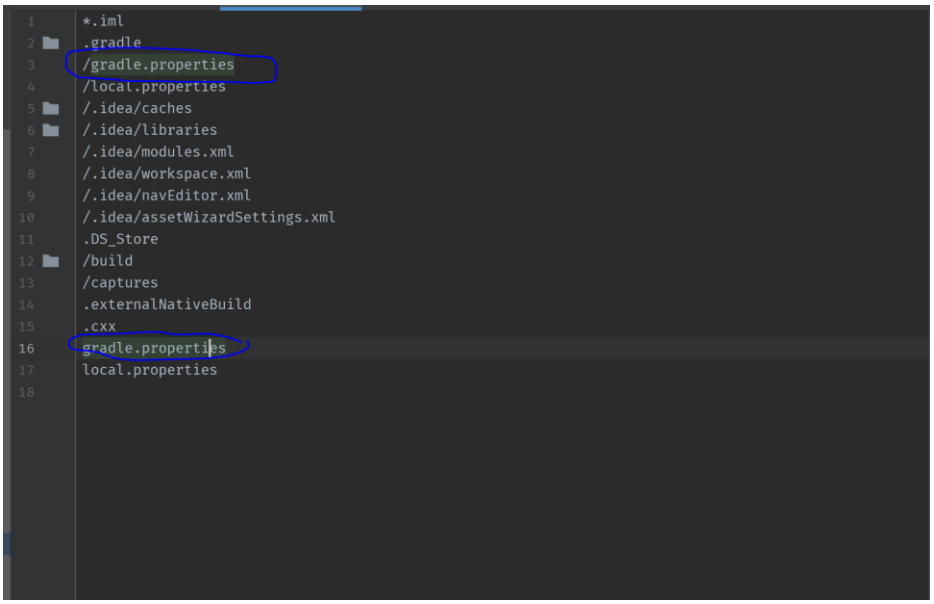
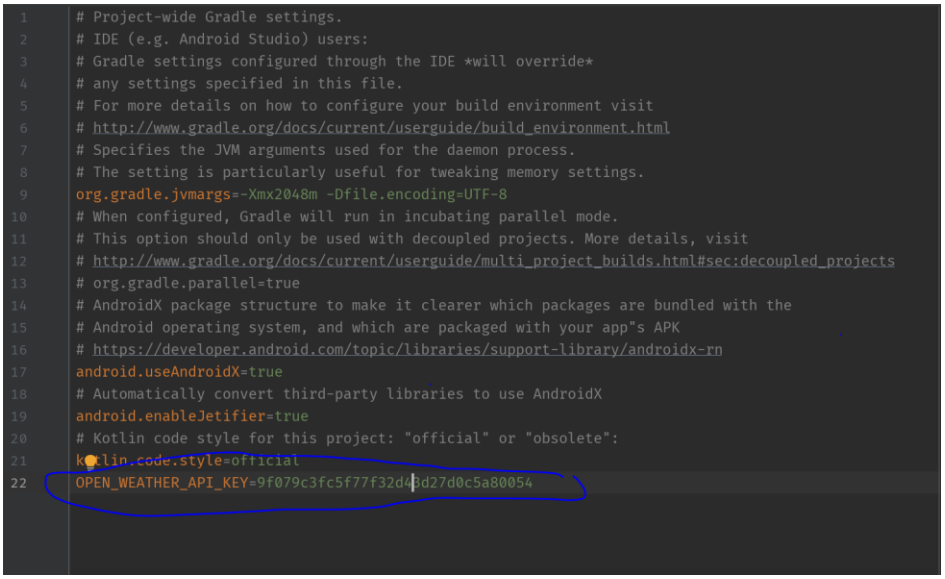
Start by doing this



```
*.iml
.gradle
/gradle.properties
/local.properties
/.idea/caches
/.idea/libraries
/.idea/modules.xml
/.idea/workspace.xml
/.idea/navEditor.xml
/.idea/assetWizardSettings.xml
.DS_Store
/build
/captures
.externalNativeBuild
.cxx
gradle.properties
local.properties
```

Go to root gradle.properties file and make a key-name and set the value the api key



```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html
# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. More details, visit
# http://www.gradle.org/docs/current/userguide/multi_project_builds.html#sec:decoupled_projects
# org.gradle.parallel=true
# AndroidX package structure to make it clearer which packages are bundled with the
# Android operating system, and which are packaged with your app"s APK
# https://developer.android.com/topic/libraries/support-library/androidx-rn
android.useAndroidX=true
# Automatically convert third-party libraries to use AndroidX
android.enableJetifier=true
# Kotlin code style for this project: "official" or "obsolete":
kotlin.code.style=official
OPEN_WEATHER_API_KEY=9f079c3fc5f77f32d43d27d0c5a80054
```

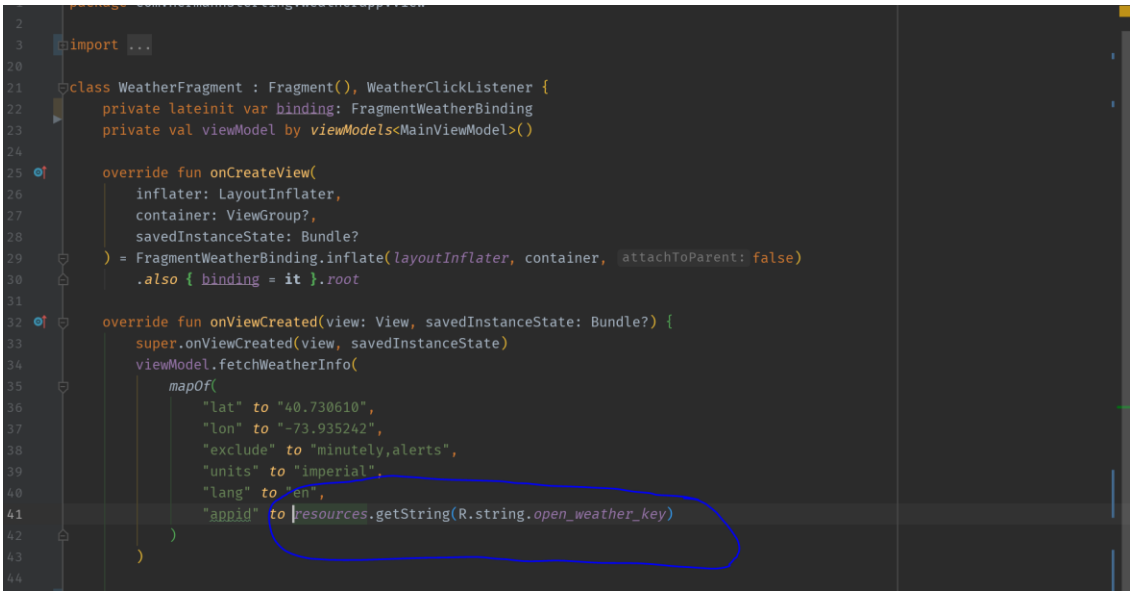
Go to your app build gradle and then add a resValue value, for the the resValue you need to indicate the type of resource you want to create , which is a string in this case, and then give it a user defined name, in this case I use “open_weather_key”. Then you would need to use project.findProperty(“name of key-name you use in gradle.properties”)

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-kapt'
    id 'kotlin-parcelize'
    id 'androidx.navigation.safeargs.kotlin'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.hermannsterling.weatherapp"
        minSdkVersion 21
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"
        resValue "string", "open_weather_key",
                (project.findProperty("OPEN_WEATHER_API_KEY") ?: "")
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
```

Go to your AndroidManifest and then add a meta-data tag within your application scope, give it a name using the android:name attribute. I used Default, and then add the string resources you created in the previous step by using the android:value

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.hermannsterling.weatherapp">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="WeatherApp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:networkSecurityConfig="@xml/network_security_config"
        android:theme="@style/Theme.MyApp">
        <meta-data
            android:name="Default"
            android:value="@string/open_weather_key" />
        <activity android:name=".view.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Then this is how you reference the key in the activity



```
import ...

class WeatherFragment : Fragment(), WeatherClickListener {
    private lateinit var binding: FragmentWeatherBinding
    private val viewModel by viewModels<MainViewModel>()

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ) = FragmentWeatherBinding.inflate(layoutInflater, container, attachToParent: false)
        .also { binding = it }.root

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        viewModel.fetchWeatherInfo(
            mapOf(
                "lat" to "40.730610",
                "lon" to "-73.935242",
                "exclude" to "minutely,alerts",
                "units" to "imperial",
                "lang" to "en",
                "appid" to resources.getString(R.string.open_weather_key)
            )
        )
```