//Ques1: Add a gradle dependency and its related repository url.

plugins **{**

id **'java'**

**}**

group **'org.example'**

version **'1.0-SNAPSHOT'**

sourceCompatibility = 1.8

repositories **{**

mavenCentral()

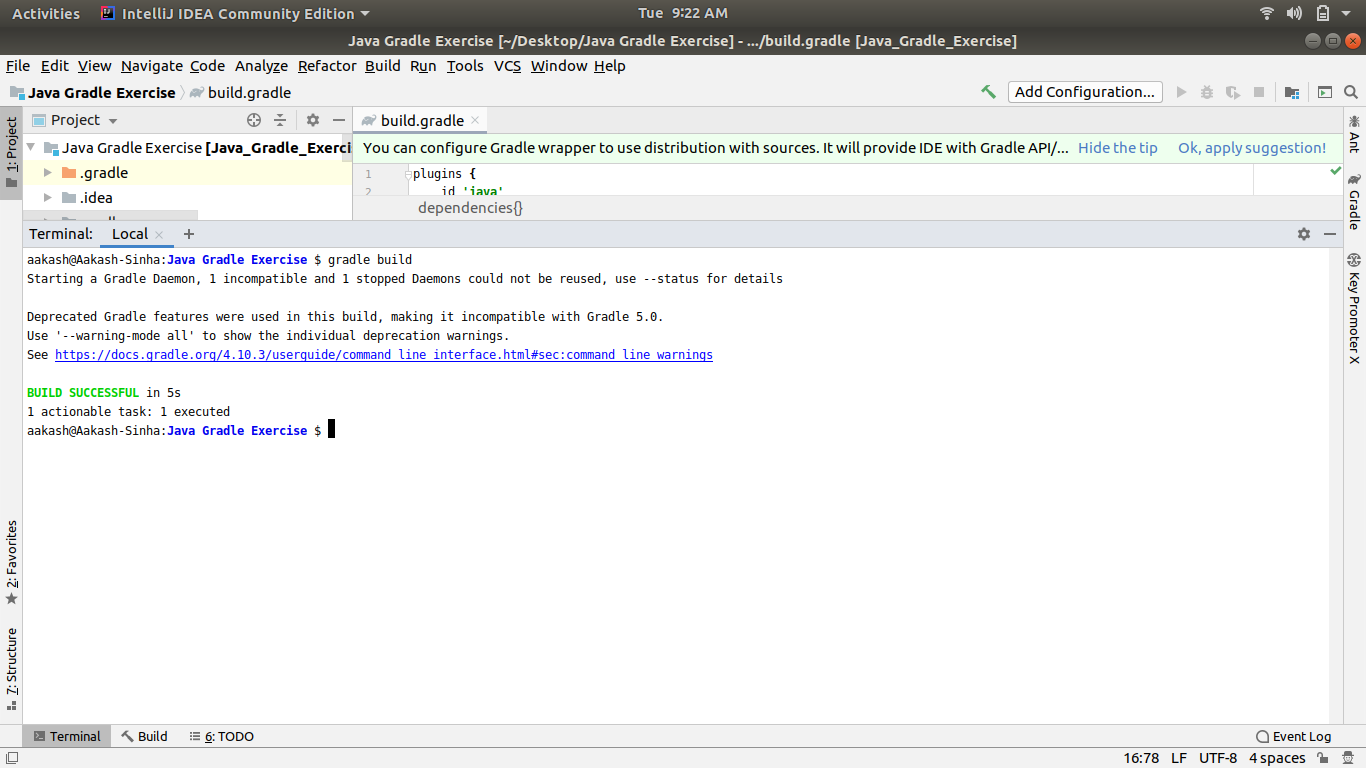
**}**

dependencies **{**

testCompile **group**: **'junit'**, **name**: **'junit'**, **version**: **'4.12'**

compileOnly **group**: **'com.google.code.gson'**, **name**: **'gson'**, **version**: **'2.8.6'**

**}**



//Ques2: Using java plugin, make changes in the manifest to make the jar executable. Using java -jar JAR\_NAME, the output should be printed as "Hello World"

plugins **{**

id **'java'**

**}**

group **'org.example'**

version **'1.0-SNAPSHOT'**

sourceCompatibility = 1.8

sourceSets**{**

main**{**

java **{**

srcDirs = [**'src/main/java'**]

**}**

**}**

**}**

jar**{**

manifest**{**

attributes(

**"Main-Class"**: **'Ques2'**,

**"Class-Path"**: configurations.**compile**.collect **{** it.getName() **}**.join(**' '**),

)

**}**

**}**

repositories **{**

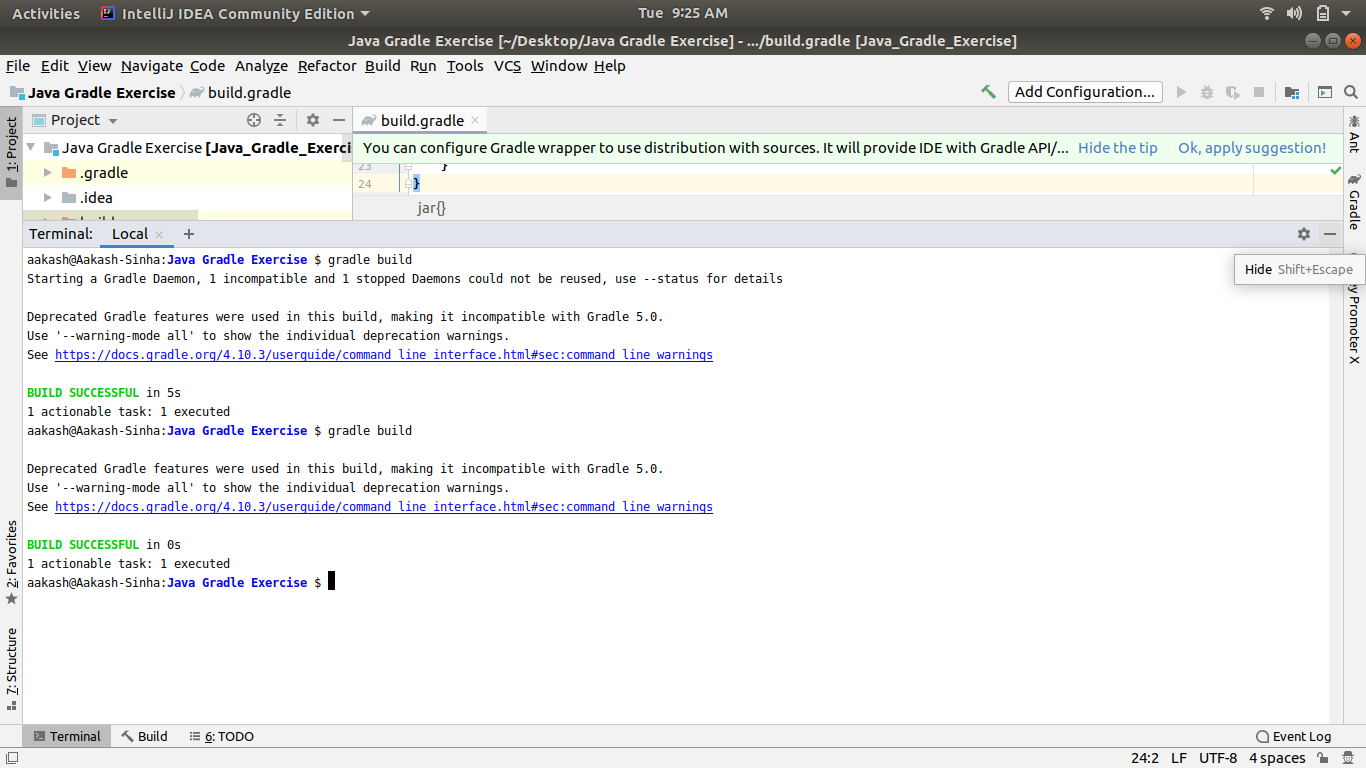
mavenCentral()

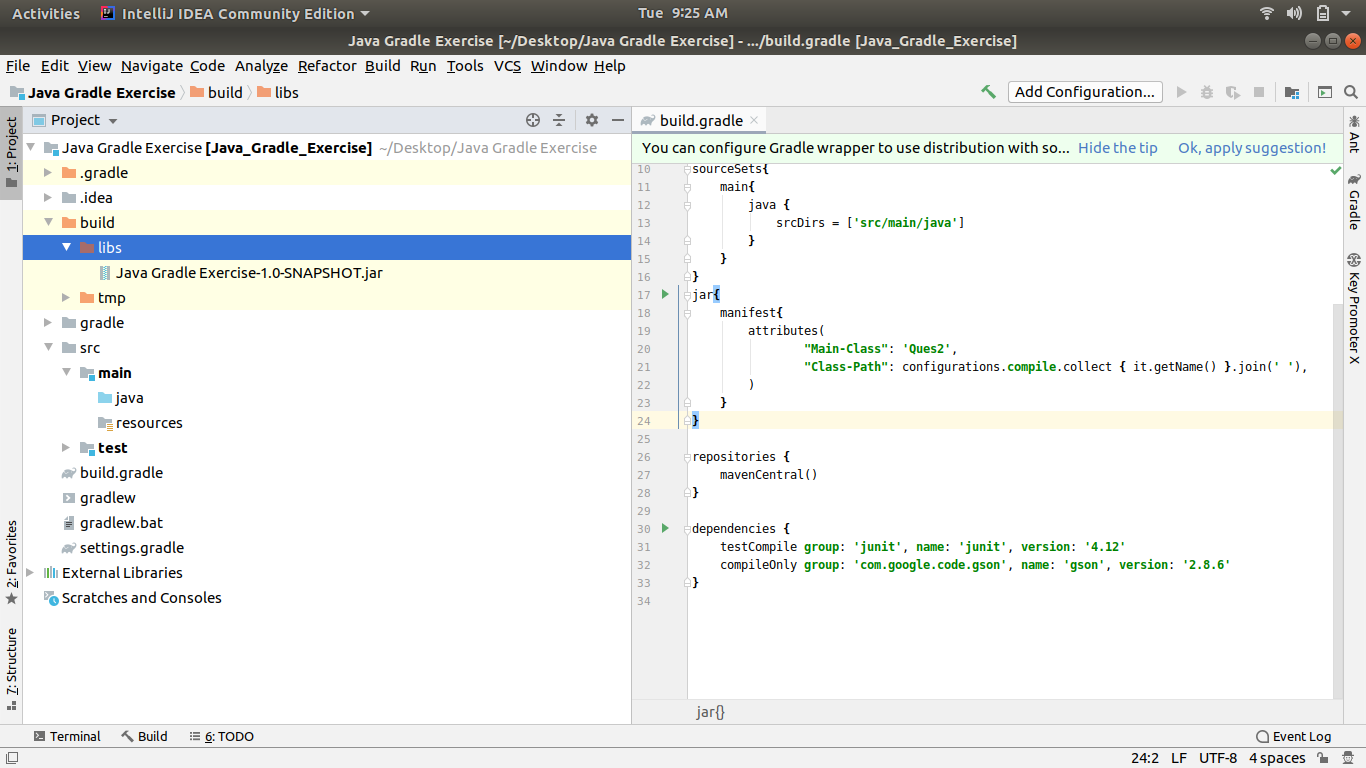
**}**

dependencies **{**

testCompile **group**: **'junit'**, **name**: **'junit'**, **version**: **'4.12'**

compileOnly **group**: **'com.google.code.gson'**, **name**: **'gson'**, **version**: **'2.8.6'**

**}**



//Ques3: Differentiate between the different dependency scopes: compile, runtime, testCompile, //testRuntime using different dependencies being defined in your build.gradle.

**Compile:​** The compile scope is the default scope. We can use it when we

have no special requirements for declaring a certain dependency.

**Runtime:** ​ We use the runtime scope for dependencies that are not needed

at compile time, like when we’re compiling against an API and only need

the implementation of that API at runtime.

**testCompile:** ​ We can use the testCompile scope for dependencies that are

only needed in tests compile time and that should not be available in

production code.

**testRuntime:** ​ We use the runRuntime scope for dependencies that are not

needed at compile time, like when we’re compiling against an API and only

need the implementation of that API at runtime.

//Ques 4: Create a custom plugin which contains a custom task which prints the current //date-time. Using that plugin in your project, execute that task after the jar task executes.

task showDate{

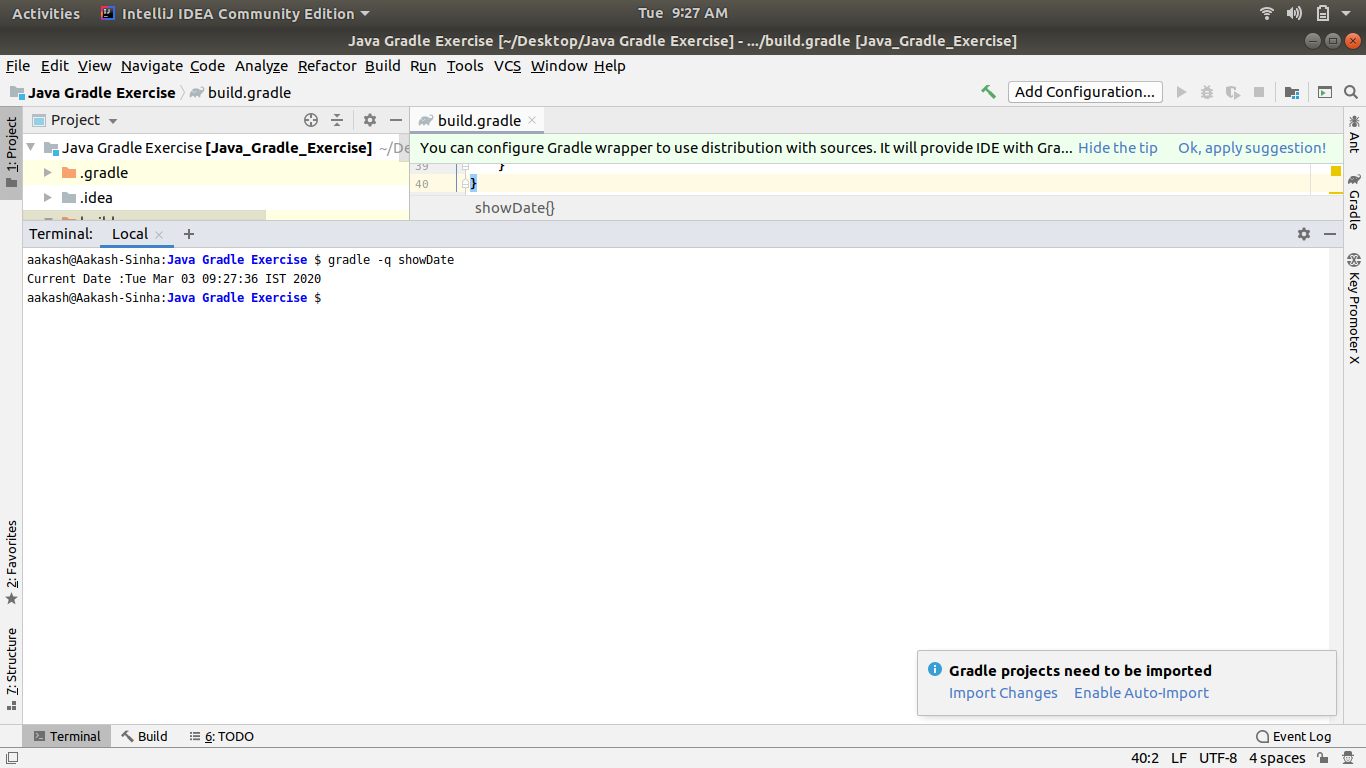
dependsOn(build)

doLast {

println 'Current Date :' + new Date();

}

}



//Ques5: Instead of using default source set, use src/main/javaCode1, src/main/javaCode2 to be //taken as code source. Make sure that the JAR created contains files from both the directories //and not from src/main/java.

sourceSets{

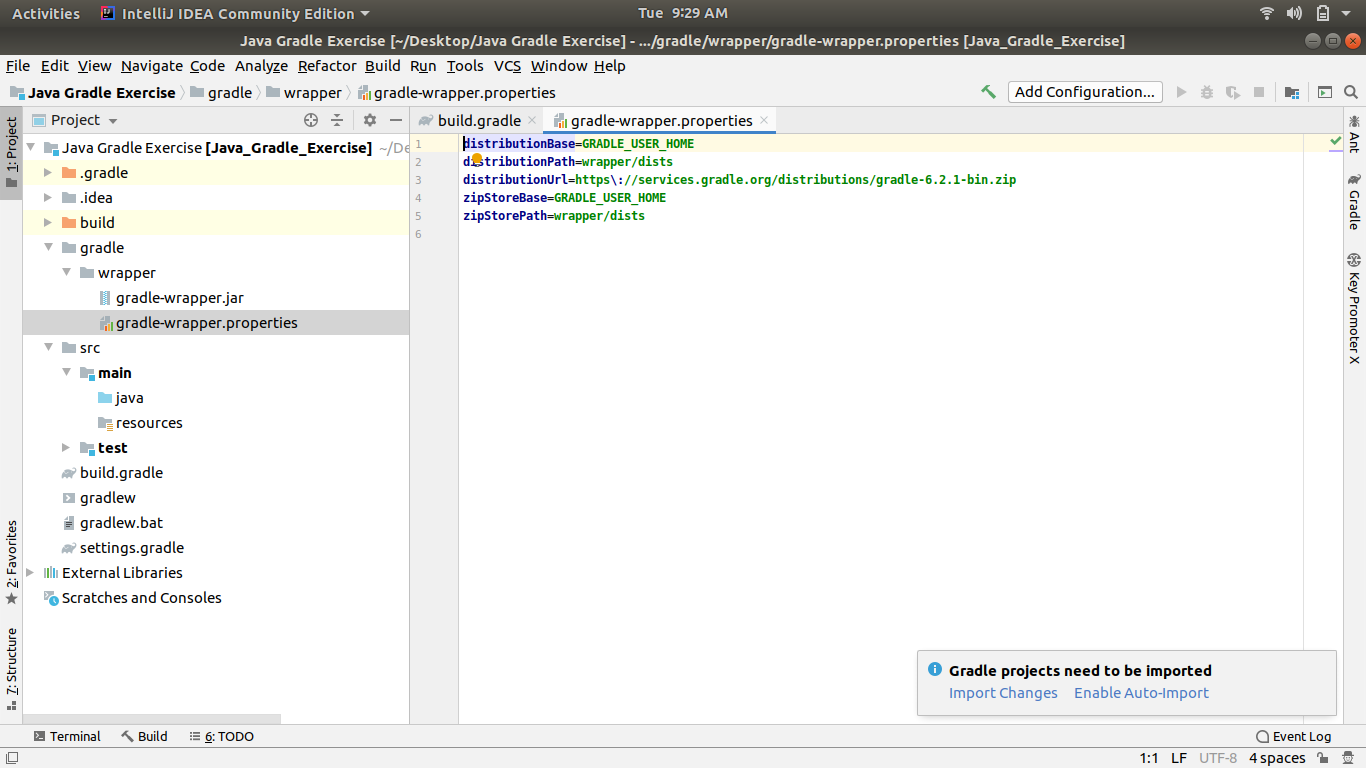
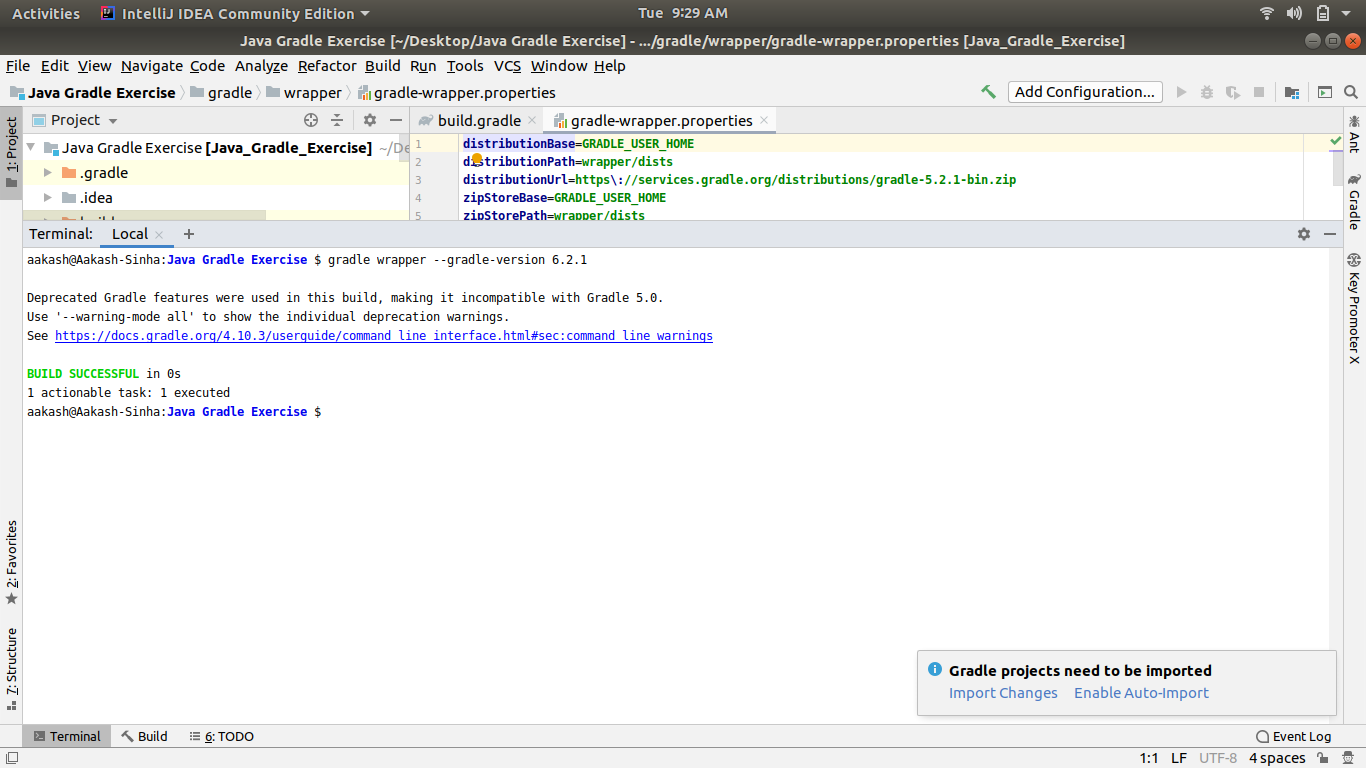
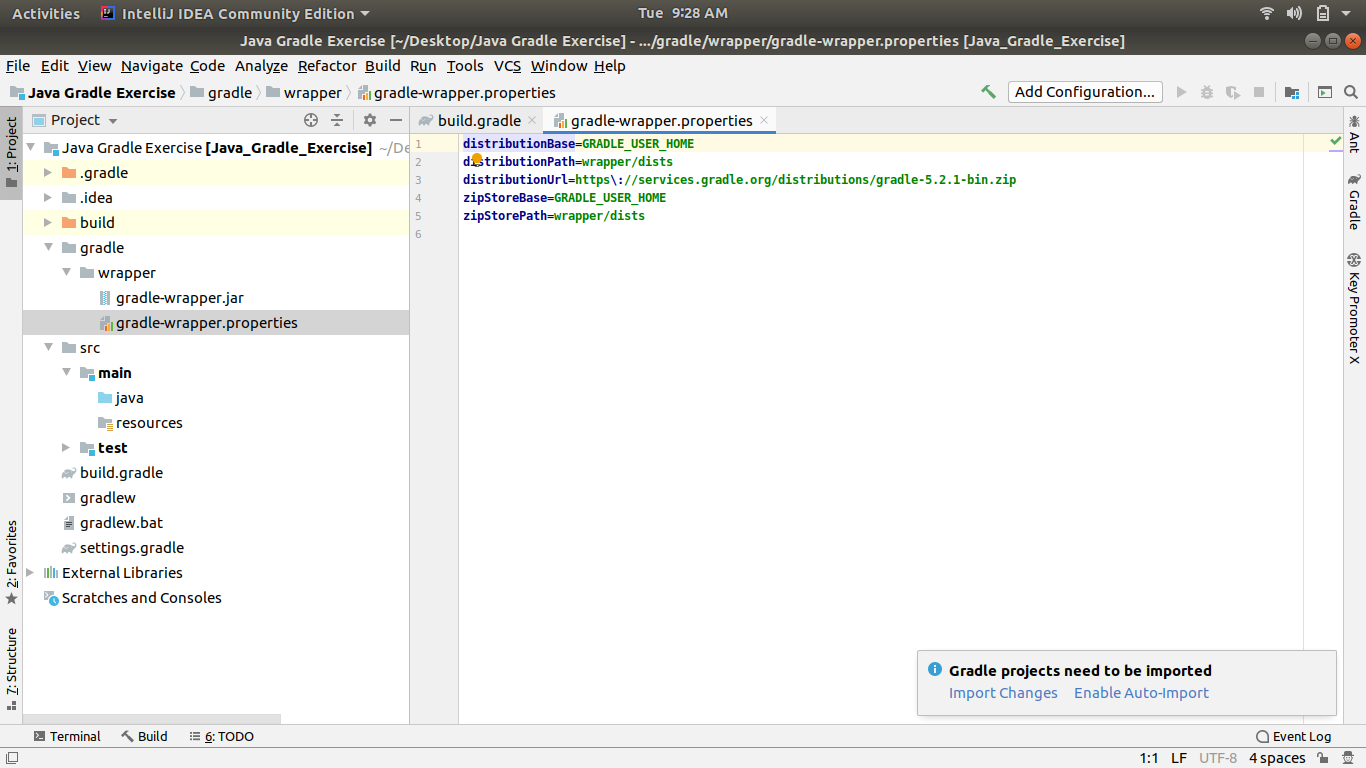
main

{

java.srcDirs= ['src/main/JavaCode1','src/main/JavaCode2']

}

}

//Ques6: Override the Gradle Wrapper task to install a different version of gradle. Make sure //that the task written in Q4 also executes with it.

//Ques7: Run the gradle profile command and attach the resulting files.