

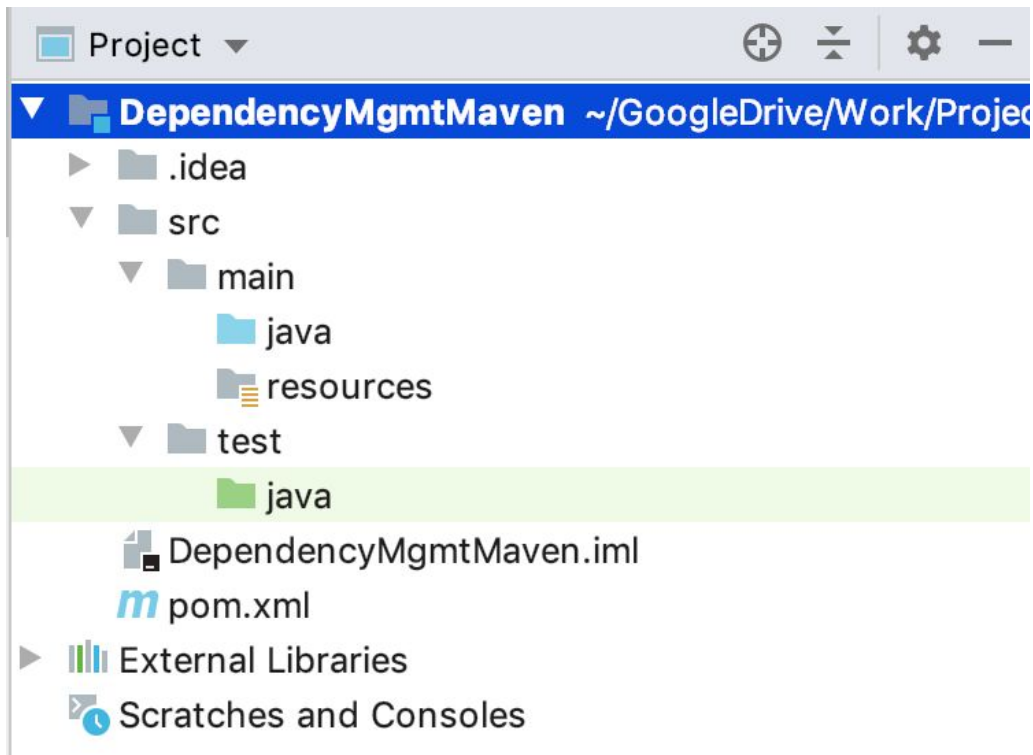
Tutorial

Dependency Management Tools: Maven and Gradle

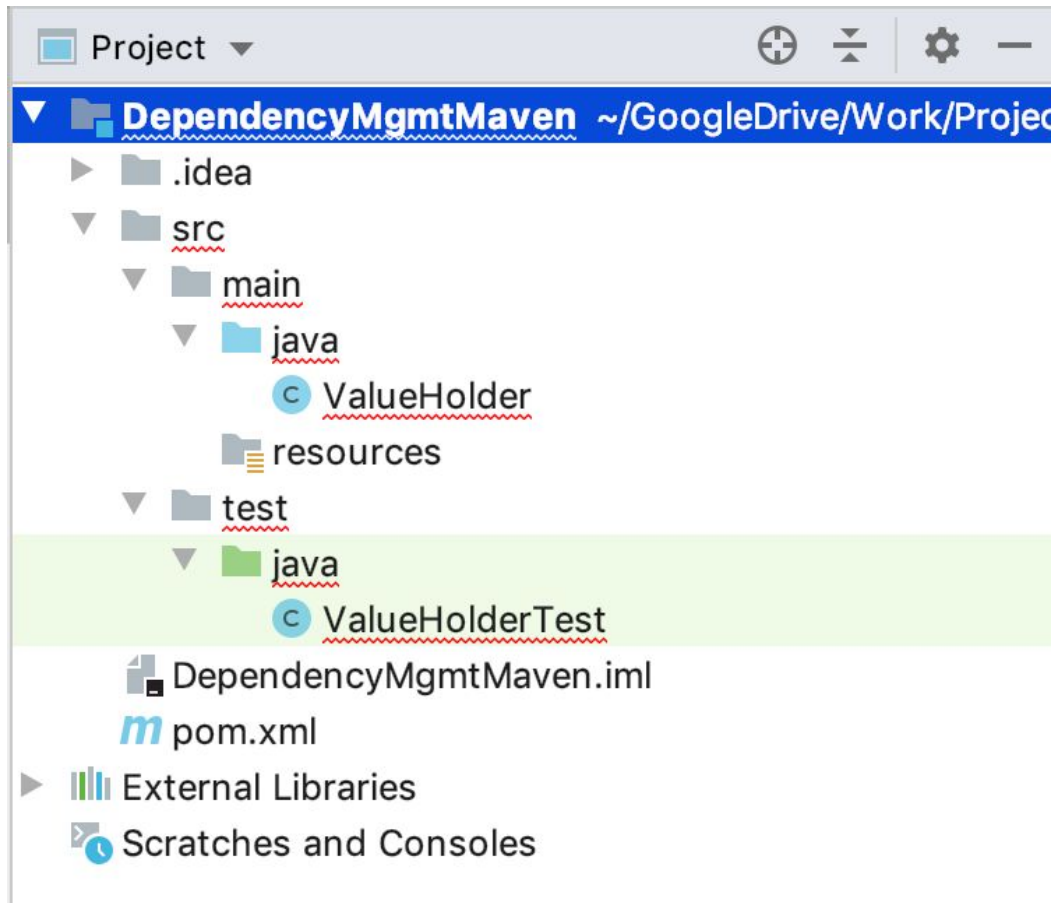
In this tutorial you will learn how to use both Maven and Gradle to manage dependencies for your project. First we will make a Maven project. Then we will make the same project but use Gradle instead. After you finish you will submit your Maven pom file and your Gradle build file.

Starting Maven

1. Create a new Maven Project in IntelliJ Idea
 - a. Click “Create new project”
 - b. Select Maven and Next
 - c. Put “com.example” as the groupid and “DependencyMgmtMaven” as the artifact id
2. Confirm that you have a project structure that looks like this:



3. Download the ValueHolder.java and ValueHolderTest.java files from [here](#) add them to the project structure as indicated below (ValueHolder in src/main/java and ValueHolderTest in src/test/java)



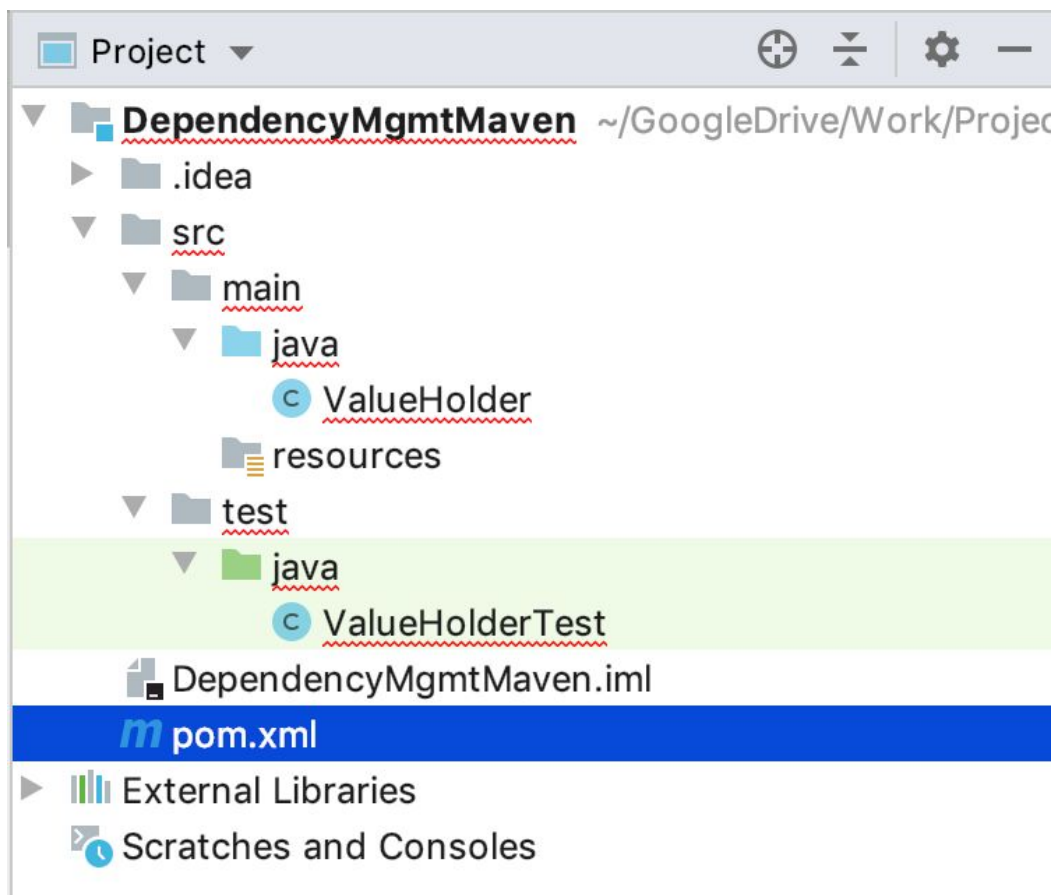
4. ValueHolder and ValueHolderTest are two java files that use external dependencies. Read about both of the files below and their functions to get a general understanding of the project.
 - a. ValueHolder is a simple class that contains a string value and has two functions
 - i. toJson: converts the object to a JSON string (you can learn more about JSON at https://www.w3schools.com/js/js_json_intro.asp if you are interested)
 - ii. fromJson: converts a JSON string back into a ValueHolder object
 - b. ValueHolderTest is a test file that verifies that the two functions mentioned above work correctly.

5. IntelliJ is not able to compile these classes (which is why they are underlined in red) because they depend on external code libraries (jar files) that are not currently available. The dependencies are described below:


- a. Gson: a JSON string to Java Object converter
- b. JUnit: a Java testing framework

We will now add these dependencies to our Maven pom.xml file

6. Open the “pom.xml” file as shown below



7. Now we will find the code snippets needed for the pom.xml file to manage our Gson dependency
8. Go to to <https://mvnrepository.com/> and search for Gson



Gson

Repository

Central 219

Sonatype 104

JCenter 72

Spring Plugins 50


Spring Lib M 48

IBiblio 10

Spring Lib Release 8

Found 321 results

Sort: **relevance** | popular | newest



1. **Gson**

[com.google.code.gson](#) » [gson](#)

Gson

Last Release on May 22, 2018

9. Click Gson to see the different versions of this dependency
10. Select the latest version of Gson (at the time of writing the document, it is version 2.8.5) as shown in the picture below

Home » [com.google.code.gson](#) » [gson](#)



Gson

Gson

License	Apache 2.0
Categories	JSON Libraries
Tags	google json
Used By	9,793 artifacts

Central (28) [Atlassian 3rd-P Old \(4\)](#) [Spring Plug](#)

2.8.x

[2.8.5](#)
[2.8.4](#)
[2.8.3](#)
[2.8.2](#)
[2.8.1](#)
[2.8.0](#)

11. Copy the code found in the “Maven” Box

Maven [Gradle](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```
<!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.8.5</version>
</dependency>
```

☒ Include comment with link to declaration

12. Go back to your Maven pom file and add the following lines

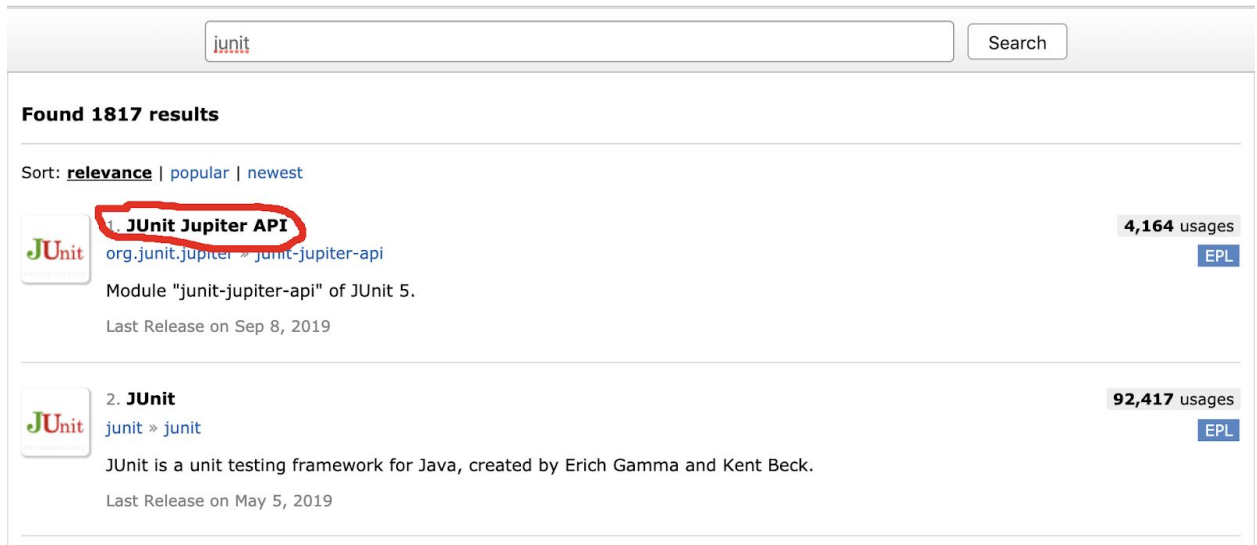


```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>com.example</groupId>
8   <artifactId>ValueHolderMaven</artifactId>
9   <version>1.0-SNAPSHOT</version>
10
11   <dependencies>
12
13   </dependencies>
14
15 </project>
```

13. Inside the “dependencies” tags paste in the code snippet you copied from the Maven Repository as shown below

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.5</version>
  </dependency>
</dependencies>
```

14. Now we will do the same thing for JUnit
15. Search for JUnit. You will see several results. We want JUnit 5, which is represented by “JUnit Jupiter API”. Select “JUnit Jupiter API” as shown below. Do not select “JUnit” this is the earlier JUnit 4 version.



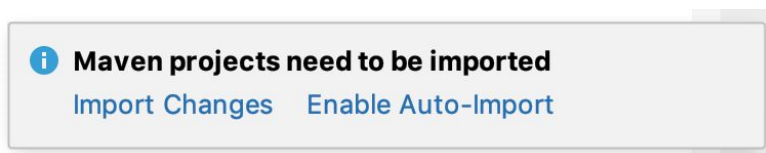
16. Select the latest version. At the time of this writing the latest version is 5.5.2.
17. Copy the Maven code snippet
18. Add that snippet to your pom.xml file under the gson dependency as shown below

```

<dependencies>
  <!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.5</version>
    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.5.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
</project>

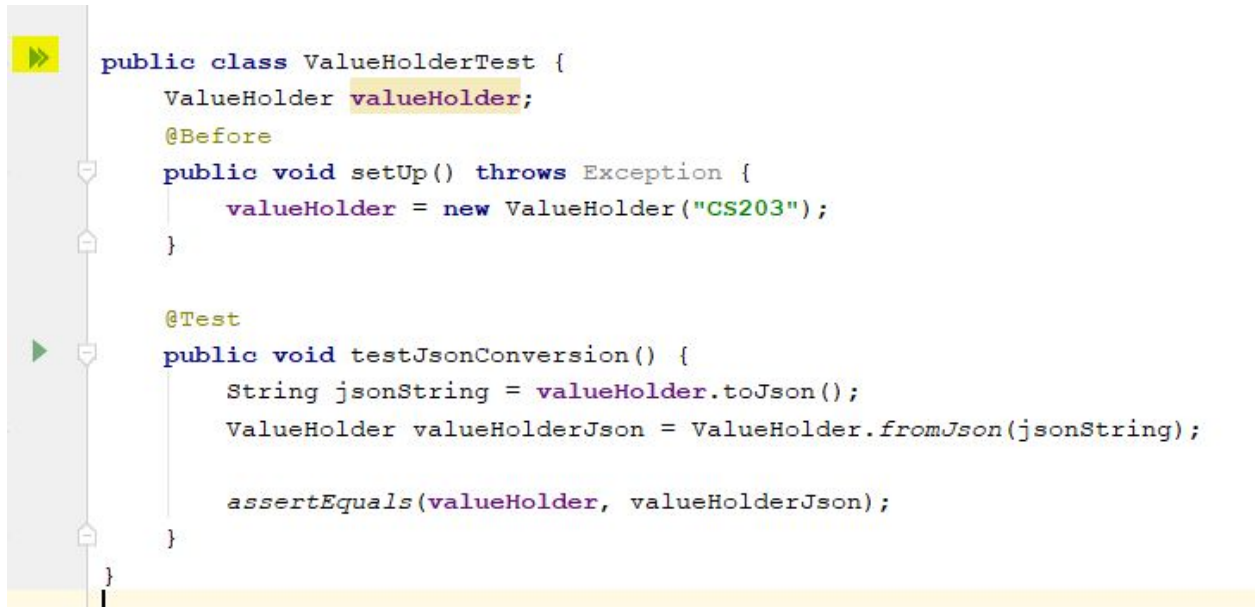
```

19. If you see a popup telling you that maven projects need to be imported, select “Enable Auto-Import”

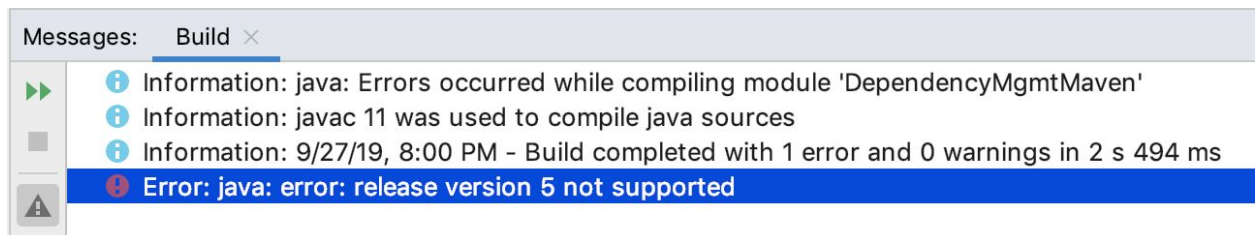


20. Confirm that your dependencies have been downloaded and made available to your project by running the ValueHolderTest (by opening the ValueHolderTest.java file and clicking the green double-arrow icon shown in the image below)

If you don't see the icon you might need to re-import the maven dependencies by right-clicking on your pom file and clicking Maven>Re-Import.



21. You will likely see the following error (if not, continue to the next step):

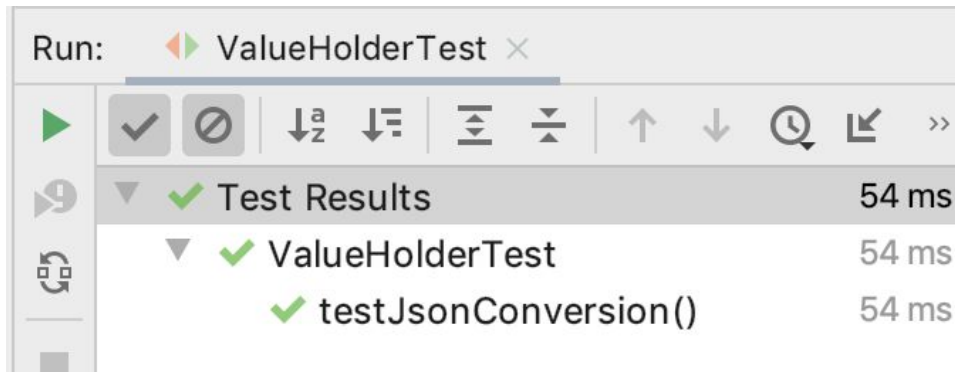


This error occurs because Maven projects in the current version of IntelliJ default to a very old version of Java. To correct this error, follow all of the steps under the “*IntelliJ Reverts to An Earlier Java Version After you Specify a Latter One*” and “*Error:java: error: release version 5 not supported*” sections of the [“Setup and Test Running Issues in IntelliJ”](#) document.

Note: When the instructions tell you to set “the desired version as the “Language Level”, select the highest version available.

After following all steps, rerun the test as described in the previous step.

22. After running the test you should see something like this



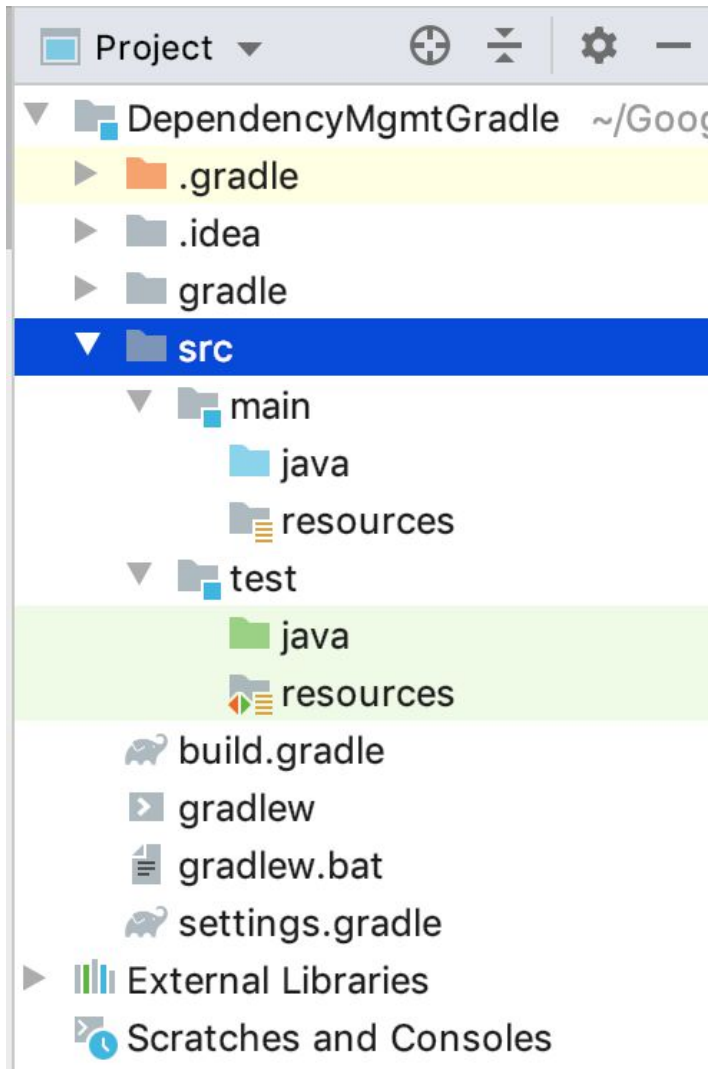
23. You now know how to add dependencies using Maven!

24. Submit a screenshot of your pom.xml file and your test completing to LearningSuite or Canvas (whichever is being used for your class).

Using Gradle

Using gradle is similar to using maven so this section should go by a bit quicker

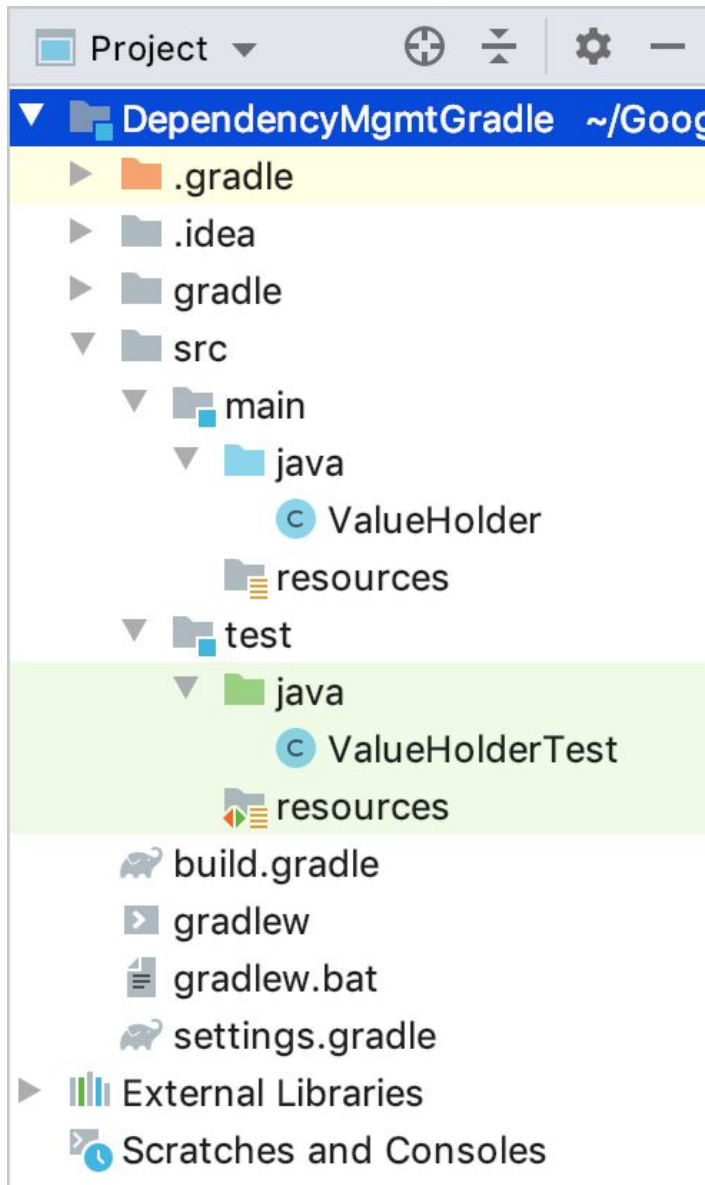
1. Create a new Gradle Project in IntelliJ IDEA
 - a. Click "Create new project"
 - b. Select Gradle and Next
 - c. Put "com.example" as the groupid and "DependencyMgmtGradle" as the artifact id
2. Wait for the syncing to finish
3. Check to make sure your project structure looks like the picture below



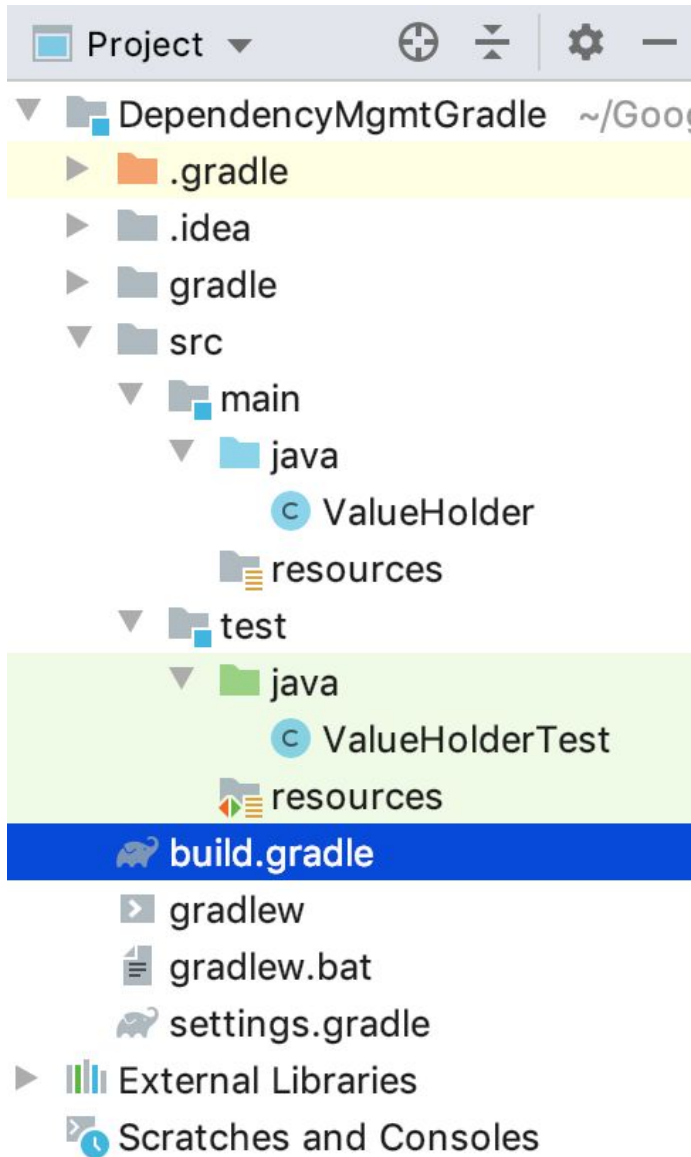
4. Download the ValueHolder and ValueHolderTest from the class website and add them to the project structure (ValueHolder in src/main/java and ValueHolderTest in src/test/java)

Note: You can use the same files you downloaded for the Maven part of this tutorial.

5. Check that your project structure now looks like the picture below



6. Open "build.gradle". This is the configuration file for Gradle projects that is equivalent to pom.xml for Maven projects.



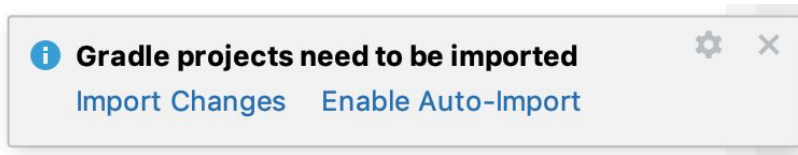
7. Now we will find the code snippets that we need to add for both of the dependencies by going to <https://mvnrepository.com/> again
8. Find the same version of Gson you used for the Maven tutorial and copy the gradle code snippet



9. Add the code snippet in the dependencies section of your build.gradle file **and change the word compile to implementation because compile is now deprecated.**

```
dependencies {  
    testCompile group: 'junit', name: 'junit', version: '4.12'  
    // https://mvnrepository.com/artifact/com.google.code.gson:gson  
    implementation group: 'com.google.code.gson', name: 'gson', version: '2.8.5'  
}
```

10. If you see a popup telling you that gradle projects need to be imported, select “Enable Auto-Import”



11. If your build.gradle file's dependencies section has a line for 'junit' version 4.12 (or any JUnit 4 version) as shown in the image for step 9, delete that line. We will replace it with JUnit 5 in the following steps.
12. Find the same version of JUnit (JUnit Jupiter API) you used for the Maven tutorial and copy the gradle code snippet

Maven
Gradle
SBT
Ivy
Grape
Leiningen
Buildr

```
// https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api
testCompile group: 'org.junit.jupiter', name: 'junit-jupiter-api', version: '5.5.2'
```

☒ Include comment with link to declaration

13. Add the code snippet in the dependencies section of your build.gradle file **and change the word testCompile to testImplementation because testCompile is now deprecated.**

```
dependencies {
    // https://mvnrepository.com/artifact/com.google.code.gson/gson
    implementation group: 'com.google.code.gson', name: 'gson', version: '2.8.5'

    // https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api
    testImplementation group: 'org.junit.jupiter', name: 'junit-jupiter-api', version: '5.5.2'
}
```

14. Verify that the dependencies have been downloaded and made available to your project by running the ValueHolderTest (by opening the ValueHolderTest.java file and clicking the green double-arrow icon shown in the image below)



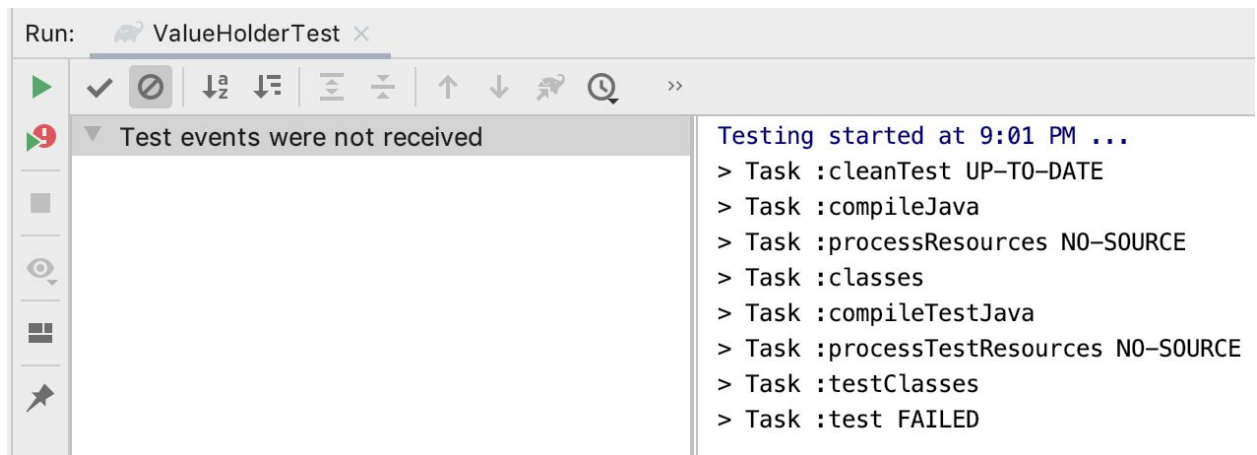
```
public class ValueHolderTest {
    ValueHolder valueHolder;

    @Before
    public void setUp() throws Exception {
        valueHolder = new ValueHolder("CS203");
    }

    @Test
    public void testJsonConversion() {
        String jsonString = valueHolder.toJson();
        ValueHolder valueHolderJson = ValueHolder.fromJson(jsonString);

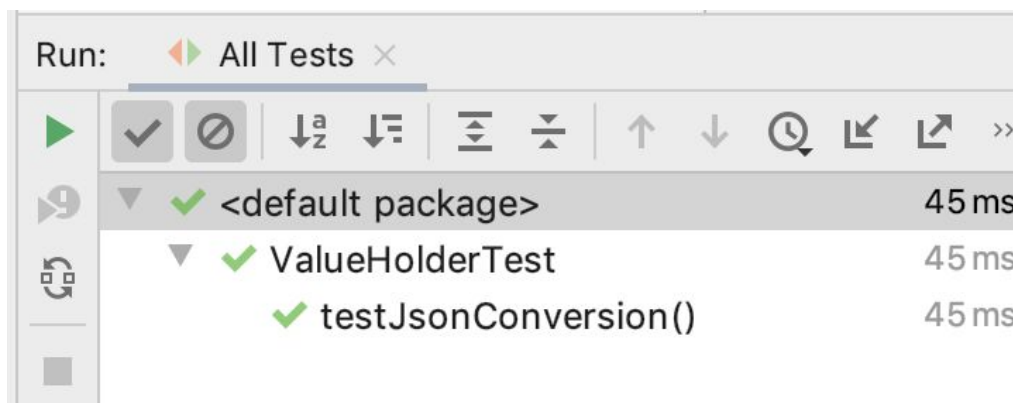
        assertEquals(valueHolder, valueHolderJson);
    }
}
```

15. You will likely see the following error (if not, continue to the next step):



This error occurs because there is a problem in the integration between the current version of IntelliJ and the latest version of Maven. To create a workaround for this issue, follow the steps under the “*Test Events Were Not Received*” section of the [“Setup and Test Running Issues in IntelliJ”](#) document to create a run configuration for your test and run it.

16. After running the test you should see something like this



17. Submit a screenshot of your build.gradle file and your test passing to LearningSuite or Canvas (whichever is being used for your class)

Lessons Learned

In this tutorial you learned how to manage dependencies using Maven and Gradle. In this simple lab we only installed 2 dependencies but in larger projects it is common to have several dependencies to external libraries.