Practical 2

Disclaimer

These examples are just to show how to use Java Streams and Java Sockets. It is not the exact way that these applications are implemented.

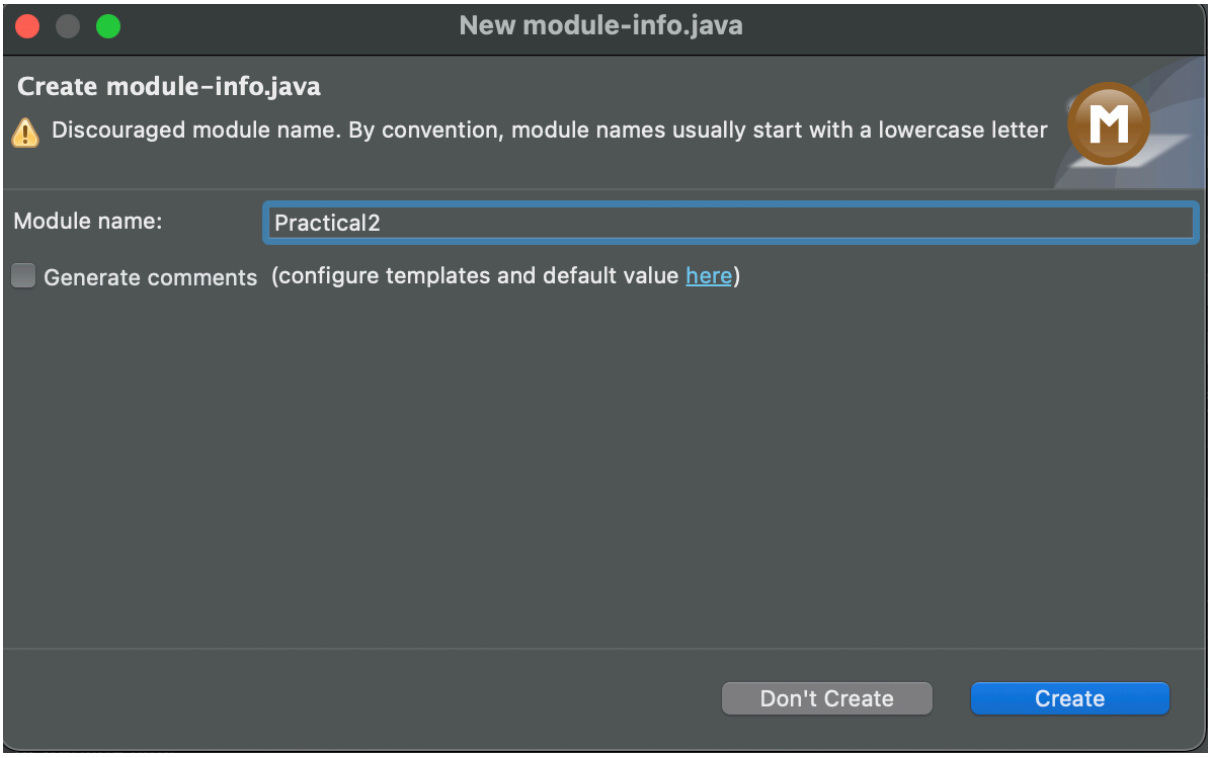
Project 1 - Copy Paste File

Description

This is an example application where it allows user to copy a selected file and paste to a target location using Java Binary Input Output.

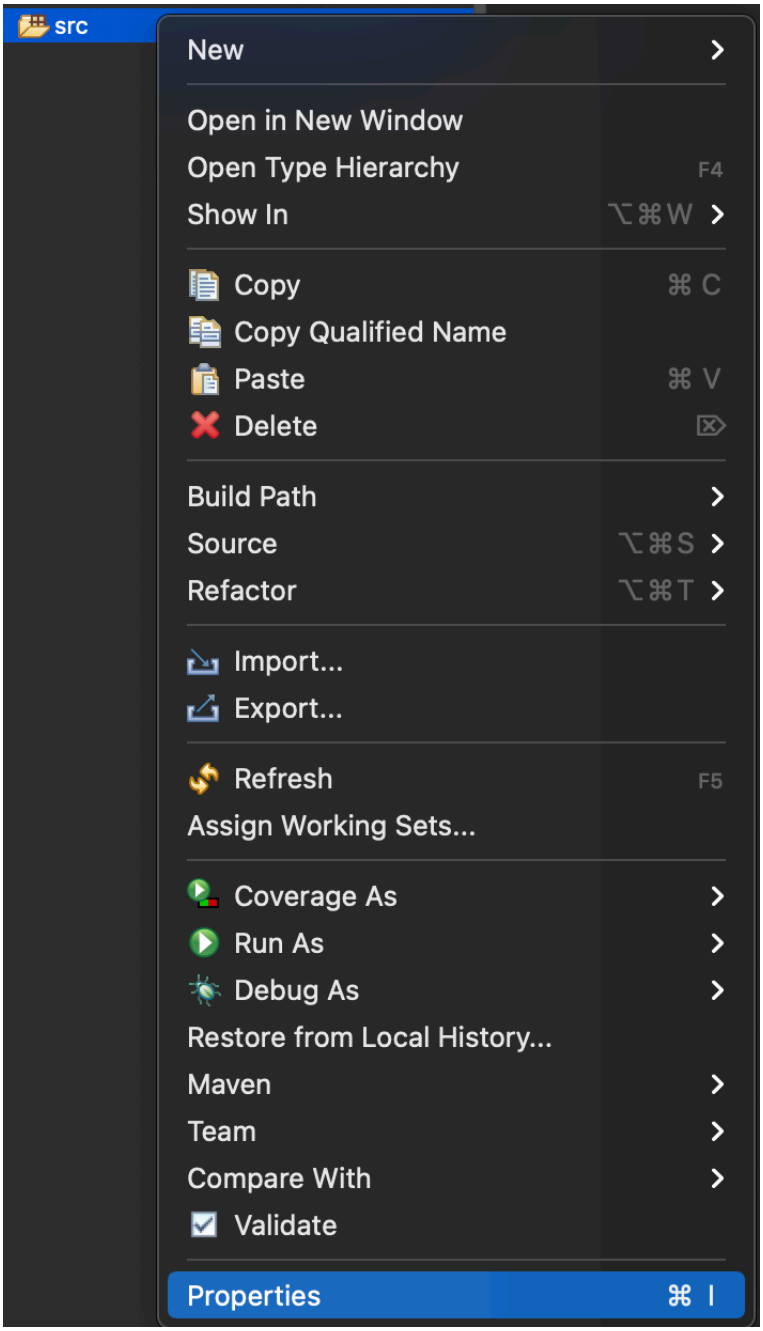
Lesson 1.1 - Setting up Java project

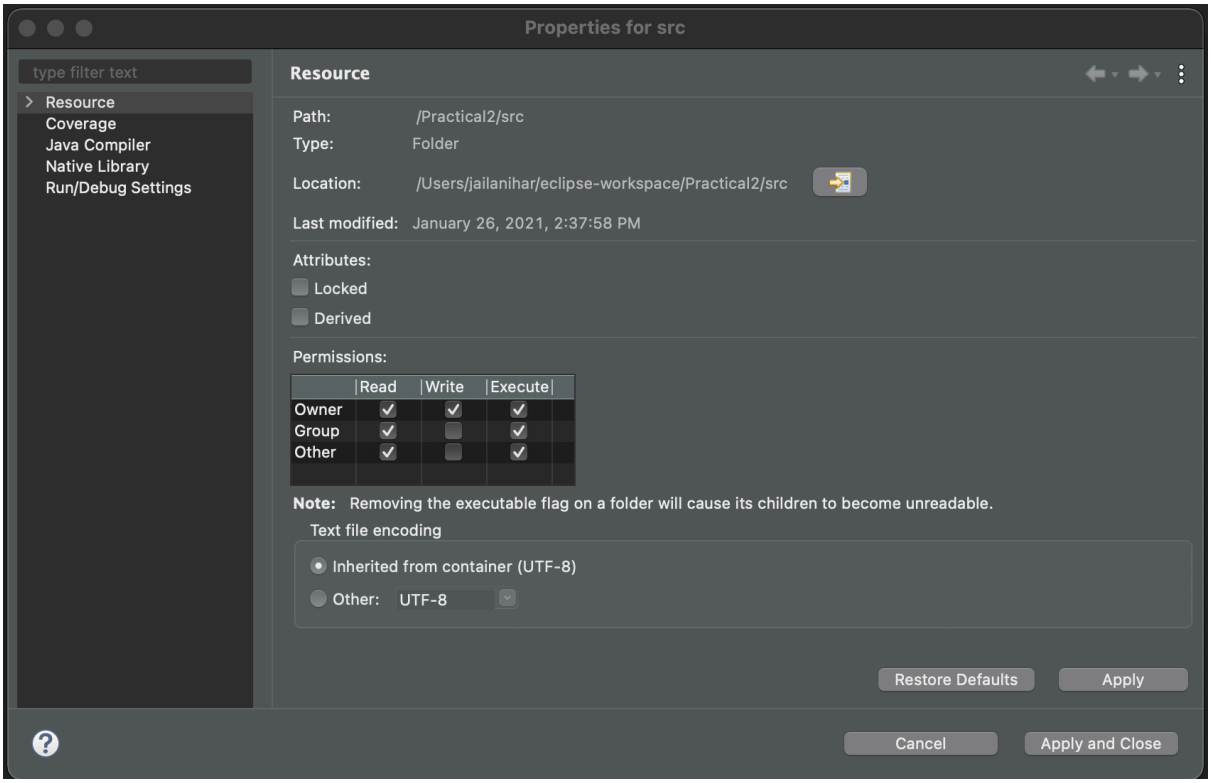
4. Create a new Java project called `Practical2`.
5. When asked to `create new module-info.java`, click `Don't Create`.



Lesson 1.2 - Setting up Git to Java project

1. Find the location of the `src` folder. You can right click the `src` folder and then go to `Properties -> Resource -> Location`. Take note / Copy the Location stated and close.


src
New
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Maven
Team
Compare With
Validate
Properties ⌘I



2. Open your `Terminal` / `Git Bash` and `cd <to the location>`. Example:

3. Then run `git init`.

Lesson 1.3 - Getting path to Source and Target File

1. Create a package called `project01`.
2. Create a Java class called `CopyPasteFile`.
3. Implement main method in the class.

```
package project01;

public class CopyPasteFile {

  public static void main(String[] args) {

  }

}
```

4. Since we need user to specify to the application the source file they want to copy. Implement `Scanner` object from `java.util` in the main method.

```
package project01;

public class CopyPasteFile {

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    scanner.close();
  }

}
```

5. Capture the user input and create a `File` object based from user input. (Note: Some part of the class are omitted.)

```
...

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Please type source file (including its path)");
    String source = scanner.nextLine();
    File sourceFile = new File(source);

    scanner.close();
  }

...
```

6. We need to make sure the user type in the correct source file. (Hint: Using the `File` object method `exists`.)

   If the source file does not exists, we need to ask the user to type in again. (Hint: We need to loop until the user type in the correct file location.)

```
...

  public static void main(String[] args) {
    ...
    File sourceFile = new File(source);
    while(!sourceFile.exists()) {
      System.out.println("Source file does not exists.");
      System.out.println("Please type source file (including its path)");
      source = scanner.nextLine();
      sourceFile = new File(source);
    }

    ...
  }

...
```

7. Once the source file has been confirmed exists then capture the user input for target file and create a `File` object based from user input.

```
...

  public static void main(String[] args) {
    ...
    while(!sourceFile.exists()) {
      ...
    }

    System.out.println("Please type target file (including its path)");
    String target = scanner.nextLine();
    File targetFile = new File(target);

    ...
  }

...
```

8. We need to make sure the user type in the correct target file. (Hint: Using the `File` object method `exists`.)

If the target file does exists, we don't want to overwrite the existing file. So we need to ask the user to type in again. (Hint: We need to loop until the user type in the correct file location.)

```
...

  public static void main(String[] args) {
    ...
    File targetFile = new File(target);
    while(targetFile.exists()) {
      System.out.println("Target file already exists.");
      System.out.println("Please type target file (including its path)");
      target = scanner.nextLine();
      targetFile = new File(target);
    }

    ...
  }

...
```

> ## ⚠️ Milestone:
>
> Commit the changes made to the `src` folder with message "Lesson 1.3 - Getting path to Source and Target File".
>
> Hint: Add the files to the staging area first, `git add <files>` and then proceed to commit, `git commit -m "Lesson 1.3 - Getting path to Source and Target File"`.

Lesson 1.4 - Copy the source file to target file

1. Since it is possible for the source file to be more than 100 MegaBytes. It would be better to use the Buffered Stream. In this case, just throw any exception thrown by the Buffered Stream and File Stream.

```
...

  public static void main(String[] args) throws FileNotFoundException,
IOException {
    ...
    while(targetFile.exists()) {
      ...
    }
```

```
    BufferedInputStream input = new BufferedInputStream(new
FileInputStream(sourceFile));
    BufferedOutputStream output = new BufferedOutputStream(new
FileOutputStream(targetFile));

    ...
  }

...
```

2. It is a good practice to close any input or output stream after using it. You could let Java auto close the stream by using the `try-with-resources`.

```
...

  public static void main(String[] args) throws FileNotFoundException,
IOException {
    ...

    try (
      BufferedInputStream input = new BufferedInputStream(new
FileInputStream(sourceFile));
      BufferedOutputStream output = new BufferedOutputStream(new
FileOutputStream(targetFile));
    ) {

    }
    ...
  }

...
```

3. Lets copy each byte of source file into target file. First we need a variable to hold the file content. Then we read the content of the file using the `InputStream` method called `read`. This will read a byte of data from the file. If the `read` method returns -1, that means it already reached the end of stream.

```
...

  public static void main(String[] args) throws FileNotFoundException,
IOException {
    ...

    try (
```

```
      BufferedInputStream input = new BufferedInputStream(new
FileInputStream(sourceFile));
      BufferedOutputStream output = new BufferedOutputStream(new
FileOutputStream(targetFile));
    ) {
      int fileContent = 0;
      while((fileContent = input.read()) != -1) {
        output.write((byte) fileContent);
      }
    }
    ...
  }

...
```

4. Additionally, lets add a variable to count the number of bytes that it copied.

```
...

  public static void main(String[] args) throws FileNotFoundException,
IOException {
    ...

    ) {
      int fileContent = 0;
      int numberOfBytesCopied = 0;
      while((fileContent = input.read()) != -1) {
        output.write((byte) fileContent);
        numberOfBytesCopied++;
      }
    }
    ...
  }

...
```

5. Print the total number of bytes copied.

```
...

  public static void main(String[] args) throws FileNotFoundException,
IOException {
    ...

    ) {
```

```
      int fileContent = 0;
      int numberOfBytesCopied = 0;
      while((fileContent = input.read()) != -1) {
        output.write((byte) fileContent);
        numberOfBytesCopied++;
      }
      System.out.println("File is copied. No of Bytes: " +
numberOfBytesCopied)
    }
    ...
  }

...
```

⚠️ Milestone:

Commit the changes made to the `src` folder with message "Lesson 1.4 - Copy the source file to target file".

Hint: Add the files to the staging area first, `git add <files>` and then proceed to commit, `git commit -m "Lesson 1.4 - Copy the source file to target file"`.