Kotlin: The Modern Crash Course

Java Boilerplate

100% Java Interoperability: Runs on the same JVM bytecode.

Versatile: Android, Spring Boot Backend, and Multiplatform.

Null Safety: Designed to prevent the Billion Dollar Mistake.

```kotlin
// Modern Kotlin
fun main() {
    val message: String? = getMessage()
    message?.let {
        println("Received: $it") // Safe call
    } ?: run {
        println("No message received")
    }

    val items = listOf("Android", "Backend", "Multiplatform")
    items.filter { it.length > 7 }
         .map { it.uppercase() }
         .forEach { println(it) }
}

fun getMessage(): String? {
    return if (System.currentTimeMillis() % 2 == 0L) "Hello Kotlin!" else null
}
```

From Hello World to Generics in 2025

The Entry Point & Variables

```kotlin
fun main() {
    // Immutable (The Standard)
    val name = "Kotlin"
    // name = "Java"  // Error: Val canno

    // Mutable (When necessary)
    var age = 13
    age = 14        // OK
}
```

Program Execution Starts Here

val

Immutable.
Read-only.

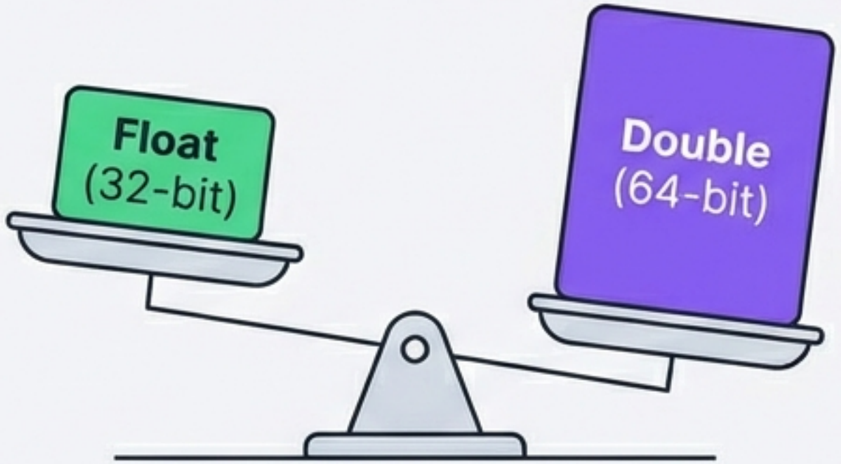
var

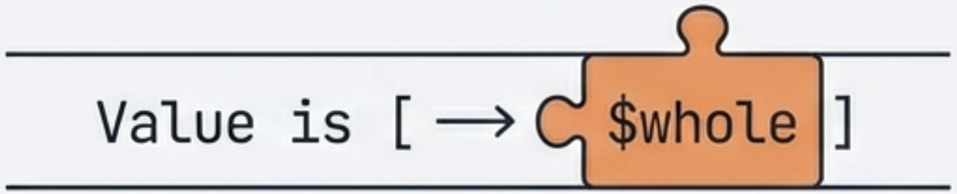
Mutable.
Reassignable.

Type Inference:
Compiler auto-detects type.

```
1  val whole: Int = 5
2  val decimal: Double = 5.0 // 64-bit preci
3  val float: Float = 5.0f // 32-bit precisi
4  val flag: Boolean = true
5
6  // String Templates
7  val message = "Value is $whole"
8  val math = "Sum: ${whole + 5}"
9
```

Primitives & Templates

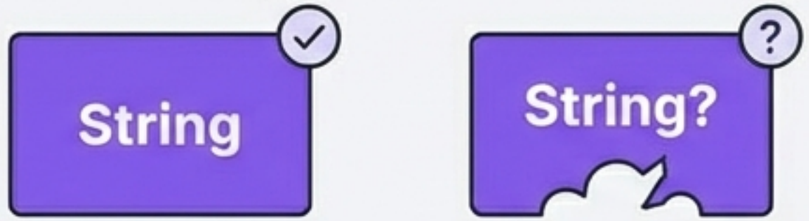




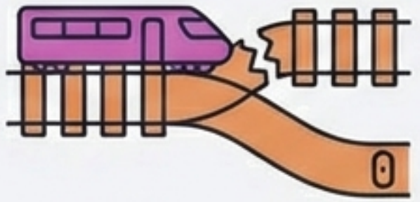
Math Warning: 5 / 9 = 0. Integer division truncates decimals.

```kotlin
var text: String? = null // Nullable

// 1. Safe Call (?.)
val length = text?.length

// 2. Elvis Operator (?:)
val safeLen = text?.length ?: 0

// 3. Assertion (!!)
// val crash = text!!.length
```

Null Safety



Safe Call (?.)
If null, stop.

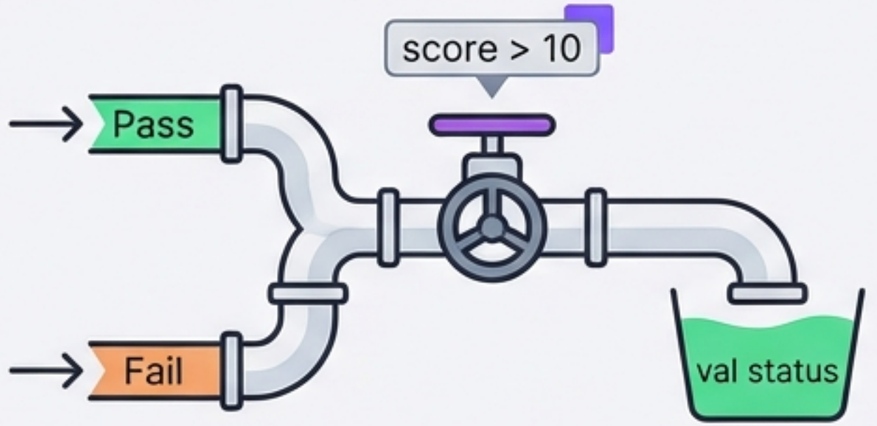


Elvis Operator (?:)
If track is broken (null), switch to default path (0).

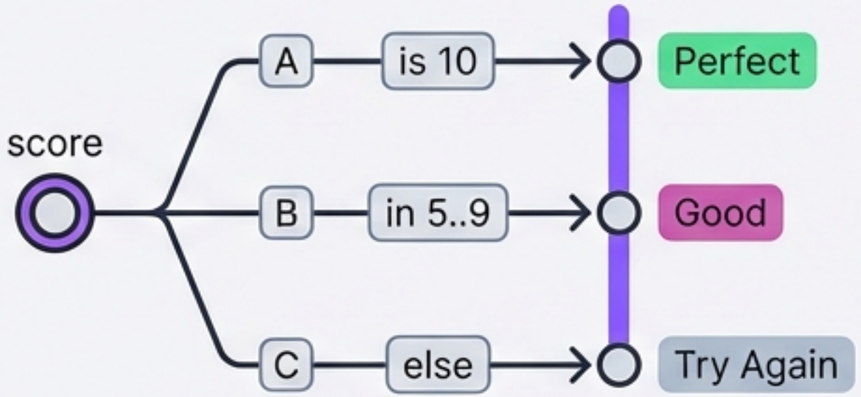
Assertion (!!)
Throws NullPointerException. Danger Zone.

Control Flow Expressions

```
1 // 'If' is an Expression
2 val status = if (score > 10) "Pass" else "Fail"
3
4 // 'When' (Better Switch)
5 val feedback = when (score) {
6     10 -> "Perfect"
7     in 5..9 -> "Good"
8     else -> "Try Again"
9 }
```



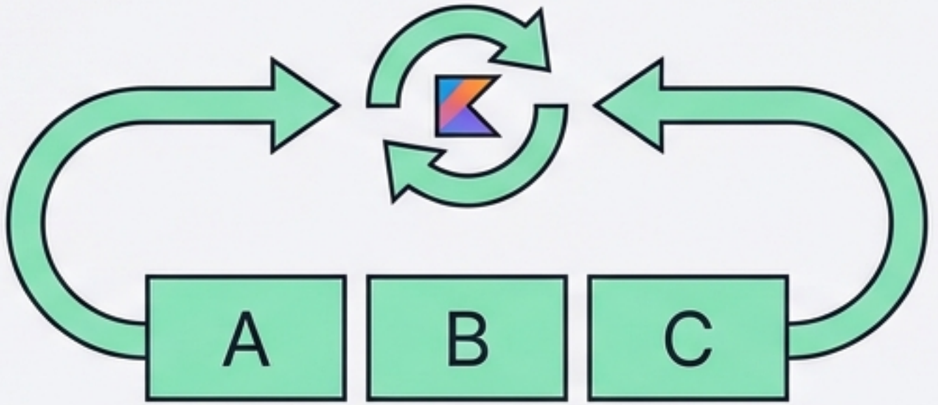
Assigns result directly.

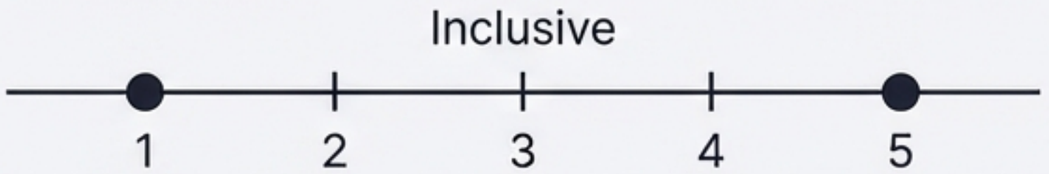


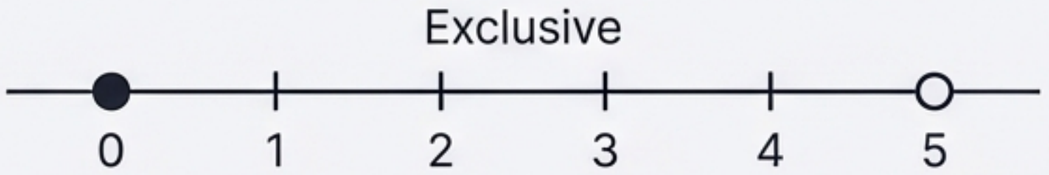
When Expression

```kotlin
val items = listOf("A", "B", "C")

// Loop Items
for (item in items) {
    println(item)
}

// Ranges
for (i in 1..5) { … } // 1, 2, 3, 4, 5
for (i in 0 until 5) { … // 0, 1, 2, 3, 4
    …
}
```

Loops & Ranges



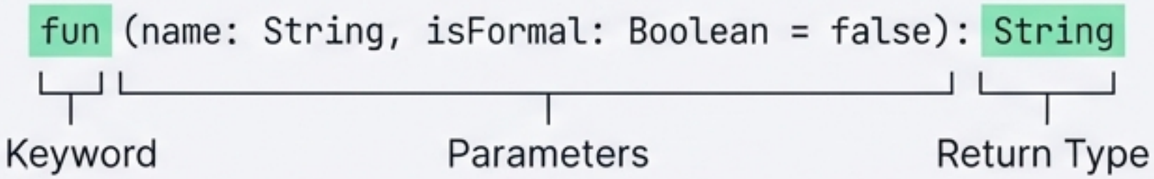




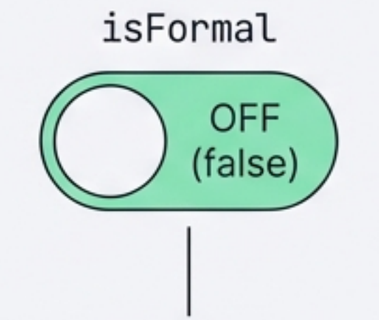
```kotlin
fun greet(
    name: String,
    isFormal: Boolean = false // Default
): String {
    return if (isFormal) "Greetings $name"
    else "Hi"
}

// Named Arguments & Defaults
greet("Jane") // isFormal = false
greet(isFormal = true, name = "John")
```

Functions 2.0

Anatomy

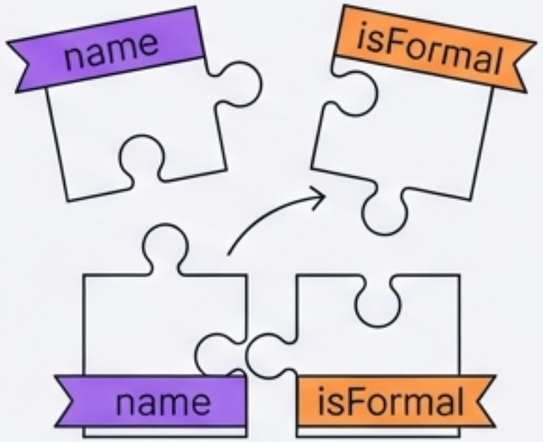


Default Values



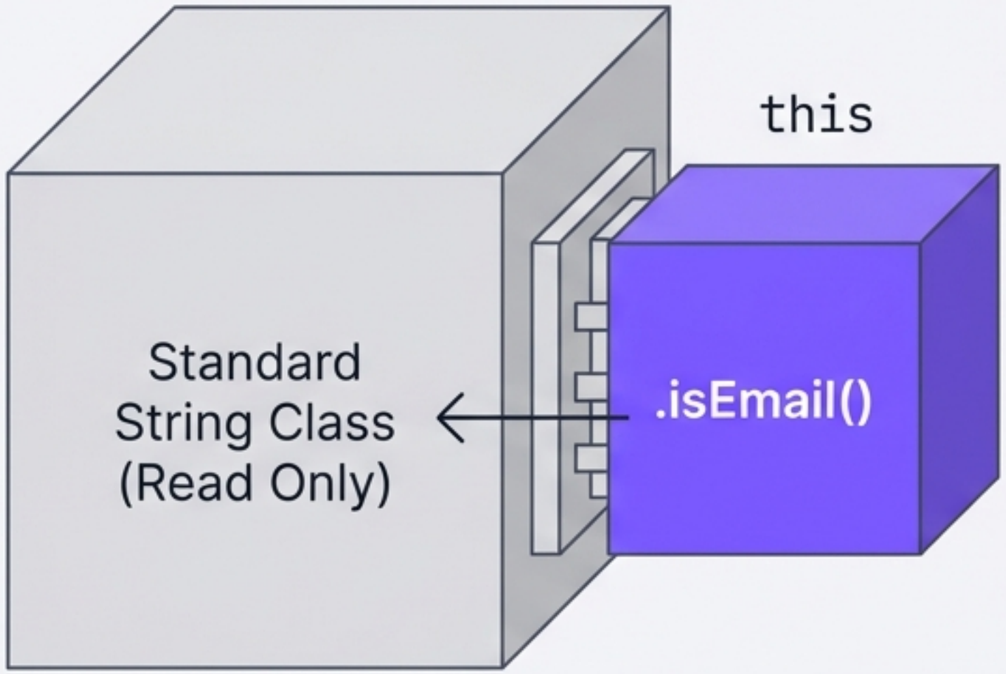
No value provided?
Switch stays default.

Named Args



Extension Functions

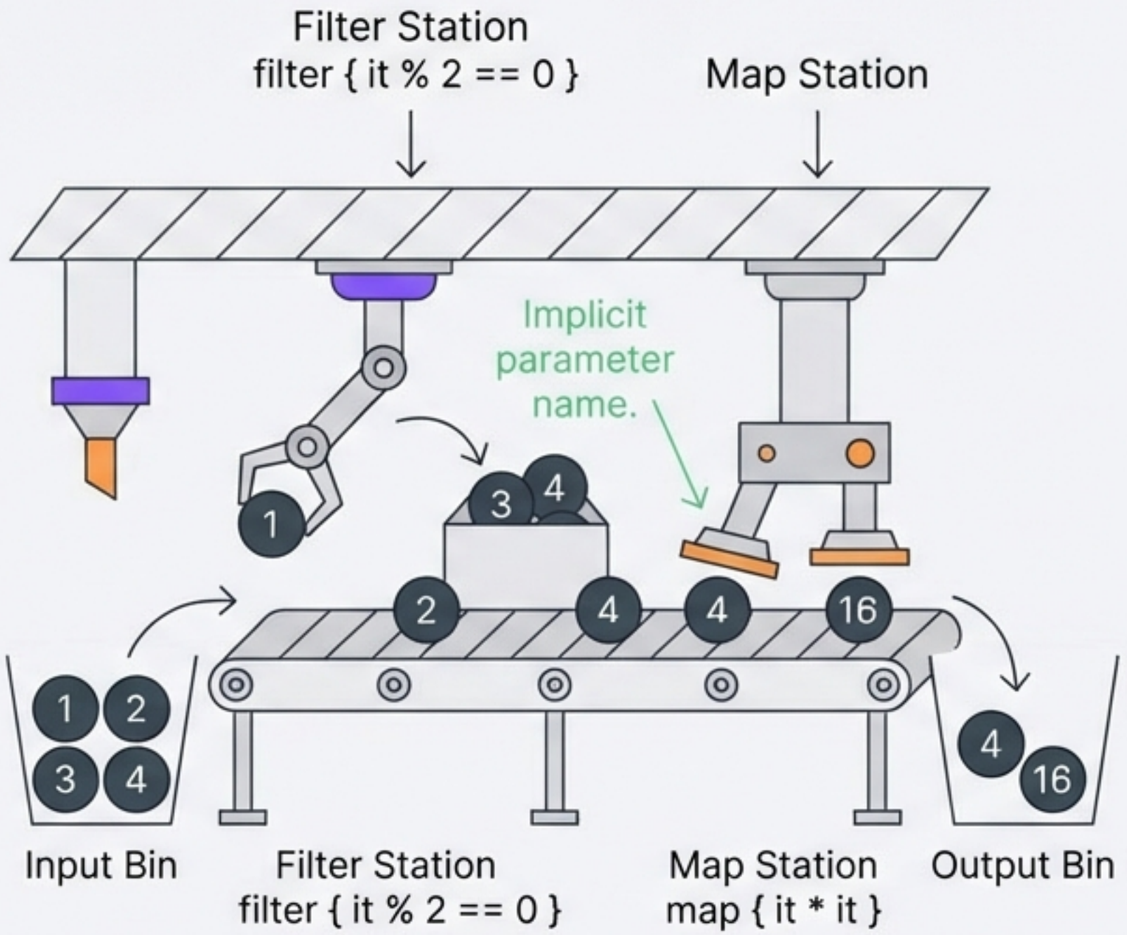
```kotlin
// Adding function to 'String' class
fun String.isEmail(): Boolean {
    return this.contains("@")
}

// Usage
val input = "user@kotlin.com"
val valid = input.isEmail()
```



Add functionality to classes you don’t own, without inheritance.

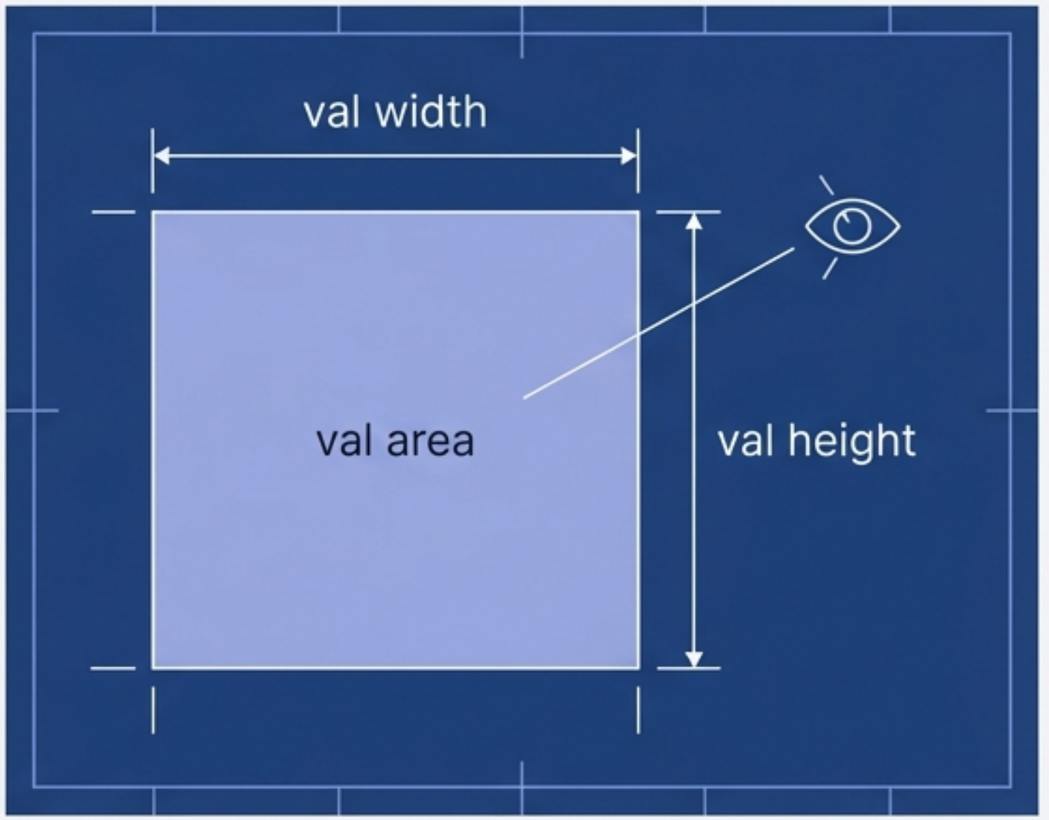
```
val numbers = listOf(1, 2, 3, 4)

val result = numbers
    .filter { it % 2 == 0 } // Lambda
    .map { it * it }
```

Lambdas & Pipelines



```
class Rectangle(
    val width: Int,  // Property
    val height: Int
) {
    // Custom Getter (Calculated)
    val area: Int
        get() = width * height
}

val box = Rectangle(5, 5)
```

Classes & Properties

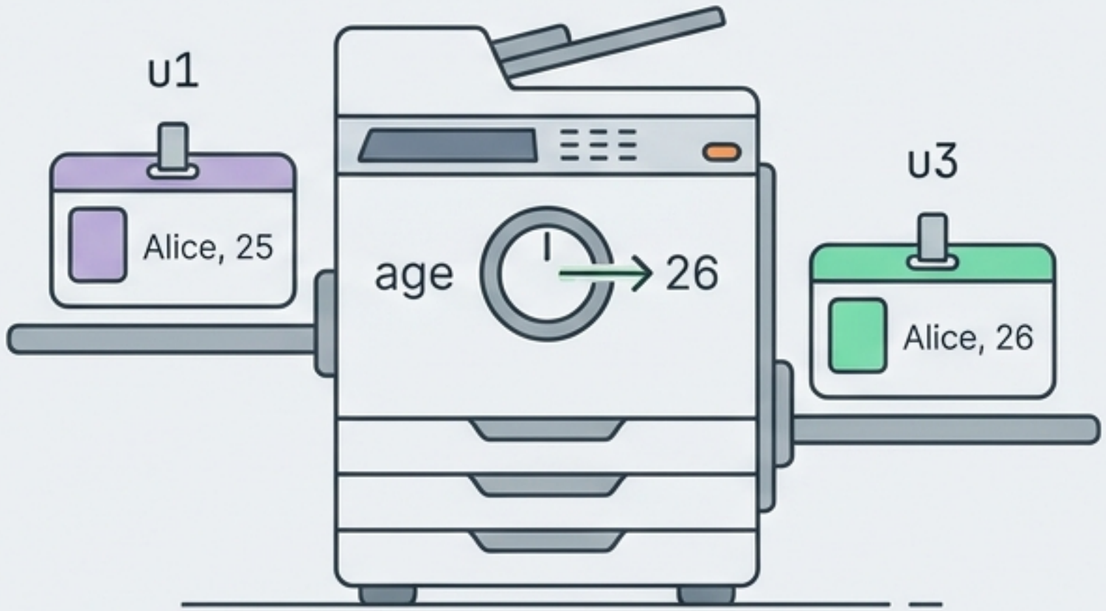


Primary Constructor defined directly in the header. No boilerplate.

```kotlin
data class User(val name: String,
  val age: Int)

val u1 = User("Alice", 25)
val u2 = User("Alice", 25)

// Checks CONTENT, not reference
println(u1 == u2) // True

// Copy
val u3 = u1.copy(age = 26)
```

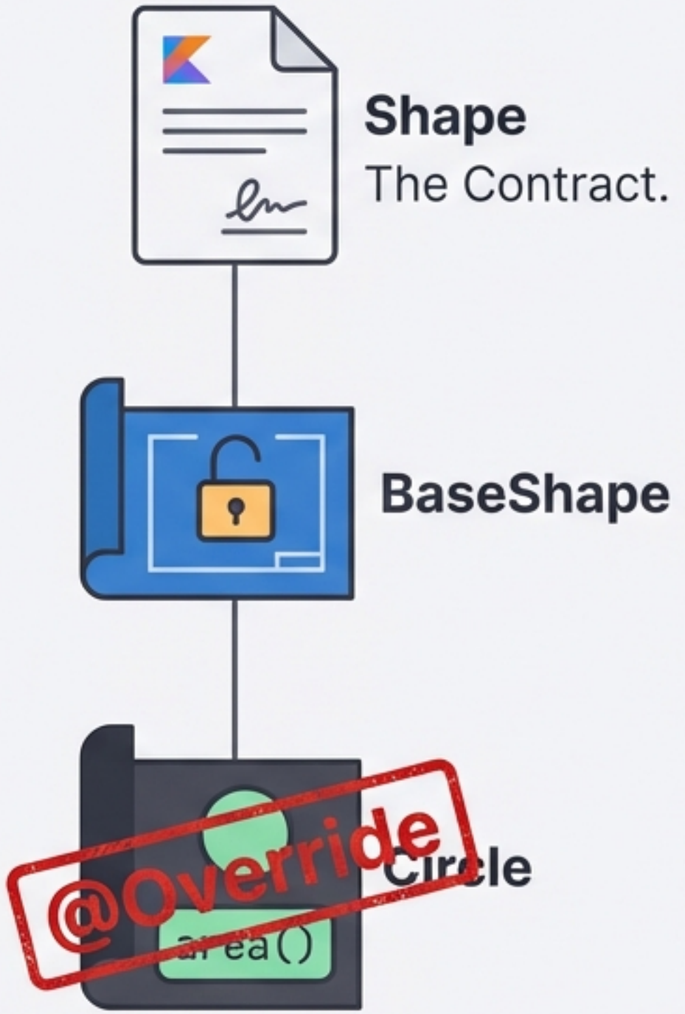
Data Classes

Structural Equality
(Data matches).



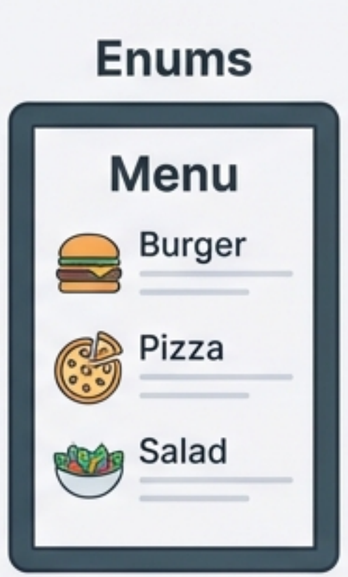
Inheritance & Interfaces

```kotlin
interface Shape {
    fun area(): Double // Contract
}

// 'open' allows inheritance
open class BaseShape : Shape {
    override fun area() = 0.0
}

class Circle(val r: Double) : BaseShape() {
    override fun area() = 3.14 * r * r
}
```

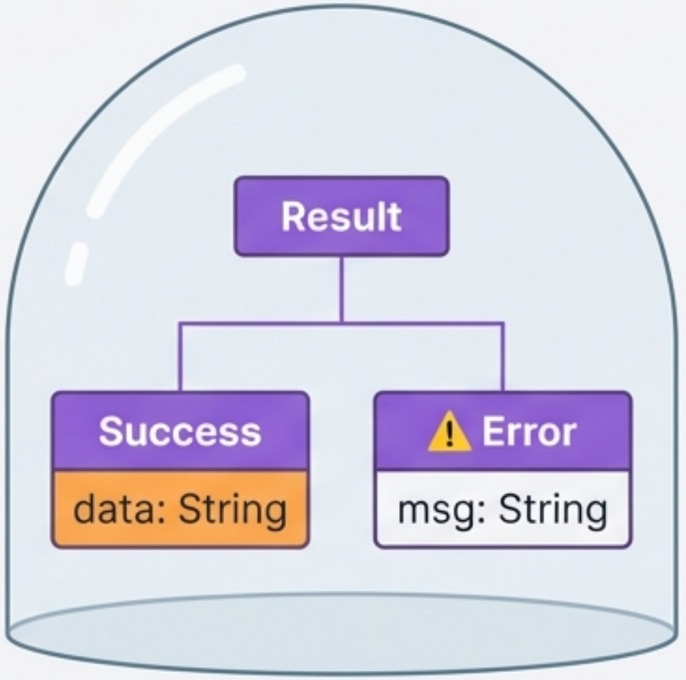


```
sealed class Result
data class Success(val data: String) : Result()
data class Error(val msg: String) : Result()


fun handle(r: Result) = when(r) {
    is Success -> print(r.data)
    is Error -> print(r.msg)
    // No 'else' needed!
}
```

Enums vs. Sealed Classes



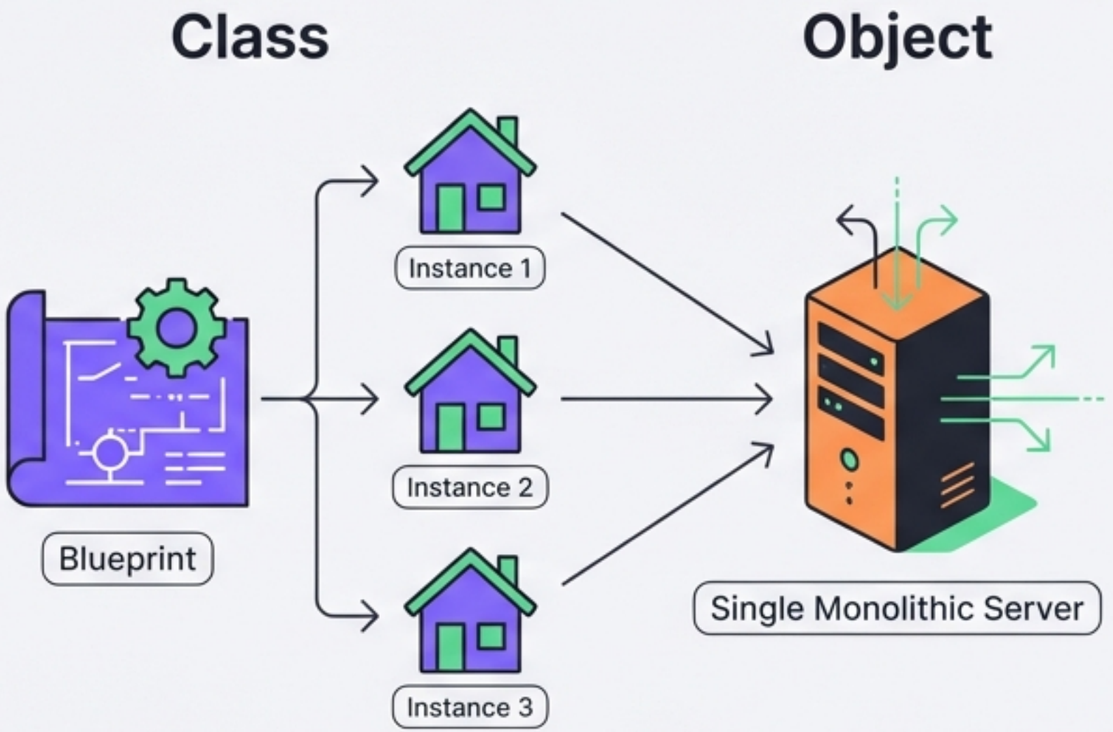
Constants.



Exhaustive.
Compiler knows all possible subclasses. No missing cases.

```
1  object Database {
2      val url = "jdbc:mysql://..."
3      fun connect() { ... }
4  }
5
6  // Usage
7  Database.connect()
```

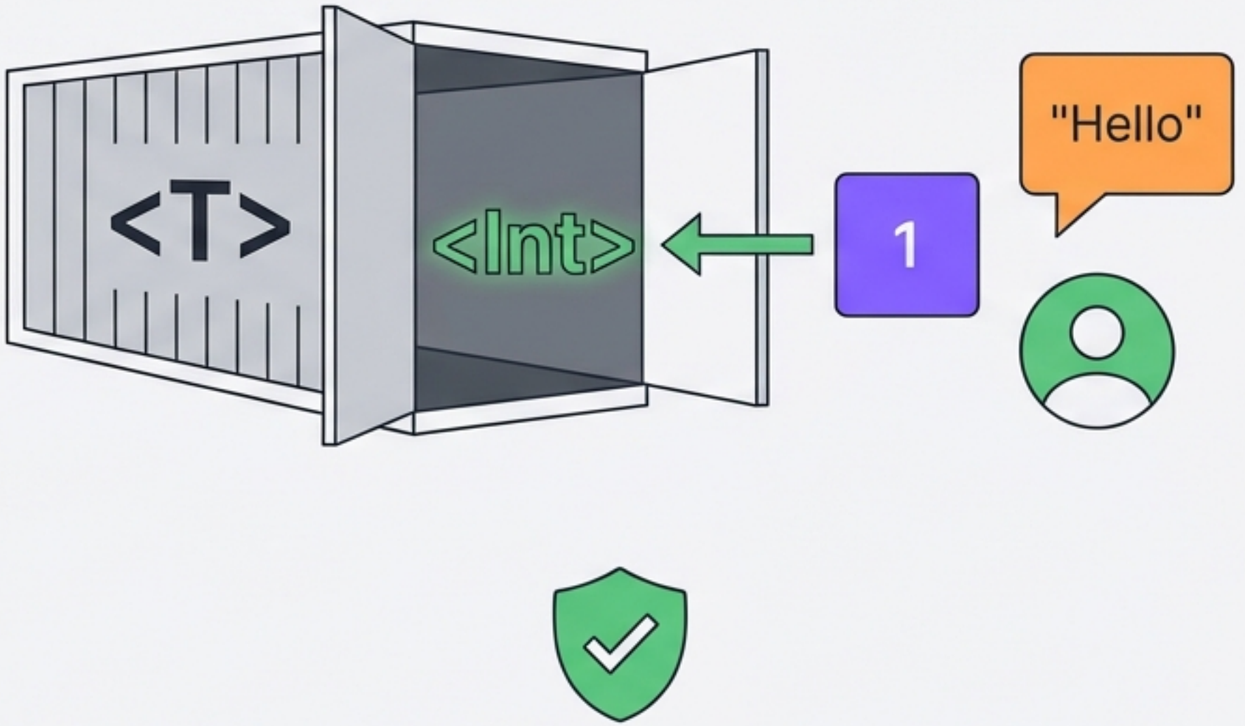
The Singleton (Object)



One instance, created immediately, shared everywhere.

Generics

```
1  class Box<T>(val item: T)
2
3  val intBox = Box(1)        // T is Int
4  val strBox = Box("Hello") // T is String
```



Type Safety: Compiler guarantees the content type.

Ready to build? Start with Android, Ktor, or Spring Boot.