CMPUT 301

Lab 6: Code Documentation and Unit Testing

University of Alberta

Michelle Deng 2026

Table of Contents

Code Documentation

KDoc

- Language used to document Kotlin code
- Without KDoc, your teammates (and future you) have no idea what functions do

KDoc

```
/**
 * A group of *members*.
 *
 * This class has no useful logic; it's just a documentation example.
 *
 * @param T the type of a member in this group.
 * @property name the name of this group.
 * @constructor Creates an empty group.
 */
class Group<T>(val name: String) {
    /**
     * Adds a [member] to this group.
     * @return the new size of the group.
     */
    fun add(member: T): Int { ... }
}
```

KDoc Syntax

- KDoc comments start with /** and end with */
- Can be applied to elements like classes and functions
- Convention:
  - Start with summary
  - Followed by detailed description
  - Then block tags: @

KDoc Syntax

```
/**
 * A group of *members*.
 *
 * This class has no useful logic; it's just a documentation example.
 *
 * @param T the type of a member in this group.
 * @property name the name of this group.
 * @constructor Creates an empty group.
 */
class Group<T>(val name: String) {
    /**
     * Adds a [member] to this group.
     * @return the new size of the group.
     */
    fun add(member: T): Int { ... }
}
```

<-- Summary

<-- Detailed description

<-- Block tags

KDoc Syntax: Block Tags

- There are many block tags we can use to describe an element:
  - @param *name*
  - @return
  - @property *name*
  - @constructor
  - @throws *class*
  - @exception *class*
  - *...*

Dokka

- An API documentation engine for Kotlin
- It takes KDoc comments and can generate documentation in many formats like HTML

Unit Testing

What is Unit Testing?

- A unit test is a piece of code that performs a test on another piece of code in isolation
- Each unit test is independent from each other
- A unit test can be run automatically

Why Write Unit Tests?

- Makes sure code is working as expected
  - When the code is written
  - And after subsequent changes
- Helps when refactoring code
- Works as a documentation of tests

Guidelines for Writing Tests

- Test the expected path
- Test alternative paths
- Test error handling
- Test boundary conditions
  - Null arguments, negative numbers, etc.

JUnit

- Modern unit testing framework for Java and Kotlin
  - We will be using JUnit 4, as Android Studio provides the most support for this version

JUnit

```
import org.junit.Assert.assertEquals
import org.junit.Test

class CalculatorTest {
    1 Usage
    private val calculator = Calculator()

    @Test
    fun add_twoIntegers_sumOfIntegersReturned() {
        assertEquals( expected = 2,  actual = calculator.add( a = 1, b = 1))
    }
}
```

JUnit

- Each class being tested usually has a corresponding test class
- The test class usually contains multiple test methods
- Each test method checks one specific behaviour or functionality of the class

JUnit

```
import org.junit.Assert.assertEquals
import org.junit.Test

class CalculatorTest {
    1 Usage
    private val calculator = Calculator()

    @Test
    fun add_twoIntegers_sumOfIntegersReturned() {
        assertEquals( expected = 2,  actual = calculator.add( a = 1, b = 1))
    }
}
```

<-- Test class
to test Calculator class

<-- Test method checking that calculator's add() works as expected

JUnit Conventions

- Name test classes by adding “Test” to the end of the class name
  - For example, CalculatorTest
- Test methods are labelled with *annotations* like @Test
  - @Test denotes that a method is a test method

JUnit Conventions

- @Before
  - Useful for running setup code *before every test method*
  - Ensures each test starts with a clean, consistent state

```
@Before
fun setUp() {
    calculator = Calculator()
}
```

JUnit Conventions

- Assertions
  - Used to test conditions inside test methods
    - assertTrue(...)
    - assertFalse(...)
    - assertEquals(...)
    - assertSame(...)
    - assertNotSame(...)
    - assertNotNull(...)
    - assertNull(...)
    - fail()

More Information

- KDoc: https://kotlinlang.org/docs/kotlin-doc.html
- Introduction to Dokka: https://kotlinlang.org/docs/dokka-introduction.html
- About JUnit 4: https://junit.org/junit4/
- Build local unit tests in Android Studio: developer.android.com/training/testing/local-tests

- Some information in these slides have been adapted from previous CMPUT 301 lab 6 material
- Images reproduced in these slides have been included under section 29 of the Copyright Act, as fair dealing for research, private study, criticism, or review. Further distribution or uses may infringe copyright.