

How to create Java Gradle project in Eclipse

qaautomation.expert/2021/05/02/how-to-create-java-gradle-project-in-eclipse/

vibssingh

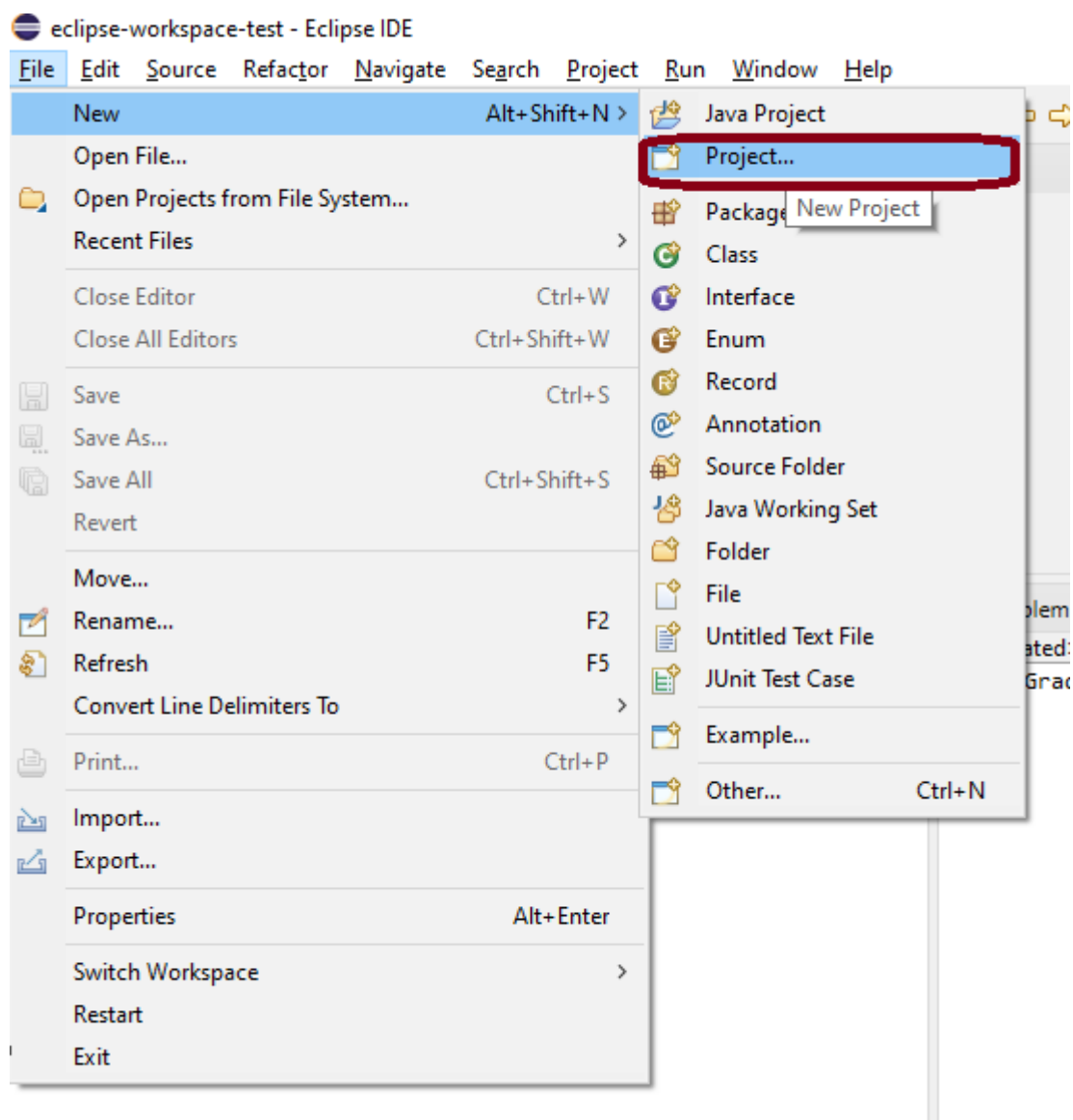
May 2, 2021

HOME

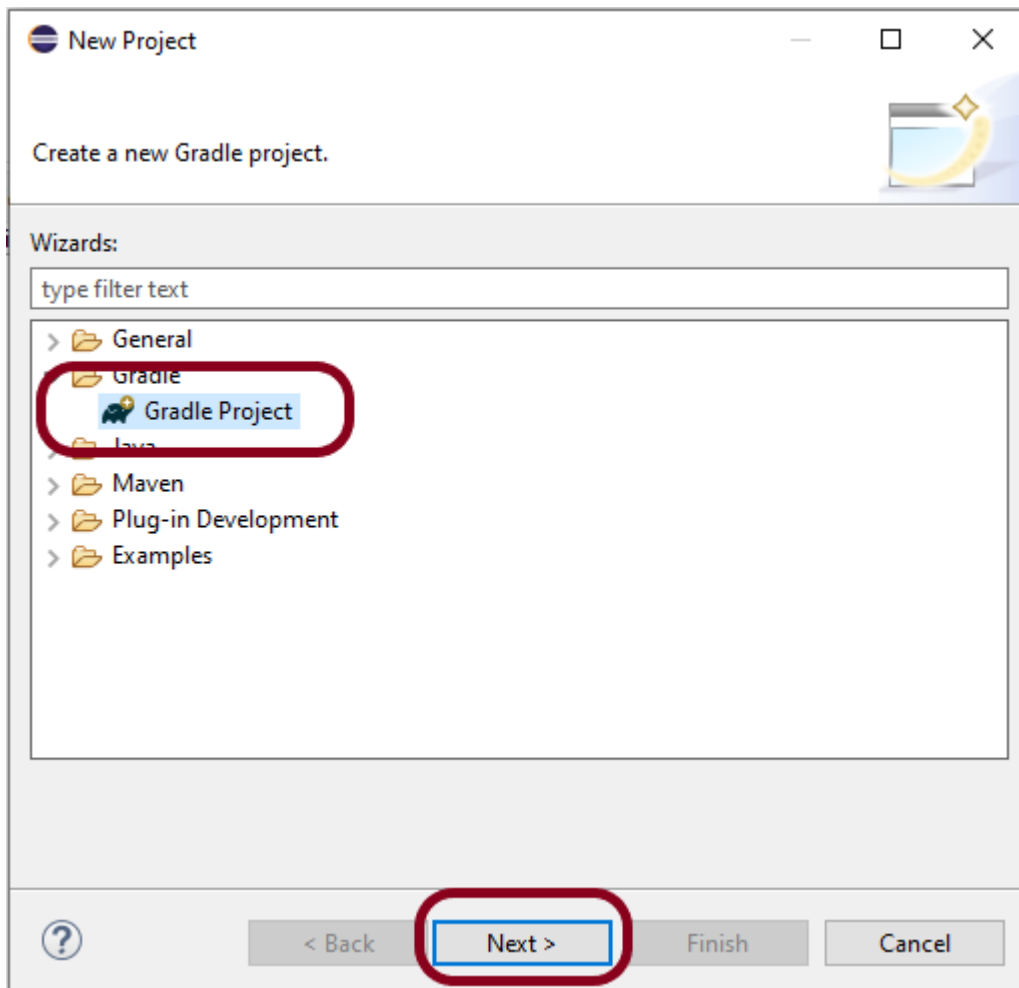
In the previous tutorial, I have explained [how to create a Java Gradle project in IntelliJ](#). In this tutorial, I will explain about creating a Java Gradle project Eclipse. I have used Gradle 6.6 to create the project.

Steps to follow:-

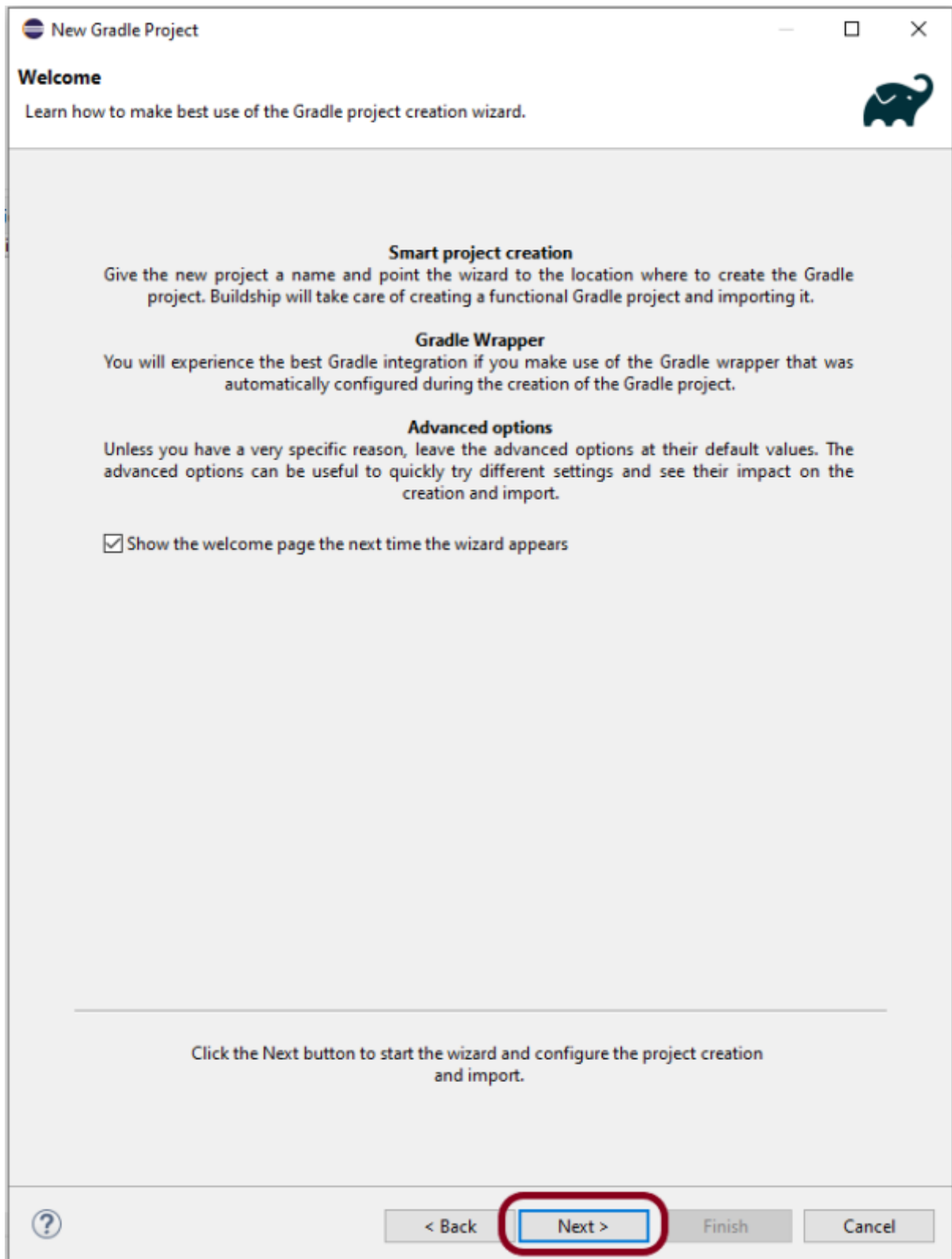
Step 1 – To create a new project – Click on the “**New**” and then select – “**Project**”.



Step 2 – Select the “**Gradle Project**” and click on the “**Next**” button.



Step 3 – A welcome screen will appear. You can uncheck the box – Show the welcome page the next time the wizard appears. This is optional. Click the **NEXT** button.



Step 4 – Below the screen will appear. Mention the “**Project Name – GradleEclipseDemo**”. Mention the location where we want to save the project in the system. Click the **NEXT** button.

New Gradle Project

New Gradle Project

Specify the name of the Gradle project to create.

Project name: GradleEclipseDemo

Project location

☐ Use default location

Location: C:\Users\long\OneDrive\Documents\Projects\Vibha\Projects\Vibha_Personal Browse...

Working sets

☐ Add project to working sets New...

Working sets: Select...

Click the Finish button to create the project and import it into the workspace. Click the Next button to select optional options.

< Back Next > Finish Cancel

Step 5 – Options screen appear. Make sure you use Gradle version 6.6 to create Gradle project in Eclipse for Version: 2021- 03 (4.19.0).

Note:- If you will try to use version higher than 6.6, then Gradle project structure will have a Gradle project with the nested project with a lib subproject in it.

New Gradle Project

Options

Specify optional options to apply when creating, importing, and interacting with the Gradle project.

☒ Override workspace settings [Configure Workspace Settings...](#)

Gradle distribution

☐ Gradle wrapper

☐ Local installation directory

☐ Remote distribution location

☒ Specific Gradle version

Advanced Options

Gradle user home

Java home

Program Arguments

JVM Arguments

☐ Offline Mode

☐ Build Scans

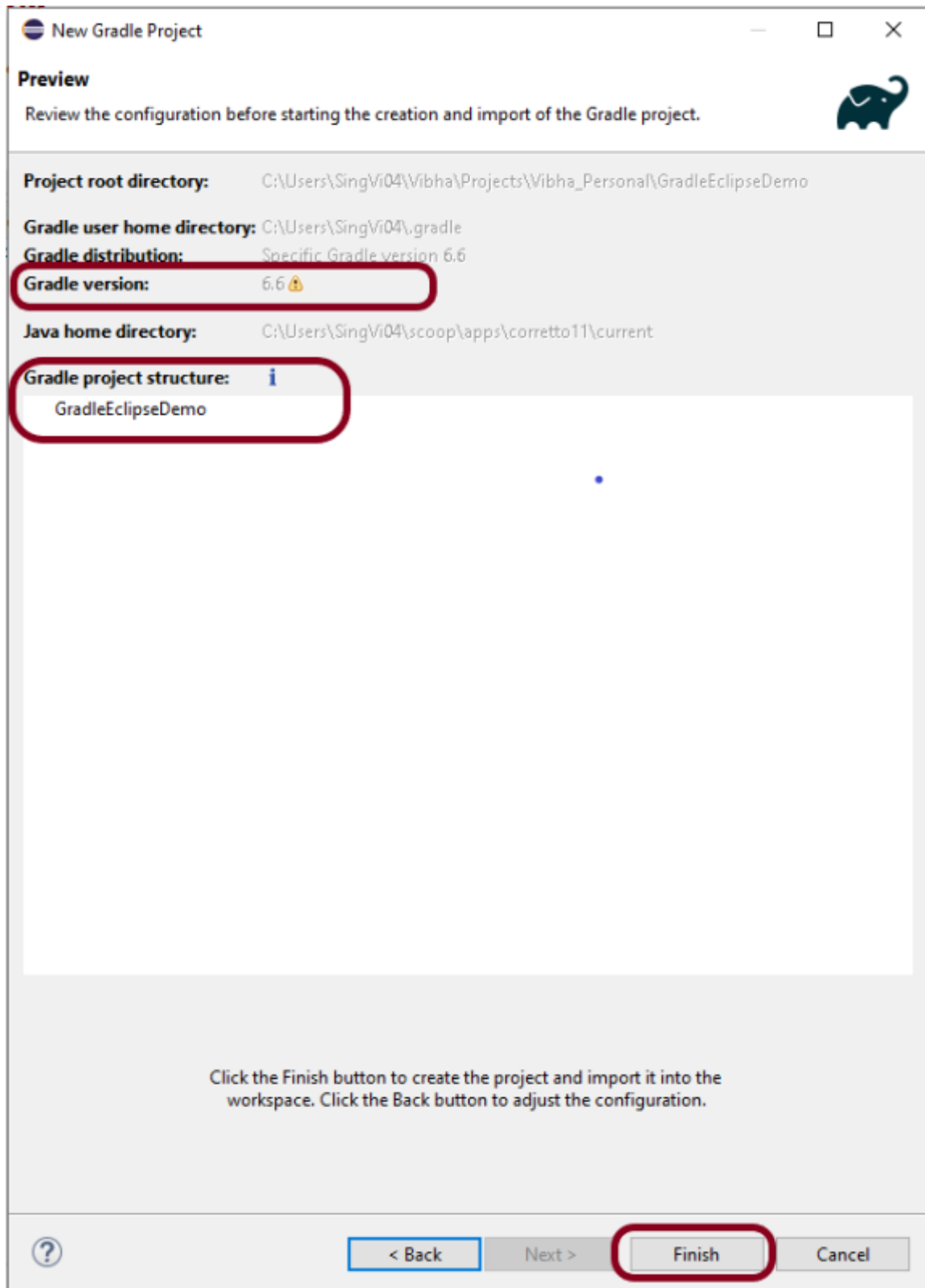
☐ Automatic Project Synchronization

☐ Show Console View

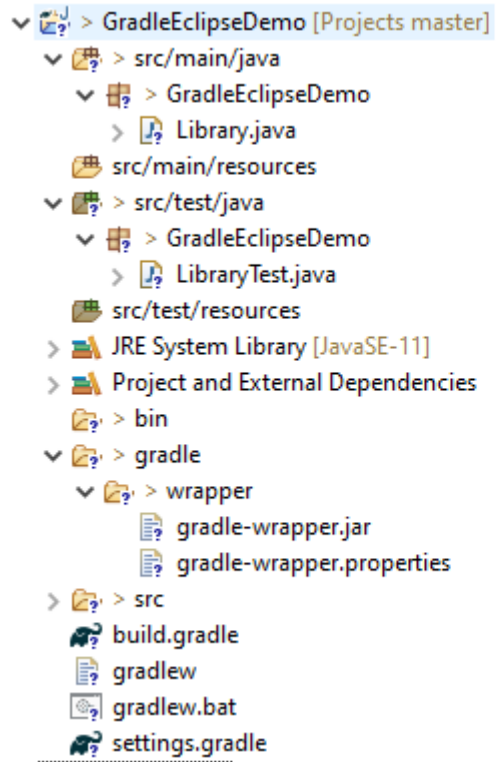
☐ Show Executions View

Click the Finish button to create the project and import it into the workspace. Click the Next button to see a summary of the configuration.

Step 6 – Verify the Gradle Version and Gradle project structure name.



Step 7 – Below is the structure of Gradle project. The `init` task generates the new project with the following structure:-



1. Generated folder for wrapper files -wrapper
2. Gradle wrapper start scripts – gradlew, gradlew.bat
3. Settings file to define build name and subprojects – settings.gradle
4. Build script of lib project – build.gradle
5. Default Java source folder – src/main/java
6. Default Java test source folder – src/test/java

Step 8 – Below is the structure and content of the **build.gradle**.

```

/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java Library project to get you started.
 * For more details take a look at the Java Libraries chapter in the Gradle
 * User Manual available at
https://docs.gradle.org/6.6/userguide/java\_library\_plugin.html
 */

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

repositories {
    // Use jcenter for resolving dependencies.
    // You can declare any Maven/Ivy/file repository here.
    jcenter()
}

dependencies {
    // This dependency is exported to consumers, that is to say found on their
    // compile classpath.
    api 'org.apache.commons:commons-math3:3.6.1'

    // This dependency is used internally, and not exposed to consumers on their
    // own compile classpath.
    implementation 'com.google.guava:guava:29.0-jre'

    // Use JUnit test framework
    testImplementation 'junit:junit:4.13'
}

```

1. **plugins** – Apply the java-library plugin for API and implementation separation.
2. **jcenter** – Use JCenter for resolving dependencies. JCenter is a central repository on JFrog Bintray platform for finding and sharing popular JVM language packages in Maven format
3. **api** – This dependency is exported to consumers, that is to say found on their compile classpath.
4. **implementation** – This dependency is used internally, and not exposed to consumers on their own compile classpath.
5. **testImplementation** – Use JUnit test framework.

Step 9 – To check if the project is created successfully. In gradle tasks tab -> navigate to the project -> expand build folder -> right click on build -> Select Run Gradle tasks.

Problems @ Javadoc Declaration Console Gradle Tasks	
Name	Description
▼ GradleEclipseDemo	
▼ build	
assemble	Assembles the outputs of this project.
build	Assembles and tests this project.
buildDependents	Assembles and tests this project and all projects that depend on it.
buildNeeded	Assembles and tests this project and all projects it depends on.
classes	Assembles main classes.
clean	Deletes the build directory.
jar	Assembles a jar archive containing the main classes.
testClasses	Assembles test classes.
▼ build setup	
init	Initializes a new Gradle build.
wrapper	Generates Gradle wrapper files.
▼ documentation	
javadoc	Generates Javadoc API documentation for the main source code.
▼ help	
buildEnvironment	Displays all buildscript dependencies declared in root project 'GradleEcli...
components	Displays the components produced by root project 'GradleEclipseDemo'....
dependencies	Displays all dependencies declared in root project 'GradleEclipseDemo'.
dependencyInsight	Displays the insight into a specific dependency in root project 'GradleEcli...
dependentComponents	Displays the dependent components of components in root project 'Gra...
help	Displays a help message.
model	Displays the configuration model of root project 'GradleEclipseDemo'. [i...
outgoingVariants	Displays the outgoing variants of root project 'GradleEclipseDemo'.

Name	Description
▼ GradleEclipseDemo	
▼ build	
assemble	Assembles the outputs of this project.
build	Assembles and tests this project.
build	Assembles and tests this project and all projects that depend on it.
build	Assembles and tests this project and all projects it depends on.
classes	Assembles main classes.
clean	Deletes the build directory.
jar	Assembles a jar archive containing the main classes.
testClasses	Assembles test classes.

- Run Gradle Tasks
- Open Gradle Run Configuration
- Run the selected tasks

This will be the output of the Gradle Run.

```
Working Directory: [redacted]\Vibha\Projects\Vibha_Personal\GradleEclipseDemo
Gradle user home: [redacted]\.gradle
Gradle Distribution: Specific Gradle version 6.6
Gradle Version: 6.6
Java Home: [redacted]\scoop\apps\corretto11\current
JVM Arguments: None
Program Arguments: None
Build Scans Enabled: false
Offline Mode Enabled: false
Gradle Tasks: build
```

```
> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE
> Task :jar
> Task :assemble
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
> Task :check
> Task :build

BUILD SUCCESSFUL in 1s
4 actionable tasks: 2 executed, 2 up-to-date
```

That's it. We have successfully created a Gradle Java project in Eclipse.

Congratulations on making it through this tutorial and hope you found it useful! Happy Learning!! Cheers!!