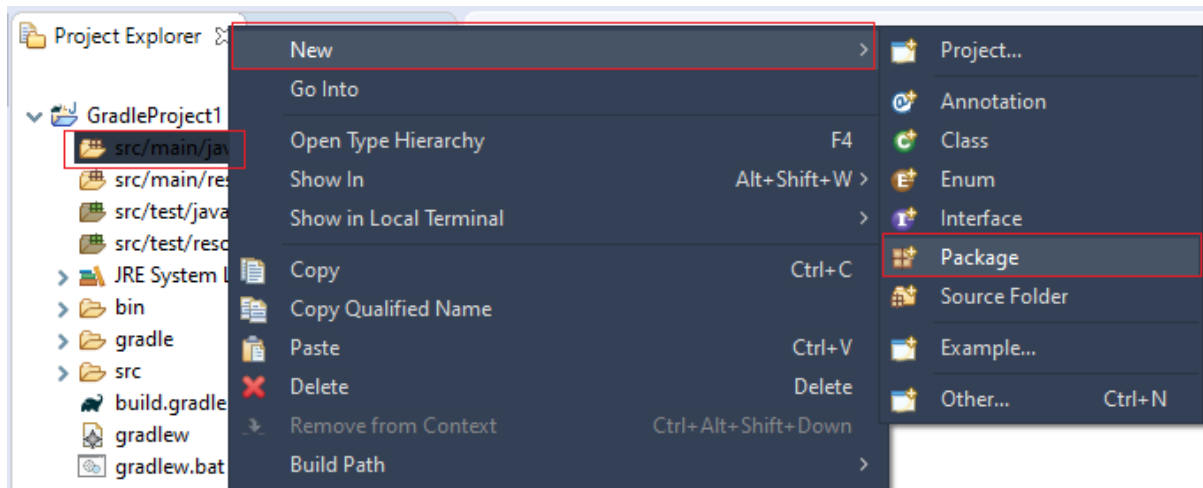


Gradle Eclipse Projects: -

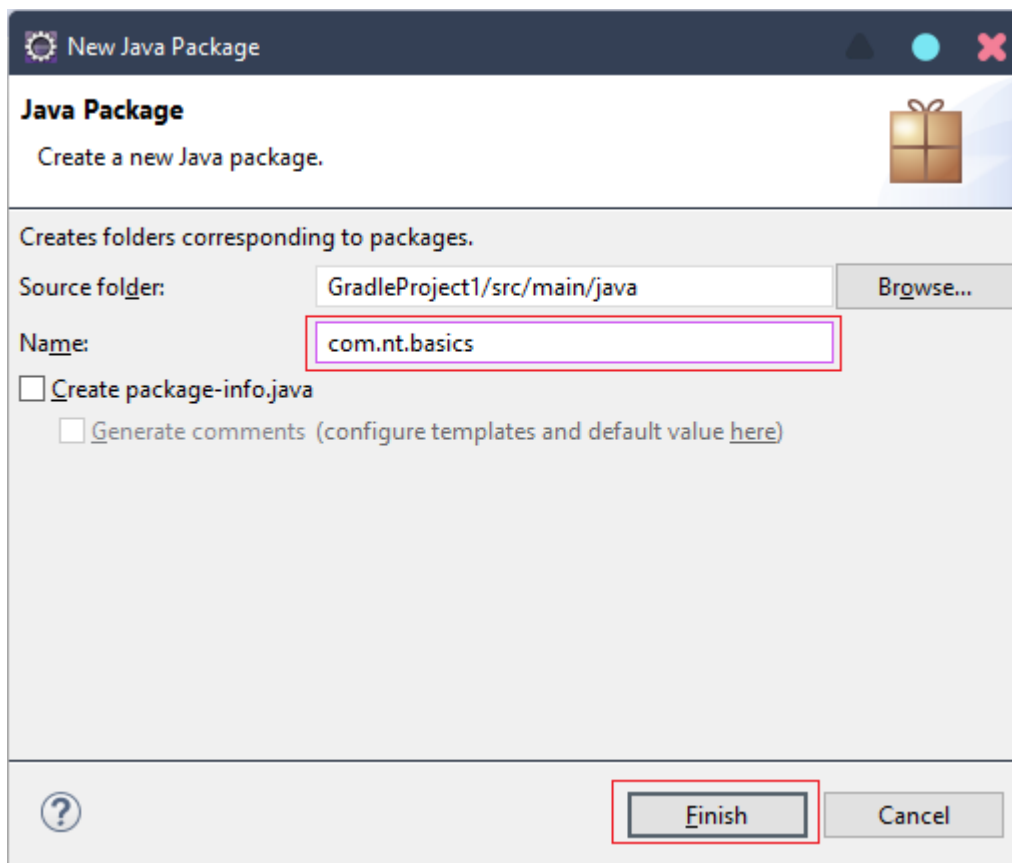
Project 1: Simple Standalone Project

Step #1: Create the project using the steps of How to a create Gradle Project using Eclipse PDF

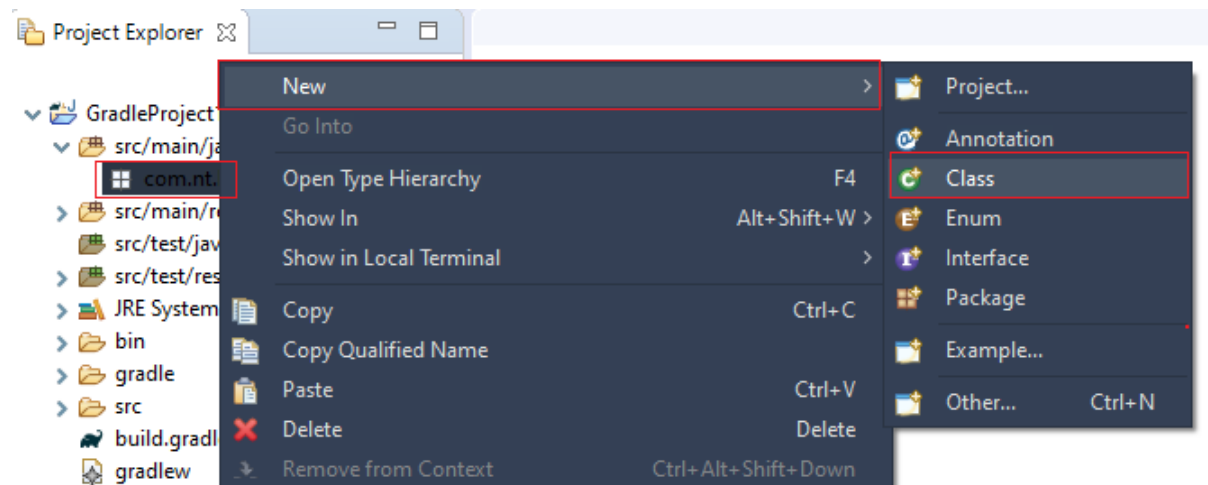
Step #2: Create a Package in src/main/java folder for the source code of your project, Right click on the folder click on New, then click on Package



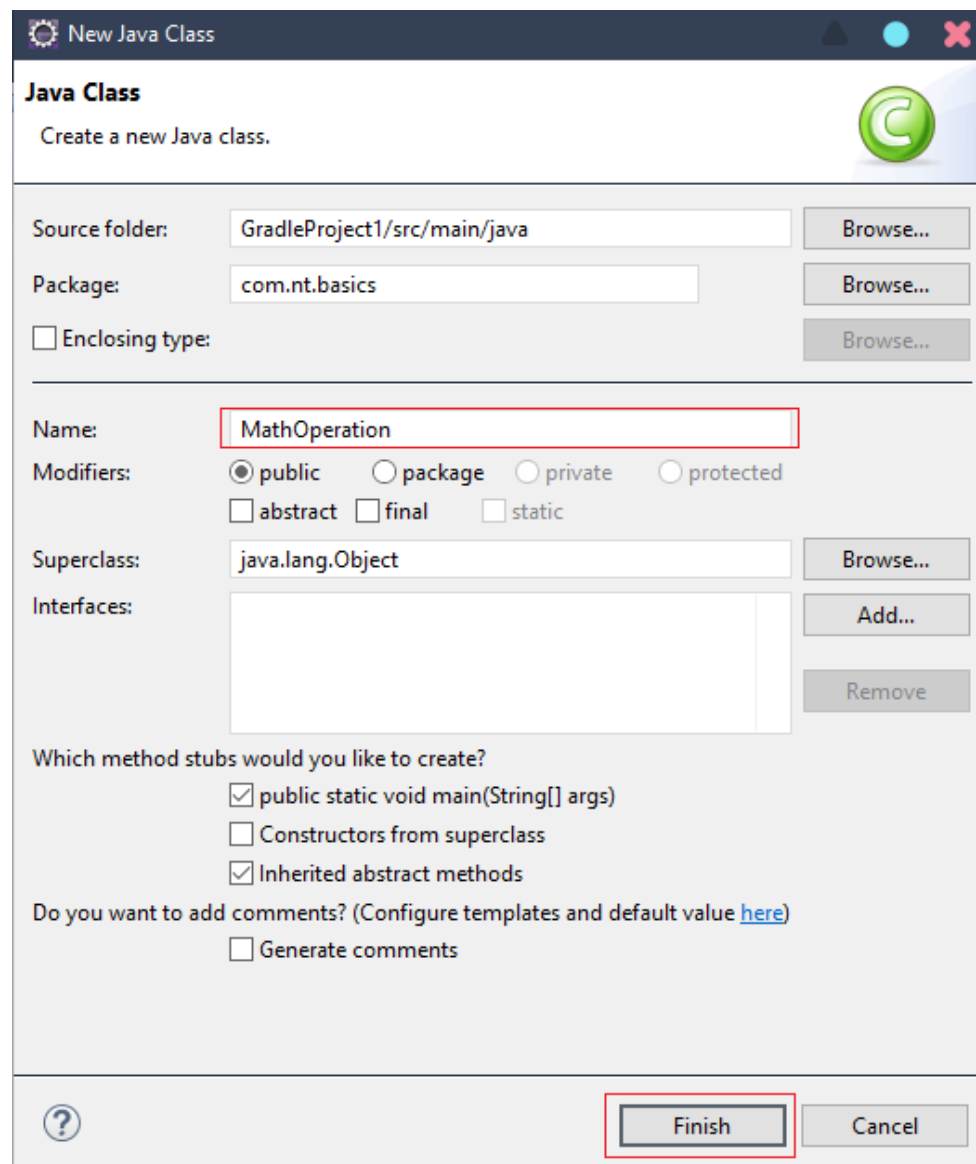
Step #3: Give the Package Name [like com.nt.basics], then click on Finish



Step #4: Create a class, Right click on Package click on New, then click on Class



Step #5: Give the Class name [like MathOperation], then click on Finish



Step #6: Write the following code in your class and save it

```
package com.nt.basics;

public class MathOperation {

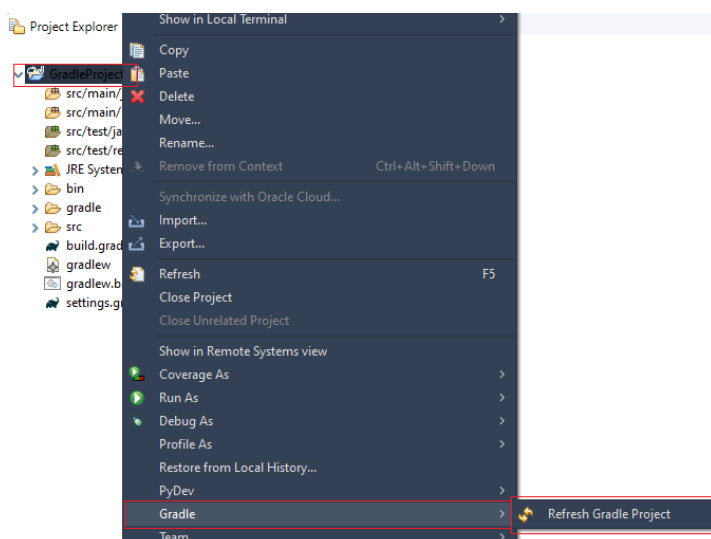
    public int sum (int x, int y) {
        return x+y;
    }
    public static void main(String[] args) {
        MathOperation operation = null;
        operation = new MathOperation();
        System.out.println("Result is: "+operation.sum(10, 20));
    }
}
```

Step #7: Go to build.gradle and modify the file with following code, and save it

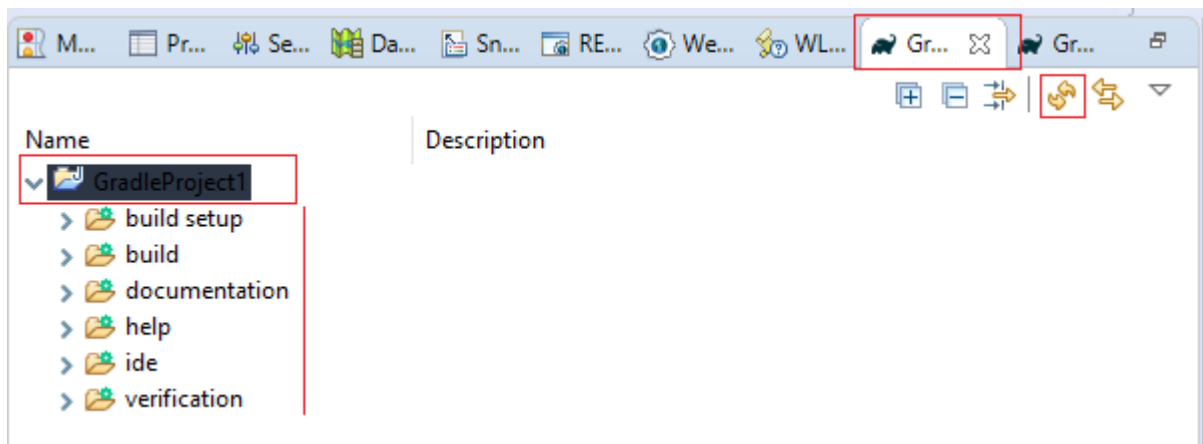
```
plugins {
    // Apply the java-library plugin to add support for Java Library
    id 'application'
}

mainClassName = 'com.nt.basics.MathOperation'
```

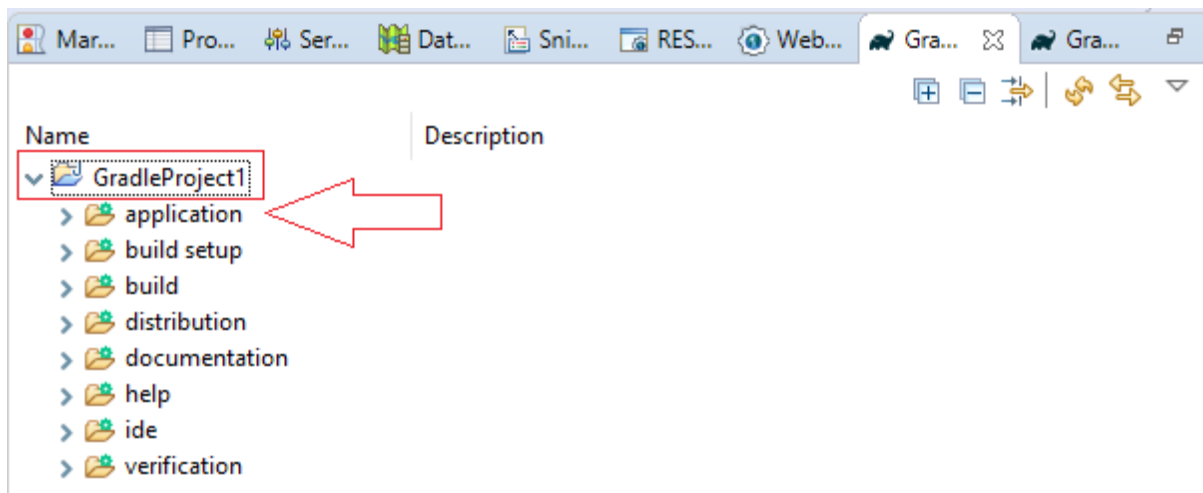
Step #8: Now time for gradle refresh, Right click on project, choose gradle then click on Refresh Gradle Project



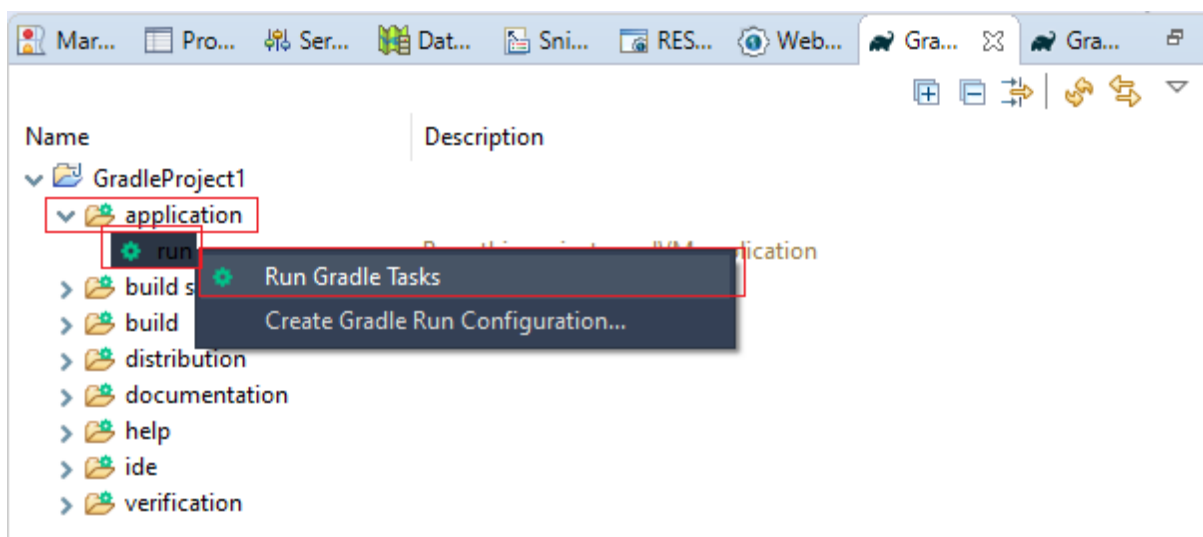
Step #9: Then go to Gradle Tasks window, choose you application and expand it but there is no application folder with “run” task, so just choose your project and click on refresh



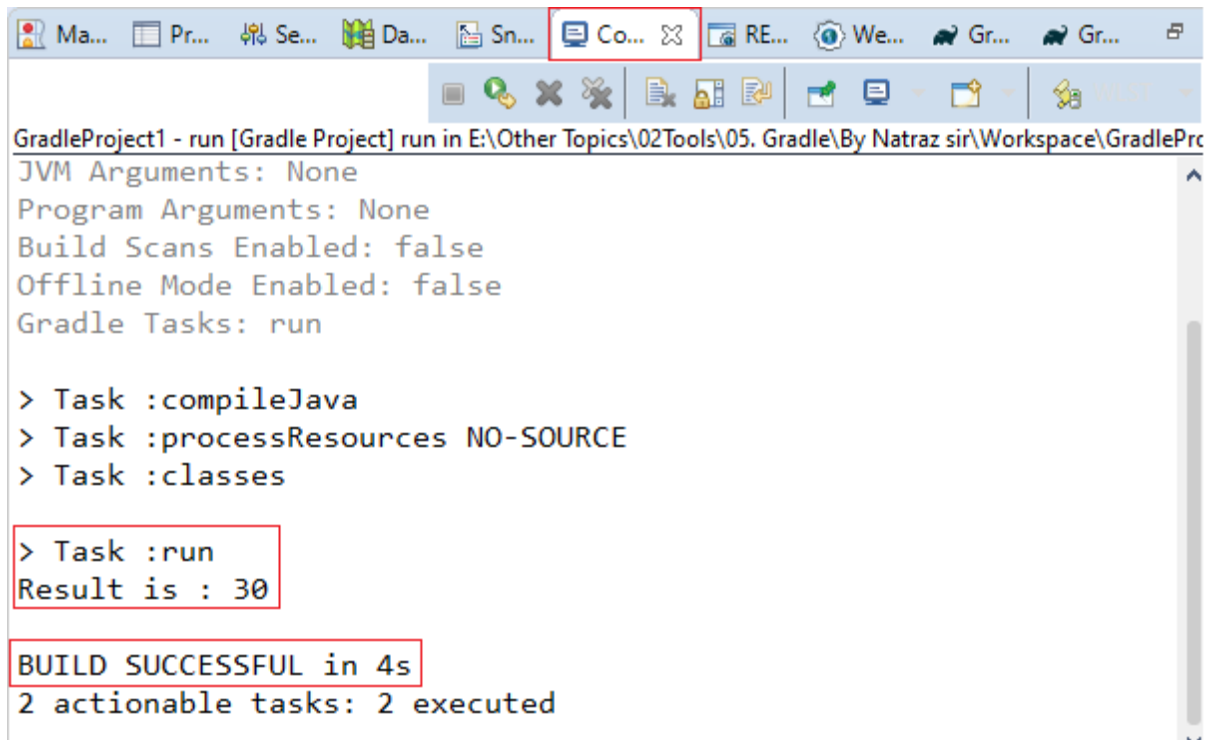
Step #10: Then you can see now an application folder had came



Step #11: Then expand the Project folder, right click on the run task, then click on Run Gradle Tasks



Step #12: Then go to your Console window you get the output with the BUILD SUCCESSFUL message



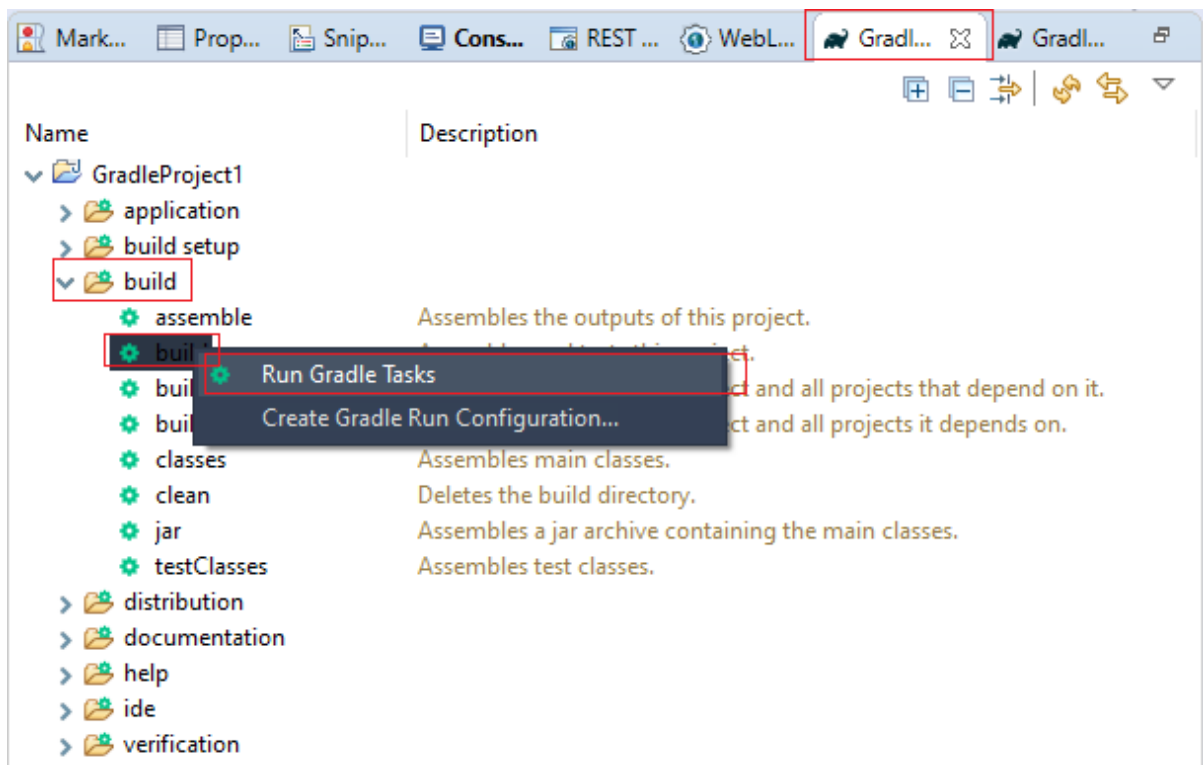
```
GradleProject1 - run [Gradle Project] run in E:\Other Topics\02Tools\05. Gradle\By Natraz sir\Workspace\GradlePr
JVM Arguments: None
Program Arguments: None
Build Scans Enabled: false
Offline Mode Enabled: false
Gradle Tasks: run

> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes

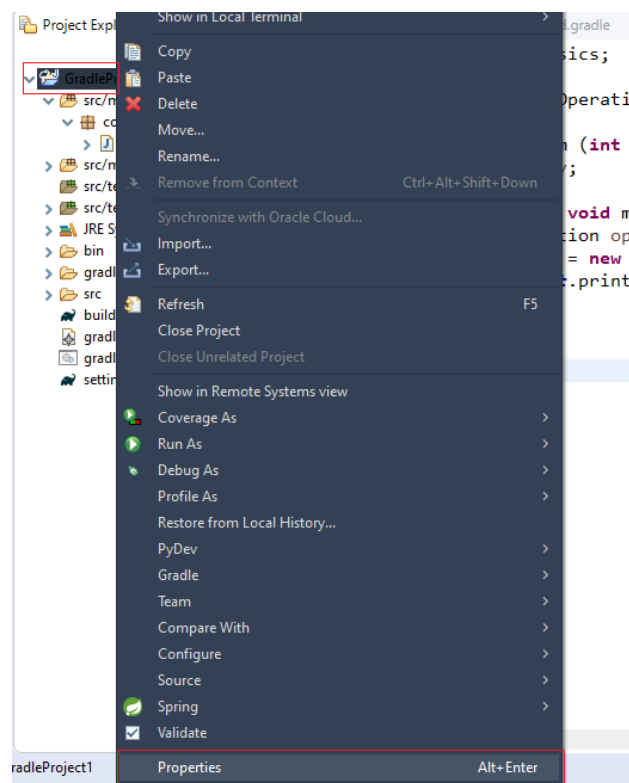
> Task :run
Result is : 30

BUILD SUCCESSFUL in 4s
2 actionable tasks: 2 executed
```

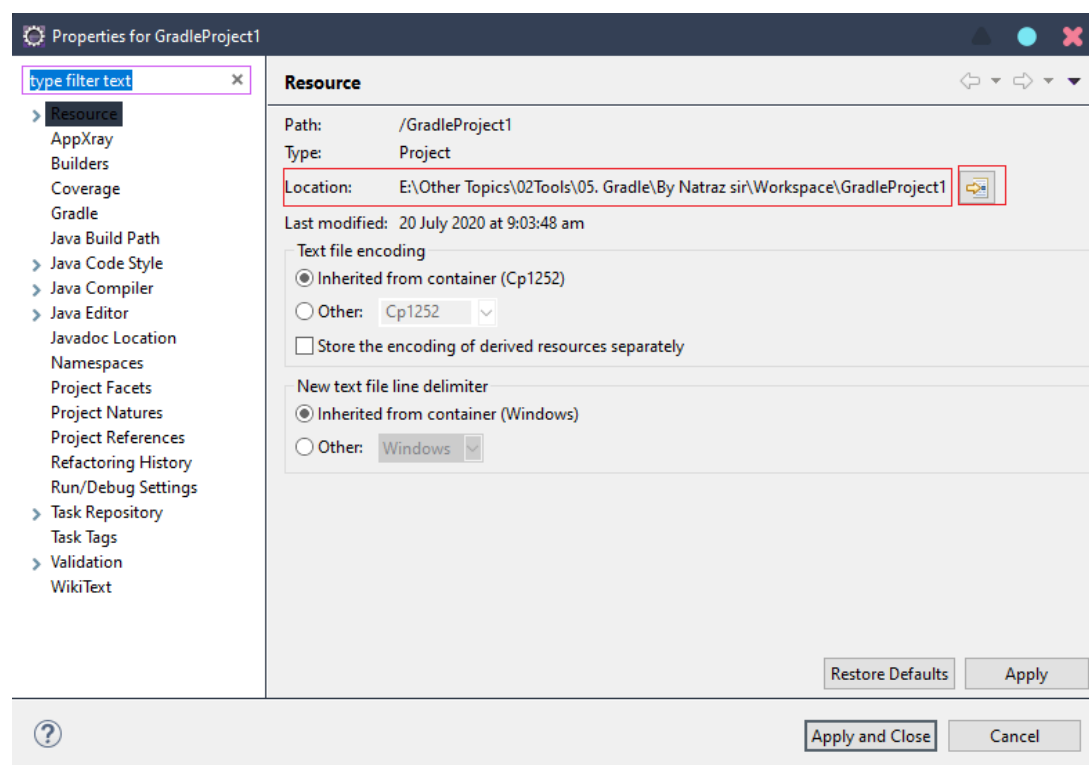
Step #13: If you want to jar file the go to Gradle Tasks window, choose your project and expand it, then expand the build folder right click on the “build” task then click on Run Gradle Tasks



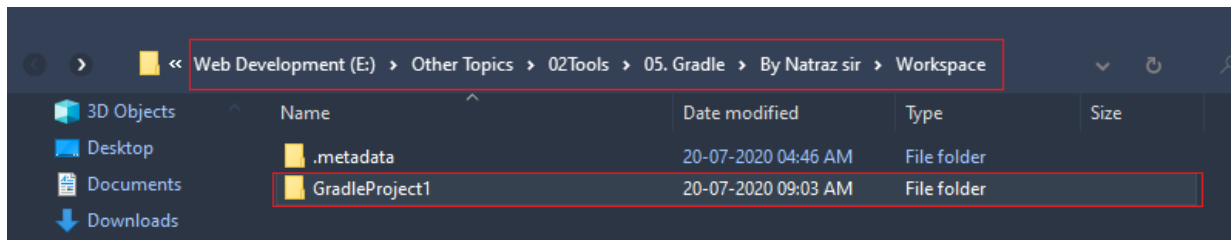
Step #14: But you can't see the Jar file in the project you have to go to the Project folder location, right click on the project then click on Properties other wish choose the project hit "alt+enter" shortcut key



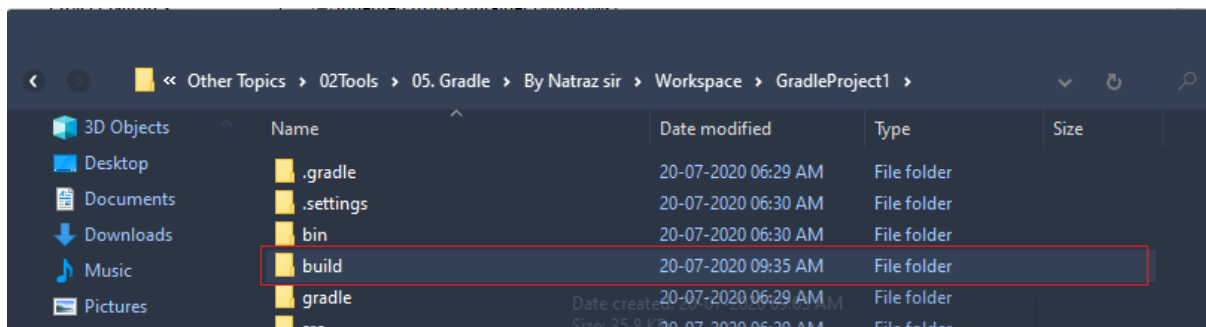
Step #15: Then the Properties window will open, in Location there is option for go to the project location [like browse option], click on that



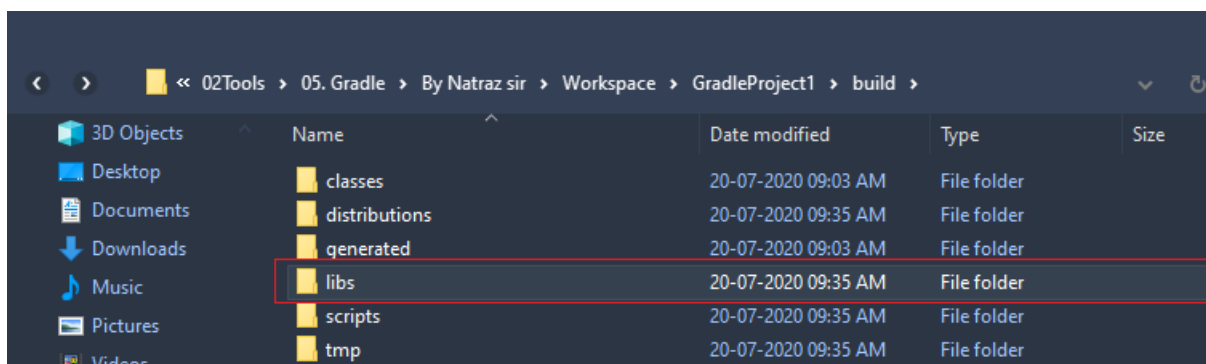
Step #16: Then the directory folder will open where you project is present, go inside the project



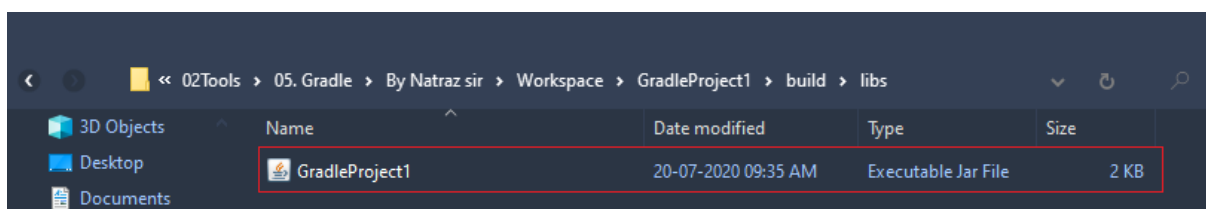
Step #17: Then go to build folder



Step #18: Then go to libs folder



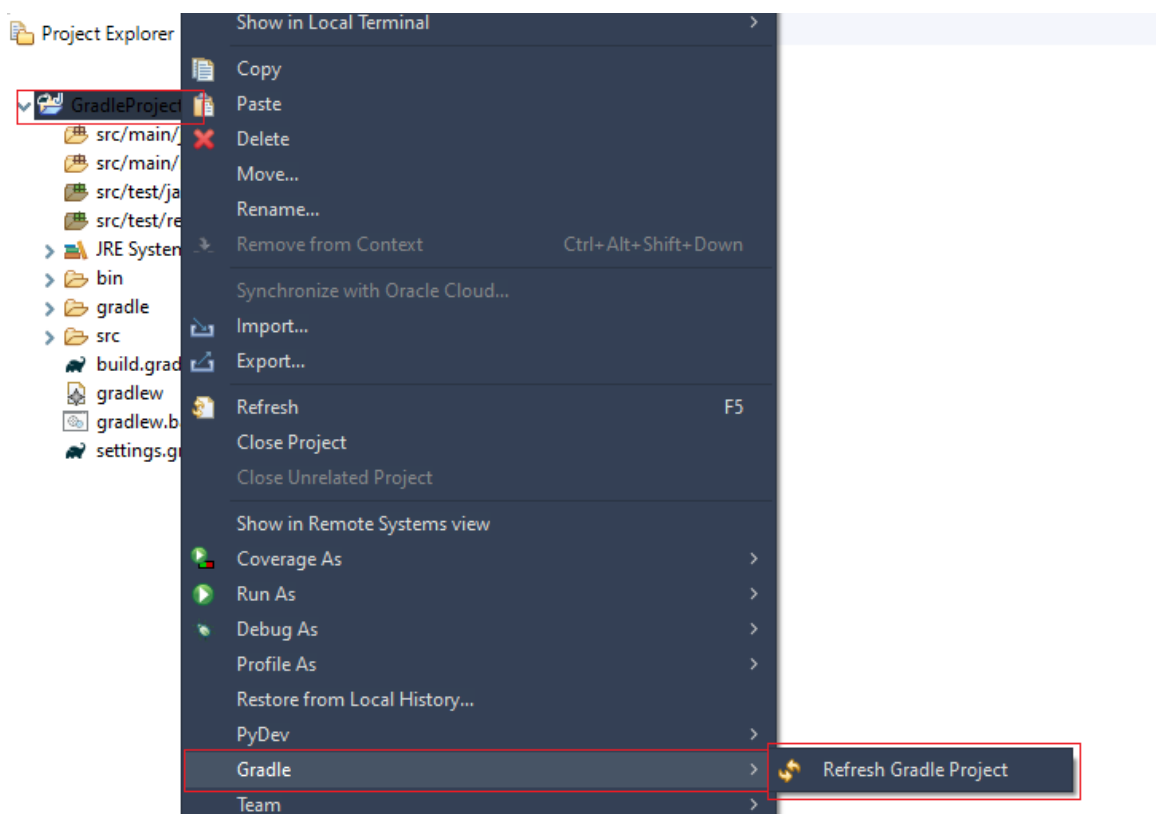
Step #19: Now you can see the jar file



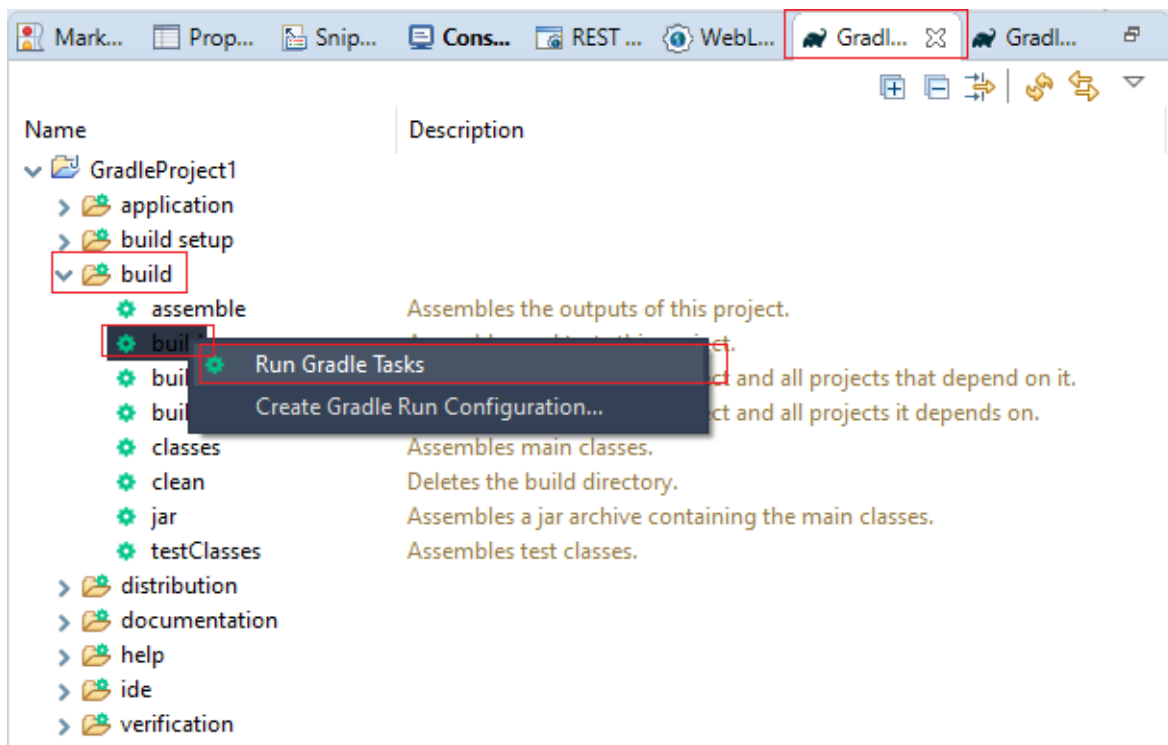
Step #20: You can add manifest attribute to make you jar as executable jar, open build.gradle file and modify the following file with following code and save it

```
plugins {  
    // Apply the java-library plugin to add support for Java Library  
    id 'application'  
}  
  
jar {  
    manifest {  
        attributes 'Main-Class':'com.nt.basics.MathOperation'  
    }  
}  
  
mainClassName = 'com.nt.basics.MathOperation'
```

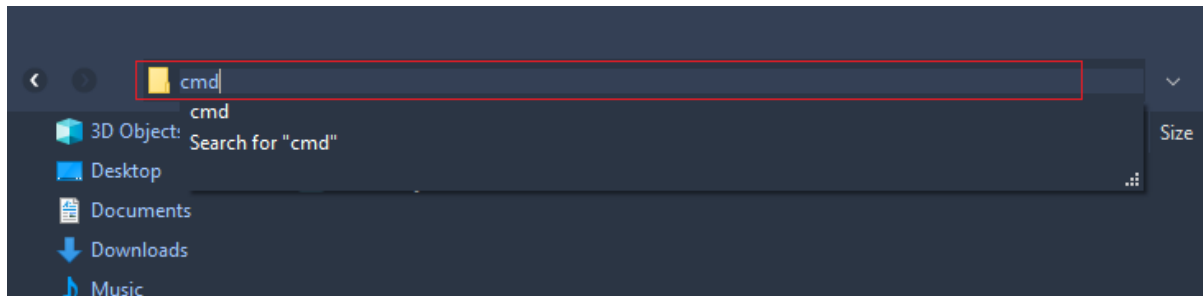
Step #21: Now time for gradle refresh, Right click on project, choose gradle then click on Refresh Gradle Project



Step #22: Then go to Gradle Tasks window, choose your project and expand it, then expand the build folder right click on the “build” task then click on Run Gradle Tasks

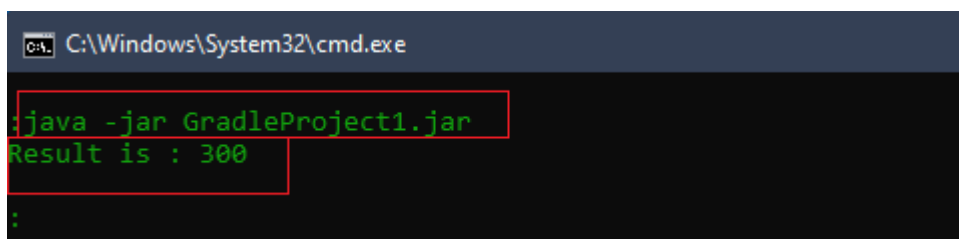


Step #23: The go to the Jar file location directory by following step-14 to step 19, then open command prompt in that location by typing cmd in the address bar of the directory



Step #24: Use following command to execute your jar

CMD> java -jar <jar_file_name>.jar



Project 2: Simple Standalone Project using jar for repositories

Step #1: Create the project using the steps of How to create Gradle Project using Eclipse PDF

Step #2: Following step-2 to step-6 from Project-1 ready the source code and all the other things.

Step #3: Go to build.gradle and modify the plugin to 'java' like below, because we want to test the project and 'test' task is available in java plugin

```
plugins {  
    // Apply the java-library plugin to add support for Java Library  
    id 'java'  
}
```

Step #4: For testing we need jars so that we have to mention repository on dependencies code in build.gradle like below

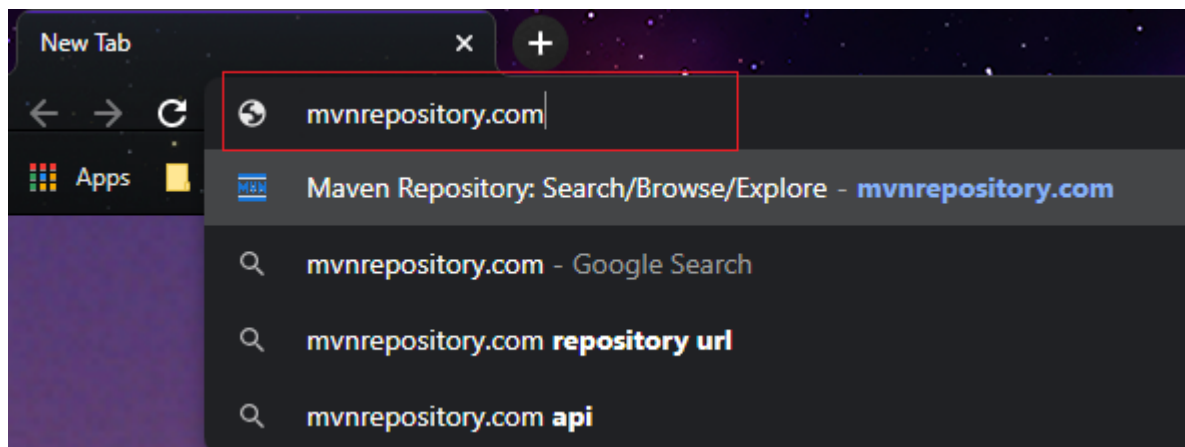
```
plugins {  
    // Apply the java-library plugin to add support for Java Library  
    id 'java'  
}  
  
repositories {  
    mavenCentral()  
}  
  
dependencies {  
    // https://mvnrepository.com/artifact/junit/junit  
    testCompile group: 'junit', name: 'junit', version: '4.13'  
}
```

Here we used Maven central repository so that we mention mavenCentral() in repositories { } enclosure.

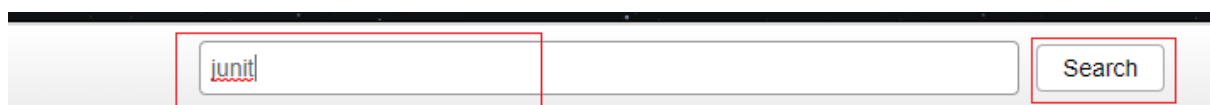
By default, the Junit jar dependency is available in dependencies { } enclosure.

But we have to know how to collect jar dependencies from internet, that's why we have to follow the following steps

Step #5: Go to browser Search mvnrepository.com and hit enter [\[click here\]](#)



Step #6: Type Junit in search bar, then click on search



What's New in Maven



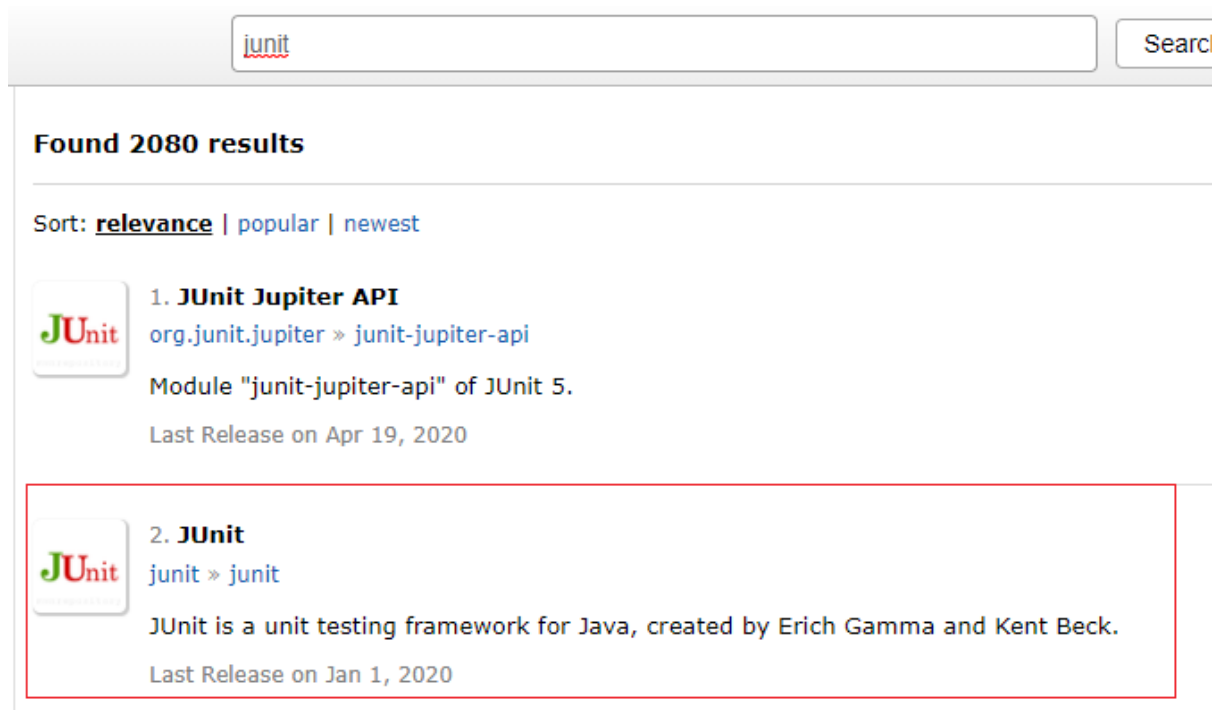
Microsoft Application Insights Logback Appender

[com.microsoft.azure](#) » [applicationinsights-logging-logback](#) » 2.6.2-BETA.2

This module provides a Microsoft Application Insights appender implementation for Logback frame

Last Release on Jul 18, 2020

Step #7: Now you get some Junit jar, click on one of them



Step #8: Click any version what you [like 4.13] from Central section, because Central jars are trustable jars

Central (30)				Redhat GA (3)	Redhat EA (3)	JBoss 3rd-party (1)	ICM (8)	Alfresco (1)
Version		Repository		Usages		Date		
4.13.x	4.13	Central		4,094		Jan, 2020		
	4.13-rc-2	Central		55		Dec, 2019		
	4.13-rc-1	Central		53		Oct, 2019		
	4.13-beta-3	Central		279		May, 2019		

Step #9: Scroll down click on Gradle section and click on the content the dependency will copy automatically

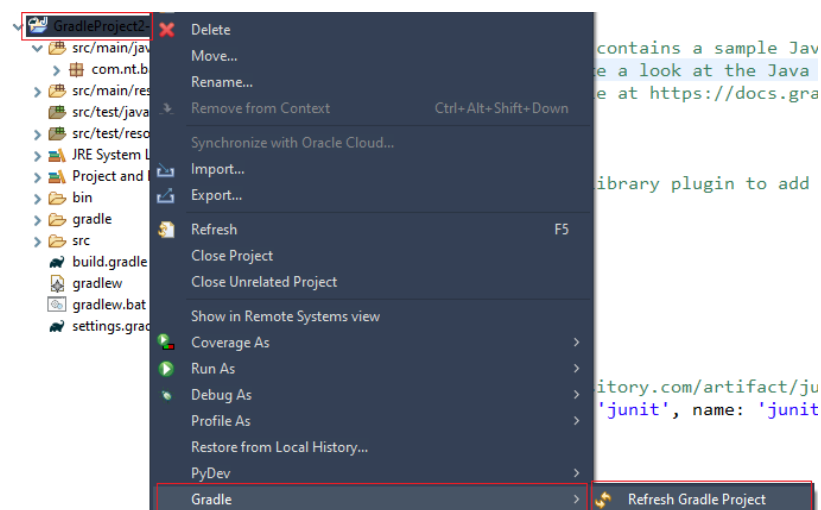
MavenGradleSBT Ivy Grape Leiningen Buildr

```
// https://mvnrepository.com/artifact/junit/junit
testCompile group: 'junit', name: 'junit', version: '4.13'
```

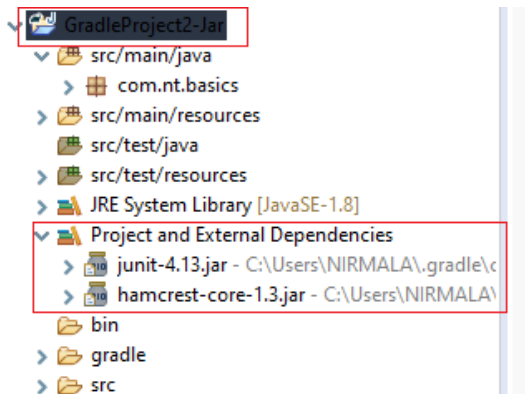
Step #10: Then go to build.gradle of your project, past the dependency in the dependencies { } enclosure, like below then save it

```
dependencies {
    // https://mvnrepository.com/artifact/junit/junit
    testCompile group: 'junit', name: 'junit', version: '4.13'
}
```

Step #11: Now time for gradle refresh, Right click on project, choose gradle then click on Refresh Gradle Project



Step #12: Now go to the Project and External Dependencies section of the project you will the Junit and other dependent jar also



Step #13: Now for writing the Test code in src/test/java folder. First create a package and the test class by following the step-2 to step-5 from Project-1

Step #14: Write the following code in your test class and save it

```
package com.nt.test;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

import com.nt.basics.MathOperation;

public class MathOperationTest {

    @Test
    public void testPositive() {
        int expected=300;
        int actual = new MathOperation().sum(100, 200);
        assertEquals("Test Positive:", expected, actual);
    }

    @Test
    public void testNegative() {
        int expected=-300;
        int actual = new MathOperation().sum(-100, -200);
        assertEquals("Test Negative:", expected, actual);
    }
}
```

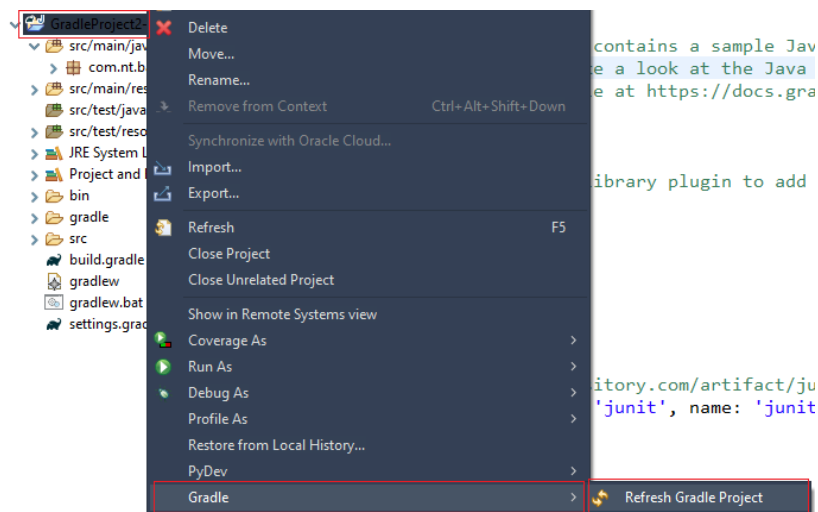
```

@Test
public void testMixed() {
    int expected=-100;
    int actual = new MathOperation().sum(100, -200);
    assertEquals("Test Positive:", expected, actual);
}

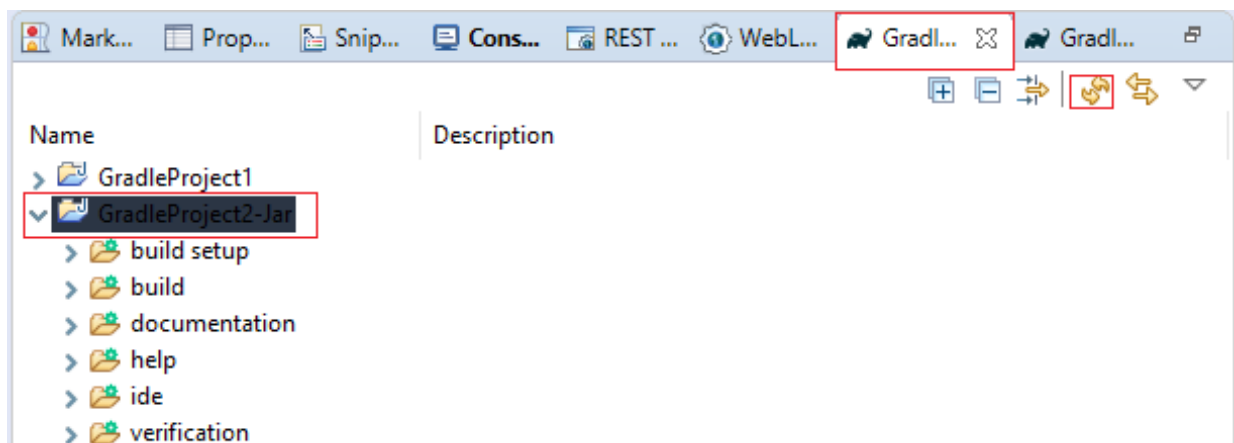
@Test
public void testZeros() {
    int expected=0;
    int actual = new MathOperation().sum(0, 0);
    assertEquals("Test Positive:", expected, actual);
}
}

```

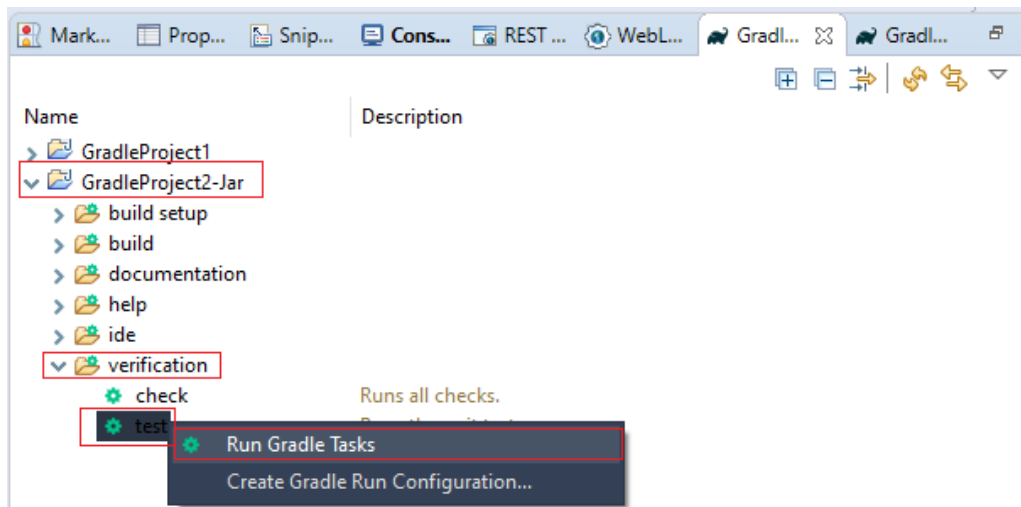
Step #15: Now time for gradle refresh, Right click on project, choose gradle then click on Refresh Gradle Project



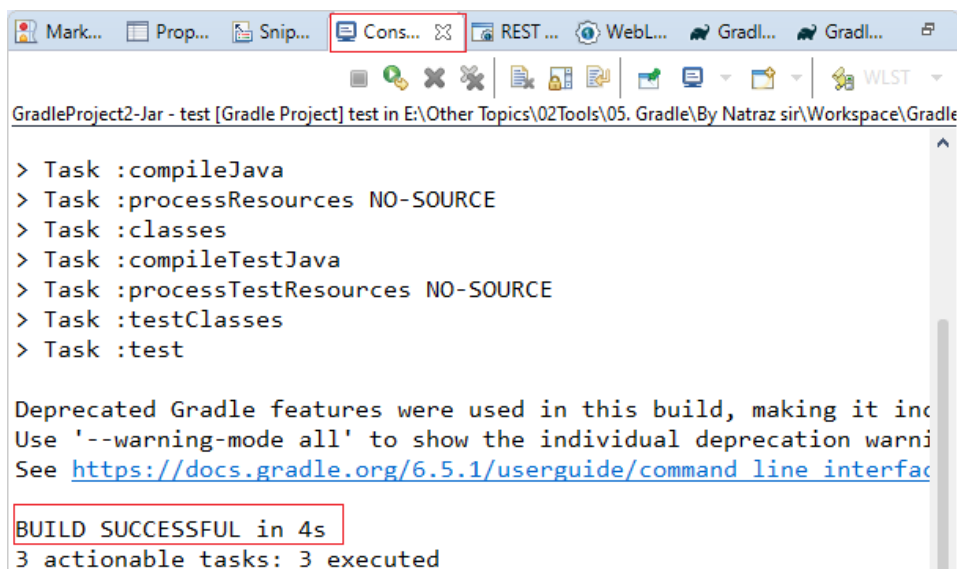
Step #16: Then go to Gradle Tasks window, choose your project and click on refresh



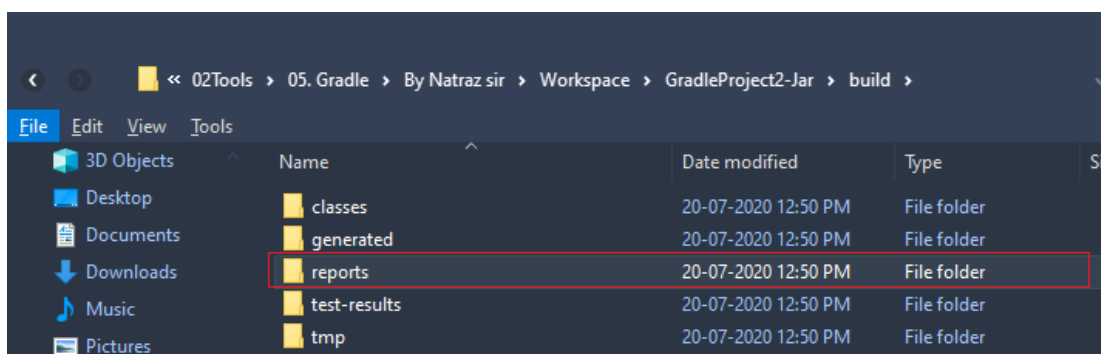
Step #17: Then expand the verification folder, right click on the test task, then click on Run Gradle Tasks



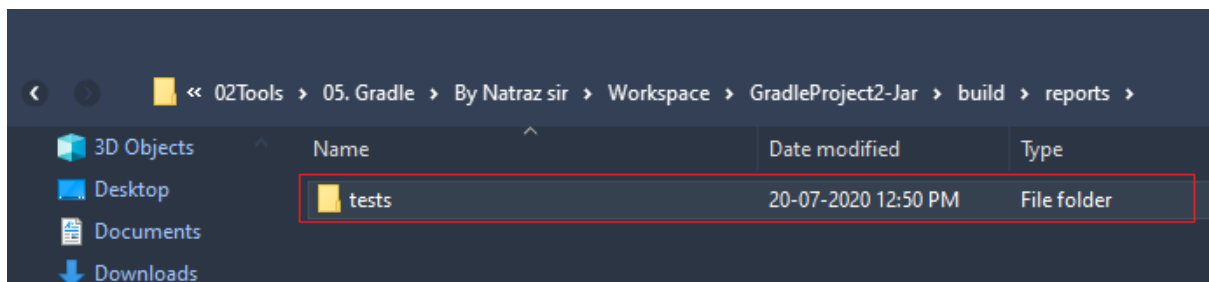
Step #18: Go to Console window you get the BUILD SUCCESSFUL message



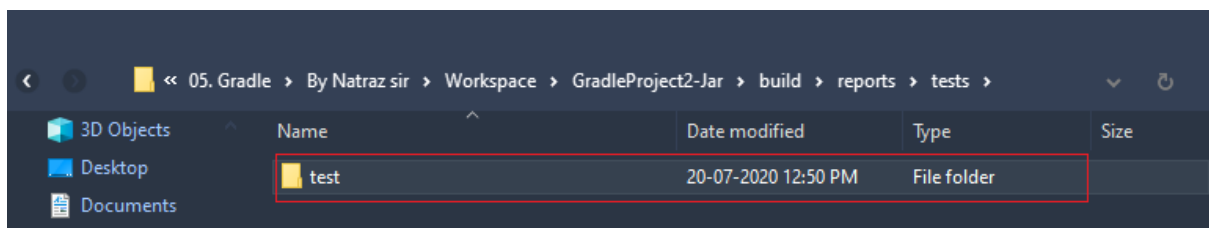
Step #19: Go to the project build folder by using step-14 to step -17 from Project-1, Then go to reports folder



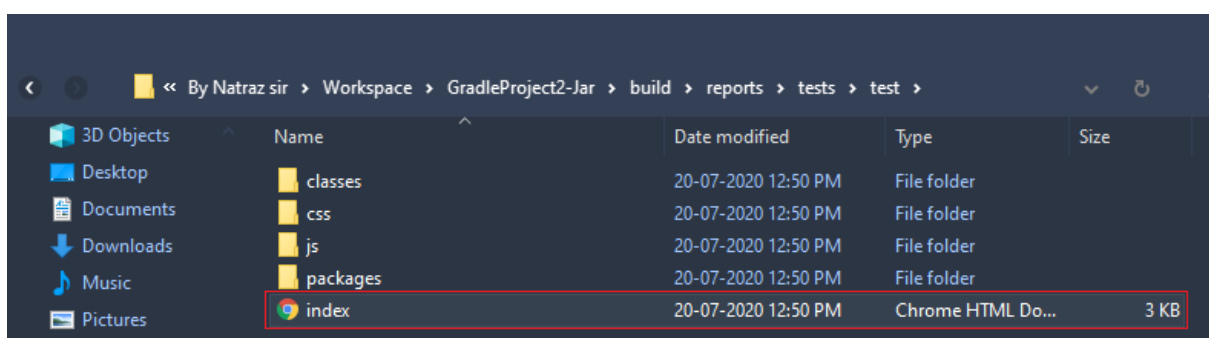
Step #20: Then go to tests folder



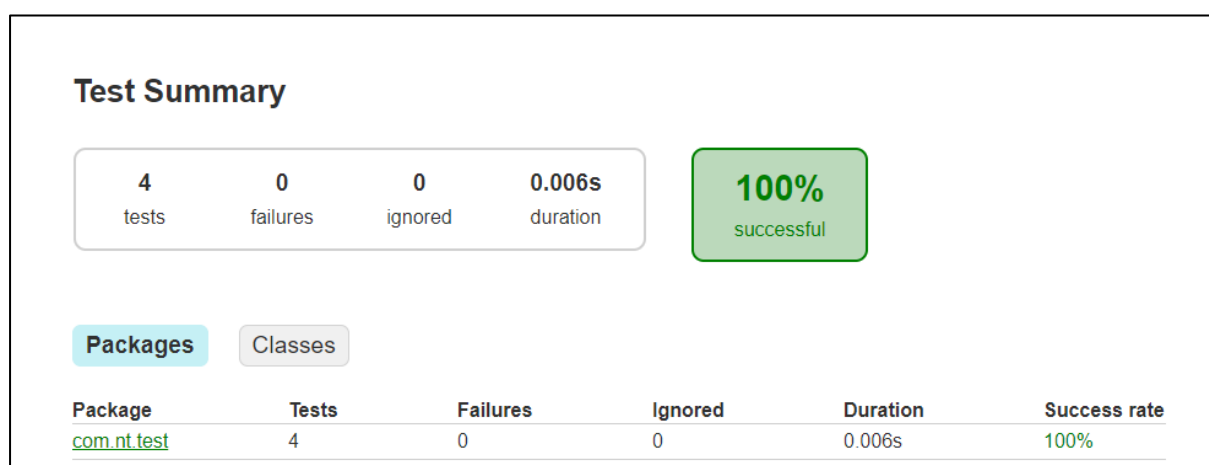
Step #21: Then go to test folder



Step #22: Inside test folder you get all the test reports, to view it click on index.html file



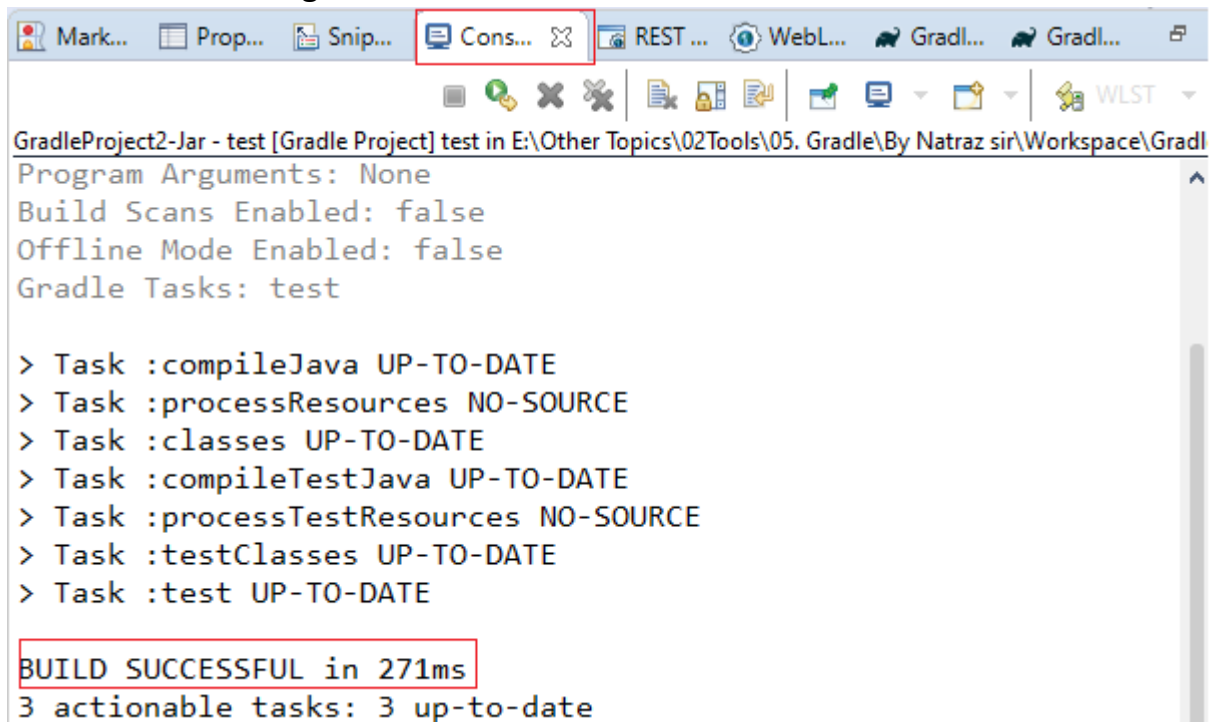
Step #23: Now it will open in any browser now you can see the generated test report



Step #24: In the console we get an warning deprecated because testCompile is deprecate now they give testImplementation, so change the testCompile with testImplementation like below in build.gradle, and save it

```
dependencies {  
    // https://mvnrepository.com/artifact/junit/junit  
    testImplementation group: 'junit', name: 'junit', version: '4.13'  
}
```

Step #25: Now follow the step-14 to step-16 from Project-2, for test but this time you don't get anu deprecated message, go to console you just get BUILD SUCCESSFUL message

A screenshot of an IDE's console window. The title bar shows tabs for 'Mark...', 'Prop...', 'Snip...', 'Cons...' (highlighted with a red box), 'REST ...', 'WebL...', and two 'Gradl...' tabs. The console output shows the following: 'GradleProject2-Jar - test [Gradle Project] test in E:\Other Topics\02Tools\05. Gradle\By Natraz sir\Workspace\Gradl', 'Program Arguments: None', 'Build Scans Enabled: false', 'Offline Mode Enabled: false', 'Gradle Tasks: test', a list of tasks (compileJava, processResources, classes, compileTestJava, processTestResources, testClasses, test) all marked as 'UP-TO-DATE', 'BUILD SUCCESSFUL in 271ms' (highlighted with a red box), and '3 actionable tasks: 3 up-to-date'.

```
GradleProject2-Jar - test [Gradle Project] test in E:\Other Topics\02Tools\05. Gradle\By Natraz sir\Workspace\Gradl  
Program Arguments: None  
Build Scans Enabled: false  
Offline Mode Enabled: false  
Gradle Tasks: test  
  
> Task :compileJava UP-TO-DATE  
> Task :processResources NO-SOURCE  
> Task :classes UP-TO-DATE  
> Task :compileTestJava UP-TO-DATE  
> Task :processTestResources NO-SOURCE  
> Task :testClasses UP-TO-DATE  
> Task :test UP-TO-DATE  
  
BUILD SUCCESSFUL in 271ms  
3 actionable tasks: 3 up-to-date
```

Step #26: Now by following step-18 to step-23 from Project-2, you can see the Test report again

Step #27: And You can also try with by failing some test and ignoring some test by using @Ignore in the test method, and you get some different test report like below, and here you get BUILD FAILED message don't worry we just fail some code that's why we get this.

Follow the step-14 to step-23 of Project-2 to get this type of repost and modify the MathOpertionTest as like below

Test Summary

4
tests

1
failures

1
ignored

0.008s
duration

66%
successful

Failed tests

Ignored tests

Packages

Classes

[MathOperationTest](#). [testNegative](#)

```
package com.nt.test;

import static org.junit.Assert.assertEquals;

import org.junit.Ignore;
import org.junit.Test;

import com.nt.basics.MathOperation;

public class MathOperationTest {
    @Ignore
    @Test
    public void testPositive() {
        int expected=300;
        int actual = new MathOperation().sum(100, 200);
        assertEquals("Test Positive:", expected, actual);
    }
    @Test
    public void testNegative() {
        int expected=300;
        int actual = new MathOperation().sum(-100, -200);
        assertEquals("Test Negative:", expected, actual);
    }
    @Test
    public void testMixed() {
        int expected=-100;
        int actual = new MathOperation().sum(100, -200);
        assertEquals("Test Positive:", expected, actual);
    }
    @Test
    public void testZeros() {
        int expected=0;
        int actual = new MathOperation().sum(0, 0);
        assertEquals("Test Positive:", expected, actual);
    }
}
```

Step #28: We can also use third parity library by collecting from internet follow the step-5 to step-10 from Project-2 to collect different jar dependency and add in build.gradle like below and save it

```
plugins {  
    // Apply the java-library plugin to add support for Java Library  
    id 'java'  
}  
  
repositories {  
    mavenCentral()  
    maven {  
        url "https://maven.repository.redhat.com/ga/"  
    }  
}  
  
dependencies {  
    // https://mvnrepository.com/artifact/junit/junit  
    testImplementation group: 'junit', name: 'junit', version: '4.13'  
  
    // https://mvnrepository.com/artifact/joda-time/joda-time  
    compile group: 'joda-time', name: 'joda-time', version: '2.9.7.redhat-1'  
}
```

When you use third party jar dependencies you have to mention the URL by collecting the URL from mvnrepository.com, where we collect jar dependency just below of that we get and kept in “url” attribute value by specifying maven { } enclosure in repositories { } enclosure.

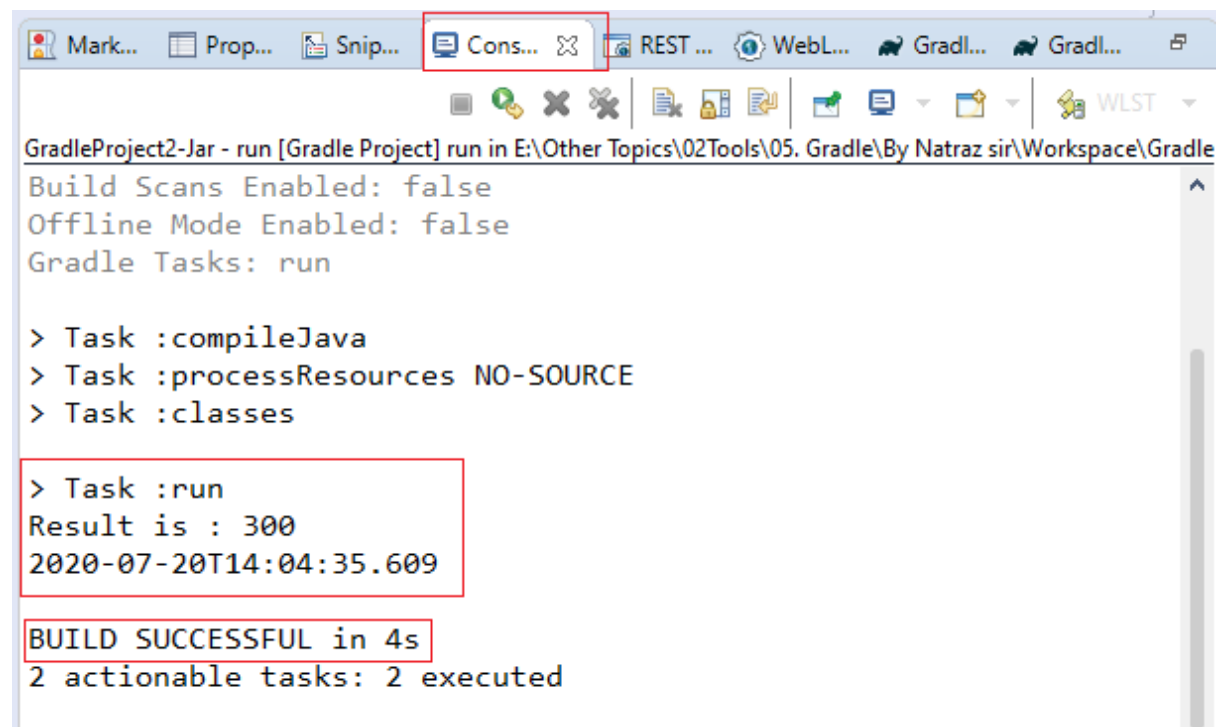
And the compile is deprecated instead we have to use implementation, like below

```
dependencies {  
    // https://mvnrepository.com/artifact/junit/junit  
    testImplementation group: 'junit', name: 'junit', version: '4.13'  
  
    // https://mvnrepository.com/artifact/joda-time/joda-time  
    implementaion group: 'joda-time', name: 'joda-time', version: '2.9.7.redhat-1'  
}
```

Step #29: Adding some the third-party jar related code, in your source code. So that you can test the jar is working or not, like below

```
LocalDateTime date=new org.joda.time.LocalDateTime();  
System.out.println(date);
```

Step #30: Use “application” plugin then only, we can execute the application and get the Date & Time output for this follow step-5 to step-12 from Project-1 to get output like below, go to console you can see the result with Joda Data & Time



The screenshot shows an IDE console window with the following content:

```
GradleProject2-Jar - run [Gradle Project] run in E:\Other Topics\02Tools\05. Gradle\By Natraz sir\Workspace\Gradle  
Build Scans Enabled: false  
Offline Mode Enabled: false  
Gradle Tasks: run  
  
> Task :compileJava  
> Task :processResources NO-SOURCE  
> Task :classes  
  
> Task :run  
Result is : 300  
2020-07-20T14:04:35.609  
  
BUILD SUCCESSFUL in 4s  
2 actionable tasks: 2 executed
```

Red boxes highlight the following sections in the original image:

- The "Cons..." tab in the IDE's top toolbar.
- The task execution list: `> Task :compileJava`, `> Task :processResources NO-SOURCE`, and `> Task :classes`.
- The execution result: `> Task :run`, `Result is : 300`, and `2020-07-20T14:04:35.609`.
- The final status: `BUILD SUCCESSFUL in 4s` and `2 actionable tasks: 2 executed`.