**CUSTOM LINTING RULES IN KOTLIN (ANDROID STUDIO)**

**What is Lint ?**

Lint in Android is a very powerful tool. The Android lint is a static code analysis tool that checks your project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.

The Android Lint API allows users to create custom lint checks. For example, if you are the author of an Android library project and your library project has certain usage requirements, you can write

additional lint rules to check that your library is used correctly, and then you can distribute those extra lint rules for users of the library. Similarly, you may have company-local rules you’d like to enforce.

**Note:** That while Android Lint has the name “Android” in it, it is no longer an Android-specific static analysis tool; it’s a general static analysis tool, and inside Google for example it is run to analyse server-side Java and Kotlin code.

**Why Should we Write Custom Lint ?**

(In my current experience, this is the main reason I see for writing custom lint rules is)  
You have a large team and you have implemented certain rules for writing code within your company (for more readable and easy-to-understand code for everyone), but you don’t want to waste time on code reviews every time, and even if you look, you don’t want to waste time on code reviews to see if these rules are followed by developers. there will be moments, at this point writing custom lint comes into play and has a positive effect on both your time and your code.  
  
**Lint Example (Sample Demo):**



**Let’s Write a Custom Lint for the Real Example**

**Sample-1 (Warning)**

**Step-1:** Create a new Android (Kotlin/Java) Project in Android Studio.

A screenshot of a computer

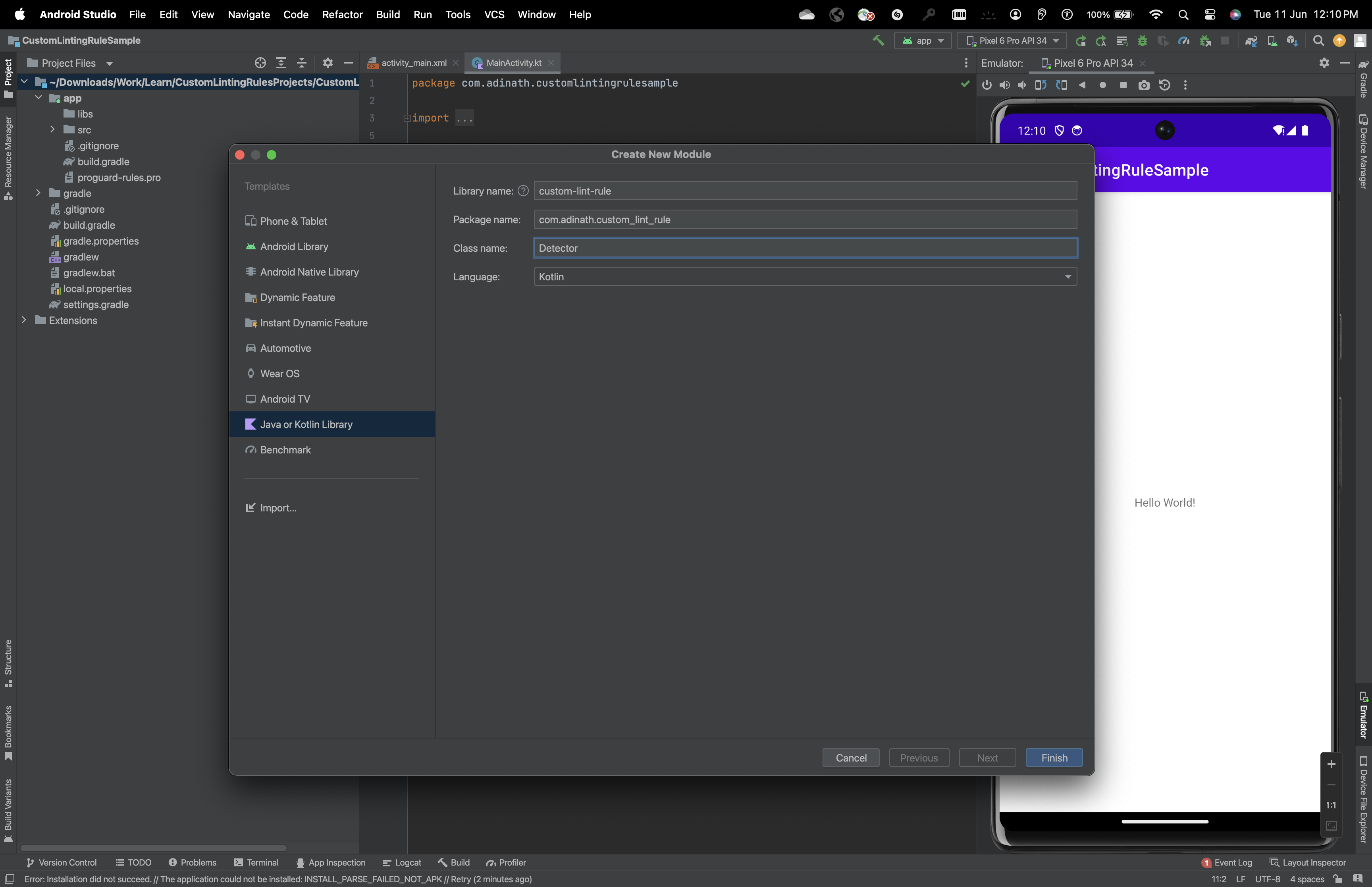
Description automatically generated

**Step-2:** Create a Custom Lint Module in Same Project.

To do that Switch the Workspace view to Project Files -> Right Click on Project Root Folder -> New -> Module -> **Select Java or Kotlin Library (Important).**

**Next** pass the library Name, Class name -> **Finish**.

You’ll see the new module appear just below the app module in Workspace of Android Studio as shown below **(Expand the src folder).**





**Step-3:** Next, Add the Following lint Dependencies in just created lint module,. -> Open Build.gradle

var lintVersion = "30.4.1" // This Lint Version should be +23 to the Current AGP

compileOnly("com.android.tools.lint:lint-api:${lintVersion}")

compileOnly("com.android.tools.lint:lint-checks:${lintVersion}")

testImplementation("com.android.tools.lint:lint:${lintVersion}")

testImplementation("com.android.tools.lint:lint-tests:${lintVersion}")

testImplementation("junit:junit:4.13.2")

Final **build.gradle (custom-lint-rule)** will look something like below, next -> **Sync Now**.  
  
A screenshot of a computer program

Description automatically generated

**Step-4:** Next we have to write a code for the Detector Class which will detect the Linting Issues in our Source code when we write.  
  
Navigate to Detector.class. I’ll rename this Detector.class to LogDetector.class for my convenience. And I’ll write the following code which will detect the Log functions in our source code and will raise a lint issue as warning to avoid the usage of Log and related functions in the source code.

A screenshot of a computer

Description automatically generated

**Step-5:** Next, we have to Register this Issue to the Lint Registry, for that we have to create a Separate Issue Registry Class in same Lint Module which will inherit the IssueRegistry Dependencies.

I Created the LogIssueRegistry Class which will take the LogDetector class as an issue to the overridden variables.

(Note: We can pass multiple Issue Detectors for the single Registry Class)

A screenshot of a computer

Description automatically generated

**Step-6:**  Next we have to pass this Issue Registry to the background linting services as an reference to let the linting service know what to raise the lint issue, for that we have to create some directories at module.

Directory Structure : Resources -> META-INF -> services -> (New File)

This new file should have a strict name as follows.  
  
**File Name: com.android.tools.lint.client.api.IssueRegistry**  
  
A screenshot of a computer

Description automatically generated

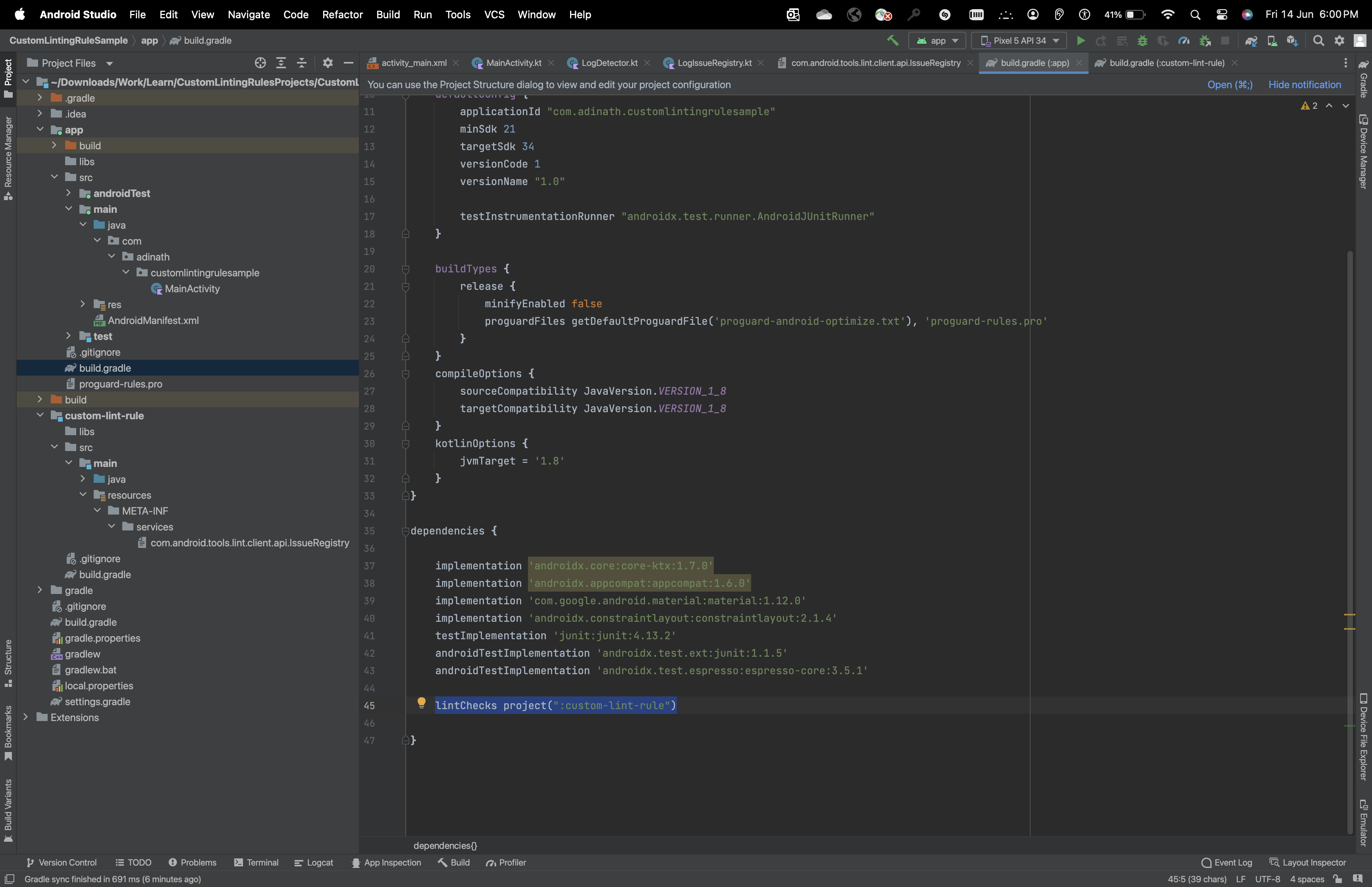
**Step-7:** Now put the following content in the just created file, this is just an reference for the background service to use the following IssueRegistry file.

A screenshot of a computer

Description automatically generated

**Step-8:** Next we have to provide the reference of this lint module to the other module(s) where we want to check the linting practices, for that open the app module build.gradle file and add the following line to the dependencies.

Syntax: lintChecks project(“:custom-lint-module-name”)

Example: lintChecks project(“:custom-lint-rule”)  
  


**Step-9:** Next we have to generate the Jar files out of the module we have created to do that we have to enter a simple command in the terminal which will do the expected.  
  
Open the Terminal :  
  
A screenshot of a computer

Description automatically generated

After this Linting Jar Files generation process will start, and once it is completed you’ll be seeing the build folder appear in the workspace in the custom lint module and that jar files will be used by specified modules.

**Step-10:** Let’s Test,

Open MainActivity.kt, and try to write a Log function  
  
Log.d(“key”, “Message”)  
  
You’ll be Seeing the yellow highlight all over the log function and if you hover over the log function you’ll see the custom message what you passed during the issue definition.

Output:

A screenshot of a computer

Description automatically generated