**Hands On Lab 1**

**Java Nio Create Directory**

Java NIO (i.e. new I/O) is an interesting file input-output mechanism introduced in Java 5 and provides different ways of working with input-output operations than standard input-output API’s. Java NIO supports a buffer-oriented, channel-based approach for the I/O operations and with the introduction of Java 7, the NIO system has expanded thereby providing the enhanced support for the file system features and the file handling mechanism. In this Hands On Lab, we will try to get an overview of what Java NIO is and work through the create directory operation.

### 2.1 Tools Used

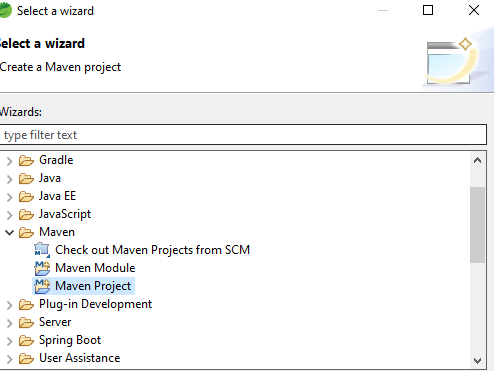
We are using Spring Tool Suite 4, JDK 11 and Maven.

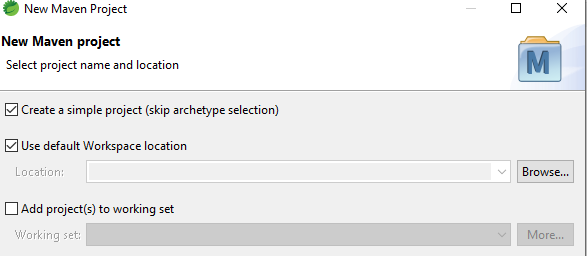
### 2.2 Project Structure

Firstly, let’s review the final project structure, in case you are confused about where you should create the corresponding files or folder later!

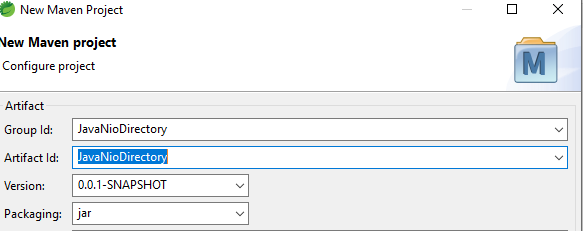
### 2.3 Project Creation

This section will demonstrate on how to create a Java-based Maven project with Eclipse. In Eclipse IDE, go to File -> New -> Other -> Maven Project.

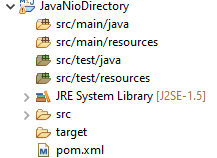


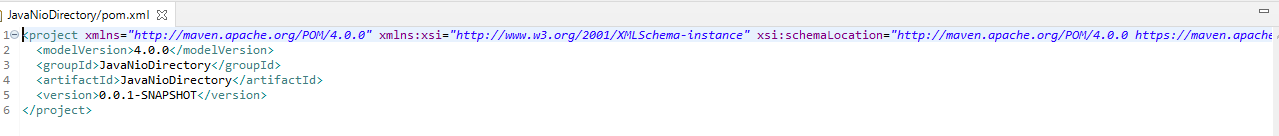
In the New Maven Project window, it will ask you to select project location. By default, ‘Use default workspace location’ will be selected. Select the ‘Create a simple project (skip archetype selection)’ checkbox and just click on next button to proceed.  
  


It will ask you to ‘Enter the group and the artifact id for the project’. We will input the details as shown in the below image. The version number will be by default: 0.0.1-SNAPSHOT.



Click on Finis. Observe that it has downloaded the maven dependencies and a pom.xml file has been created. It should have the following code:





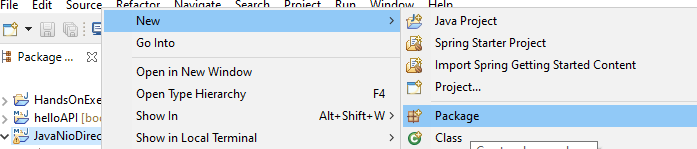
Developers can start adding the dependencies that they want like JUnit etc. Let’s start building the application!

## 3. Application Building

Below are the steps involved in developing this application.

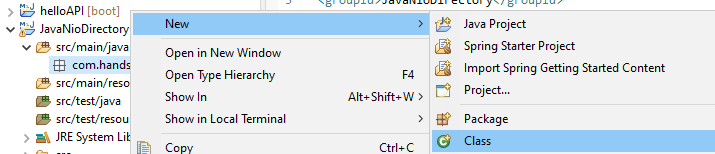
### 3.1 Java Class Creation

Let’s create the required Java files. Right-click on src/main/java folder, New -> Package.

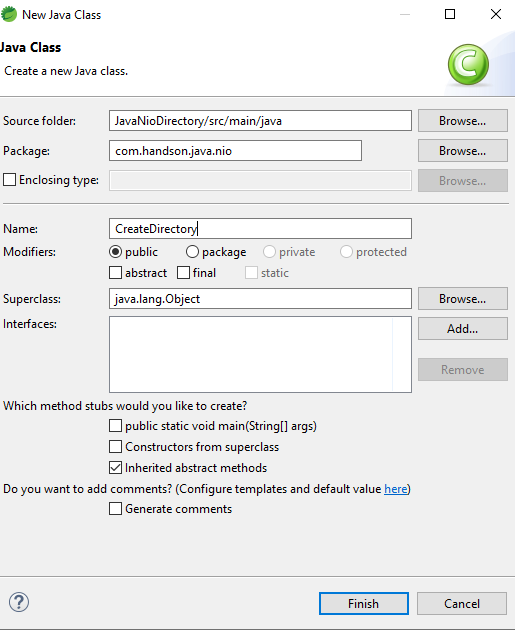


A new pop window will open where we will enter the package name as: com.handson.java.nio.

Once the package is created in the application, we will need to create the implementation class. Right-click on the newly created package: New -> Class.



A new pop window will open and enter the file name as: CreateDirectory. The implementation class will be created inside the package: com.handson.java.nio.

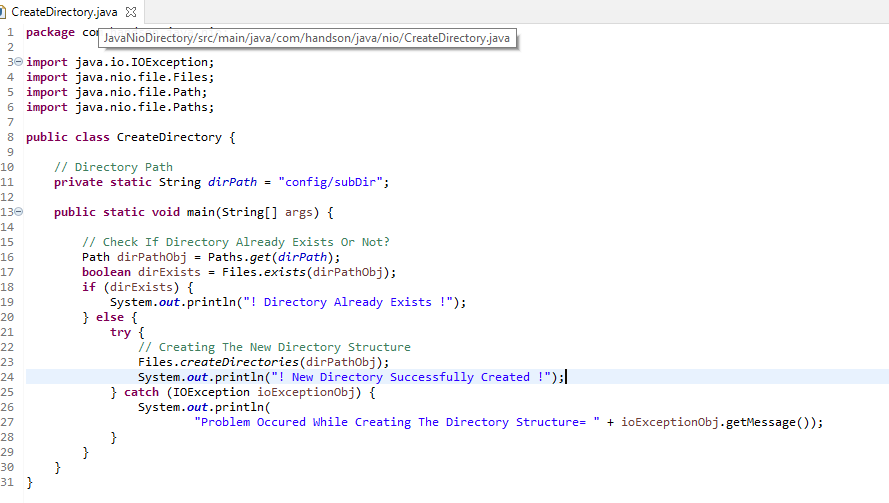


#### 3.1.1 Implementation of Utility Class

The Files.createDirectory() method creates a new directory from the Path instance. If creating the directory succeeds, a Path instance is returned which points to the newly created path. If the directory already exists, a java.nio.file.FileAlreadyExistsException will be thrown and if something else goes wrong, an IOException may get thrown. For e.g., if the parent directory of the desired new directory does not exist, an IOException may get thrown.

Do note, the parent directory is the directory in which developers want to create the new directory. Add the following code to it:

*CreateDirectory.java*

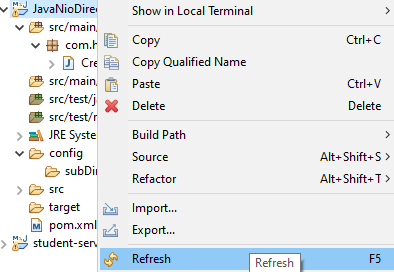


## 4. Run the Application

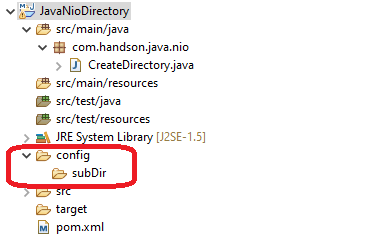
To run the Java Nio application, Right-click on the CreateDirectory class -> Run As -> Java Application. Developers can debug the example and see what happens after every step!

## 5. Project Demo

Right click on your project and click Refresh.



In the above code, we have used the Files.createDirectories() method to create a new directory in the project and the code creates the config/subDir directories in the project after execution.



## 6. Conclusion

This Hands On Lab uses a simple example to illustrate the functionality of the Java Nio package and helps developers understand the basic configuration required to use the create directory operation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*