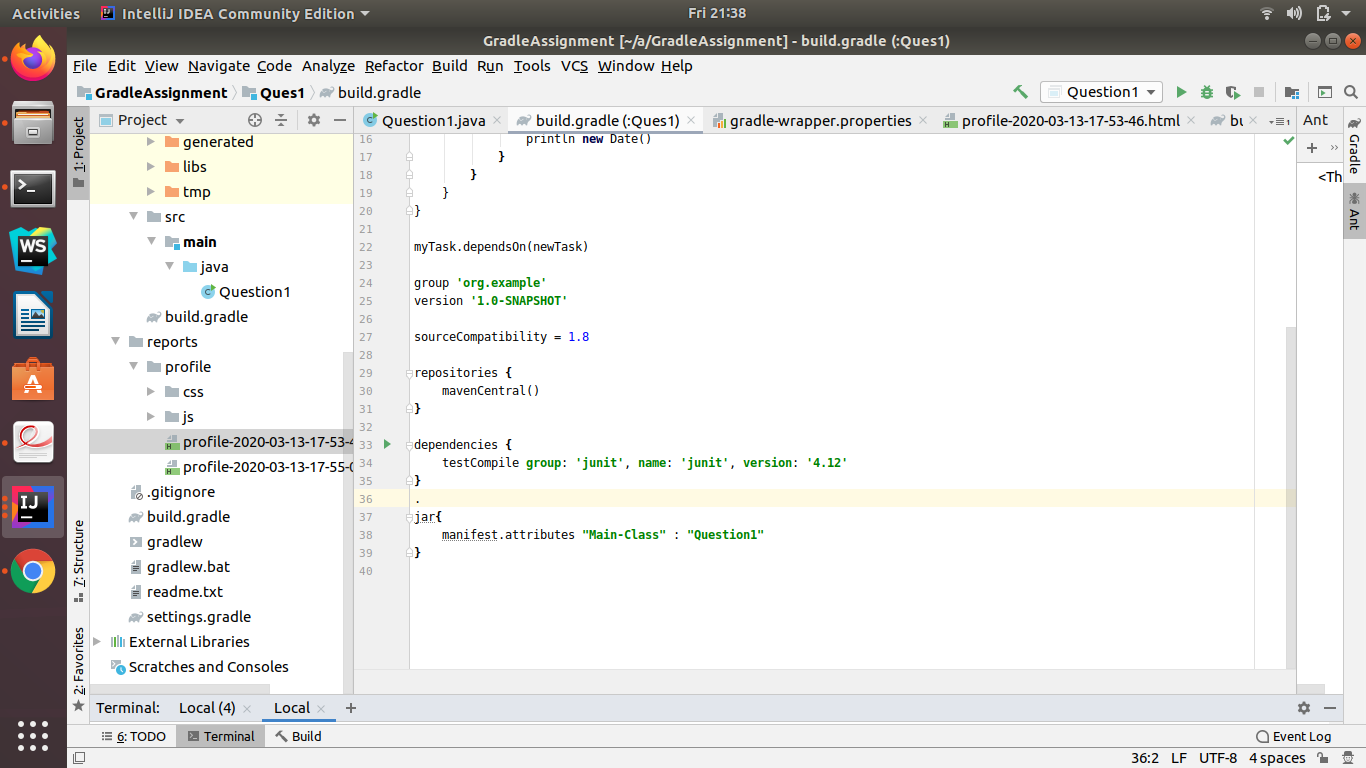
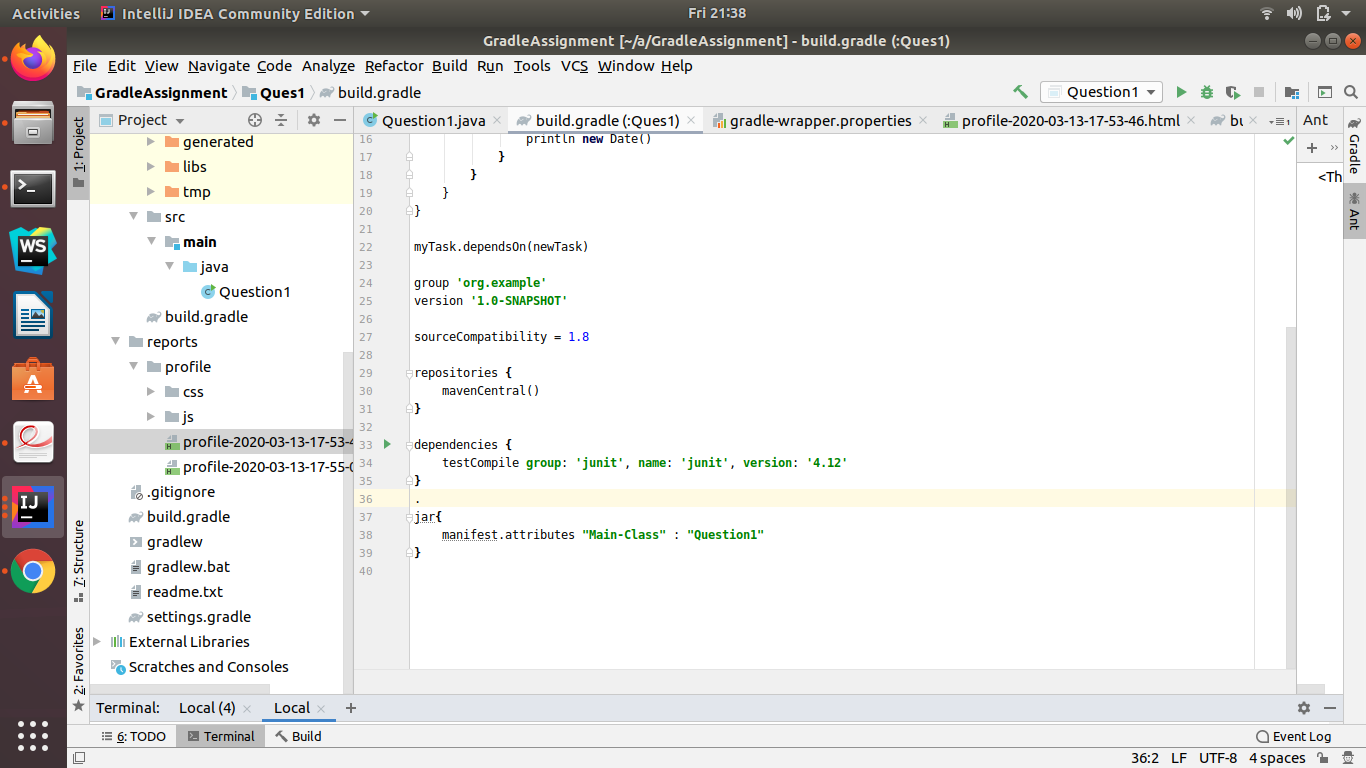
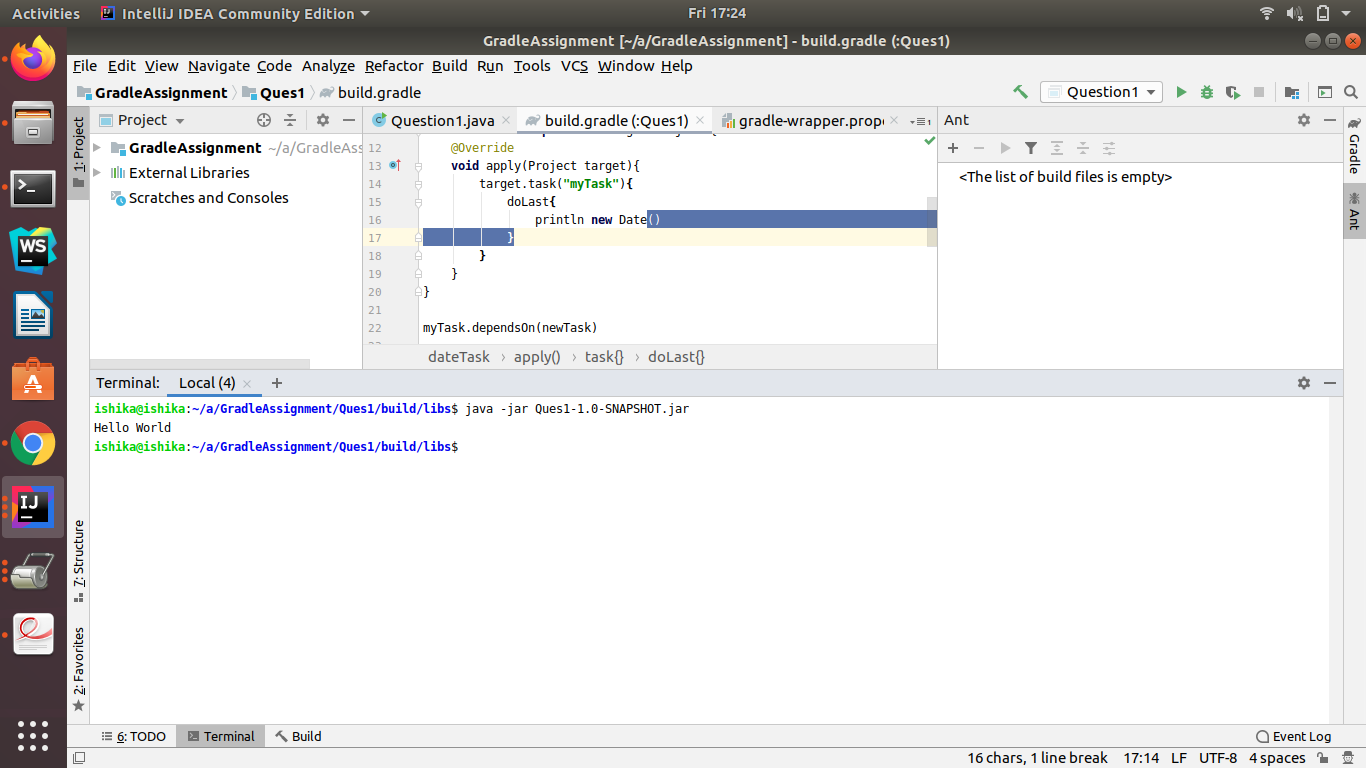
Ques1 ) Add a gradle dependency and its related repository url.



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".





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

dependencies {

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

compile group: 'joda-time', name: 'joda-time', version: '2.10.5'

runtime group: 'commons-collections' ,name: 'commons-collections' ,version: '3.2.2'

testRuntime group: 'commons-lang' ,name: 'commons-lang' ,version: '2.5'

}

testCompile are required for testing purpose during compilation.

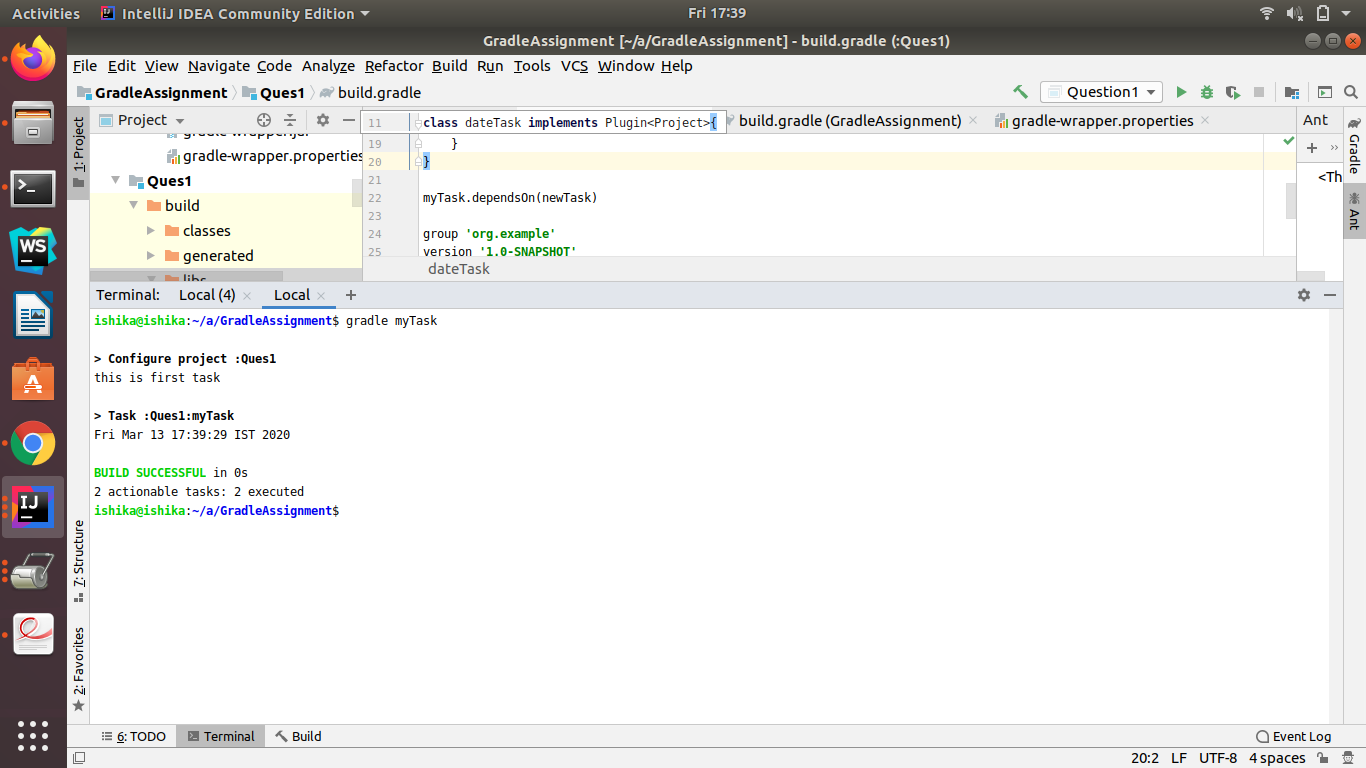
compile dependencies are required while compiling.

runtime dependencies are required while executing the project.

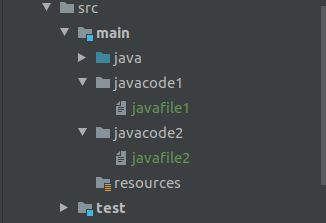
testRuntime are required for testing purpose.

Ques4 ) Create a custom plugin which contains a custom task which prints the current

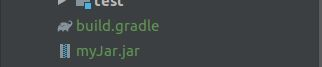
date-time. Using that plugin in your project, execute that task after the jar task executes.



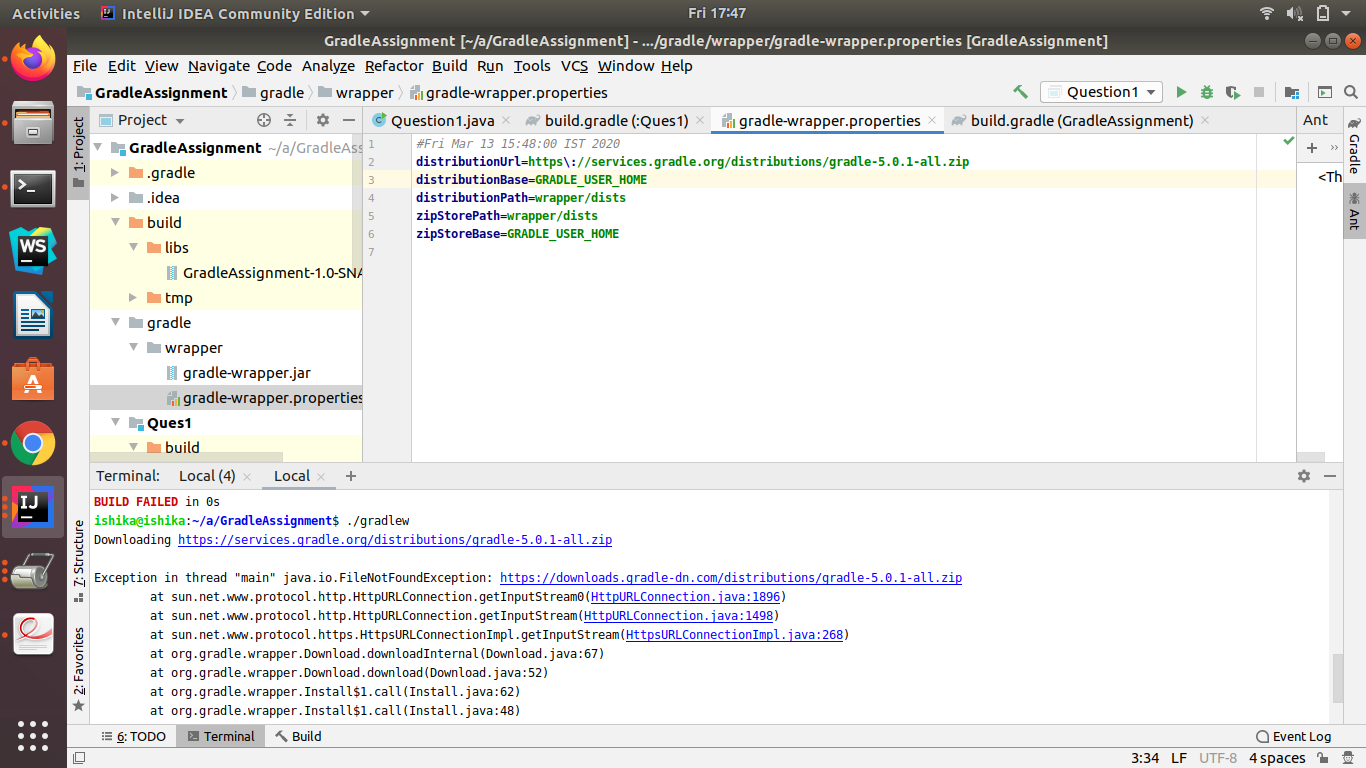
Ques5) Instead of using default source set, use src/main/javaCode1, src/main/javaCode2 to be taken as code source. Make sure that the JAR created contains files from both the directories and not from src/main/java.



jar cf myJar.jar src/main/javacode1/javafile1 src/main/javacode2/javafile2



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.



Ques 7) Run the gradle profile command and attach the resulting files.

