Kotlin Cheat Sheet

Current release line — Kotlin 2.4.x (JVM)

TYPES & VARIABLES

```
val x = 5          // read-only
var y = 0          // mutable
val s: String = "hi"
val n: Int? = null  // nullable
```

`Int` `Long` `Double` `Float` `Boolean` `Char` — no boxed/primitive split

`Any` root type · `Unit` = void · `Nothing` = never returns

`UInt` `ULong` unsigned · `IntArray` primitive array

STRINGS

```
"Hi, $name! ${a+b}"
"""raw
multi-line"""
str.trimIndent()
```

`==` structural · `===` referential (see Equality)

OPERATORS & RANGES

```
1..10   1 until 10   10 downTo 1
1..10 step 2
x in 1..5     x !in 1..5
```

Bitwise: `and` `or` `xor` `shl` `shr` `inv` (no symbols)

`infix fun` → call without dot/parens

CONTROL FLOW

```
val m = if (a > b) a else b

val d = when (x) {
  1, 2 -> "small"
  in 3..9 -> "medium"
  is String -> "text"
  else -> "other"
}
```

`for (i in 1..5)` · `break@label` / `continue@label`

FUNCTIONS

```
fun add(a: Int, b: Int) = a + b
fun f(n: Int, admin: Boolean = false) {}
f(5, admin = true)
// named arg
fun sum(vararg xs: Int) = xs.sum()
tailrec fun fact(n: Long, acc: Long = 1): Long =
  if (n <= 1) acc else fact(n - 1, n * acc)
```

CLASSES & DATA CLASSES

```
class Person(val name: String, var age: Int)

data class Point(val x: Int, val y: Int)
// equals/hashCode/toString/copy/componentN generated
val (x, y) = Point(1, 2)
// destructuring
p.copy(y = 9)
```

Classes are **final** by default → `open class` to allow subclassing

INHERITANCE & INTERFACES

```
open class Shape { open fun area() = 0.0 }
class Circle(val r: Double) : Shape() {
  override fun area() = Math.PI * r * r
}
interface Greet { val g: String; fun hi() = println(g) }
class Loud(d: Repo) : Repo by d      // delegation
```

SEALED & ENUM

```
sealed interface Result
data class Ok(val v: String) : Result
object Pending : Result

when (r) {
// exhaustive, no else needed
  is Ok -> r.v
  Pending -> "..."
}

enum class Dir(val deg: Int) { N(0), E(90), S(180), W(270) }
```

OBJECTS & COMPANIONS

```
object Config { var env = "prod" }

class User private constructor(val id: Int) {
  companion object {
    fun create() = User(1)
// factory, called User.create()
  }
}

val l = object : Listener {
override fun onClick() {} }
```

EXTENSION & SCOPE FUNCTIONS

```
fun String.shout() = uppercase() + "!"
"hi".shout()
```

`let`	it → result	null-check/transform
`run`	this → result	init + compute
`with`	this → result	several calls, one obj
`apply`	this → self	configure & return
`also`	it → self	side effect in a chain

LAMBDAS & HIGHER-ORDER FNS

```
val sq: (Int) -> Int = { x -> x * x }
list.map { it * 2 }
inline fun  time(block: () -> T): T { ... }
```

`inline` pastes body at call site → allows non-local `return`

`noinline` / `crossinline` refine which lambdas inline

GENERICS & VARIANCE

```
class Box(var v: T)
interface Producer { fun get(): T }
interface Consumer { fun put(t: T) }
inline fun  isA(v: Any) = v is T
fun show(b: Box<*>) {}
// star projection
```

NULL SAFETY

```
var s: String? = null
val len = s?.length ?: 0
// safe call + Elvis
val forced = s!!.length
// throws NPE if null
val i = (any as? Int) ?: -1
// safe cast
if (s != null) println(s.length) // smart cast
```

EQUALITY & OPERATORS

```
a == b     // structural (.equals)
a === b    // referential (same instance)

operator fun plus(o: Vec) = Vec(x+o.x, y+o.y)
operator fun invoke(x: Int) = x * factor
operator fun get(i: Int) = items[i]
```

COLLECTIONS & SEQUENCES

```
listOf(1,2,3)
// read-only
mutableListOf(1,2,3).add(4)
nums.map{it*2}.filter{it>2}.fold(0){a,n->a+n}
nums.groupBy{it%2};
nums.associateBy{it.id}
list.asSequence().map{}.first{}
// lazy
```

EXCEPTIONS

```
val n = try { s.toInt() } catch (e: NumberFormatException) { 0 }
// NO checked exceptions in Kotlin
val r: Result = runCatching { fetch() }
r.onSuccess{}.onFailure{}
r.getOrNull();
r.getOrDefault(GUEST)
```

COROUTINES

```
suspend fun load(): User { delay(100); return u }

fun main() = runBlocking {
  launch { ... }
// fire-and-forget
  val d = async { compute() }
  println(d.await())
}
scope.launch(Dispatchers.IO) { ... }
```

Structured concurrency: cancel parent → cancels all children

CONTEXT, CANCELLATION & ERRORS

```
withContext(Dispatchers.Default) { compute() }
withTimeoutOrNull(2_000) { fetch() }   // null on timeout
val scope = CoroutineScope(SupervisorJob() + handler)
val handler = CoroutineExceptionHandler { _, e -> log(e) }
```

Dispatchers: `Default` CPU · `IO` blocking I/O · `Main` UI

FLOWS

```
fun nums(): Flow = flow {
  for (i in 1..3) {
delay(100); emit(i) }
}
nums().map{it*2}.collect{ println(it) }

val _s = MutableStateFlow(0)
// hot, always has a value
val shared = MutableSharedFlow() // hot, broadcast events
flow.buffer();
flow.conflate()
```

ANDROID & TOOLING ONE-LINERS

- Google's preferred Android language since 2019
- KTX: `bundleOf()`, `prefs.edit {}`
- `viewModelScope` / `lifecycleScope` — auto-cancelled coroutine scopes
- Jetpack: `ViewModel`, Room, Compose
- `build.gradle.kts` — Gradle Kotlin DSL
- `kotlin-maven-plugin` for Maven builds
- Static analysis: `ktlint`, `detekt`
- Testing: `kotlin.test`, JUnit 5, MockK, `runTest`
- Spring Boot: `kotlin-spring` / `kotlin-jpa` plugins open classes automatically