Kotlin Cheatsheet

Basic Structure and Variables

Kotlin is a modern, concise, and safe programming language, fully interoperable with Java.

- Declaration:
  - `val`: Read-only (constant).
  - `var`: Mutable variable.
- Null Safety:
  - `String`: Cannot be null.
  - `String?`: Can be null.
- Type Inference: `val s = "Hello"` (String).

Control Flow

- `when` (Powerful replacement for `switch`):

```
when (x) {
    1 → print("one")
    in 2..10 → print("range")
    is String → print("type check")
    else → print("others")
}
```

- `if` as an expression:

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

- Loops:
  - `for (item in list) { ... }`
  - `for (i in 1..10) { ... }` (Range)

Functions

- Standard: `fun add(a: Int, b: Int): Int { return a + b }`
- Single-expression: `fun add(a: Int, b: Int) = a + b`
- Default arguments: `fun greet(name: String = "Guest") { ... }`
- Extension Functions:

```
fun String.hello() = "Hello $this"
```

Classes and Objects

- Data Class: Automatically generates `equals`, `hashCode`, `toString`.

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

- Constructor:

```
class Person(val name: String) {
    init { println("Init block") }
}
```

- Inheritance: All classes are `final` by default. Use `open` to allow inheritance.
- Object (Singleton): `object Database { ... }`

Collections

- Immutable: `listOf()`, `mapOf()`, `setOf()`.
- Mutable: `mutableListOf()`, `mutableMapOf()`.
- Functional Processing: `list.filter { it > 0 }.map { it * 2 }`.

Null Handling Safe Calls

- `name?.length`: Returns length or null.
- `name?.length ?: 0`: Elvis operator (default value).
- `name!!`: Assert not null (throws exception if null).
- `name?.let { ... }`: Execute block if not null.

Coroutines (Concurrency)

```
launch {
    val data = async { fetchData() }
    val result = data.await()
    updateUI(result)
}
```

Interoperability with Java

Kotlin can call Java code and vice-versa seamlessly. Most Java libraries and frameworks work out of the box.

Google Style Guide

Naming Conventions

- Class/Object/Interface: `PascalCase`.
- Function/Property/Local Variable: `camelCase`.
- Constants (val with no custom getter): `UPPER_SNAKE_CASE`.
- File naming: `PascalCase.kt` (usually matches the main class).

Formatting

- Indentation: 4 spaces.
- Line Length: Max 100 characters.
- Braces: Same-line opening brace `{`.
- Parameter List: Use trailing commas in multi-line parameter lists (Kotlin 1.4+).

Programming Practices

- Use `val` by default.
- Prefer `when` over `if` for multiple branches.
- Use `data class` for simple data holders.
- Minimize the use of the `!!` operator. Prefer safe calls or Elvis operator.
- Avoid unnecessary `return` in single-expression functions.

Documentation (KDoc)

- Use `/** ... */` for documentation.
- Use `@param`, `@return`, `@throws` for tags.
- The first paragraph should be a brief summary.