream.of("a1", "a2", "a3")
    .map(s -> s.substring(1))
    .mapToInt(Integer::parseInt)
    .max()
    .ifPresent(System.out::println); // 3
```

```
// Kotlin:
sequenceOf("a1", "a2", "a3")
    .map { it.substring(1) }
    .map(String::toInt)
    .max().apply(::println)
```

## 6 - Ints

```
// Java:
IntStream.range(1, 4)
    .mapToObj(i -> "a" + i)
    .forEach(System.out::println);

// a1
// a2
// a3
```

```
// Kotlin: (inclusive range)
(1..3).map { "a$it" }.forEach(::println)
```

## 7 - IntString

```
// Java:
Stream.of(1.0, 2.0, 3.0)
    .mapToInt(Double::intValue)
    .mapToObj(i -> "a" + i)
    .forEach(System.out::println);

// a1
// a2
// a3
```

```
// Kotlin:
sequenceOf(1.0, 2.0, 3.0).map(Double::toInt).map { "a$it" }.forEach(::println)
```

```
// Java:
long count = items.stream().filter( item -> item.startsWith("t")).count();
```

```
// Kotlin:
val count = items.filter { it.startsWith('t') }.size
// but better to not filter, but count with a predicate
val count = items.count { it.startsWith('t') }
```

```
// Java:
List<String> myList = Arrays.asList("a1", "a2", "b1", "c2", "c1");

myList.stream()
    .filter(s -> s.startsWith("c"))
    .map(String::toUpperCase)
    .sorted()
    .forEach(System.out::println);

// C1
// C2
```

```
// Kotlin:
```

```
val list = listOf("a1", "a2", "b1", "c2", "c1")
list.filter { it.startsWith('c') }.map (String::toUpperCase).sorted()
    .forEach (::println)
```

1 -

```
// Java:
Arrays.asList("a1", "a2", "a3")
    .stream()
    .findFirst()
    .ifPresent(System.out::println);
```

```
// Kotlin:
listOf("a1", "a2", "a3").firstOrNull()?.apply(::println)
```

## StringifPresent

```
// Kotlin:
inline fun String?.ifPresent(thenDo: (String)->Unit) = this?.apply { thenDo(this) }

// now use the new extension function:
listOf("a1", "a2", "a3").firstOrNull().ifPresent(::println)
```

[apply\(\)](#)

?. <http://stackoverflow.com/questions/34498562/in-kotlin-what-is-the-idiomatic-way-to-deal-with-nullable-values-referencing-o/34498563> 34498563

5 -

```
// Java:
String phrase = persons
    .stream()
    .filter(p -> p.age >= 18)
    .map(p -> p.name)
    .collect(Collectors.joining(" and ", "In Germany ", " are of legal age.));

System.out.println(phrase);
// In Germany Max and Peter and Pamela are of legal age.
```

```
// Kotlin:
val phrase = persons
    .filter { it.age >= 18 }
    .map { it.name }
    .joinToString(" and ", "In Germany ", " are of legal age.")

println(phrase)
// In Germany Max and Peter and Pamela are of legal age.
```

## Kotlin

```
// Kotlin:
// data class has equals, hashCode, toString, and copy methods automagically
```



```

data class Person(val name: String, val age: Int)

val persons = listOf(Person("Tod", 5), Person("Max", 33),
                    Person("Frank", 13), Person("Peter", 80),
                    Person("Pamela", 18))

```

## 6 -

```

// Java:
Map<Integer, String> map = persons
    .stream()
    .collect(Collectors.toMap(
        p -> p.age,
        p -> p.name,
        (name1, name2) -> name1 + ";" + name2));

System.out.println(map);
// {18=Max, 23=Peter;Pamela, 12=David}

```

### Kotlin ◦ /Map

```

// Kotlin:
val map1 = persons.map { it.age to it.name }.toMap()
println(map1)
// output: {18=Max, 23=Pamela, 12=David}
// Result: duplicates overridden, no exception similar to Java 8

val map2 = persons.toMap({ it.age }, { it.name })
println(map2)
// output: {18=Max, 23=Pamela, 12=David}
// Result: same as above, more verbose, duplicates overridden

val map3 = persons.toMapBy { it.age }
println(map3)
// output: {18=Person(name=Max, age=18), 23=Person(name=Pamela, age=23), 12=Person(name=David,
age=12)}
// Result: duplicates overridden again

val map4 = persons.groupBy { it.age }
println(map4)
// output: {18=[Person(name=Max, age=18)], 23=[Person(name=Peter, age=23), Person(name=Pamela,
age=23)], 12=[Person(name=David, age=12)]}
// Result: closer, but now have a Map<Int, List<Person>> instead of Map<Int, String>

val map5 = persons.groupBy { it.age }.mapValues { it.value.map { it.name } }
println(map5)
// output: {18=[Max], 23=[Peter, Pamela], 12=[David]}
// Result: closer, but now have a Map<Int, List<String>> instead of Map<Int, String>

```

```

// Kotlin:
val map6 = persons.groupBy { it.age }.mapValues { it.value.joinToString(";") { it.name } }

println(map6)
// output: {18=Max, 23=Peter;Pamela, 12=David}
// Result: YAY!!

```

joinToStringPersonPerson.name

o

## 7a -

```
// Java (verbose):
Collector<Person, StringJoiner, String> personNameCollector =
Collector.of(
    () -> new StringJoiner(" | "),           // supplier
    (j, p) -> j.add(p.name.toUpperCase()), // accumulator
    (j1, j2) -> j1.merge(j2),              // combiner
    StringJoiner::toString);               // finisher

String names = persons
    .stream()
    .collect(personNameCollector);

System.out.println(names); // MAX | PETER | PAMELA | DAVID

// Java (concise)
String names = persons.stream().map(p -> p.name.toUpperCase()).collect(Collectors.joining(" | "));
```

```
// Kotlin:
val names = persons.map { it.name.toUpperCase() }.joinToString(" | ")
```

## 7b - SummarizingInt

```
// Java:
IntSummaryStatistics ageSummary =
    persons.stream()
        .collect(Collectors.summarizingInt(p -> p.age));

System.out.println(ageSummary);
// IntSummaryStatistics{count=4, sum=76, min=12, average=19.000000, max=23}
```

```
// Kotlin:

// something to hold the stats...
data class SummaryStatisticsInt(var count: Int = 0,
                                var sum: Int = 0,
                                var min: Int = Int.MAX_VALUE,
                                var max: Int = Int.MIN_VALUE,
                                var avg: Double = 0.0) {

    fun accumulate(newInt: Int): SummaryStatisticsInt {
        count++
        sum += newInt
        min = min.coerceAtMost(newInt)
        max = max.coerceAtLeast(newInt)
        avg = sum.toDouble() / count
        return this
    }
}

// Now manually doing a fold, since Stream.collect is really just a fold
val stats = persons.fold(SummaryStatisticsInt()) { stats, person ->
    stats.accumulate(person.age) }
```

```
println(stats)
// output: SummaryStatisticsInt(count=4, sum=76, min=12, max=23, avg=19.0)
```

## 2Kotlin stdlib

```
// Kotlin:
inline fun Collection<Int>.summarizingInt(): SummaryStatisticsInt
    = this.fold(SummaryStatisticsInt()) { stats, num -> stats.accumulate(num) }

inline fun <T: Any> Collection<T>.summarizingInt(transform: (T)->Int): SummaryStatisticsInt =
    this.fold(SummaryStatisticsInt()) { stats, item -> stats.accumulate(transform(item)) }
```

summarizingInt

```
val stats2 = persons.map { it.age }.summarizingInt()

// or

val stats3 = persons.summarizingInt { it.age }
```

◦ Sequence◦

**Java 8 Stream** <https://riptutorial.com/zh-TW/kotlin/topic/707/java-8-stream>

---

# 4: JUnit

## Examples

### JUnit

```
@Rule @JvmField val myRule = TemporaryFolder()
```

@JvmField myRule ◦ **JUnit**◦

**JUnit** <https://riptutorial.com/zh-TW/kotlin/topic/6973/junit>

# 5: Kotlin Android

KotlinAndroidButterKnife◦ ◦

## Examples

gradle◦

Kotlin build.gradle◦

```
buildscript {  
    ...  
}  
  
apply plugin: "com.android.application"  
...  
apply plugin: "kotlin-android"  
apply plugin: "kotlin-android-extensions"  
...
```

activity\_main.xmlactivity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <Button  
        android:id="@+id/my_button"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="My button"/>  
</LinearLayout>
```

Kotlin

```
import kotlinx.android.synthetic.main.activity_main.my_button  
  
class MainActivity: Activity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
        // my_button is already casted to a proper type of "Button"  
        // instead of being a "View"  
        my_button.setText("Kotlin rocks!")  
    }  
}
```

\*ID

```
// my_button can be used the same way as before  
import kotlinx.android.synthetic.main.activity_main.*
```

```
//
```

```
import kotlinx.android.synthetic.main.activity_main.my_button

class NotAView {
    init {
        // This sample won't compile!
        my_button.setText("Kotlin rocks!")
    }
}
```

**AndroidAndroid**◦ build.gradle

```
android {
    productFlavors {
        paid {
            ...
        }
        free {
            ...
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/buy_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Buy full version"/>
</LinearLayout>
```

```
import kotlinx.android.synthetic.free.main_activity.buy_button
```

## Kotlin

```
mView.afterMeasured {
    // inside this block the view is completely drawn
    // you can get view's height/width, it.height / it.width
}
```

```
inline fun View.afterMeasured(crossinline f: View.() -> Unit) {
    viewTreeObserver.addOnGlobalLayoutListener(object : ViewTreeObserver.OnGlobalLayoutListener {
        override fun onGlobalLayout() {
            if (measuredHeight > 0 && measuredWidth > 0) {
                viewTreeObserver.removeOnGlobalLayoutListener(this)
                f()
            }
        }
    })
}
```

```
}
```

Kotlin Android <https://riptutorial.com/zh-TW/kotlin/topic/9474/kotlin-android>

# 6: Kotlin for Java Developers

KotlinJava

JavaKotlinKotlinJava

## Examples

KotlinJava

```
val i : Int = 42
```

- valvarfinal “val ue”var iable
- ::
- Kotlin

Java	
<code>int i = 42;</code>	<code>var i = 42</code> <code>var i : Int = 42</code>
<code>final int i = 42;</code>	<code>val i = 42</code>

- ;
- Kotlin
- Kotlin**100Java**
- Kotlin JVM
- Kotlin
- Kotlin`equals` / `hashCode`
- Kotlin
- Kotlin `new` ◦ ◦
- Kotlin ◦ `val a = someMap["key"]`
- KotlinJVMJava **Script** Kotlin
- Kotlin **Java 6** Android
- Kotlin**Android**
- Kotlin
- Kotlin**Coroutines**

Kotlin`==equals` `===`

Java	
<code>a.equals(b);</code>	<code>a == b</code>
<code>a == b;</code>	<code>a === b</code>



## Java

```
a != b;
```

```
a !== b
```

<https://kotlinlang.org/docs/reference/equality.html>

## IFTRY

Kotlin `if` `try`void。

KotlinJava*Elvis*

```
val i = if (someBoolean) 33 else 42
```

`try`

```
val i = try {  
    Integer.parseInt(someString)  
}  
catch (ex : Exception)  
{  
    42  
}
```

Kotlin for Java Developers <https://riptutorial.com/zh-TW/kotlin/topic/10099/kotlin-for-java-developers>

# 7: Kotlin

Kotlin.

1. Kotlin.kt.
2. KotlinAnyJavaObject.
3. val - var -
4. .
5. Unit.It. 6.===. aba === btrue.

## Examples

1. .

```
fun printHello(name: String?): Unit {
    if (name != null)
        println("Hello ${name}")
}

fun printHello(name: String?) {
    ...
}
```

### 2.Single-Expression=

```
fun double(x: Int): Int = x * 2
```

```
fun double(x: Int) = x * 2
```

### 3.String.

```
In java:
int num=10
String s = "i =" + i;

In Kotlin
val num = 10
val s = "i = $num"
```

### 4.Kotlinnullnullnull. Stringnull

```
var a: String = "abc"
a = null // compilation error
```

### String

```
var b: String? = "abc"
```

```
b = null // ok
```

## 5. Kotlin == 与 == b

```
a?.equals(b) ?: (b === null)
```

**Kotlin** <https://riptutorial.com/zh-TW/kotlin/topic/10648/kotlin>

---

# 8: Kotlin

## Examples

### toString

**Kotlin** `toString` `nullString?String?` ◦

**Android** `EditText` ◦

```
// Incorrect:  
val text = view.textfield?.text.toString() ?: ""
```

"null" ◦

```
// Correct:  
val text = view.textfield?.text?.toString() ?: ""
```

**Kotlin** <https://riptutorial.com/zh-TW/kotlin/topic/6608/kotlin>

# 9:

## Examples

### try-catch-finally

#### KotlinJava

```
try {
    doSomething()
}
catch(e: MyException) {
    handle(e)
}
finally {
    cleanup()
}
```

```
try {
    doSomething()
}
catch(e: FileSystemException) {
    handle(e)
}
catch(e: NetworkException) {
    handle(e)
}
catch(e: MemoryException) {
    handle(e)
}
finally {
    cleanup()
}
```

try

```
val s: String? = try { getString() } catch (e: Exception) { null }
```

#### Kotlin。

```
fun fileToString(file: File) : String {
    //readAllBytes throws IOException, but we can omit catching it
    fileContent = Files.readAllBytes(file)
    return String(fileContent)
}
```

<https://riptutorial.com/zh-TW/kotlin/topic/7246/>

# 10: Vararg

- **Vararg** `vararg`
- **Spread Operator** `*` “”。

## Examples

### vararg

`vararg`

```
fun printNumbers(vararg numbers: Int) {  
    for (number in numbers) {  
        println(number)  
    }  
}
```

◦

```
printNumbers(0, 1)           // Prints "0" "1"  
printNumbers(10, 20, 30, 500) // Prints "10" "20" "30" "500"
```

`Vararg`

### Spread Operator `vararg`

#### Spread Operator `*vararg`

.....

```
fun printNumbers(vararg numbers: Int) {  
    for (number in numbers) {  
        println(number)  
    }  
}
```

...

```
val numbers = intArrayOf(1, 2, 3)  
printNumbers(*numbers)  
  
// This is the same as passing in (1, 2, 3)
```

...

```
val numbers = intArrayOf(1, 2, 3)  
printNumbers(10, 20, *numbers, 30, 40)
```

```
// This is the same as passing in (10, 20, 1, 2, 3, 30, 40)
```

Vararg <https://riptutorial.com/zh-TW/kotlin/topic/5835/vararg>

# 11:

- **Params** = ...
- **Params** {...}
- **PARAMS** {...}
- **< > PARAMS** {...}
- {...}
- { **ArgName ArgType** -> ... }
- { **ArgName** -> ... }
- { **ArgNames** -> ... }
- { **ArgName ArgType** -> ... }

PARAMS	<i>Name : Type</i>
ArgName	
ArgType	<b>ArgName</b>
ArgNames	ArgName

## Examples

“Lambda”。 “”

```
# Takes no parameters and returns anything
() -> Any?

# Takes a string and an integer and returns ReturnType
(arg1: String, arg2: Int) -> ReturnType
```

() -> Any? **lambda**

```
fun twice(x: () -> Any?) {
    x(); x();
}

fun main() {
    twice {
        println("Foo")
    } # => Foo
    # => Foo
}
```

## Lambda



Lambda. {braces} - ->.

```
{ name: String ->
  "Your name is $name" //This is returned
}
```

lambda.

lambda.

```
{ argumentOne:String, argumentTwo:String ->
  "$argumentOne - $argumentTwo"
}
```

lambdait.

```
{ "Your name is $it" }
```

lambda.

```
# These are identical
listOf(1, 2, 3, 4).map { it + 2 }
listOf(1, 2, 3, 4).map({ it + 2 })
```

::.

```
fun addTwo(x: Int) = x + 2
listOf(1, 2, 3, 4).map(::addTwo) # => [3, 4, 5, 6]
```

(ParamTypeA, ParamTypeB, ...) -> ReturnType ParamTypeA ParamTypeB ...`ReturnType1`.

```
fun foo(p0: Foo0, p1: Foo1, p2: Foo2): Bar {
  //...
}
println(::foo::class.java.genericInterfaces[0])
// kotlin.jvm.functions.Function3<Foo0, Foo1, Foo2, Bar>
// Human readable type: (Foo0, Foo1, Foo2) -> Bar
```

.

```
class Foo
fun Foo.foo(p0: Foo0, p1: Foo1, p2: Foo2): Bar {
  //...
}
val ref = Foo::foo
println(ref::class.java.genericInterfaces[0])
// kotlin.jvm.functions.Function4<Foo, Foo0, Foo1, Foo2, Bar>
// Human readable type: (Foo, Foo0, Foo1, Foo2) -> Bar
// takes 4 parameters, with receiver as first and actual parameters following, in their order

// this function can't be called like an extension function, though
```

```

val ref = Foo::foo
Foo().ref(Foo0(), Foo1(), Foo2()) // compile error

class Bar {
    fun bar()
}
print(Bar::bar) // works on member functions, too.

```

◦

```

object Foo
fun Foo.foo(p0: Foo0, p1: Foo1, p2: Foo2): Bar {
    //...
}
val ref = Foo::foo
println(ref::class.java.genericInterfaces[0])
// kotlin.jvm.functions.Function3<Foo0, Foo1, Foo2, Bar>
// Human readable type: (Foo0, Foo1, Foo2) -> Bar
// takes 3 parameters, receiver not needed

object Bar {
    fun bar()
}
print(Bar::bar) // works on member functions, too.

```

**kotlin 1.1** ◦

## 1.1.0

```

fun makeList(last: String?): List<String> {
    val list = mutableListOf("a", "b", "c")
    last?.let(list::add)
    return list
}

```

◦ ◦

◦ ◦

```

class Foo
class Bar {
    fun Foo.foo() {}
    val ref = Foo::foo // compile error
}

```

fun ◦ Unit ◦ {} ◦ Unit **return** ◦

```

fun sayMyName(name: String): String {
    return "Your name is $name"
}

```

```

fun sayMyName(name: String): String = "Your name is $name"

```

```
fun sayMyName(name: String) = "Your name is $name"
```

◦ ◦

```
fun sayMyName(name: String): String = "Your name is $name"
```

**inline inline C** - ◦ **lambdas** ◦

```
inline fun sayMyName(name: String) = "Your name is $name"
```

**C**

```
inline fun sayMyName() = "Your name is $name"

fun main() {
    val name = "Foo"
    sayMyName() # => Unresolved reference: name
}
```

**Kotlin+\*** ◦ ◦ operator

```
data class IntListWrapper (val wrapped: List<Int>) {
    operator fun get(position: Int): Int = wrapped[position]
}

val a = IntListWrapper(listOf(1, 2, 3))
a[1] // == 2
```

<https://riptutorial.com/zh-TW/kotlin/topic/1280/>

---

# 12:

Kotlin

## Examples

1

```
fun main(args: Array<String>) {  
    launch(CommonPool) { // create new coroutine in common thread pool  
        delay(1000L) // non-blocking delay for 1 second (default time unit is ms)  
        println("World!") // print after delay  
    }  
    println("Hello,") // main function continues while coroutine is delayed  
    Thread.sleep(2000L) // block main thread for 2 seconds to keep JVM alive  
}
```

```
Hello,  
World!
```

<https://riptutorial.com/zh-TW/kotlin/topic/10936/>

# 13:

- 
- 

JVMJAR kotlin-reflect.jar ◦ JS◦

## Examples

### KClass

```
val c1 = String::class
val c2 = MyClass::class
```

### Kotlin◦

```
fun isPositive(x: Int) = x > 0

val numbers = listOf(-2, -1, 0, 1, 2)
println(numbers.filter(::isPositive)) // [1, 2]
```

## Java

KotlinKClassJavaClass KClass.java

```
val stringKClass: KClass<String> = String::class
val c1: Class<String> = stringKClass.java

val c2: Class<MyClass> = MyClass::class.java
```

### KClass◦

BaseExampleExample

```
open class BaseExample(val baseField: String)

class Example(val field1: String, val field2: Int, baseField: String):
    BaseExample(baseField) {

    val field3: String
        get() = "Property without backing field"

    val field4 by lazy { "Delegated value" }

    private val privateField: String = "Private value"
}
```

```
val example = Example(field1 = "abc", field2 = 1, baseField = "someText")
```

```
example::class.memberProperties.forEach { member ->
    println("${member.name} -> ${member.get(example)}")
}
```

◦ private val privateField **private** member.get(example) ◦ ◦ **Java getter** ◦ private val **getter** ◦

```
fun isFieldAccessible(property: KProperty1<*, *>): Boolean {
    return property.javaGetter?.modifiers?.let { !Modifier.isPrivate(it) } ?: false
}

val example = Example(field1 = "abc", field2 = 1, baseField = "someText")

example::class.memberProperties.filter { isFieldAccessible(it) }.forEach { member ->
    println("${member.name} -> ${member.get(example)}")
}
```

```
example::class.memberProperties.forEach { member ->
    member.isAccessible = true
    println("${member.name} -> ${member.get(example)}")
}
```

```
class TestClass {
    val readOnlyProperty: String
        get() = "Read only!"

    var readWriteString = "asd"
    var readWriteInt = 23

    var readWriteBackedStringProperty: String = ""
        get() = field + '5'
        set(value) { field = value + '5' }

    var readWriteBackedIntProperty: Int = 0
        get() = field + 1
        set(value) { field = value - 1 }

    var delegatedProperty: Int by TestDelegate()

    private var privateProperty = "This should be private"

    private class TestDelegate {
        private var backingField = 3

        operator fun getValue(thisRef: Any?, prop: KProperty<*>): Int {
            return backingField
        }

        operator fun setValue(thisRef: Any?, prop: KProperty<*>, value: Int) {
            backingField += value
        }
    }
}
```

◦ ◦

```

val instance = TestClass()
TestClass::class.memberProperties
    .filter{ prop.visibility == KVisibility.PUBLIC }
    .filterIsInstance<KMutableProperty<*>>()
    .forEach { prop ->
        System.out.println("${prop.name} -> ${prop.get(instance)}")
    }

```

String"Our Value"◦ **KotlinJava VMType Erasure**List<String>PropertiesList<Any>◦ ◦

```

val instance = TestClass()
TestClass::class.memberProperties
    .filter{ prop.visibility == KVisibility.PUBLIC }
    // We only want strings
    .filter{ it.returnType.isSubtypeOf(String::class.starProjectedType) }
    .filterIsInstance<KMutableProperty<*>>()
    .forEach { prop ->
        // Instead of printing the property we set it to some value
        prop.setter.call(instance, "Our Value")
    }

```

<https://riptutorial.com/zh-TW/kotlin/topic/2402/>

---

# 14:

## Kotlin4.

- 
- 
- 
- 
- `<visibility modifier> val/var <variable name> = <value>`

## Examples

```
public val name = "Avijit"
```

```
private val name = "Avijit"
```

```
protected val name = "Avijit"
```

```
internal val name = "Avijit"
```

<https://riptutorial.com/zh-TW/kotlin/topic/10019/>



# 15:

object◦ javaSingletons◦ java◦ Javakotlin◦

## Examples

### java/

```
object CommonUtils {  
  
    var anyname: String ="Hello"  
  
    fun dispMsg(message: String) {  
        println(message)  
    }  
}
```

```
CommonUtils.anyname  
CommonUtils.dispMsg("like static call")
```

Kotlin◦ SomeSingleton.INSTANCE◦

### java

```
public enum SharedRegistry {  
    INSTANCE;  
    public void register(String key, Object thing) {}  
}  
  
public static void main(String[] args) {  
    SharedRegistry.INSTANCE.register("a", "apple");  
    SharedRegistry.INSTANCE.register("b", "boy");  
    SharedRegistry.INSTANCE.register("c", "cat");  
    SharedRegistry.INSTANCE.register("d", "dog");  
}
```

### kotlin

```
object SharedRegistry {  
    fun register(key: String, thing: Object) {}  
}  
  
fun main(Array<String> args) {  
    SharedRegistry.register("a", "apple")  
    SharedRegistry.register("b", "boy")  
    SharedRegistry.register("c", "cat")  
    SharedRegistry.register("d", "dog")  
}
```

◦

<https://riptutorial.com/zh-TW/kotlin/topic/10152/>

# 16: KotlinRecyclerView

KotlinRecyclerView。

## Examples

Kotlinactivity\_main.xmlRecyclerView。

```
class MainActivity : AppCompatActivity() {

    lateinit var mRecyclerView : RecyclerView
    val mAdapter : RecyclerViewAdapter = RecyclerViewAdapter()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val toolbar = findViewById(R.id.toolbar) as Toolbar
        setSupportActionBar(toolbar)

        mRecyclerView = findViewById(R.id.recycler_view) as RecyclerView

        mRecyclerView.setHasFixedSize(true)
        mRecyclerView.layoutManager = LinearLayoutManager(this)
        mAdapter.RecyclerViewAdapter(getList(), this)
        mRecyclerView.adapter = mAdapter
    }

    private fun getList(): ArrayList<String> {
        var list : ArrayList<String> = ArrayList()
        for (i in 1..10) { // equivalent of 1 <= i && i <= 10
            println(i)
            list.add("$i")
        }
        return list
    }
}
```

Recyclermain\_item.xml

```
class RecyclerViewAdapter : RecyclerView.Adapter<RecyclerViewAdapter.ViewHolder>() {

    var mItems: ArrayList<String> = ArrayList()
    lateinit var mClick : OnClickListener

    fun RecyclerViewAdapter(item : ArrayList<String>, mClick : OnClickListener){
        this.mItems = item
        this.mClick = mClick;
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val item = mItems[position]
        holder.bind(item, mClick, position)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
```

```
        val inflater = LayoutInflater.from(parent.context)
        return ViewHolder(inflater.inflate(R.layout.main_item, parent, false))
    }

    override fun getItemCount(): Int {
        return mItems.size
    }

    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val card = view.findViewById(R.id.card) as TextView
        fun bind(str: String, mClick: OnClickListener, position: Int) {
            card.text = str
            card.setOnClickListener { view ->
                mClick.onClickListner(position)
            }
        }
    }
}
```

**KotlinRecyclerView** <https://riptutorial.com/zh-TW/kotlin/topic/10143/kotlinrecyclerview>

# 17: Lambdas

- 
- {parameterNameParameterTypeotherParameterNameOtherParameterType -> anExpression}
- 
- val addition: Int -> Int = {a, b -> a + b}
- it
- val square: Int -> Int = {it \* it}
- 
- -> ResultType
- InputType -> ResultType
- InputType1InputType2 -> ResultType

```
data class User(val firstName: String, val lastName: String) {  
    fun username(userNameGenerator: (String, String) -> String) =  
        userNameGenerator(firstName, secondName)  
}
```

## lambdalambda

```
val user = User("foo", "bar")  
println(user.username { firstName, secondName ->  
    "${firstName.toUpperCase}"_"${secondName.toUpperCase}"  
}) // prints FOO_BAR
```

## lambda

```
//valid:  
val addition: (Int, Int) = { a, b -> a + b }  
//valid:  
val addition = { a: Int, b: Int -> a + b }  
//error (type inference failure):  
val addition = { a, b -> a + b }
```

## lambdait

```
listOf(1,2,3).map { it * 2 } // [2,4,6]
```

# Examples

## Lambda

```
val allowedUsers = users.filter { it.age > MINIMUM_AGE }
```

## Lambda

```
val isOfAllowedAge = { user: User -> user.age > MINIMUM_AGE }  
val allowedUsers = users.filter(isOfAllowedAge)
```

## Lambda

```
object Benchmark {  
    fun realtime(body: () -> Unit): Duration {  
        val start = Instant.now()  
        try {  
            body()  
        } finally {  
            val end = Instant.now()  
            return Duration.between(start, end)  
        }  
    }  
}
```

```
val time = Benchmark.realtime({  
    // some long-running code goes here ...  
})  
println("Executed the code in $time")
```

Lambdas <https://riptutorial.com/zh-TW/kotlin/topic/5878/lambdas>

# 18:

Kotlin ◦ ◦ ◦

## Examples

```
val foo : Int by lazy { 1 + 1 }
println(foo)
```

2 ◦

```
var foo : Int by Delegates.observable("1") { property, oldValue, newValue ->
    println("${property.name} was changed from $oldValue to $newValue")
}
foo = 2
```

foo was changed from 1 to 2foo was changed from 1 to 2

```
val map = mapOf("foo" to 1)
val foo : String by map
println(foo)
```

1

```
class MyDelegate {
    operator fun getValue(owner: Any?, property: KProperty<*>): String {
        return "Delegated value"
    }
}

val foo : String by MyDelegate()
println(foo)
```

Delegated value

KotlinNull TypeWeakReference<T> ◦

WeakReference◦

```
class MyMemoryExpensiveClass {
    companion object {
        var reference: WeakReference<MyMemoryExpensiveClass>? = null

        fun doWithReference(block: (MyMemoryExpensiveClass) -> Unit) {
            reference?.let {
                it.get()?.let(block)
            }
        }
    }

    init {
```

```
        reference = WeakReference(this)
    }
}
```

## WeakReference

```
class WeakReferenceDelegate<T>(initialValue: T? = null) : ReadWriteProperty<Any, T?> {
    var reference = WeakReference(initialValue)
    private set

    override fun getValue(thisRef: Any, property: KProperty<*>): T? = reference.get()

    override fun setValue(thisRef: Any, property: KProperty<*>, value: T?) {
        reference = WeakReference(value)
    }
}
```

WeakReference

```
class MyMemoryExpensiveClass {
    companion object {
        var reference: MyMemoryExpensiveClass? by
        WeakReferenceDelegate<MyMemoryExpensiveClass>()

        fun doWithReference(block: (MyMemoryExpensiveClass) -> Unit) {
            reference?.let(block)
        }
    }

    init {
        reference = this
    }
}
```

<https://riptutorial.com/zh-TW/kotlin/topic/10571/>



# 19:

## Examples

`String` `string[index]` `string[index]` ◦

```
val str = "Hello, World!"
println(str[1]) // Prints e
```

`for` ◦

```
for (c in str) {
    println(c)
}
```

Kotlin

- 
- 

◦ ◦ `\t \b \n \r \' \" \\$` ◦ **Unicode** `\uFF00` ◦

```
val s = "Hello, world!\n"
```

"""

```
val text = """
    for (c in "foo")
        print(c)
    """
```

`trimMargin` ◦

```
val text = """
    |Tell me and I forget.
    |Teach me and I remember.
    |Involve me and I learn.
    |(Benjamin Franklin)
    """.trimMargin()
```

| `trimMargin;` `trimMargin(">")` ◦

◦ ◦ \$

```
val i = 10
val s = "i = $i" // evaluates to "i = 10"
```

```
val s = "abc"
```

```
val str = "$s.length is ${s.length}" // evaluates to "abc.length is 3"
```

```
val str = "\$foo" // evaluates to "$foo"
```

◦ ◦

```
val price = """  
${'$'}9.99  
"""
```

## Kotlin==◦

```
val str1 = "Hello, World!"  
val str2 = "Hello," + " World!"  
println(str1 == str2) // Prints true
```

===◦

```
val str1 = """  
|Hello, World!  
|""".trimMargin()  
  
val str2 = """  
#Hello, World!  
#""".trimMargin("#")  
  
val str3 = str1  
  
println(str1 == str2) // Prints true  
println(str1 === str2) // Prints false  
println(str1 === str3) // Prints true
```

<https://riptutorial.com/zh-TW/kotlin/topic/8285/>

# 20:

## Examples

### DTOPOJO / POCO

kotlin data

```
data class User(var firstname: String, var lastname: String, var age: Int)
```

User

- val **S**
- equals()
- hashCode()
- toString()
- copy()
- componentN() N

```
data class User(var firstname: String = "Joe", var lastname: String = "Bloggs", var age: Int = 20)
```

◦

```
val list = listOf(1,2,3,4,5,6)

//filter out even numbers

val even = list.filter { it % 2 == 0 }

println(even) //returns [2,4]
```

◦ ◦ ◦

```
class MyTable private constructor(table: Table<Int, Int, Int>) : Table<Int, Int, Int> by table
{
    constructor() : this(TreeBasedTable.create()) // or a different type of table if desired
}
```

MyTable MyTable() ◦ MyTableTable<Int, Int, Int> ◦ ◦

SO ◦

### KotlinSerializableserialVersionUID

KotlinserialVersionUID ◦

```
private const val MySpecialCase
```

```
class MySpecialCase : Serializable {  
    companion object {  
        private const val serialVersionUID: Long = 123  
    }  
}
```

## getter / setter...

```
class MySpecialCase : Serializable {  
    companion object {  
        private val serialVersionUID: Long = 123  
    }  
}
```

getSerialVersionUID **getter** getSerialVersionUID ◦

```
class MySpecialCase : Serializable {  
    companion object {  
        @JvmStatic private val serialVersionUID: Long = 123  
    }  
}
```

MySpecialCase **getter** getSerialVersionUID ◦

serialVersionUID Serializable ◦

## Kotlin

### KotlinJava

```
fun doSomething() {  
    someOtherAction()  
    return this  
}
```

```
fun <T: Any> T.fluently(func: ()->Unit): T {  
    func()  
    return this  
}
```

```
fun doSomething() {  
    return fluently { someOtherAction() }  
}
```

## let

let Kotlin ◦

```
val str = "foo"
str.let {
    println(it) // it
}
```

"foo"Unit ◦

*let* also *let* ◦ *also* *return* Unit ◦

null null *let* also

```
val str: String? = someFun()
str?.let {
    println(it)
}
```

str null *let* ◦ null ? ◦

## apply

apply

this this ◦

**kdoc** apply ◦ apply this apply ◦ ◦

```
File(dir).apply { mkdirs() }
```

```
fun makeDir(String path): File {
    val result = new File(path)
    result.mkdirs()
    return result
}
```

<https://riptutorial.com/zh-TW/kotlin/topic/2273/>

# 21:

## Kotlin

### Examples

#### Kotlin

```
interface MyInterface {
    fun bar()
}
```

```
class Child : MyInterface {
    override fun bar() {
        print("bar() was called")
    }
}
```

#### Kotlin

```
interface MyInterface {
    fun withImplementation() {
        print("withImplementation() was called")
    }
}
```

```
class MyClass: MyInterface {
    // No need to reimplement here
}
val instance = MyClass()
instance.withImplementation()
```

#### gettersetter

```
interface MyInterface2 {
    val helloWorld
    get() = "Hello World!"
}
```

```
interface MyInterface3 {
    // this property won't compile!
    var helloWorld: Int
    get() = field
    set(value) { field = value }
}
```

```
interface A {
    fun notImplemented()
    fun implementedOnlyInA() { print("only A") }
```

```

    fun implementedInBoth() { print("both, A") }
    fun implementedInOne() { print("implemented in A") }
}

interface B {
    fun implementedInBoth() { print("both, B") }
    fun implementedInOne() // only defined
}

class MyClass: A, B {
    override fun notImplemented() { print("Normal implementation") }

    // implementedOnlyInA() can by normally used in instances

    // class needs to define how to use interface functions
    override fun implementedInBoth() {
        super<B>.implementedInBoth()
        super<A>.implementedInBoth()
    }

    // even if there's only one implementation, there multiple definitions
    override fun implementedInOne() {
        super<A>.implementedInOne()
        print("implementedInOne class implementation")
    }
}

```

◦ ◦

```

interface MyInterface {
    val property: Int // abstract

    val propertyWithImplementation: String
        get() = "foo"

    fun foo() {
        print(property)
    }
}

class Child : MyInterface {
    override val property: Int = 29
}

```

◦ ◦ ◦

```

interface FirstTrait {
    fun foo() { print("first") }
    fun bar()
}

interface SecondTrait {
    fun foo() { print("second") }
    fun bar() { print("bar") }
}

class ClassWithConflict : FirstTrait, SecondTrait {
    override fun foo() {

```

```
    super<FirstTrait>.foo() // delegate to the default implementation of FirstTrait
    super<SecondTrait>.foo() // delegate to the default implementation of SecondTrait
}

// function bar() only has a default implementation in one interface and therefore is ok.
}
```

```
interface MyInterface {
    fun funcOne() {
        //optional body
        print("Function with default implementation")
    }
}
```

**super.**

```
super.funcOne()
```

<https://riptutorial.com/zh-TW/kotlin/topic/900/>



## 22:

- `fun TypeName.extensionNameparams...{/ * body * /} //`
- `fun <TAny> TypeNameWithGenerics <T> .extensionNameparams...{/ * body * /} //`
- `myObj.extensionNameargs...//`

◦ reference-type;◦ ◦

## Examples

◦

```
fun IntArray.addTo(dest: IntArray) {
    for (i in 0 .. size - 1) {
        dest[i] += this[i]
    }
}
```

`IntArrayIntArray ◦ this ◦`

```
val myArray = intArrayOf(1, 2, 3)
intArrayOf(4, 5, 6).addTo(myArray)
```

◦ ◦

```
open class Super

class Sub : Super()

fun Super.myExtension() = "Defined for Super"

fun Sub.myExtension() = "Defined for Sub"

fun callMyExtension(myVar: Super) {
    println(myVar.myExtension())
}

callMyExtension(Sub())
```

`"Defined for Super" myVarSuper ◦`

`IntLong`

```
fun Long.humanReadable(): String {
    if (this <= 0) return "0"
    val units = arrayOf("B", "KB", "MB", "GB", "TB", "EB")
    val digitGroups = (Math.log10(this.toDouble())/Math.log10(1024.0)).toInt();
    return DecimalFormat("#,##0.#").format(this/Math.pow(1024.0, digitGroups.toDouble())) + "
" + units[digitGroups];
}
```

```
fun Int.humanReadable(): String {
    return this.toLong().humanReadable()
}
```

```
println(1999549L.humanReadable())
println(someInt.humanReadable())
```

## Java 7+ Path

API◦ exist notExistsdeleteRecursivelyJava 7+ Path

```
fun Path.exists(): Boolean = Files.exists(this)
fun Path.notExists(): Boolean = !this.exists()
fun Path.deleteRecursively(): Boolean = this.toFile().deleteRecursively()
```

```
val dir = Paths.get(dirName)
if (dir.exists()) dir.deleteRecursively()
```

## Kotlin

```
val x: Path = Paths.get("dirName").apply {
    if (Files.notExists(this)) throw IllegalStateException("The important file does not exist")
}
```

apply◦ ◦

```
infix inline fun <T> T.verifiedBy(verifyWith: (T) -> Unit): T {
    verifyWith(this)
    return this
}

infix inline fun <T: Any> T.verifiedWith(verifyWith: T.() -> Unit): T {
    this.verifyWith()
    return this
}
```

```
val x: Path = Paths.get("dirName") verifiedWith {
    if (Files.notExists(this)) throw IllegalStateException("The important file does not exist")
}
```

lambda◦

TverifiedByT: Any?◦ verifiedWith◦

## Java 8 TemporalISO

```
fun Temporal.toIsoString(): String = DateTimeFormatter.ISO_INSTANT.format(this)
```

```
val dateAsString = someInstant.toIsoString()
```

## Companion

- SomethingfromString

```
class Something {
    companion object {}
}

class SomethingElse {
}

fun Something.Companion.fromString(s: String): Something = ...

fun SomethingElse.fromString(s: String): SomethingElse = ...

fun main(args: Array<String>) {
    Something.fromString("") //valid as extension function declared upon the
                            //companion object

    SomethingElse().fromString("") //valid, function invoked on instance not
                                    //statically

    SomethingElse.fromString("") //invalid
}
```

◦ this [KotlinKT-9686KT-13053](#) ◦ ◦

color ◦ colorCache thislazy

```
class KColor(val value: Int)

private val colorCache = mutableMapOf<KColor, Color>()

val KColor.color: Color
    get() = colorCache.getOrPut(this) { Color(value, true) }
```

View◦

Anko

```
inline fun <reified T : View> View.find(id: Int): T = findViewById(id) as T
inline fun <reified T : View> Activity.find(id: Int): T = findViewById(id) as T
inline fun <reified T : View> Fragment.find(id: Int): T = view?.findViewById(id) as T
inline fun <reified T : View> RecyclerView.ViewHolder.find(id: Int): T =
    itemView?.findViewById(id) as T

inline fun <reified T : View> View.findOptional(id: Int): T? = findViewById(id) as? T
inline fun <reified T : View> Activity.findOptional(id: Int): T? = findViewById(id) as? T
inline fun <reified T : View> Fragment.findOptional(id: Int): T? = view?.findViewById(id) as?
T
inline fun <reified T : View> RecyclerView.ViewHolder.findOptional(id: Int): T? =
    itemView?.findViewById(id) as? T
```

```
val yourButton by lazy { find<Button>(R.id.yourButtonId) }  
val yourText by lazy { find<TextView>(R.id.yourTextId) }  
val yourEdittextOptional by lazy { findOptional<EditText>(R.id.yourOptionEdittextId) }
```

<https://riptutorial.com/zh-TW/kotlin/topic/613/>

# 23:

## Examples

KotlinArray<T> ◦

emptyArray<T>()

```
val empty = emptyArray<String>()
```

```
var strings = Array<String>(size = 5, init = { index -> "Item #${index}" })
print(Arrays.toString(a)) // prints "[Item #0, Item #1, Item #2, Item #3, Item #4]"
print(a.size) // prints 5
```

get(index: Int): Tset(index: Int, value: T)

```
strings.set(2, "ChangedItem")
print(strings.get(2)) // prints "ChangedItem"
```

```
// You can use subscription as well:
strings[2] = "ChangedItem"
print(strings[2]) // prints "ChangedItem"
```

Array<T> ◦

Kotlin		JVM
BooleanArray	booleanArrayOf(true, false)	boolean[]
ByteArray	byteArrayOf(1, 2, 3)	byte[]
CharArray	charArrayOf('a', 'b', 'c')	char[]
DoubleArray	doubleArrayOf(1.2, 5.0)	double[]
FloatArray	floatArrayOf(1.2, 5.0)	float[]
IntArray	intArrayOf(1, 2, 3)	int[]
LongArray	longArrayOf(1, 2, 3)	long[]
ShortArray	shortArrayOf(1, 2, 3)	short[]

average()Byte Int Long Short Double FloatDouble

```
val doubles = doubleArrayOf(1.5, 3.0)
print(doubles.average()) // prints 2.25
```

```
val ints = intArrayOf(1, 4)
println(ints.average()) // prints 2.5
```

component1() component2() ... component5()

**index**getOrNull(index: Int) **null**

first() last()

toHashSet() HashSet<T>

sortedArray() sortedArrayDescending() **current**

sort() sortDescending

min() max()

**Java:** in ◦

```
val asc = Array(5, { i -> (i * i).toString() })
for(s : String in asc){
    println(s);
}
```

**for** ◦

```
val asc = Array(5, { i -> (i * i).toString() })
for(s in asc){
    println(s);
}
```

```
val a = arrayOf(1, 2, 3) // creates an Array<Int> of size 3 containing [1, 2, 3].
```

```
val a = Array(3) { i -> i * 2 } // creates an Array<Int> of size 3 containing [0, 2, 4]
```

```
val a = arrayOfNulls<Int>(3) // creates an Array<Int?> of [null, null, null]
```

◦ ◦

<https://riptutorial.com/zh-TW/kotlin/topic/5722/>

# 24:

## JavaKotlin ◦ EnumClass

```
EnumClass.valueOf(value: String): EnumClass  
EnumClass.values(): Array<EnumClass>
```

valueOf() IllegalArgumentException ◦

```
val name: String  
val ordinal: Int
```

## Comparable ◦

## Examples

```
enum class Color(val rgb: Int) {  
    RED(0xFF0000),  
    GREEN(0x00FF00),  
    BLUE(0x0000FF)  
}
```

◦ ; ◦

abstract ◦

```
enum class Color {  
    RED {  
        override val rgb: Int = 0xFF0000  
    },  
    GREEN {  
        override val rgb: Int = 0x00FF00  
    },  
    BLUE {  
        override val rgb: Int = 0x0000FF  
    }  
;  
  
    abstract val rgb: Int  
  
    fun colorString() = "%06X".format(0xFFFFFFFF and rgb)  
}
```

```
enum class Color {  
    RED, GREEN, BLUE  
}
```

◦ ◦

```
enum class Planet(var population: Int = 0) {  
    EARTH(7 * 100000000),  
    MARS();  
  
    override fun toString() = "$name[population=$population]"  
}  
  
println(Planet.MARS) // MARS[population=0]  
Planet.MARS.population = 3  
println(Planet.MARS) // MARS[population=3]
```

<https://riptutorial.com/zh-TW/kotlin/topic/2286/>



# 25:

## Java switch when break behaviors

```
when (x) {  
    "foo", "bar" -> println("either foo or bar")  
    else -> println("didn't match anything")  
}
```

## Examples

### if

```
val str = "Hello!"  
if (str.length == 0) {  
    print("The string is empty!")  
} else if (str.length > 5) {  
    print("The string is short!")  
} else {  
    print("The string is long!")  
}
```

### else-branches if

### if

### if

```
val str = if (condition) "Condition met!" else "Condition not met!"
```

### elseif

### else if

```
val str = if (condition1) {  
    "Condition1 met!"  
} else if (condition2) {  
    "Condition2 met!"  
} else {  
    "Conditions not met!"  
}
```

Kotlin `val str: String =`

### when-if-else-if

### when-else-if-branches if

```

when {
  str.length == 0 -> print("The string is empty!")
  str.length > 5  -> print("The string is short!")
  else            -> print("The string is long!")
}

```

## if-else-if

```

if (str.length == 0) {
  print("The string is empty!")
} else if (str.length > 5) {
  print("The string is short!")
} else {
  print("The string is long!")
}

```

## ifelse-branch

```

when {
  condition -> {
    doSomething()
    doSomeMore()
  }
  else -> doSomethingElse()
}

```

## when

when **-statement** ◦ ==equals◦ ◦

```

when (x) {
  "English" -> print("How are you?")
  "German"  -> print("Wie geht es dir?")
  else -> print("I don't know that language yet :(")
}

```

## when

```

val names = listOf("John", "Sarah", "Tim", "Maggie")
when (x) {
  in names -> print("I know that name!")
  !in 1..10 -> print("Argument was not in the range from 1 to 10")
  is String -> print(x.length) // Due to smart casting, you can use String-functions here
}

```

```

val greeting = when (x) {
  "English" -> "How are you?"
  "German"  -> "Wie geht es dir?"
  else -> "I don't know that language yet :("
}
print(greeting)

```

## whenelse

whenever

```
enum class Day {
    Sunday,
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday
}

fun doOnDay(day: Day) {
    when(day) {
        Day.Sunday -> // Do something
        Day.Monday, Day.Tuesday -> // Do other thing
        Day.Wednesday -> // ...
        Day.Thursday -> // ...
        Day.Friday -> // ...
        Day.Saturday -> // ...
    }
}
```

Tuesday MondayTuesday enum

◦ else

```
fun doOnDay(day: Day) {
    when(day) {
        Day.Monday -> // Work
        Day.Tuesday -> // Work hard
        Day.Wednesday -> // ...
        Day.Thursday -> //
        Day.Friday -> //
        else -> // Party on weekend
    }
}
```

if-then-elsewhenever

**kotlin** enum

<https://riptutorial.com/zh-TW/kotlin/topic/2685/>

# 26:

## Examples

“” when “” - °

```
import kotlin.text.regex

var string = /* some string */

val regex1 = Regex( /* pattern */ )
val regex2 = Regex( /* pattern */ )
/* etc */

when {
    regex1.matches(string) -> /* do stuff */
    regex2.matches(string) -> /* do stuff */
    /* etc */
}
```

“” whenwhen °

```
import kotlin.text.regex

var string = /* some string */

when {
    Regex( /* pattern */ ).matches(string) -> /* do stuff */
    Regex( /* pattern */ ).matches(string) -> /* do stuff */
    /* etc */
}
```

whenwhen “argument-ful” whenwhenEntrywhen “immutable locals” “anonymous temporaries” °

```
import kotlin.text.regex

var string = /* some string */

when (RegexWhenArgument(string)) {
    Regex( /* pattern */ ) -> /* do stuff */
    Regex( /* pattern */ ) -> /* do stuff */
    /* etc */
}
```

when

```
class RegexWhenArgument (val whenArgument: CharSequence) {
    operator fun equals(whenEntry: Regex) = whenEntry.matches(whenArgument)
    override operator fun equals(whenEntry: Any?) = (whenArgument == whenEntry)
}
```

## Kotlin

RegexRegexnull◦

# RegEx

KotlinRegex(pattern: String)regexfind(..)replace(..)◦

Regexinputcdtrue

```
val regex = Regex(pattern = "c|d")
val matched = regex.containsMatchIn(input = "abc") // matched: true
```

Regexpatterninput◦ ◦ containsMatchIn(..)◦

find(..)matchEntire(..)MatchResult?◦ ? MatchResultKotlin null◦

find(..)nullKotlinRegexnull

```
val matchResult =
    Regex("c|d").find("efg") // matchResult: null
val a = matchResult?.value // a: null
val b = matchResult?.value?.orEmpty() // b: ""
a?.toUpperCase() // Still needs question mark. => null
b.toUpperCase() // Accesses the function directly. => ""
```

orEmpty() bnull?b◦

KotlinJava!!

```
a!!.toUpperCase() // => KotlinNullPointerException
```

KotlinJavaJava◦

```
"""\d{3}-\d{3}-\d{4}"" // raw Kotlin string
"\d{3}-\d{3}-\d{4}" // standard Java string
```

# findCharSequencestartIndexIntMatchResult

inputRegexpattern◦ Matchresult?startIndexinputnull◦ MatchResult?MatchResult?value◦ startIndex0◦

```
val phoneNumber :String? = Regex(pattern = """\d{3}-\d{3}-\d{4}""")
```

```
.find(input = "phone: 123-456-7890, e..")?.value // phoneNumber: 123-456-7890
```

inputphoneNumbernull ◦

---

## findAllInputCharSequencestartIndexInt Sequence

patterninput ◦

```
val matchedResults = Regex(pattern = "\\d+").findAll(input = "ab12cd34ef")
val result = StringBuilder()
for (matchedText in matchedResults) {
    result.append(matchedText.value + " ")
}

println(result) // => 12 34
```

matchedResultsMatchResult ◦ inputfindAll(..) ◦

---

## matchEntireCharSequenceMatchResult

inputpatterninput ◦ null ◦

```
val a = Regex("\\d+").matchEntire("100")?.value // a: 100
val b = Regex("\\d+").matchEntire("100 dollars")?.value // b: null
```

---

## matchesCharSequenceBoolean

true ◦ ◦

```
val regex = Regex(pattern = "\\d+")
regex.matches(input = "50") // => true
regex.matches(input = "50 dollars") // => false
```

---

## containsMatchIninputCharSequenceBoolean

true ◦ ◦

```
Regex("\\d+").containsMatchIn("50 dollars") // => true
Regex("\\d+").containsMatchIn("Fifty dollars") // => false
```

# splitCharSequencelimitIntList

◦

```
val a = Regex("""\d+""").split("ab12cd34ef") // a: [ab, cd, ef]
val b = Regex("""\d+""").split("This is a test") // b: [This is a test]
```

◦ input ◦

---

# replaceCharSequencereplacementString String

inputpattern◦

X

```
val result = Regex("""\d+""").replace("ab12cd34ef", "x") // result: abxcdxef
```

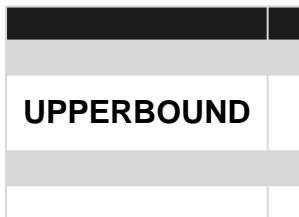
<https://riptutorial.com/zh-TW/kotlin/topic/8364/>

# 27:

◦ List◦

◦

- class *ClassName* < **TypeName** >
- class *ClassName* <\*>
- *ClassName* <in **UpperBound** >
- *ClassName* <out **LowerBound** >
- class *Name* < **TypeName UpperBound** >



Kotlin Generics TAny? ◦

```
class Consumer<T>
```

T: Any? ◦ T: Any ◦

```
class Consumer<T: Any>
```

## Examples

-

- ◦

```
class Consumer<in T> { fun consume(t: T) { ... } }

fun charSequencesConsumer() : Consumer<CharSequence>() = ...

val stringConsumer : Consumer<String> = charSequenceConsumer() // OK since in-projection
val anyConsumer : Consumer<Any> = charSequenceConsumer() // Error, Any cannot be passed

val outConsumer : Consumer<out CharSequence> = ... // Error, T is `in`-parameter
```

- List<out T> TComparator<in T> T◦

Java



```
val takeList : MutableList<out SomeType> = ... // Java: List<? extends SomeType>
val takenValue : SomeType = takeList[0] // OK, since upper bound is SomeType
takeList.add(takenValue) // Error, lower bound for generic is not specified
```

```
val putList : MutableList<in SomeType> = ... // Java: List<? super SomeType>
val valueToPut : SomeType = ...
putList.add(valueToPut) // OK, since lower bound is SomeType
putList[0] // This expression has type Any, since no upper bound is specified
```

```
val starList : MutableList<*> = ... // Java: List<?>
starList[0] // This expression has type Any, since no upper bound is specified
starList.add(someValue) // Error, lower bound for generic is not specified
```

- [JavaKotlinVariant Generics](#)。

<https://riptutorial.com/zh-TW/kotlin/topic/1147/>

# 28:

## Examples

### NullableNon Nullable

String ◦ ? String?

```
var string          : String = "Hello World!"
var nullableString: String? = null

string = nullableString // Compiler error: Can't assign nullable to non-nullable type.
nullableString = string // This will work however!
```

◦

? . `nullnull`;

```
val string: String? = "Hello World!"
print(string.length) // Compile error: Can't directly access property of nullable type.
print(string?.length) // Will print the string's length, or "null" if the string is null.
```

## null

### `nullKotlin`[apply](#)

```
obj?.apply {
    foo()
    bar()
}
```

objfooobj thisapplythis objnull◦

`let`[apply](#)

```
nullable?.let { notnull ->
    notnull.foo()
    notnull.bar()
}
```

notnull[lambda](#)[it](#) ◦

## null

```
var string: String? = "Hello!"
print(string.length) // Compile error
if(string != null) {
    // The compiler now knows that string can't be null
}
```

```
print(string.length) // It works now!
}
```

◦ null

◦

## Iterable

Collection<T?>Collections<T> ◦ filterNotNull ◦

```
val a: List<Int?> = listOf(1, 2, 3, null)
val b: List<Int> = a.filterNotNull()
```

## Null Coalescing / Elvis

if-else ◦ elvis?: Kotlin ◦

```
val value: String = data?.first() ?: "Nothing here."
```

data?.first() data null "Nothing here" data?.first() data?.first() ◦

◦

```
val value: String = data?.second()
?: throw IllegalArgumentException("Value can't be null!")
```

NullPointerException data!!.second()!!

!!nullability null ◦ null Kotlin NullPointerException ◦

```
val message: String? = null
println(message!!) // Kotlin NullPointerException thrown, app crashes
```

:)

Kotlin null reference ◦ a “null<sub>x</sub>”

```
var a: String? = "Nullable String Value"
```

a null ◦ a ◦ if...else ◦

```
val b: Int = if (a != null) a.length else -1
```

Elvis ?:if...else Elvis

```
val b = a?.length ?: -1
```

?: a?.length nullelvis -1 ◦ ◦

<https://riptutorial.com/zh-TW/kotlin/topic/2080/>

---

# 29:

Kotlin ◦ ◦

## Examples

```
interface Foo {
    fun example()
}

class Bar {
    fun example() {
        println("Hello, world!")
    }
}

class Baz(b : Bar) : Foo by b

Baz(Bar()).example()
```

Hello, world!

<https://riptutorial.com/zh-TW/kotlin/topic/10575/>

---

# 30: kotlin

Kotlin

## Examples

### kotlin.logging

```
class FooWithLogging {
    companion object: KLogging()

    fun bar() {
        logger.info { "hello $name" }
    }

    fun logException(e: Exception) {
        logger.error(e) { "Error occurred" }
    }
}
```

[kotlin.logging](#)

[kotlin https://riptutorial.com/zh-TW/kotlin/topic/3258/kotlin](https://riptutorial.com/zh-TW/kotlin/topic/3258/kotlin)

# 31:

Kotlin。 plain int。

## Examples

x

```
repeat(10) { i ->
    println("This line will be printed 10 times")
    println("We are on the ${i + 1}. loop iteration")
}
```

## iterables

### foriterable

```
val list = listOf("Hello", "World", "!")
for(str in list) {
    print(str)
}
```

### Kotlin

```
for(i in 0..9) {
    print(i)
}
```

```
for((index, element) in iterable.withIndex()) {
    print("$element at index $index")
}
```

### forEach

```
iterable.forEach {
    print(it.toString())
}
```

it [Lambda](#)

### whiledo-while

```
while(condition) {
    doSomething()
}

do {
    doSomething()
}
```

```
} while (condition)
```

## do-while

◦

```
while(true) {  
    if(condition1) {  
        continue // Will immediately start the next iteration, without executing the rest of  
the loop body  
    }  
    if(condition2) {  
        break // Will exit the loop completely  
    }  
}
```

## breakcontinue

```
outer@ for(i in 0..10) {  
    inner@ for(j in 0..10) {  
        break // Will break the inner loop  
        break@inner // Will break the inner loop  
        break@outer // Will break the outer loop  
    }  
}
```

forEach

## kotlin

```
//iterates over a map, getting the key and value at once  
  
var map = hashMapOf(1 to "foo", 2 to "bar", 3 to "baz")  
  
for ((key, value) in map) {  
    println("Map[$key] = $value")  
}
```

## Kotlin

```
fun factorial(n: Long): Long = if (n == 0) 1 else n * factorial(n - 1)  
  
println(factorial(10)) // 3628800
```

factorial

## Kotlin

map

```
val numbers = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
```



```
val numberStrings = numbers.map { "Number $it" }
```

- ◦ `filter`◦

```
val numbers = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
val numberStrings = numbers.filter { it % 2 == 0 }.map { "Number $it" }
```

<https://riptutorial.com/zh-TW/kotlin/topic/2727/>

## 32:

rangeTo..in

## Examples

IntRangeLongRangeCharRange。Javafor-loop

```
for (i in 1..4) print(i) // prints "1234"  
for (i in 4..1) print(i) // prints nothing
```

## downTo

◦ downTo

```
for (i in 4 downTo 1) print(i) // prints "4321"
```

## step

1step

```
for (i in 1..4 step 2) print(i) // prints "13"  
for (i in 4 downTo 1 step 2) print(i) // prints "42"
```

## enduntil

```
for (i in 1 until 10) { // i in [1, 10), 10 is excluded  
    println(i)  
}
```

<https://riptutorial.com/zh-TW/kotlin/topic/10121/>

# 33:

## Examples

◦

```
annotation class Strippable
```

```
@Target (AnnotationTarget.CLASS, AnnotationTarget.FUNCTION,  
AnnotationTarget.VALUE_PARAMETER, AnnotationTarget.EXPRESSION)  
annotation class Strippable
```

```
annotation class Strippable(val importanceValue: Int)
```

- JavalntLong;
- 
- Foo :: class
- 
- 
- 
- @Target
- @Retention◦
- @Repeatable◦
- @MustBeDocumentedAPIAPI◦

```
@Target (AnnotationTarget.CLASS, AnnotationTarget.FUNCTION,  
AnnotationTarget.VALUE_PARAMETER, AnnotationTarget.EXPRESSION)  
@Retention (AnnotationRetention.SOURCE)  
@MustBeDocumented  
annotation class Fancy
```

<https://riptutorial.com/zh-TW/kotlin/topic/4074/>

---

## 34:

- (String) -> BooleanPair<Person, Person> ◦
- ◦
- **typealias** *alias-name = existing-type*
- JVM ◦ ◦

## Examples

```
 typealias StringValidator = (String) -> Boolean
 typealias Reductor<T, U, V> = (T, U) -> V
```

```
 typealias Parents = Pair<Person, Person>
 typealias Accounts = List<Account>
```

<https://riptutorial.com/zh-TW/kotlin/topic/9453/>

# 35: Kotlin

## Examples

### Gradle

kotlin-gradle-plugin Gradle Kotlin Kotlin 1.0.3 kotlin-gradle-plugin 1.0.3

[gradle.propertiesExtraPropertiesExtension](#)

```
buildscript {
    ext.kotlin_version = '1.0.3'

    repositories {
        mavenCentral()
    }

    dependencies {
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
```

◦

### JVM

```
apply plugin: 'kotlin'
```

### Android

```
apply plugin: 'kotlin-android'
```

### JS

```
apply plugin: 'kotlin2js'
```

- **kotlin** src/main/kotlin
- **java** src/main/java
- **kotlin** src/test/kotlin
- **java** src/test/java
- src/main/resources
- src/test/resources

[SourceSets](#) ◦

### Kotlin

```
dependencies {  
    compile "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"  
}
```

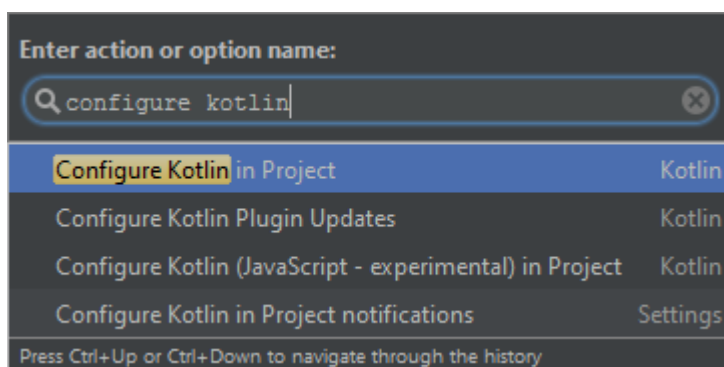
**Kotlin Reflection** `compile "org.jetbrains.kotlin:kotlin-reflect:$kotlin_version"`

## Android Studio

Android Studio Android Kotlin

Kotlin >>> JetBrains... > Kotlin > Android Studio

Android Studio `Ctrl + Shift + A`. "Kotlin" Enter



Android Studio Gradle

## Java

Java Kotlin `Ctrl + Shift + A` "Java Kotlin" .kt Kotlin

```
package com.orangeflash81.myapplication;

public class Foo {
    private String name = "Joe Bloggs";

    String getName() { return name; }

    void setName(String value) { name = value; }
}
```

## GroovyGradleKotlin

- [gradle-script-kotlin](#)
- /
  - build.gradle.kts
  - gradlew
  - gradlew.bat
  - settings.gradle
- build.gradle.kts
- IntelliJGradle◦
- IntelliJbuild.gradle.kts◦ /IntelliJ
- Gradle

Windows Gradle 3.3◦ ◦

OSXUbuntu。

MavenJitpack Groovy。。

Kotlin <https://riptutorial.com/zh-TW/kotlin/topic/2501/kotlin>



# 36:

Kotlin API.

- listOfmapOfsetOf.
- arrayListOfhashMapOfhashSetOflinkedMapOfLinkedHashMaplinkedSetOfLinkedHashSet mutableListOfKotlin MutableListmutableMapOfThe Kotlin MutableListmutableSetOfThe Kotlin MutableListSet sortedMapOfsortedSetOf
- firstlastgetlambdafiltermapjoinreduce.

## Examples

```
// Create a new read-only List<String>
val list = listOf("Item 1", "Item 2", "Item 3")
println(list) // prints "[Item 1, Item 2, Item 3]"
```

```
// Create a new read-only Map<Integer, String>
val map = mapOf(Pair(1, "Item 1"), Pair(2, "Item 2"), Pair(3, "Item 3"))
println(map) // prints "{1=Item 1, 2=Item 2, 3=Item 3}"
```

## set

```
// Create a new read-only Set<String>
val set = setOf(1, 3, 5)
println(set) // prints "[1, 3, 5]"
```

<https://riptutorial.com/zh-TW/kotlin/topic/8846/>

# 37:

◦

3

1. ◦
2. lambda◦
3. ◦

## Kotlin

Kotlin

- 
- 
- Ktor
- 

## Examples

lambda◦ JFrame

```
import javax.swing.*

fun JFrame.menuBar(init: JMenuBar.() -> Unit) {
    val menuBar = JMenuBar()
    menuBar.init()
    setJMenuBar(menuBar)
}

fun JMenuBar.menu(caption: String, init: JMenu.() -> Unit) {
    val menu = JMenu(caption)
    menu.init()
    add(menu)
}

fun JMenu.menuItem(caption: String, init: JMenuItem.() -> Unit) {
    val menuItem = JMenuItem(caption)
    menuItem.init()
    add(menuItem)
}
```

```
class MyFrame : JFrame() {
    init {
        menuBar {
            menu("Menu1") {
                menuItem("Item1") {
                    // Initialize MenuItem with some Action
                }
                menuItem("Item2") {}
            }
        }
    }
}
```

```
    }  
    menu("Menu2") {  
        menuItem("Item3") {}  
        menuItem("Item4") {}  
    }  
}  
}
```

<https://riptutorial.com/zh-TW/kotlin/topic/6010/>

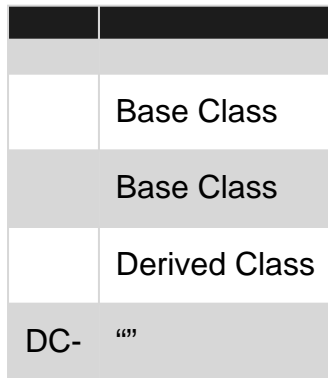
# 38:

Apples OrangesPears ◦ ◦

```
Fruit getFruit()
```

Fruit ◦

- {}
- class {Derived Class}{Base Class}{Init Arguments}
- {}
- {DC-Object}{Base Class} == true



## Examples

### 'open'

Kotlin **final**.

open ◦

```
open class Thing {  
    // I can now be extended!  
}
```

open ◦

```
open class BaseClass {  
    val x = 10  
}
```

```
class DerivedClass: BaseClass() {  
    fun foo() {  
        println("x is equal to " + x)  
    }  
}
```

```
}
```

```
fun main(args: Array<String>) {  
    val derivedClass = DerivedClass()  
    derivedClass.foo() // prints: 'x is equal to 10'  
}
```

```
open class Person {  
    fun jump() {  
        println("Jumping...")  
    }  
}
```

```
class Ninja: Person() {  
    fun sneak() {  
        println("Sneaking around...")  
    }  
}
```

## Person

```
fun main(args: Array<String>) {  
    val ninja = Ninja()  
    ninja.jump() // prints: 'Jumping...'  
    ninja.sneak() // prints: 'Sneaking around...'  
}
```

```
abstract class Car {  
    abstract val name: String;  
    open var speed: Int = 0;  
}  
  
class BrokenCar(override val name: String) : Car() {  
    override var speed: Int  
        get() = 0  
        set(value) {  
            throw UnsupportedOperationException("The car is broken")  
        }  
}  
  
fun main(args: Array<String>) {  
    val car: Car = BrokenCar("Lada")  
    car.speed = 10  
}
```

```
interface Ship {  
    fun sail()  
    fun sink()  
}  
  
object Titanic : Ship {  
  
    var canSail = true
```

```
override fun sail() {  
    sink()  
}  
  
override fun sink() {  
    canSail = false  
}  
}
```

<https://riptutorial.com/zh-TW/kotlin/topic/5622/>

S. No		Contributors
1	Kotlin	<a href="#">babedev</a> , <a href="#">Community</a> , <a href="#">cyberscientist</a> , <a href="#">ganesshkumar</a> , <a href="#">Ihor Kucherenko</a> , <a href="#">Jayson Minard</a> , <a href="#">mnoronha</a> , <a href="#">newworld</a> , <a href="#">Parker Hoyes</a> , <a href="#">Ruckus T-Boom</a> , <a href="#">Sach</a> , <a href="#">Sean Reilly</a> , <a href="#">Sheigutn</a> , <a href="#">Simón Oroño</a> , <a href="#">UnKnown</a> , <a href="#">Urko Pineda</a>
2	DSL	<a href="#">Dmitriy L</a> , <a href="#">ice1000</a>
3	Java 8 Stream	<a href="#">Brad</a> , <a href="#">Gerson</a> , <a href="#">Jayson Minard</a> , <a href="#">Piero Divasto</a> , <a href="#">Sam</a>
4	JUnit	<a href="#">jenglert</a>
5	Kotlin Android	<a href="#">Jemo Mgebrishvili</a> , <a href="#">Ritave</a>
6	Kotlin for Java Developers	<a href="#">Thorsten Schleinzer</a>
7	Kotlin	<a href="#">Shinoo Goyal</a>
8	Kotlin	<a href="#">Grigory Konushev</a> , <a href="#">Spidfire</a>
9		<a href="#">Brad Larson</a> , <a href="#">jereksel</a> , <a href="#">Sapan Zaveri</a>
10	Vararg	<a href="#">byxor</a> , <a href="#">piotrek1543</a> , <a href="#">Sam</a>
11		<a href="#">Aaron Christiansen</a> , <a href="#">baha</a> , <a href="#">Caelum</a> , <a href="#">glee8e</a> , <a href="#">Jayson Minard</a> , <a href="#">KeksArmee</a> , <a href="#">madhead</a> , <a href="#">Spidfire</a>
12		<a href="#">Jemo Mgebrishvili</a>
13		<a href="#">atok</a> , <a href="#">Kirill Rakhman</a> , <a href="#">madhead</a> , <a href="#">Ritave</a> , <a href="#">Sup</a>
14		<a href="#">Avijit Karmakar</a>
15		<a href="#">Divya</a> , <a href="#">glee8e</a>
16	KotlinRecyclerView	<a href="#">Mohit Suthar</a>
17	Lambdas	<a href="#">memoizr</a> , <a href="#">Rich Kuzsma</a>
18		<a href="#">Sam</a> , <a href="#">Seaskyways</a>
19		<a href="#">Januson</a> , <a href="#">Sam</a>
20		<a href="#">Aaron Christiansen</a> , <a href="#">Adam Arold</a> , <a href="#">Brad Larson</a> , <a href="#">Héctor</a> , <a href="#">Jayson Minard</a> , <a href="#">Konrad Jamrozik</a> , <a href="#">madhead</a> , <a href="#">mayojava</a> , <a href="#">razzledazzle</a> , <a href="#">Sapan Zaveri</a> , <a href="#">Serge Nikitin</a> , <a href="#">yole</a>

21		<a href="#">Divya</a> , <a href="#">Jan Vladimir Mostert</a> , <a href="#">Jayson Minard</a> , <a href="#">Ritave</a> , <a href="#">Robin</a>
22		<a href="#">Dávid Tímár</a> , <a href="#">Jayson Minard</a> , <a href="#">Kevin Robatel</a> , <a href="#">Konrad Jamrozik</a> , <a href="#">olivierlemasle</a> , <a href="#">Parker Hoyes</a> , <a href="#">razzledazzle</a>
23		<a href="#">egor.zhdan</a> , <a href="#">Sam</a> , <a href="#">UnKnown</a>
24		<a href="#">David Soroko</a> , <a href="#">Kirill Rakhman</a> , <a href="#">SerCe</a>
25		<a href="#">Abdullah</a> , <a href="#">Alex Facciorusso</a> , <a href="#">jpmcosta</a> , <a href="#">Kirill Rakhman</a> , <a href="#">Robin</a> , <a href="#">Spidfire</a>
26		<a href="#">Espen</a> , <a href="#">Travis</a>
27		<a href="#">hotkey</a> , <a href="#">Jayson Minard</a> , <a href="#">KeksArmee</a>
28		<a href="#">KeksArmee</a> , <a href="#">Kirill Rakhman</a> , <a href="#">piotrek1543</a> , <a href="#">razzledazzle</a> , <a href="#">Robin</a> , <a href="#">SerCe</a> , <a href="#">Spidfire</a> , <a href="#">technerd</a> , <a href="#">Thorsten Schleinzer</a>
29		<a href="#">Sam</a>
30	kotlin	<a href="#">Konrad Jamrozik</a> , <a href="#">olivierlemasle</a> , <a href="#">oshai</a>
31		<a href="#">Ben Leggiero</a> , <a href="#">JaseAnderson</a> , <a href="#">mayojava</a> , <a href="#">razzledazzle</a> , <a href="#">Robin</a>
32		<a href="#">Nihal Saxena</a>
33		<a href="#">Brad Larson</a> , <a href="#">Caelum</a> , <a href="#">Héctor</a> , <a href="#">Mood</a> , <a href="#">piotrek1543</a> , <a href="#">Sapan Zaveri</a>
34		<a href="#">Kevin Robatel</a>
35	Kotlin	<a href="#">Aaron Christiansen</a> , <a href="#">elect</a> , <a href="#">madhead</a>
36		<a href="#">Ascension</a>
37		<a href="#">Slav</a>
38		<a href="#">byxor</a> , <a href="#">KeksArmee</a> , <a href="#">piotrek1543</a> , <a href="#">Slav</a>