

# INTRODUCTION TO GRADLE

---

## **What is a gradle?**

- > open source
- > build automation tool
- > focused on flexibility and performance
- > Gradle build scripts are written in groovy or kotlin DSL(domain specific language).
- > Easier to maintain when compared to XMLs
- > It has become official build tool for android studio.
- > Android SDK tools mainly provide
  - > gradle android plugin
  - > Android studio as official tools
- > It's initial release was in the year 2007.

## **Why we need gradle?**

- > Highly customizable(we can customize based on the technologies, ex: if we are working on java and need gradle as build tool, then we have to use plugin java. And if we are using it in Android studio we need to use android plugin.) so this is the reason where the gradle can be applied to many of the programming languages hence it is highly customizable.
- > Fast(it has high performance, because, it uses the outputs from the previous executions, and processes only the inputs that are changed, and executes tasks parallel).
- > powerful(it is so powerful as it's been used by/ supports many popular languages and technologies).

## **Projects and tasks in gradle:**

- > There are mainly two basic concepts on gradle
  - > projects
  - > tasks
- > each of the gradle build is made up of one or more projects.

## **Gradle projects:**

- > project could be a jar or a web application

- > it could also be a ZIP file that contain the jars produced by other projects.
- > the gradle project could be deploying the application to production environment.
- > each and every project has one or more tasks.
- > each task are atomic(they have specific purpose)

#### **Gradle tasks:**

- > compiling class could be a task of a project
- > creating a jar could be a task
- > generating javadoc
- > publishing the archives into the repositories.

#### **Installing gradle and setting up in our system:**

##### **step-1:**

- > go to the link "<https://gradle.org/install/> "
- > scroll down to see "Installing manually" section
- > Under it choose "Binary-only".
- > then installation starts automatically

##### **step-2:**

- > once installed extract the zip file.
- > then goto control panel
- > goto system properties and select advance
- > in this select environment variable
- > click on new under user variables section, then give variable name as "GRADLE\_HOME"
- > and variable value as path where the gradle installed[note: you need to place the path expect opening the bin folder.
- > Then select "path" and click "edit" and click on "new"
- > add the path including the bin folder here.

## Screen shots of these steps:

### Step-1:

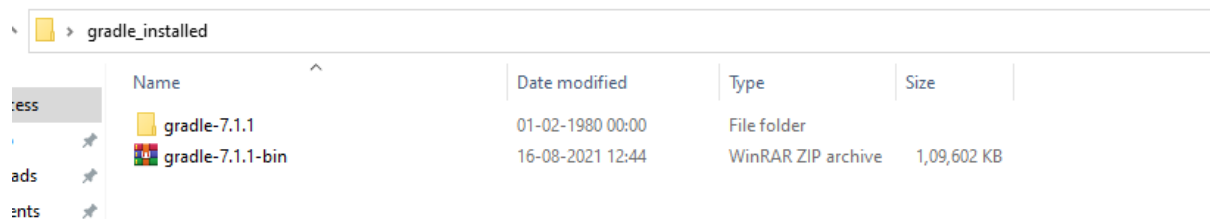
## Installing manually

### Step 1. [Download](#) the latest Gradle distribution

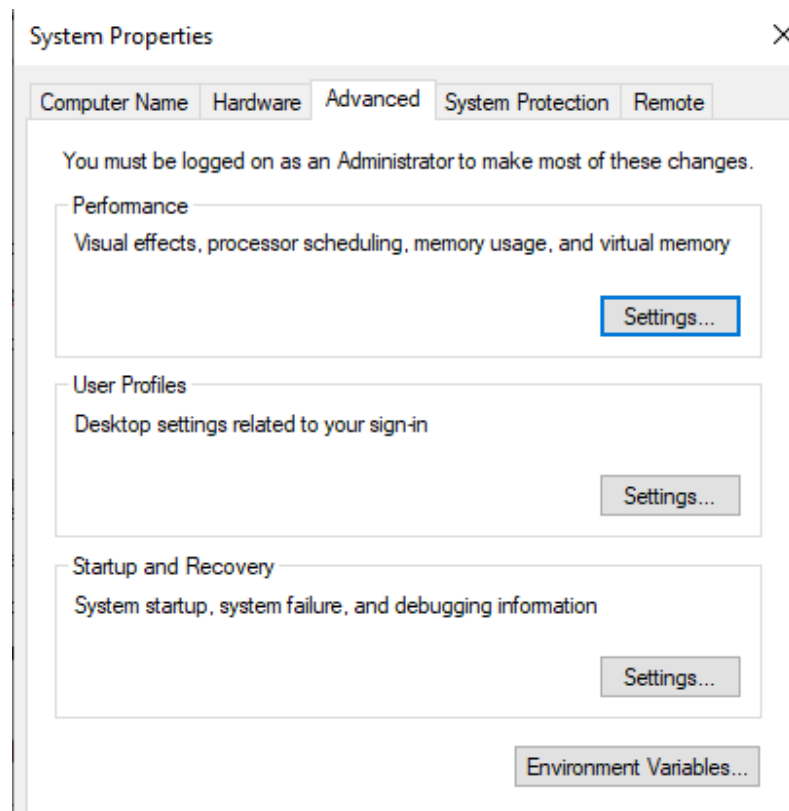
The current Gradle release is version 7.1.1, released on 02 Jul 2021. The distribution zip file comes in two flavors:

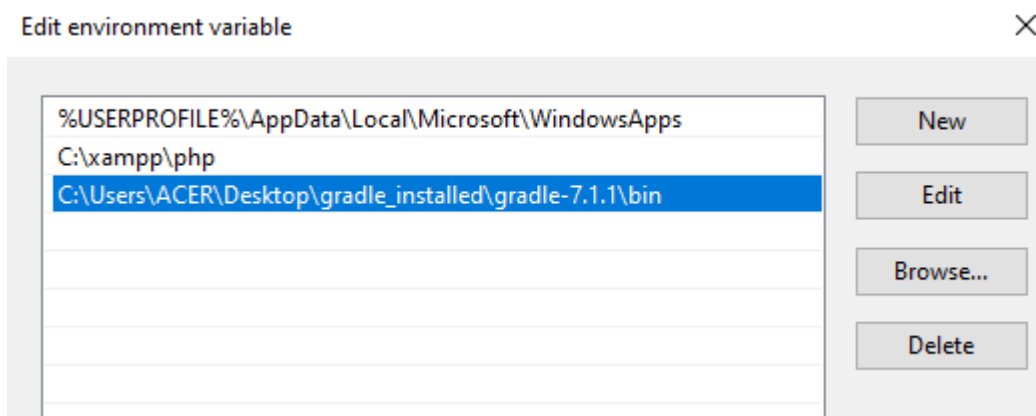
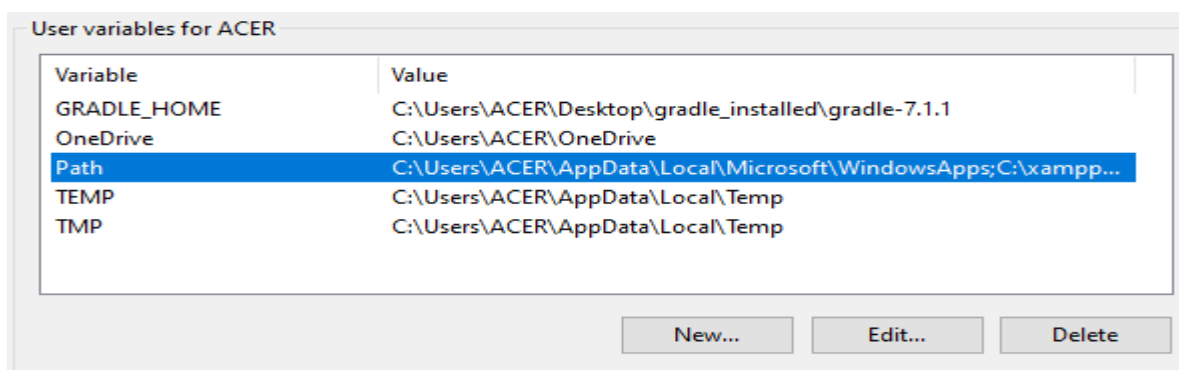
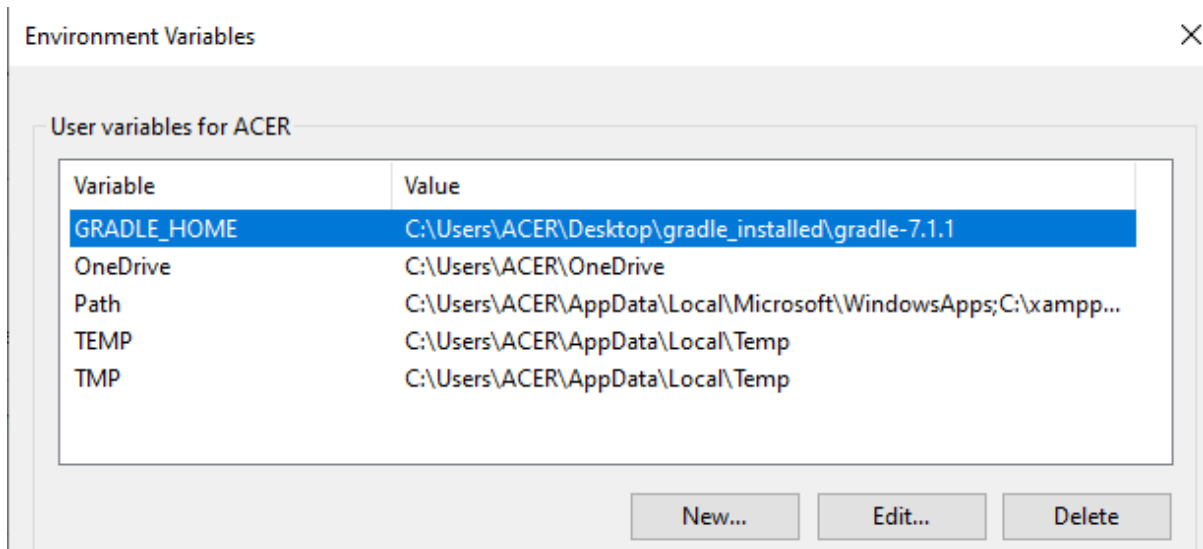
- [Binary-only](#)
- [Complete](#), with docs and sources

### Step-2:



Name	Date modified	Type	Size
gradle-7.1.1	01-02-1980 00:00	File folder	
gradle-7.1.1-bin	16-08-2021 12:44	WinRAR ZIP archive	1,09,602 KB





-> now everything is done.

### Adding gradle plugin in eclipse:

-> After setting everything on the system, you have to go to "c:" and inside which select "users" folder

within that select check that ".gradle" folder is generated automatically.

-> This is the "local repository" of gradle.[what all dependencies jar files that will be downloading will be stored

some where inside .gradle.

-> Then if we want gradle to be used in eclipse we have to add the gradle plugin inside the eclipse.

-> i.e the steps in adding gradle plugin inside the eclipse are:

step-1:

-> open eclipse

step-2:

-> goto "help" at the top

step-3:

-> select "eclipse marketplace" [NOTE: you need to have internet while this process needs to be carried out]

-> Then inside this goto find and type "gradle"

step-4:

-> you will be displayed with one plugin named "Buildship gradle integration"

-> all you need to do is install it by pressing on install button given below.

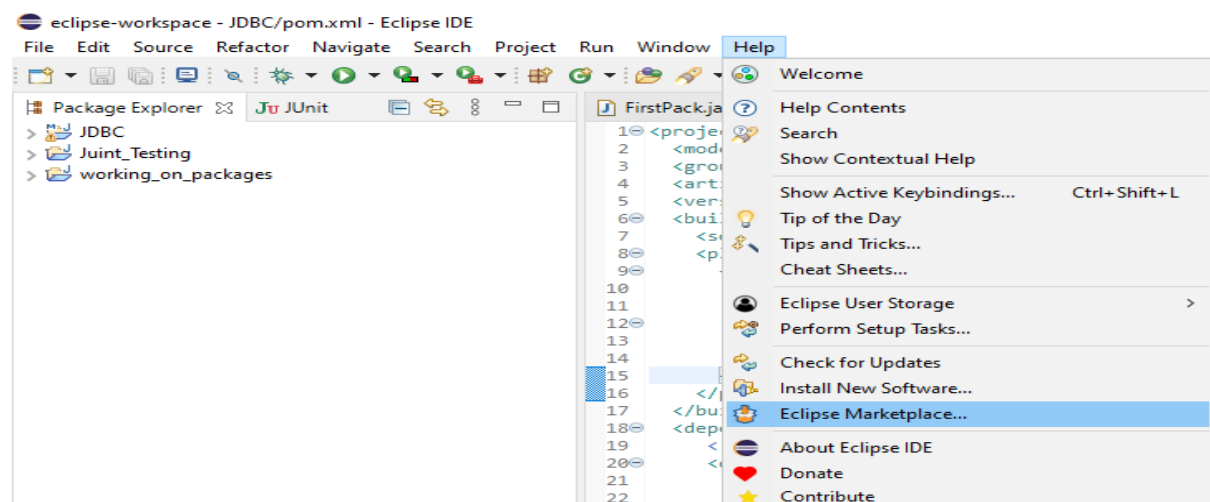
-> so now gradle plugin is incorporated inside the eclipse.

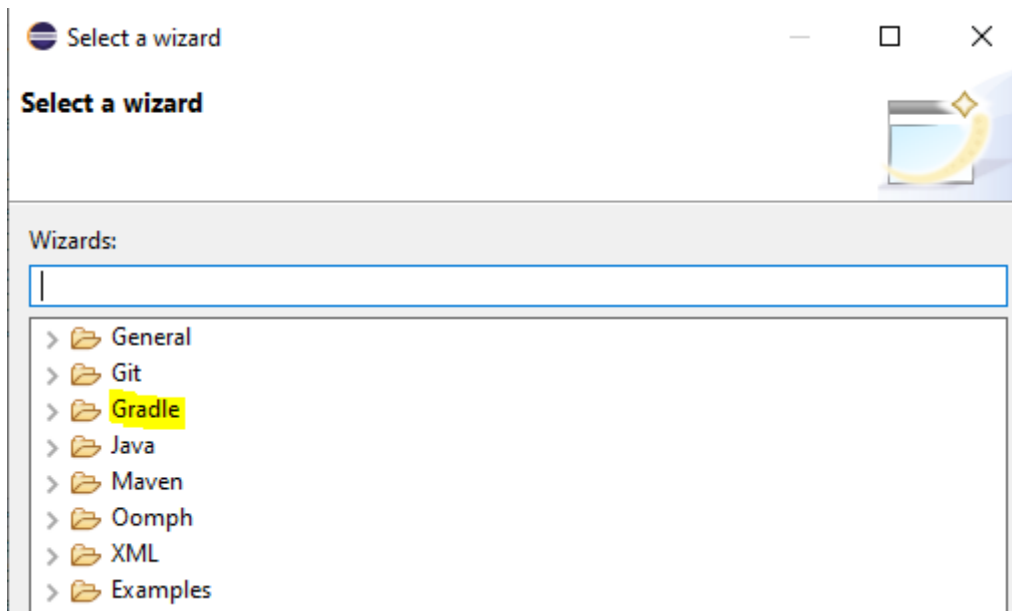
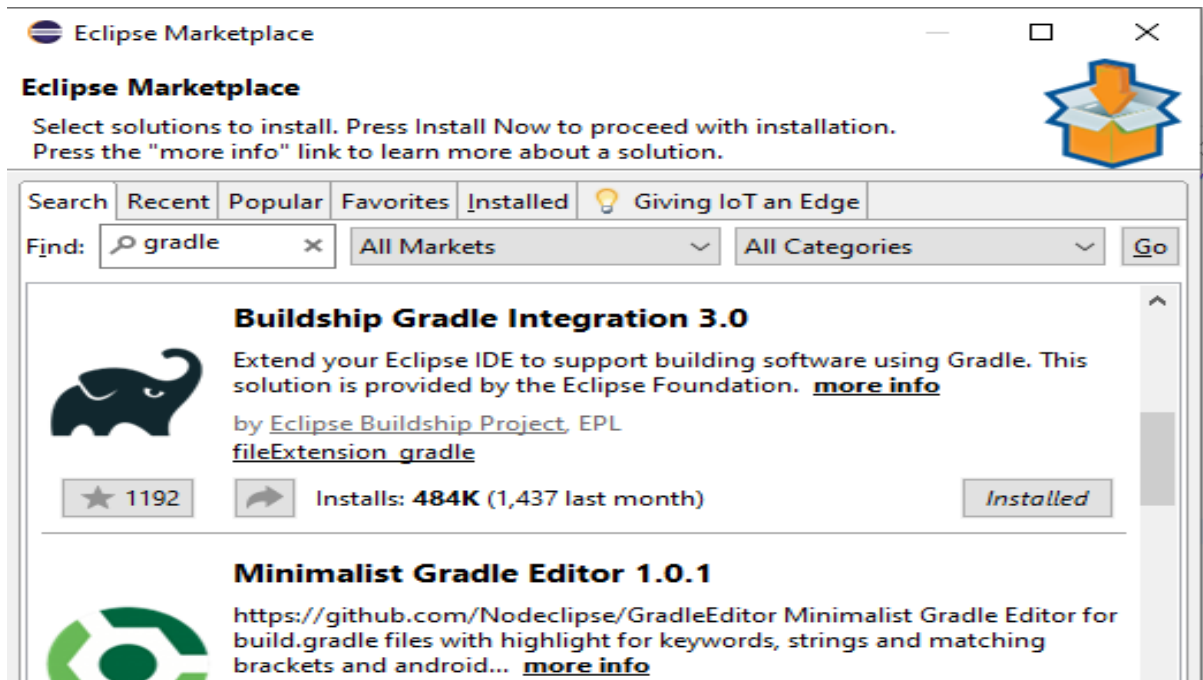
-> to check whether the gradle plugin is installed or not, goto "files" then select "new"

-> then select "others" you can see a wizard

-> that wizard should have the "Gradle" folder.

### Screen shots of the steps:





### Creating a gradle project:

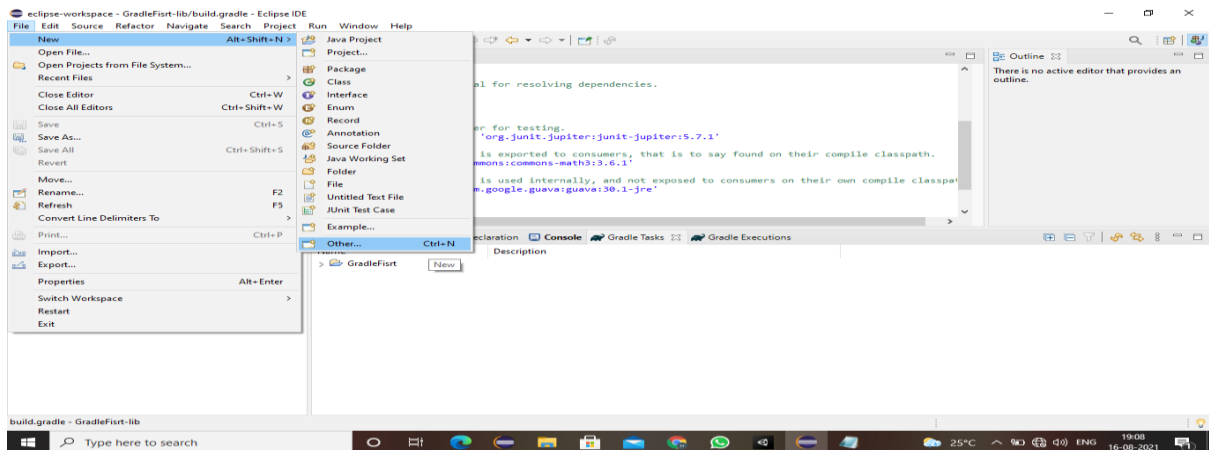
steps:

step-1:

-> goto "files"

-> select "new"

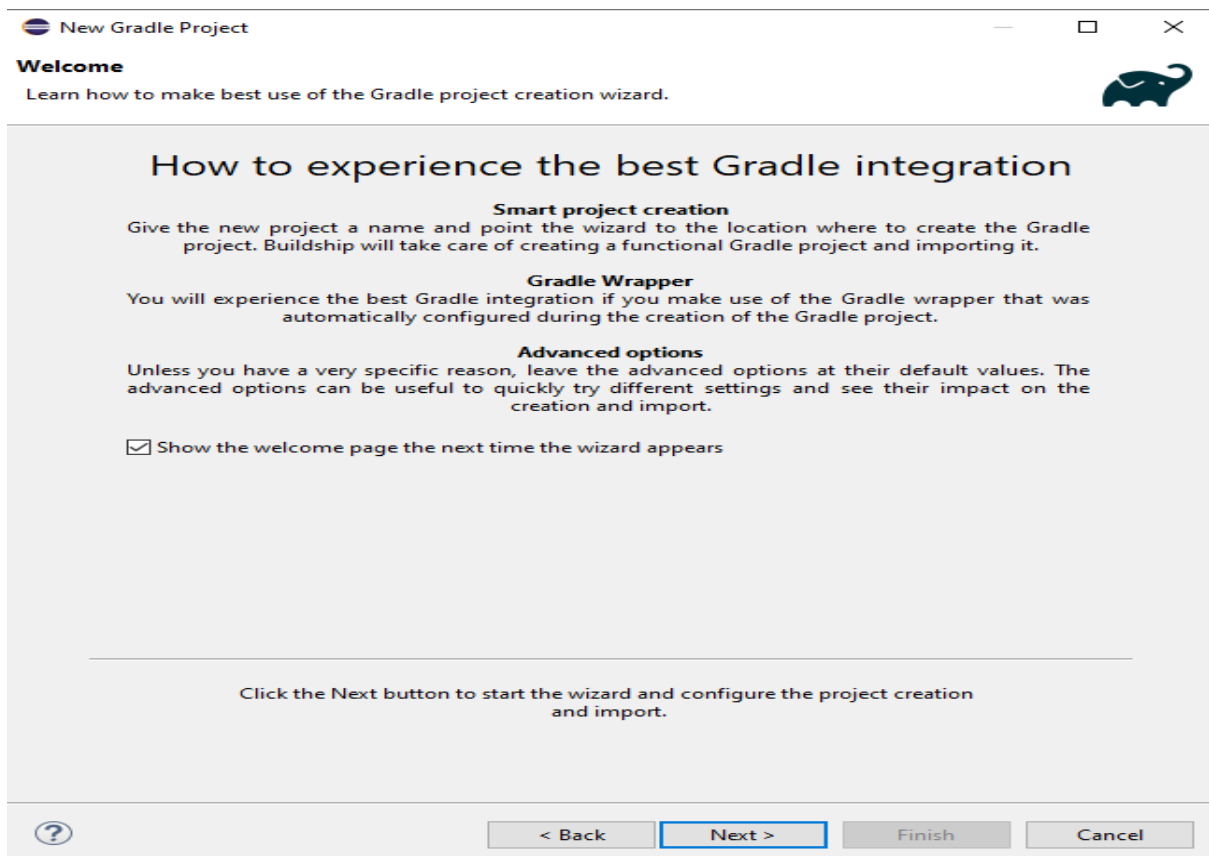
-> goto "others"



step-2:

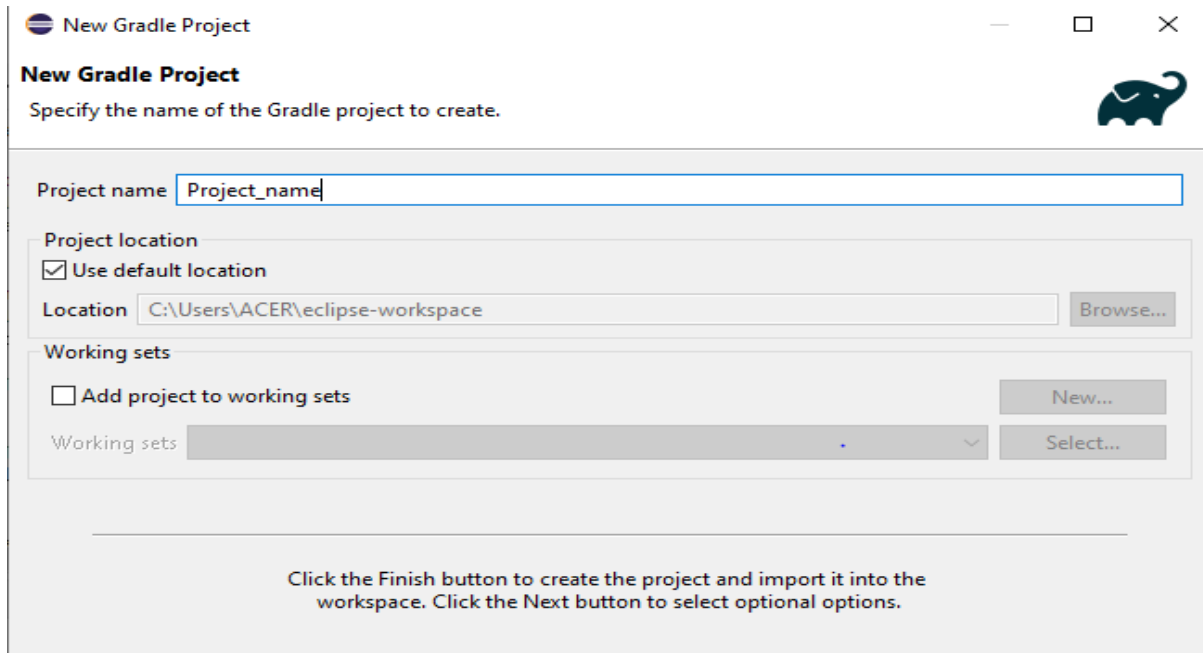
-> choose the "gradle folder" and click on "gradle project"

-> then you will see a wizard showing some information, click on next



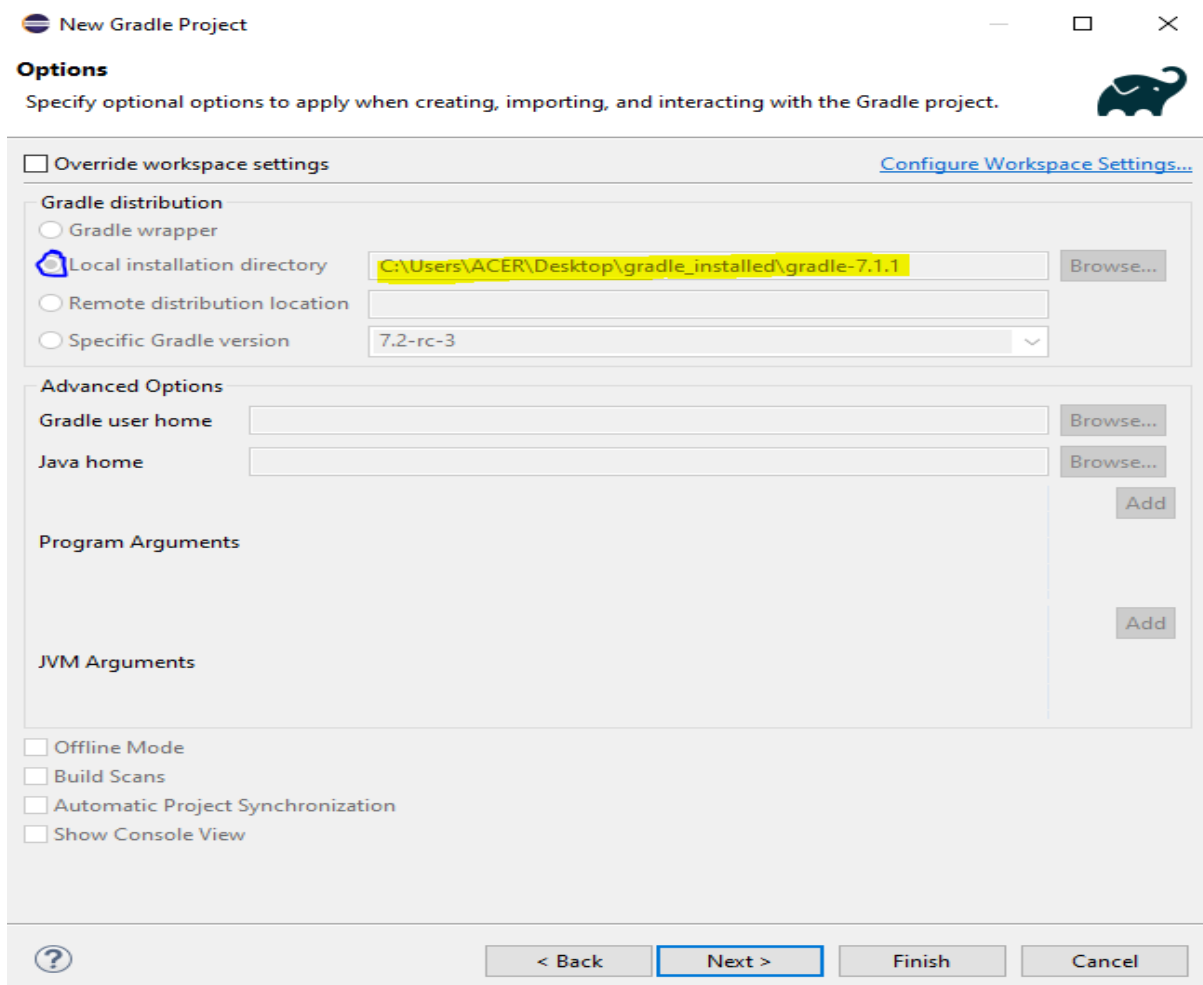
step-3:

-> give the project name and click on next



The 'New Gradle Project' dialog box is shown. It has a title bar with a minus, maximize, and close button. The title is 'New Gradle Project' with a Gradle elephant icon on the right. Below the title is the instruction 'Specify the name of the Gradle project to create.' The main area contains three sections: 'Project name' with a text field containing 'Project\_name'; 'Project location' with a checked 'Use default location' checkbox and a 'Location' text field containing 'C:\Users\ACER\eclipse-workspace' with a 'Browse...' button; and 'Working sets' with an unchecked 'Add project to working sets' checkbox, a 'Working sets' dropdown menu, and 'New...' and 'Select...' buttons. At the bottom, a message says: 'Click the Finish button to create the project and import it into the workspace. Click the Next button to select optional options.'

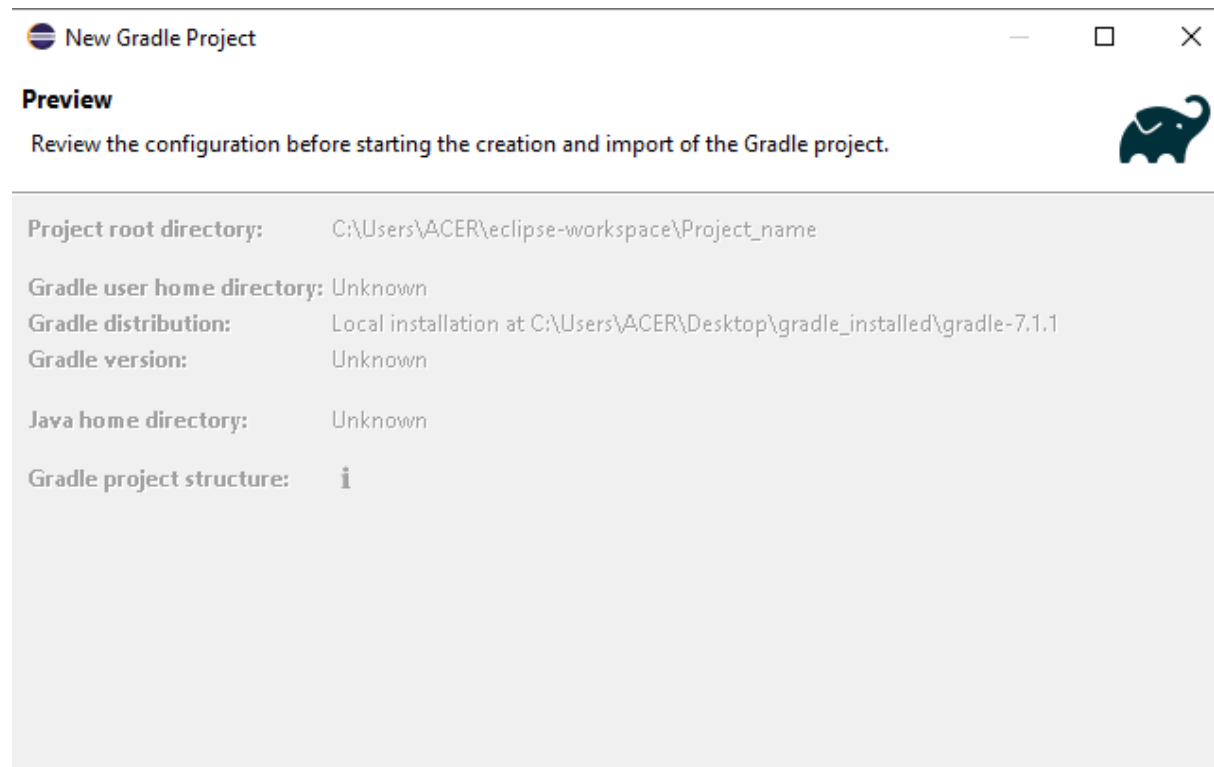
-> you will have to enter the option as shown in the image below



The 'Options' dialog box is shown. It has a title bar with a minus, maximize, and close button. The title is 'Options' with a Gradle elephant icon on the right. Below the title is the instruction 'Specify optional options to apply when creating, importing, and interacting with the Gradle project.' The main area contains several sections: 'Override workspace settings' with an unchecked checkbox and a 'Configure Workspace Settings...' link; 'Gradle distribution' with three radio buttons: 'Gradle wrapper' (unchecked), 'Local installation directory' (checked, with a text field containing 'C:\Users\ACER\Desktop\gradle\_installed\gradle-7.1.1' and a 'Browse...' button), 'Remote distribution location' (unchecked), and 'Specific Gradle version' (unchecked, with a text field containing '7.2-rc-3' and a dropdown arrow); 'Advanced Options' with 'Gradle user home' and 'Java home' text fields and 'Browse...' buttons, and 'Program Arguments' and 'JVM Arguments' sections with 'Add' buttons; and a bottom section with four unchecked checkboxes: 'Offline Mode', 'Build Scans', 'Automatic Project Synchronization', and 'Show Console View'. At the bottom of the dialog are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'.



-> after this you will be shown with the preview before the project is built



-> after this you will see a project that is created and visible on the eclipse.

### Gradle tasks:

Default gradle task-> the tasks that gradle is already providing.

we can also write our own customised tasks.

so we have two types of tasks:

1. default tasks
2. custom task

Default gradle tasks in gradle.

-> open cmd

-> type the command gradle tasks

-> when enter is pressed the gradle will list down all the tasks available in gradle.

-> [NOTE: before typing the command "gradle tasks" you basically have to specify the root project  
i.e it expects the settings.gradle file to be present in order to display the default or built in tasks]

```

C:\Users\ACER>cd GradleFisrt

C:\Users\ACER\GradleFisrt>gradle tasks

> Task :tasks

-----
Tasks runnable from root project 'GradleFisrt'
-----

Build Setup tasks
-----
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Help tasks
-----
buildEnvironment - Displays all buildscript dependencies declared in root project 'GradleFisrt'.
dependencies - Displays all dependencies declared in root project 'GradleFisrt'.
dependencyInsight - Displays the insight into a specific dependency in root project 'GradleFisrt'.
help - Displays a help message.
javaToolchains - Displays the detected java toolchains.
outgoingVariants - Displays the outgoing variants of root project 'GradleFisrt'.
projects - Displays the sub-projects of root project 'GradleFisrt'.
properties - Displays the properties of root project 'GradleFisrt'.
tasks - Displays the tasks runnable from root project 'GradleFisrt' (some of the displayed tasks may belong to subprojects).

To see all tasks and more detail, run gradle tasks --all

To see more detail about a task, run gradle help --task <task>

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
C:\Users\ACER\GradleFisrt>

```

This is how it basically looks when you type the command “gradle tasks”.

- I also made a point that we get an build failed exception i.e shown below:

```

C:\Users\ACER>gradle tasks
Starting a Gradle Daemon, 1 stopped Daemon could not be reused, use --status for details

FAILURE: Build failed with an exception.

* What went wrong:
Executing Gradle tasks as part of a build without a settings file is not supported. Make sure that you are executing Gradle from a directory within your Gradle project.
Your project should have a 'settings.gradle(.kts)' file in the root directory.

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run with --scan to get full insights.

* Get more help at https://help.gradle.org

BUILD FAILED in 17s

```

- To know the complete tasks in the gradle we need to use the command “ gradle tasks –all”.
- This will display all the tasks that are default in the gradle and i.e is shown below.

```

Build Setup tasks
-----
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Help tasks
-----
buildEnvironment - Displays all buildscript dependencies declared in root project 'GradleFisrt'.
lib:buildEnvironment - Displays all buildscript dependencies declared in project ':lib'.
dependencies - Displays all dependencies declared in root project 'GradleFisrt'.
lib:dependencies - Displays all dependencies declared in project ':lib'.
dependencyInsight - Displays the insight into a specific dependency in root project 'GradleFisrt'.
lib:dependencyInsight - Displays the insight into a specific dependency in project ':lib'.
help - Displays a help message.
lib:help - Displays a help message.
javaToolchains - Displays the detected java toolchains.
lib:javaToolchains - Displays the detected java toolchains.
outgoingVariants - Displays the outgoing variants of root project 'GradleFisrt'.
lib:outgoingVariants - Displays the outgoing variants of project ':lib'.
projects - Displays the sub-projects of root project 'GradleFisrt'.
lib:projects - Displays the sub-projects of project ':lib'.
properties - Displays the properties of root project 'GradleFisrt'.
lib:properties - Displays the properties of project ':lib'.
tasks - Displays the tasks runnable from root project 'GradleFisrt' (some of the displayed tasks may belong to subprojects).
lib:tasks - Displays the tasks runnable from project ':lib'.

Other tasks
-----
components - Displays the components produced by root project 'GradleFisrt'. [deprecated]
lib:components - Displays the components produced by project ':lib'. [deprecated]
dependentComponents - Displays the dependent components of components in root project 'GradleFisrt'. [deprecated]
lib:dependentComponents - Displays the dependent components of components in project ':lib'. [deprecated]
model - Displays the configuration model of root project 'GradleFisrt'. [deprecated]
lib:model - Displays the configuration model of project ':lib'. [deprecated]
prepareKotlinBuildScriptModel

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
C:\Users\ACER\GradleFisrt>

```

### Custom gradle tasks:

- We can specify our own tasks to be executed apart from the default tasks.
- To do that we have to go to the eclipse and create a gradle project (use the guide lines to create the gradle project given above in the same doc.).
- Once the project is built go to the build.gradle file.
- You can see there are different sections like plugin, repository and dependencies.
- Where plugin are used to specify the plugin for the language that we want to use the gradle i.e say if we want to use gradle for java then we need to add the java-libraries plugin inside the plugin section. i.e

```
plugins {  
    // Apply the java-library plugin for API and implementation separation.  
    id 'java-library'  
}
```

- Then repository section hold stuff, from where the dependencies are downloaded i.e

```
repositories {  
    // Use Maven Central for resolving dependencies.  
    mavenCentral()  
}
```

- Then dependencies section, we use this section to add certain dependencies that our project may require

```
dependencies {  
    // Use JUnit Jupiter for testing.  
    testImplementation 'org.junit.jupiter:junit-jupiter:5.7.1'  
  
    // 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 classp  
    implementation 'com.google.guava:guava:30.1-jre'  
}
```

### Now we will see how to write our own task:

- So if you are writing your task, note you need to write in the build.gradle file only.
- The syntax for to write the custom task is:  
task task\_name{  
 group "group\_name" //specifies under this group name the task need to be/shown.  
 Description "write the description if the task"  
 doLast{  
 //write the logic that is to be performed i.e the task you need to perform as customized  
 }  
}
- Example : I have written a task that just prints some statement

```

task customTask{
    group "custom"
    description "this task prints the task running"
    doLast{
        println " This is my custom task running"
    }
}

```

- So after wiring the task we need to run the task.
- Then command to run the task is "gradle task\_name".

```

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed

> Task :lib:customTask
  This is my custom task running
BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
C:\Users\ACER\eclipse-workspace\GradleFisrt>

```

Now let's have a discussion on the topic **COPY TASK**:

- Here we will write a task that copies the file from one folder to another.
- So how it is done?
- Here is the syntax:  
task task\_name(type : copy){  
from "specify the location of the file"  
to "specify where the file needs to be copied"  
}
- Example:

```

task copyFile(type: Copy){
    from 'D:\\copytaskcopied\\CopytaskFile.txt'
    into 'D:\\'
}

```

- So when this task is made run [NOTE: command to run task: "gradle copyFile" in case of above example]
- So here the most important parameters are:  
type: Copy // this specifies that we are performing copy operation  
from // this specifies from where the file needs to be copied  
into // this specifies to where the file needs to be copied
- The output:

```

C:\Users\ACER\eclipse-workspace\GradleFisrt>gradle copyFile

BUILD SUCCESSFUL in 12s
1 actionable task: 1 executed
C:\Users\ACER\eclipse-workspace\GradleFisrt>

```

### doFirst and doLast:

- Now we will discuss some simple topics named doFirst and doLast.
- So by the name itself it say, doFirst indicates you need to do this stuff first
- And doLast indicates you need to perform that particular task at the last
- Let us see this with an example:

```
task doFirstanddoLast{  
  
    doFirst{  
        println " hello, this is from doFirst"  
    }  
  
    doLast{  
        println "this is from doLast"  
    }  
}
```

#### Output:

```
> Task :lib:doFirstanddoLast  
hello, this is from doFirst  
this is from doLast  
  
BUILD SUCCESSFUL in 8s  
1 actionable task: 1 executed  
C:\Users\ACER\eclipse-workspace\GradleFisrt>_
```

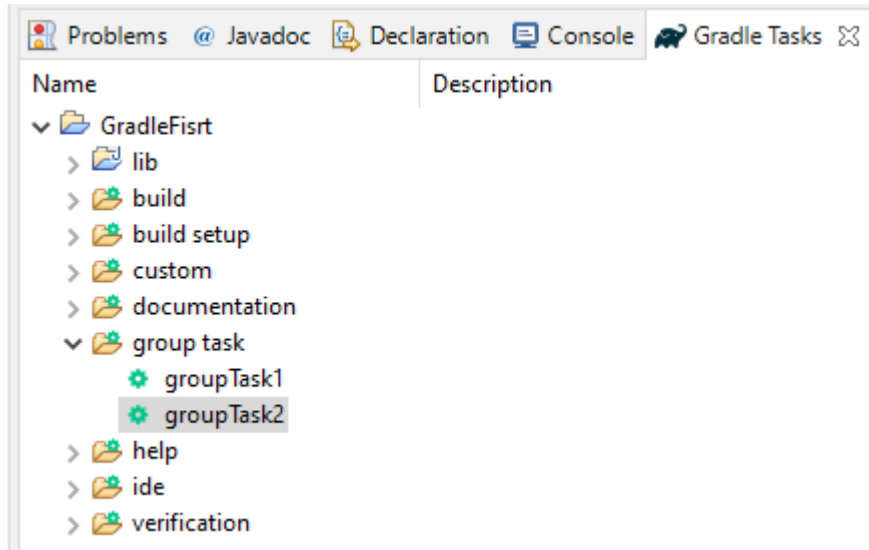
### Grouping tasks:

- So here we will see how we can group different tasks under same group.
- This is basically done by giving same group name.
- Example for this:

```
task groupTask1{  
    group "Group Task"  
    doLast{  
        println "This is from group task1"  
    }  
}  
  
task groupTask2{  
    group "Group Task"  
    doLast{  
        println "This is from group task2"  
    }  
}
```

#### Output:

```
Group Task tasks  
-----  
groupTask1  
groupTask2
```



- Make sure that in projects the build automatically is ticked.
- Then once the above tasks are written you can see the groups in the gradle tasks section.

### Skipping Task:

- Here we are going to talk about skipping some tasks.
- Where say we want to skip some task, then this option is also available in gradle.
- Let us see an example:

```
task skipping{
    group "skipping example"
    doLast{
        println "this is skipping task"
    }
}
skipping.onlyIf{
    project.hasProperty('skip')
}
```

- In the above example we can see how we can skip some tasks.
- Where we have a task named skipping which is placed under the group skipping example.
- And the task will be skipped if the program argument is not specified
- In command prompt we execute this type of tasks as follows:

```
C:\Users\ACER\eclipse-workspace\GradleFisrt>gradle skipping -P skip

> Task :lib:skipping
this is skipping task

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
C:\Users\ACER\eclipse-workspace\GradleFisrt>
```

- Where -P stands for property and it should be followed by the property name.
- Here the property name given is skip.

### Task dependency:

- Here we will be discussing about how dependencies are been handled.
- Say for example:

```
task usedTask{
    group "dependencyTask"
    doLast{
        println "This is the used task"
    }
}
task userTask(dependsOn: 'usedTask'){
    group "dependencyTask"
    doLast{
        println "This is user task"
    }
}
```

- In the above example we have two tasks, one is the userTask and other is usedTask, where the userTask is dependent on the usedTask.
- This is specified in the above example using [dependsOn: "task\_name"]
- Which basically mean that before executing the usedTask the userTask cannot be.
- So if you try to execute the userTask it will first execute the usedTask and then will execute the userTask.

### Output:

```
C:\Users\ACER\eclipse-workspace\GradleFisrt>gradle userTask

> Task :lib:usedTask
This is the used task

> Task :lib:userTask
This is user task

BUILD SUCCESSFUL in 2s
2 actionable tasks: 2 executed
C:\Users\ACER\eclipse-workspace\GradleFisrt>
```

```
> Task :lib:usedTask
This is the used task

> Task :lib:userTask
This is user task

BUILD SUCCESSFUL in 662ms
2 actionable tasks: 2 executed
```

### Zippping task:

- In this task what we need to do is, we have to choose a folder that is to be converted as zip file.
- And then we need to specify the destination directory to where the zipped file has to stored.
- And we also have to define the name of the zipped file.
- So let us see this with an example:

```

task zippin(type: Zip){
    group "zip"
    from "D:\\copytaskcopied"
    destinationDir = file("D:\\zippedTask")
    archiveName = "zipTask.zip"
}

```

## Output:

```

C:\Users\ACER\eclipse-workspace\GradleFisrt>gradle zippin
Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.
You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.
See https://docs.gradle.org/7.1.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 3s
1 actionable task: 1 up-to-date
C:\Users\ACER\eclipse-workspace\GradleFisrt>

```

copytaskcopied	18-08-2021 18:10	File folder	
lab programs cg	06-07-2021 14:42	File folder	
zippedTask	18-08-2021 21:57	File folder	

PC > New Volume (D:) > zippedTask

Name	Date modified	Type	Size
zipTask	18-08-2021 21:57	WinRAR ZIP archive	1 KB