Software Design & Architecture

Kotlin & Android Toolchain

Pengyu Nie

Acknowledgements: referenced Jeff Avery's CS 346 Application Development.

Plan for the next few weeks

4	Jan 27 Mon	Kotlin, Android Toolchain	implementation
	Jan 29 Wed	Arch Styles 1 - MVVM	architrecture
	Jan 31 Fri	**P2** Project Proposal [requirements]	
5	Feb 03 Mon	Building Android Apps (Guest Lecture)	implementation
	Feb 05 Wed	Arch Styles 2 - Server/Client	architrecture
6	Feb 10 Mon	Arch Styles 3	architrecture
	Feb 12 Wed	**P3** Iteration 1 Demo [requirements]	
7		*Reading Week*	
8	Feb 24 Mon	Design Patterns 1 - Creational	design
	Feb 26 Wed	Design Patterns 2 - Structural	design
9	Mar 03 Mon	Design Patterns 3 - Behavioral	design
	Mar 05 Wed	**P4** Iteration 2 Demo [requirements]	
10	Mar 10 Mon	Design Patterns 4	design
	Mar 12 Wed	Testing	quality assurance
11	Mar 17 Mon	Continuous Integration, Release	quality assurance
	Mar 19 Wed	**P5** Iteration 3 Demo [requirements]	

Android-related technologies

Agenda

- Programming language: Kotlin
- IDE: Android Studio
- Build system: Gradle
- UI framework: Jetpack Compose

Kotlin

- Modern, general-purpose language
- Designed by JetBrains as a drop-in replacement for Java
  - compatible with Java source code and ecosystem
- Google’s recommended language for Android development
  - if not the only possible language as of now…

Kotlin – Grammar Sugars Overview

```kotlin
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Greeting(name = "Android")
        }
    }
}

@Composable
fun Greeting(name: String) {
    Text("Hello $name!")
}
```

Nullability in the type system helps prevent NullPointExceptions

Semicolons are optional

Lambdas allow you to pass code to a function as a parameter

Named parameters make code easier to read

String templates simplify concatenation

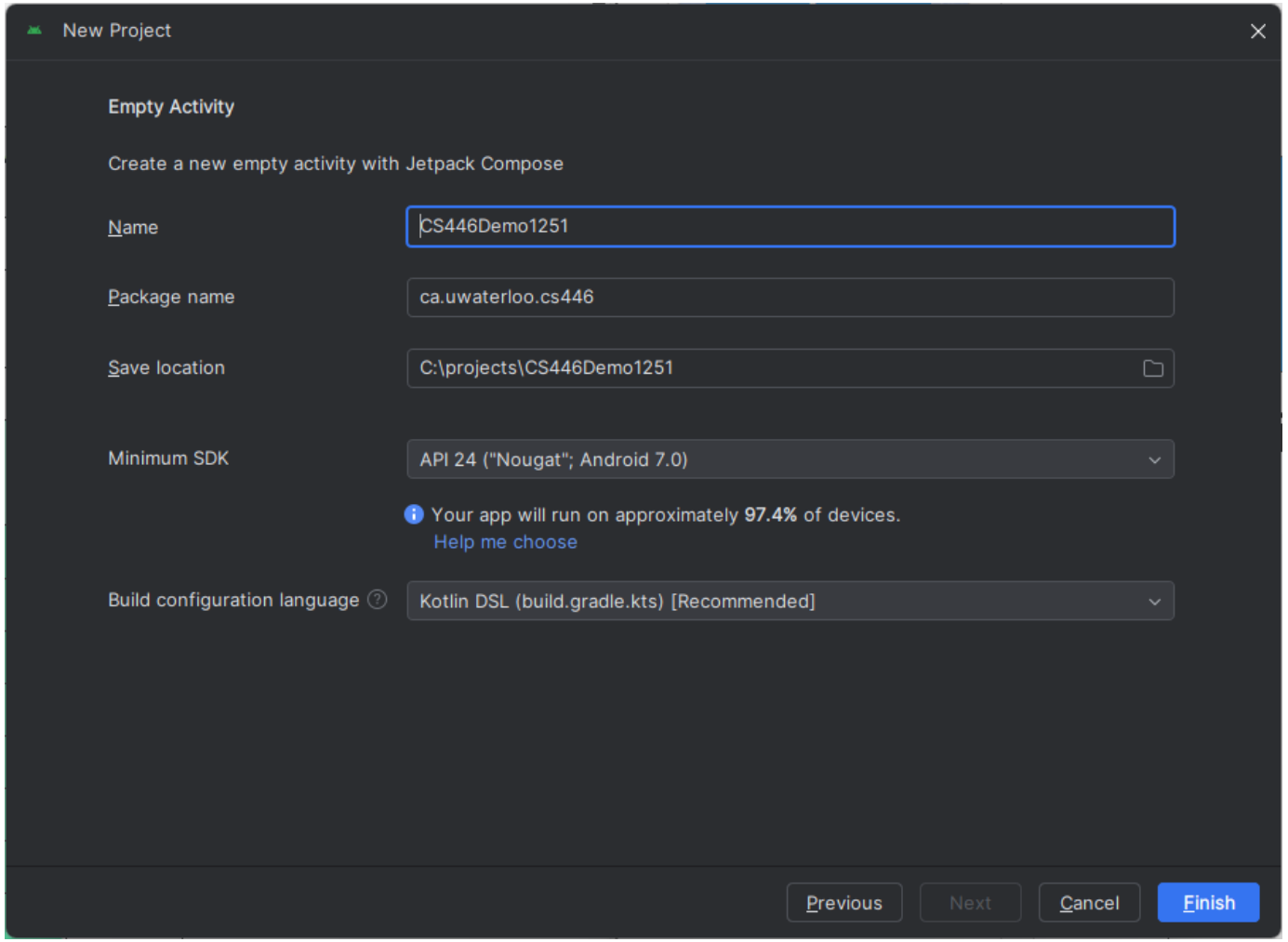
Some other key differences from Java:

- `fun` keyword to start function (method) declaration;
- `val` keyword to start variable declaration;
- `name: type` syntax for type annotations;
- `type` cannot hold a null value; `type?` can
- `when` instead of `switch`;

Kotlin – Resources

- Official documentation: https://kotlinlang.org/docs/home.html
- Introduction and samples of Kotlin for Android: https://developer.android.com/kotlin
- Kotlin programming section in CS346: https://student.cs.uwaterloo.ca/~cs346/1251/course-notes/development/kotlin/index.html

Android Studio – New Project



avoid using underscore "_" in package name

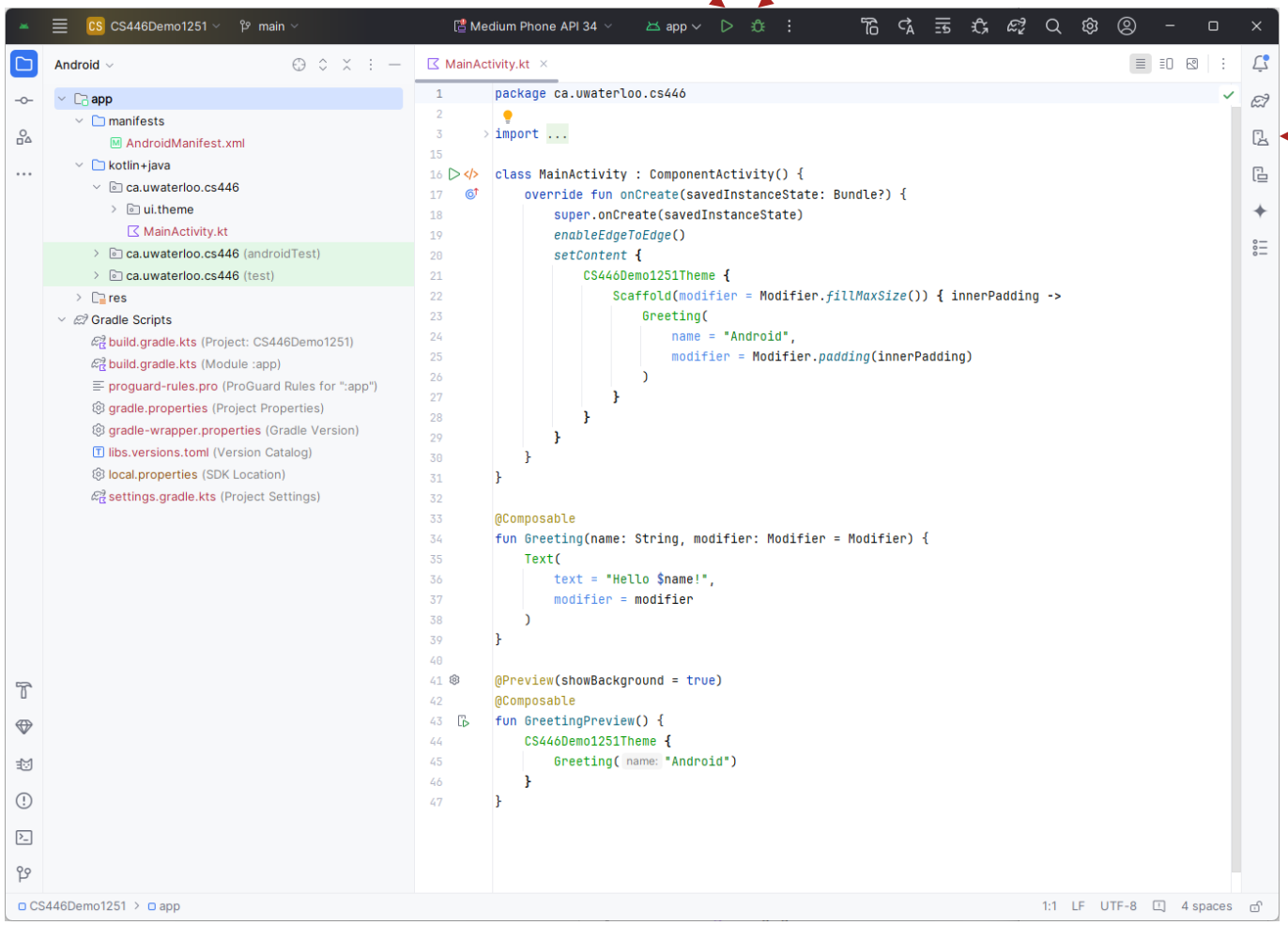
minimum sdk, setting it lower can:

- let your app run on older devices
- complicate the usage of newer APIs

21 is the absolute minimum for supporting Jetpack Compose

Android Studio – First Glance

run *debug*

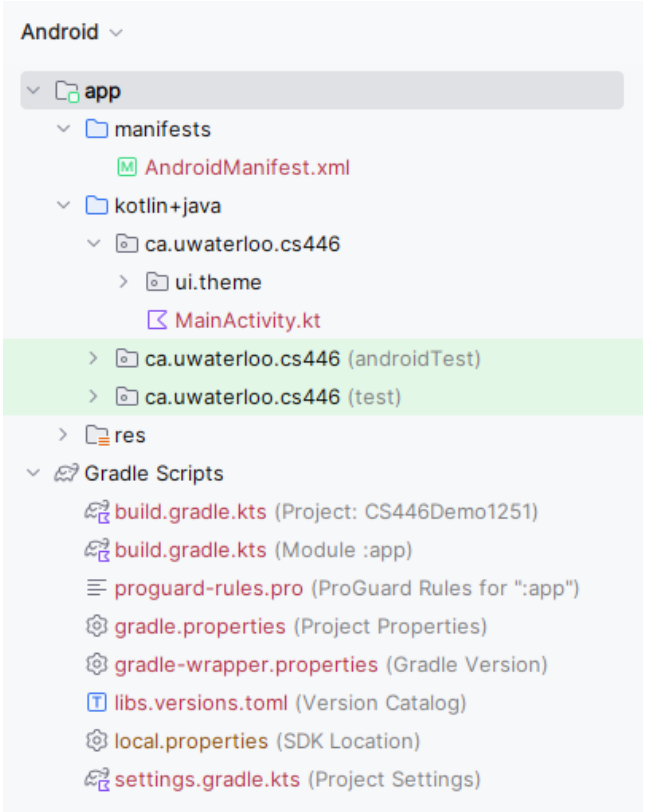


device manager

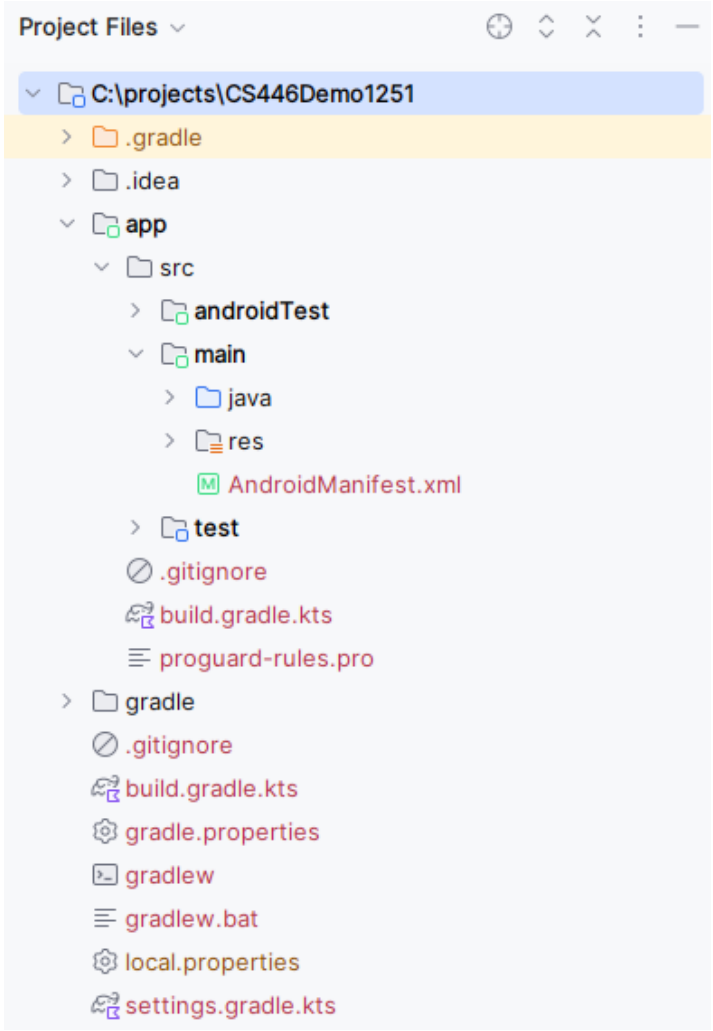
install an Android emulator here
(or connect your Android phone as the debugging device)

Android Studio – Project Files

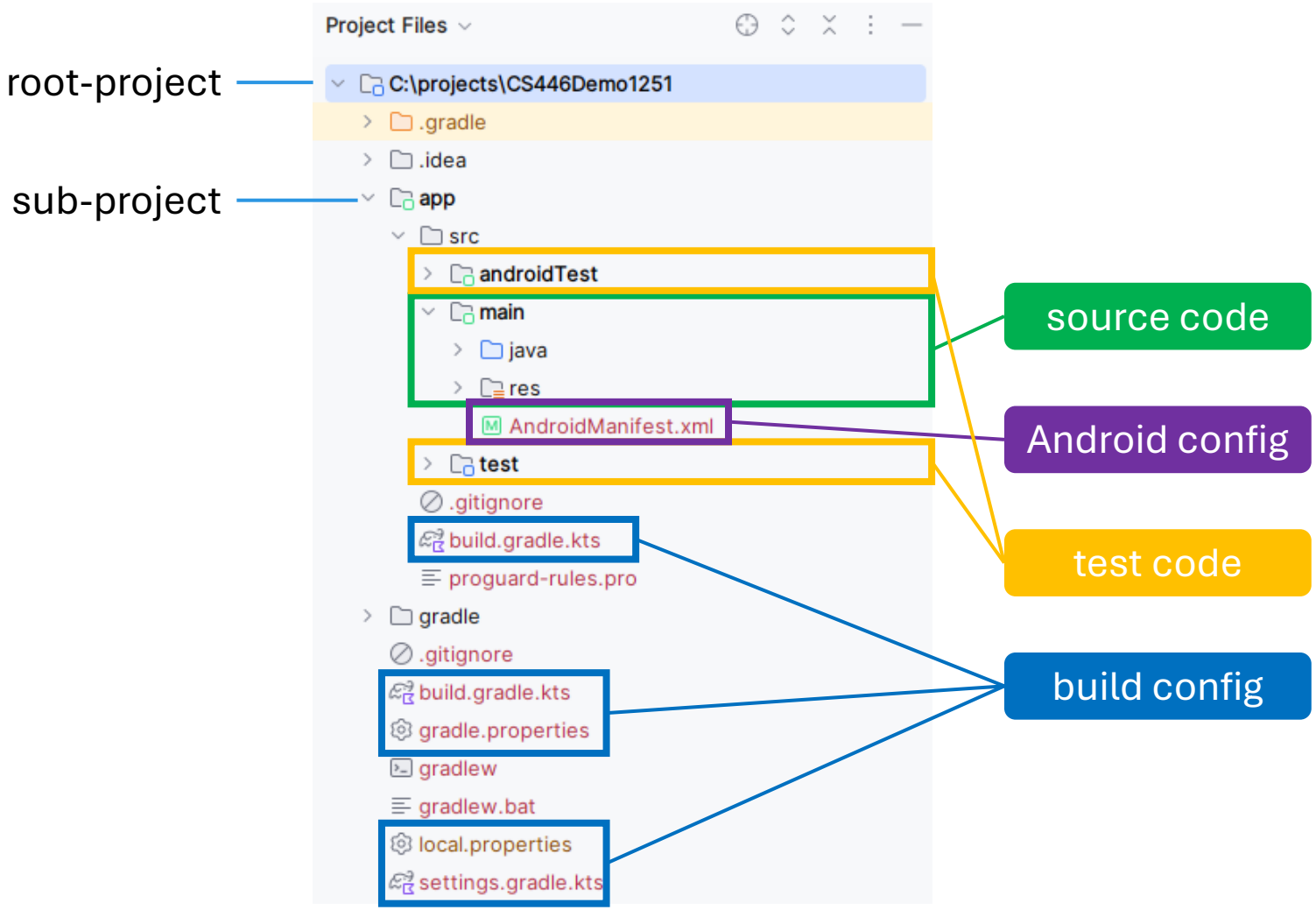
The “logical” view of project files



The “physical” view of project files

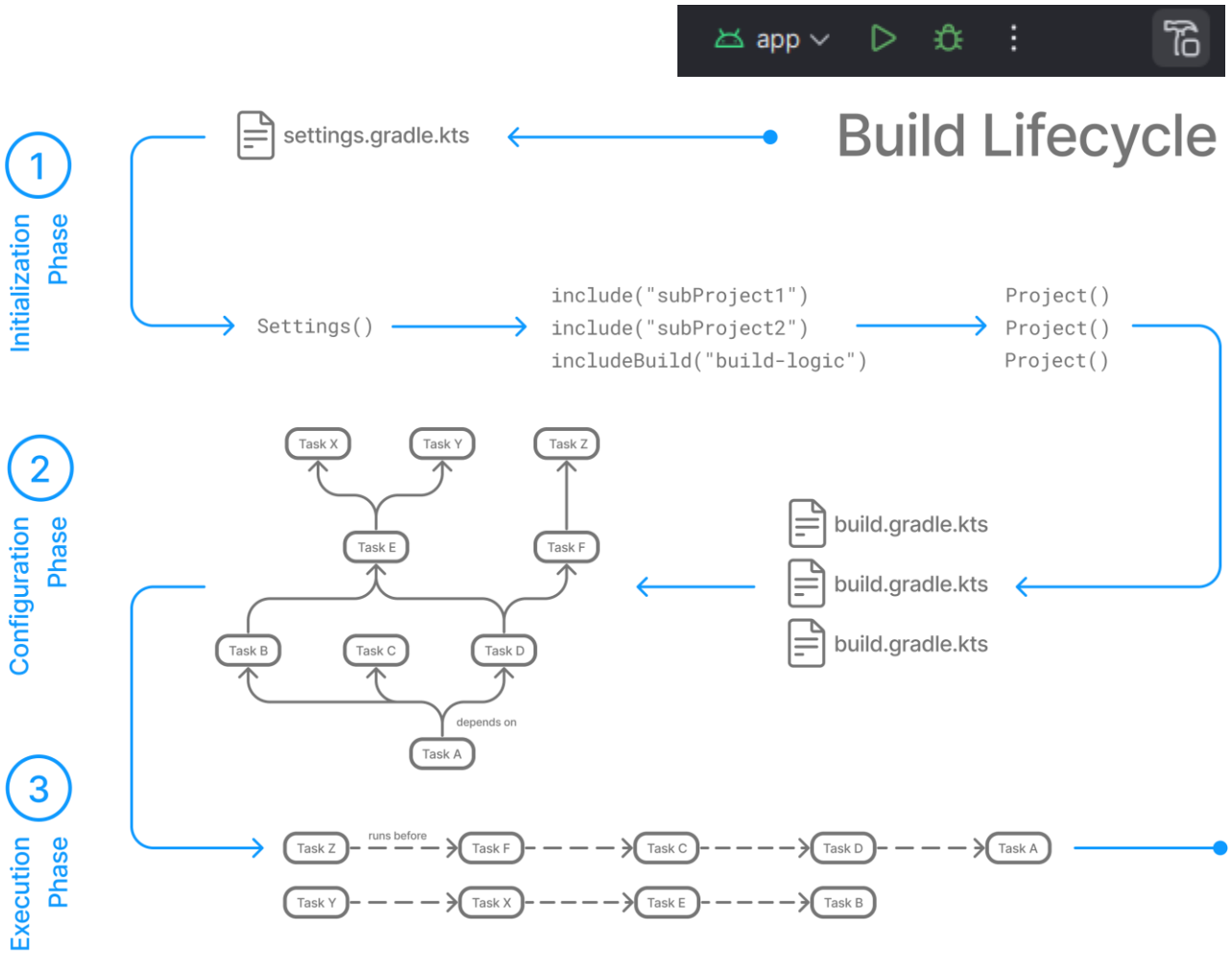


Android Studio – Project Files (cont.)



More about AndroidManifest.xml config file: https://developer.android.com/guide/topics/manifest/manifest-intro

Gradle Build System



https://docs.gradle.org/current/userguide/build_lifecycle.html

Gradle – Important Notes

- Source code / test code: assume a directory structure
  - source code: src/main/java/...packages.../*.kt
  - test code: src/{test,androidTest}/java/...packages.../*.kt

- Build config file

dependencies

declare 3rd party libraries used

```
dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation(libs.androidx.ui)
    implementation(libs.androidx.ui.graphics)
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
}
```

plugins

pre- and post-processing

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose)
}
```

other configs

```
namespace = "ca.uwaterloo.cs446"
compileSdk = 35

defaultConfig {
    applicationId = "ca.uwaterloo.cs446"
    minSdk = 24
    targetSdk = 35
    versionCode = 1
    versionName = "1.0"

    testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
}
```

target sdk:

- the highest version your app is designed to run on
- should equal to `compileSdk`

Use 34 or higher if you plan to publish to GooglePlay!

More on API levels: https://apilevels.com/

Gradle – Resources

- Official documentation: https://docs.gradle.org/current/userguide/userguide.html
- Gradle for Android notes: https://cookbook.gradle.org/android/
- Android Gradle plugin documentation: https://developer.android.com/build

Jetpack Compose

- Android's recommended modern toolkit for building native UI
- Key features:
  - intuitive domain specific language (in Kotlin) for describing UI
  - live previews to speed up debugging iterations
  - modern UI components
- See the demo… And more resources:
  - Tutorial: https://developer.android.com/develop/ui/compose/tutorial
  - Official documentation: https://developer.android.com/develop/ui/compose/documentation
  - Android app samples: https://developer.android.com/develop/ui/compose/samples
  - CS346 notes on mobile app: https://student.cs.uwaterloo.ca/~cs346/1251/course-notes/development/application-styles/mobile.html

Agenda (recap)

- Programming language: Kotlin
- IDE: Android Studio
- Build system: Gradle
- UI framework: Jetpack Compose

- Review the linked resources
- Prepare questions to ask in next week's guest lecture
- Let me know if you want to see demos/analysis on specific topics in class