

Lesson 06 Demo 01

Utilizing File Methods in Java

Objective: To utilize file methods in Java and efficiently handle file operations within a directory

Tools required: Eclipse IDE

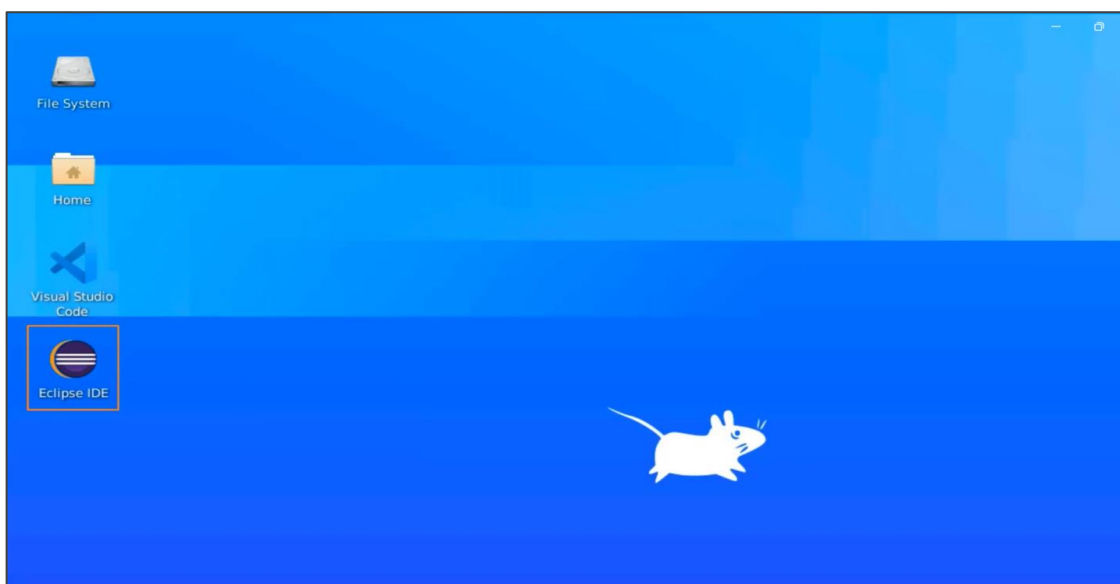
Prerequisites: None

Steps to be followed:

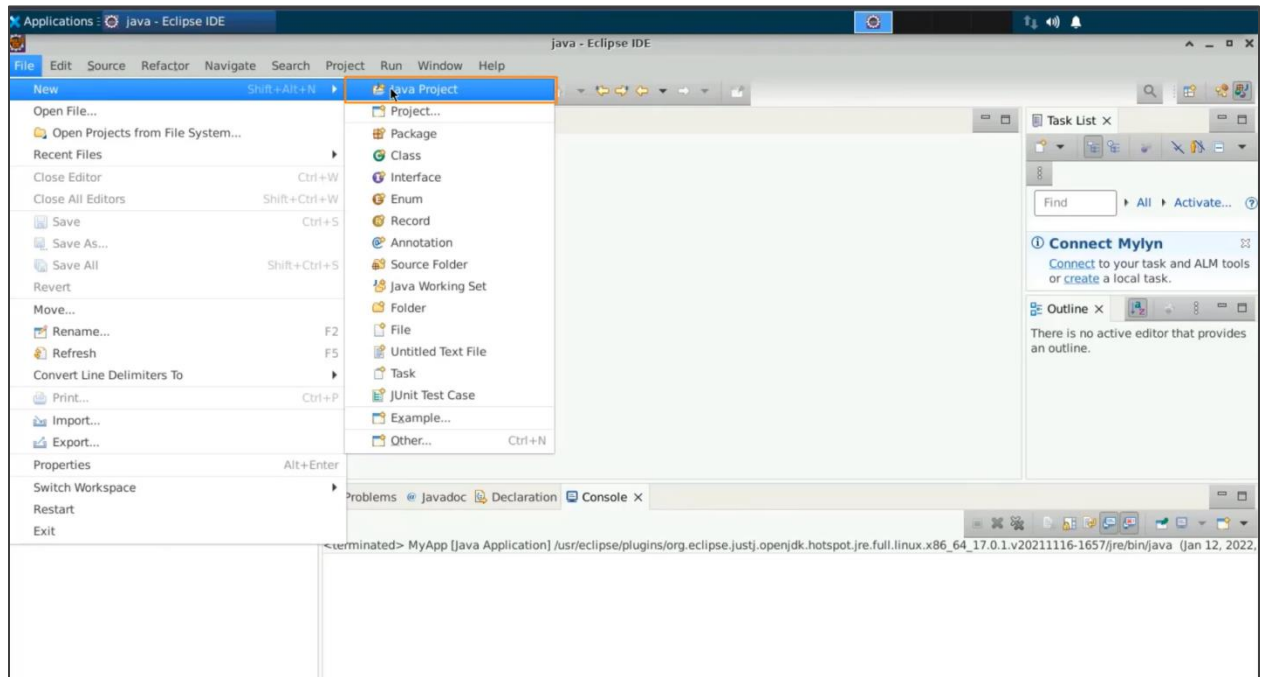
1. Open the Eclipse IDE and create a new Java project
2. Create a temporary file in a directory
3. Use the try and catch block
4. Create another path for the file in the temporary directory path
5. Write the string content
6. Use this static method to directly write the string and execute the code

Step 1: Open the Eclipse IDE and create a new Java project

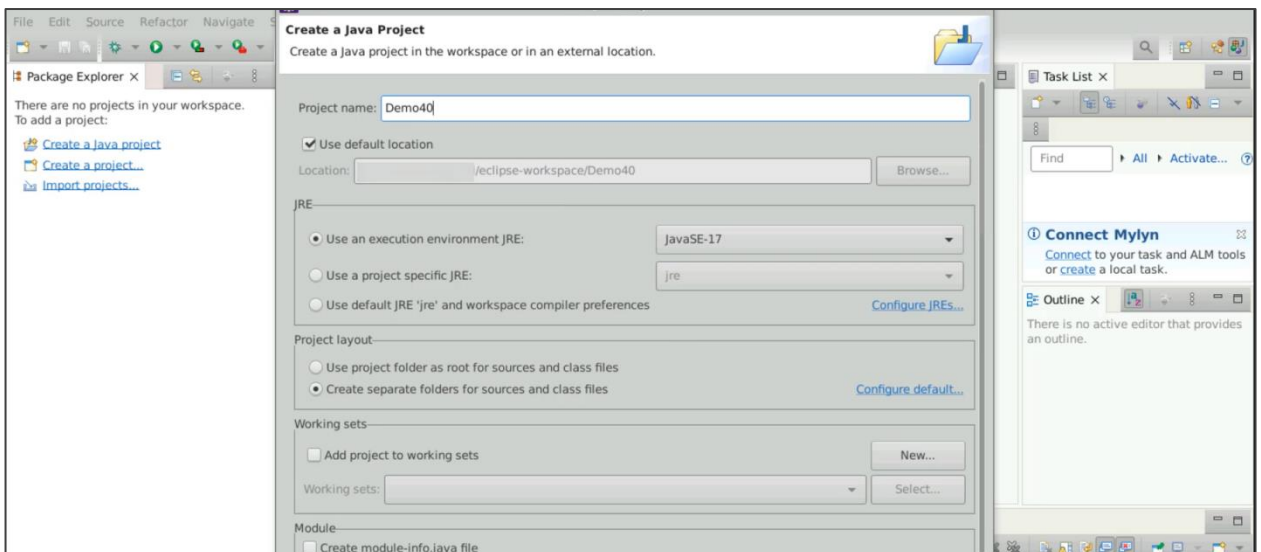
1.1 Open the Eclipse IDE



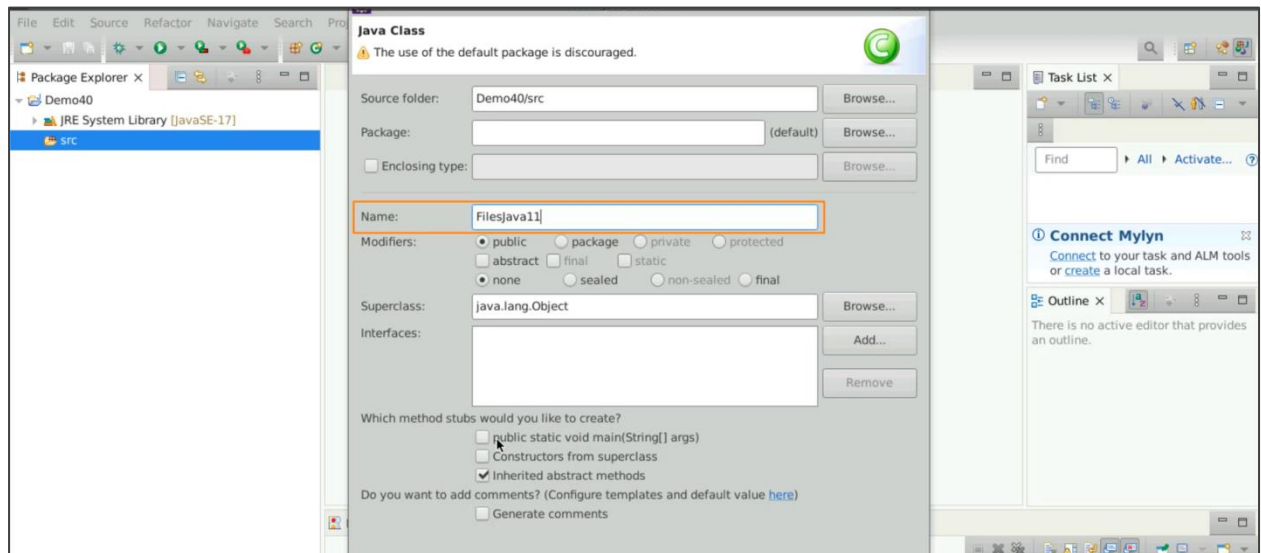
1.2 Select **File**, then **New**, and then **Java project**



1.3 Name the project **Demo40**, uncheck **Create a module-info.java file**, and press **Finish**

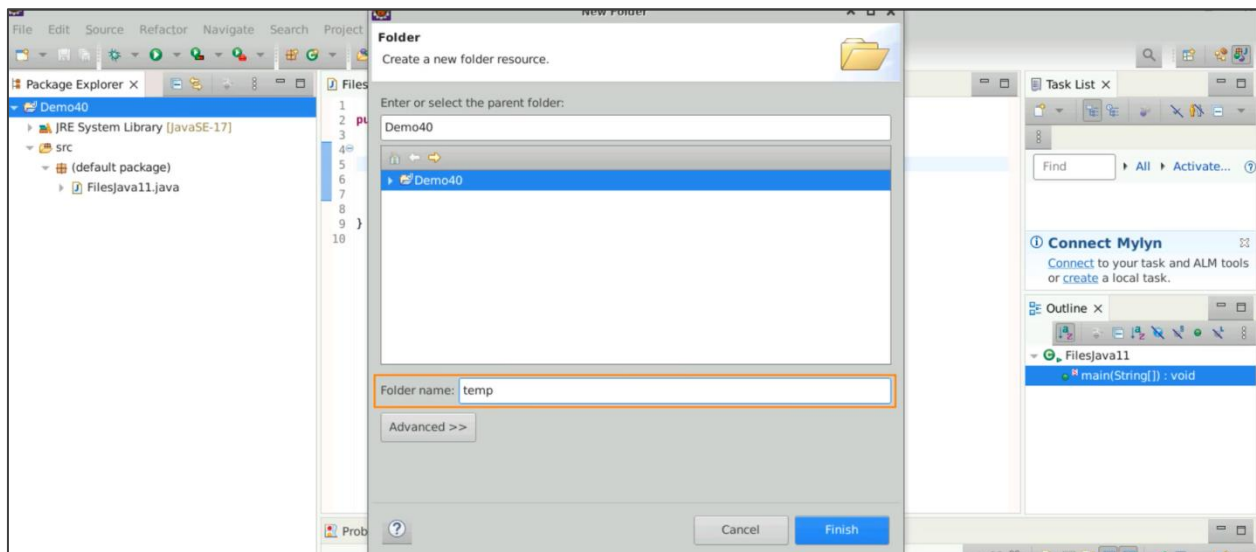


1.3 With **Demo40** selected, right-click on the **src**, and create a **new class**. Name this class **FilesJava11**, select the **main method**, and then select **finish**



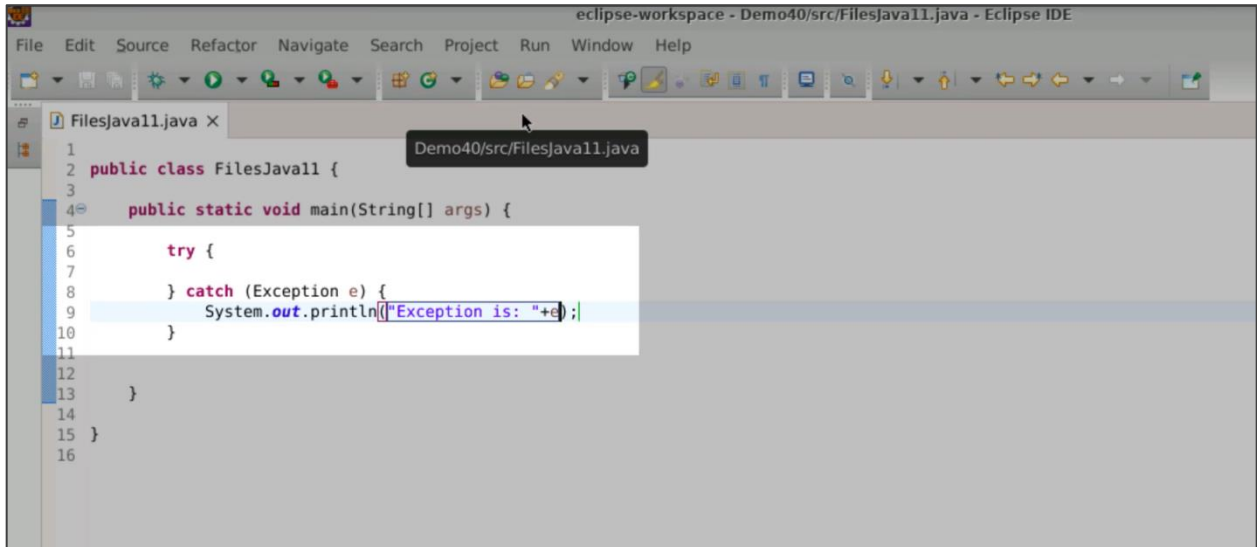
Step 2: Create a temporary file in a directory

2.1 Create a directory in the package. Select the project and create a new folder named **temp**, which stands for temporary and can contain some files



Step 3: Use the try and the catch block

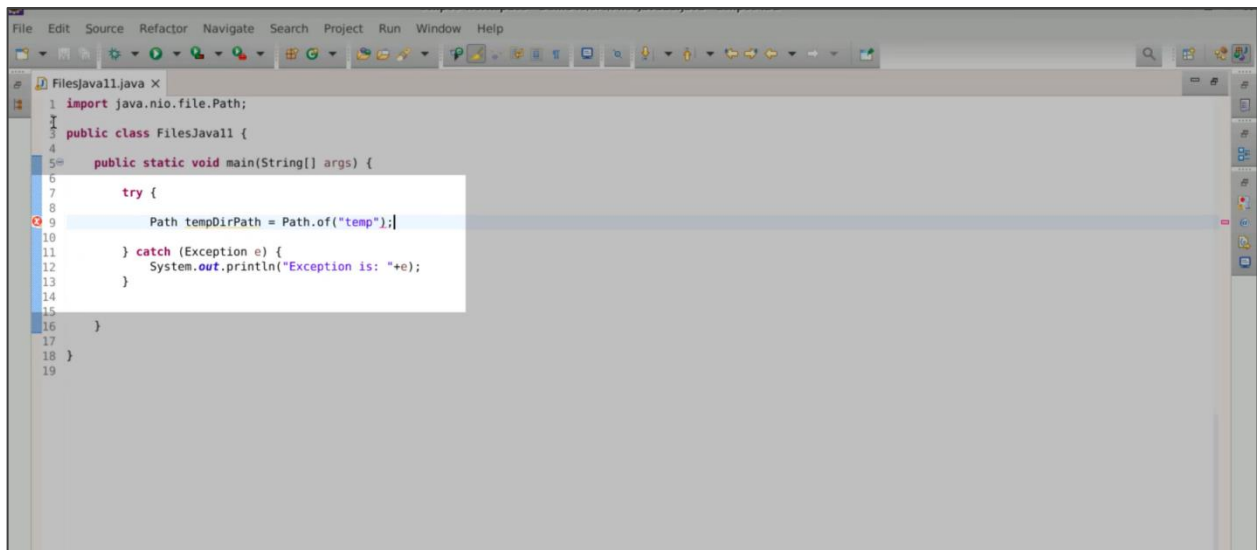
3.1 To handle potential IO exceptions, it is advised to wrap your file IO operations within a try-catch block, as many APIs may throw IO exceptions, ensuring proper exception handling



```

eclipse-workspace - Demo40/src/FilesJava11.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
FilesJava11.java x
1 public class FilesJava11 {
2
3
4 public static void main(String[] args) {
5
6     try {
7
8     } catch (Exception e) {
9         System.out.println("Exception is: "+e);
10    }
11
12
13 }
14
15 }
16
  
```

3.2 Utilize the **Java.io.file** API's **Path** method to create a temporary directory path using the static method **File.createTempDirectory** and provide the name of the directory as the input

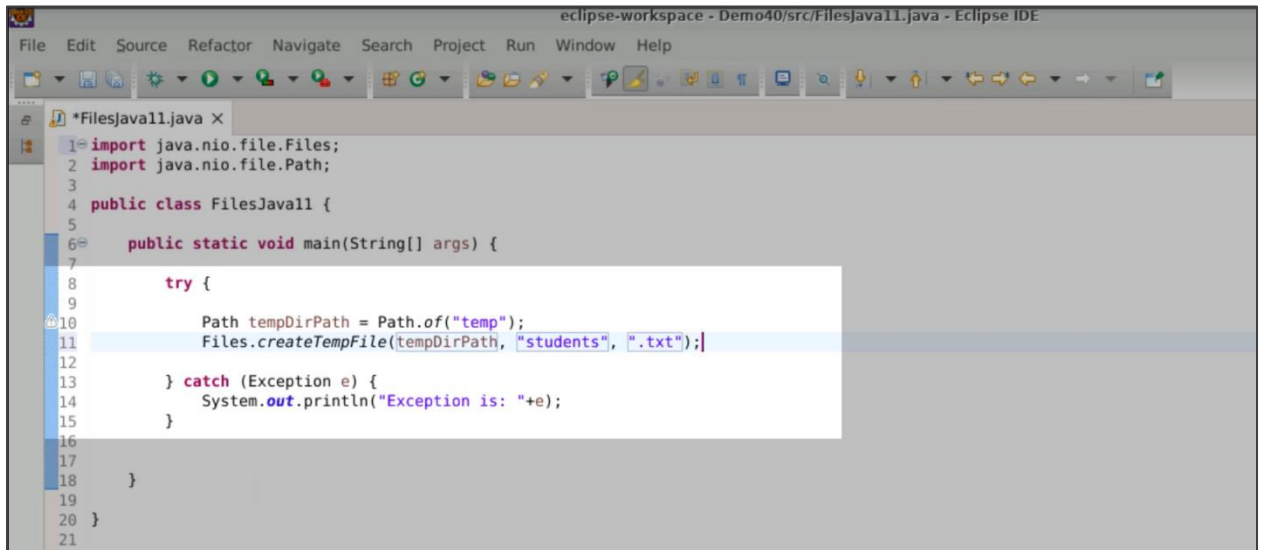


```

File Edit Source Refactor Navigate Search Project Run Window Help
FilesJava11.java x
1 import java.nio.file.Path;
2
3 public class FilesJava11 {
4
5 public static void main(String[] args) {
6
7     try {
8
9         Path tempDirPath = Path.of("temp");
10
11     } catch (Exception e) {
12         System.out.println("Exception is: "+e);
13     }
14
15 }
16
17 }
18
19
  
```

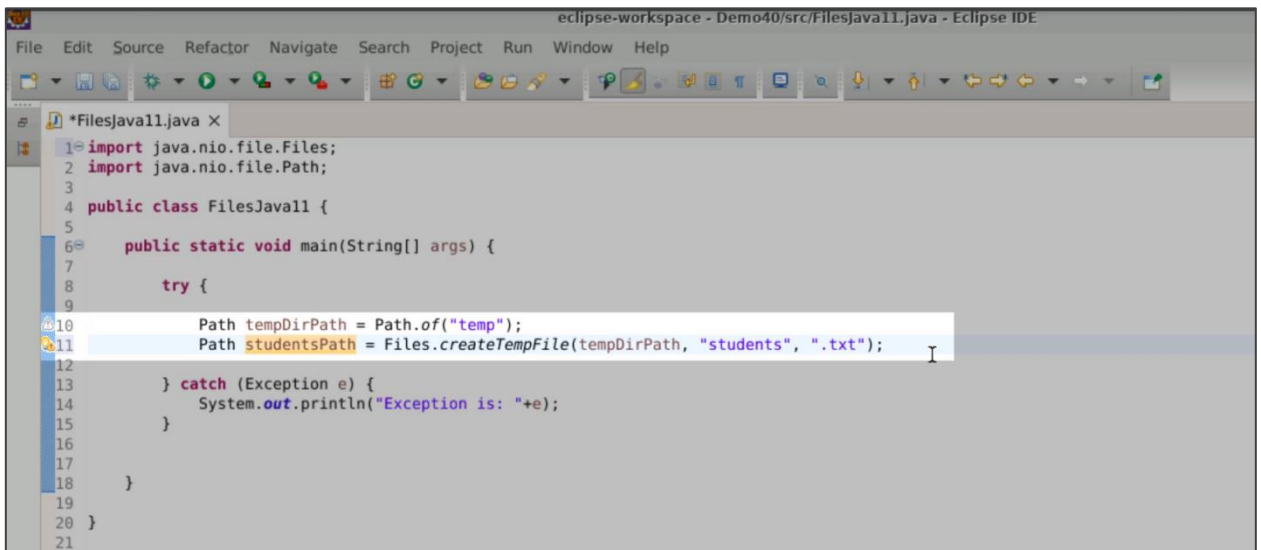
Step 4: Create one more path for the file in the temporary directory path

- 4.1 Create another path for a file within the temporary directory using the **Java.io.files.createTempFile** method, providing the directory path as the temporary directory path and setting the filename as **students** with a **.txt** extension



```
eclipse-workspace - Demo40/src/FilesJavall.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*FilesJavall.java X
1 import java.nio.file.Files;
2 import java.nio.file.Path;
3
4 public class FilesJavall {
5
6     public static void main(String[] args) {
7
8         try {
9
10             Path tempDirPath = Path.of("temp");
11             Files.createTempFile(tempDirPath, "students", ".txt");
12
13         } catch (Exception e) {
14             System.out.println("Exception is: "+e);
15         }
16
17     }
18 }
19
20 }
21
```

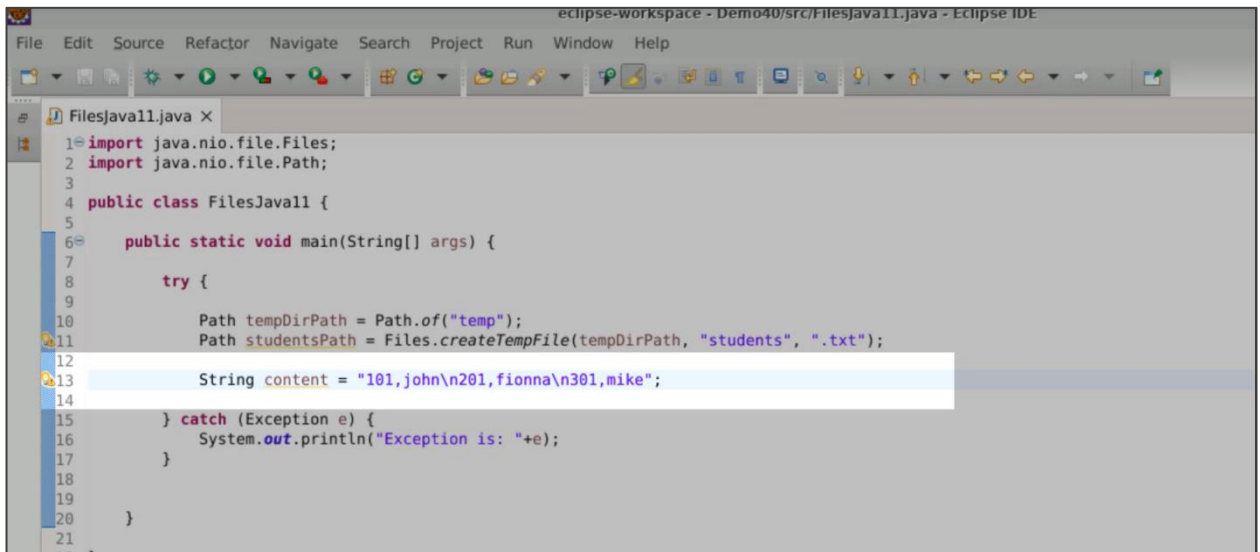
- 4.2 Store it in one of the other paths, which is called **studentsPath**. This is the name of your temporary file on the path



```
eclipse-workspace - Demo40/src/FilesJavall.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*FilesJavall.java X
1 import java.nio.file.Files;
2 import java.nio.file.Path;
3
4 public class FilesJavall {
5
6     public static void main(String[] args) {
7
8         try {
9
10             Path tempDirPath = Path.of("temp");
11             Path studentsPath = Files.createTempFile(tempDirPath, "students", ".txt");
12
13         } catch (Exception e) {
14             System.out.println("Exception is: "+e);
15         }
16
17     }
18 }
19
20 }
21
```

Step 5: Write the string content

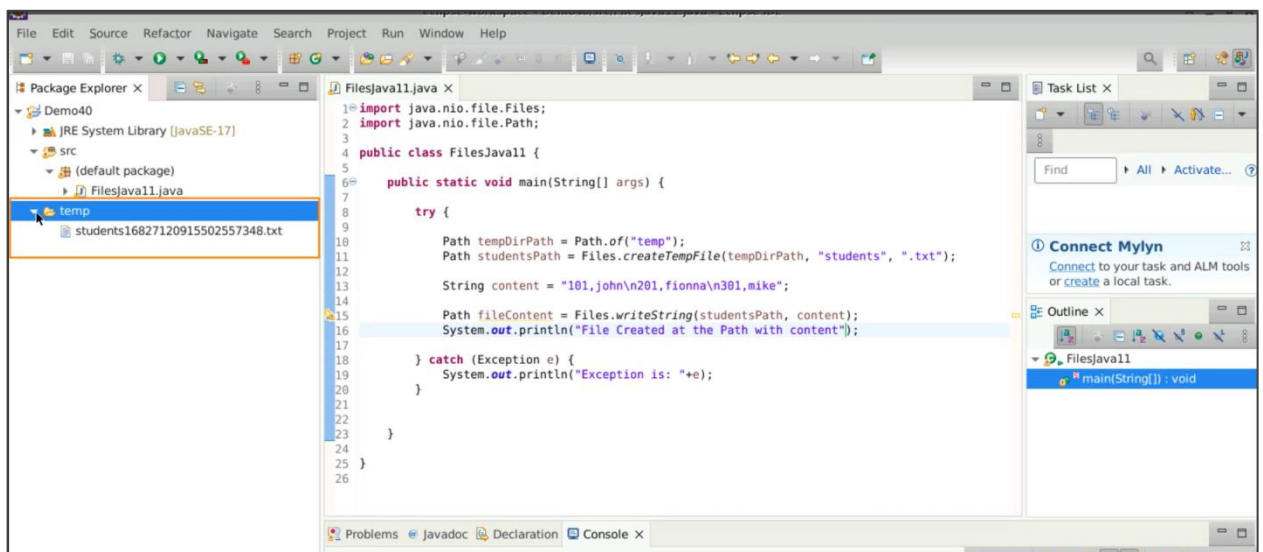
- 5.1 The content to be written into the file can be defined as a string with comma-separated values, including names like **John** and **Fionna** along with corresponding roll numbers like **101** and **201**, and additional records like **301, mike** with newlines



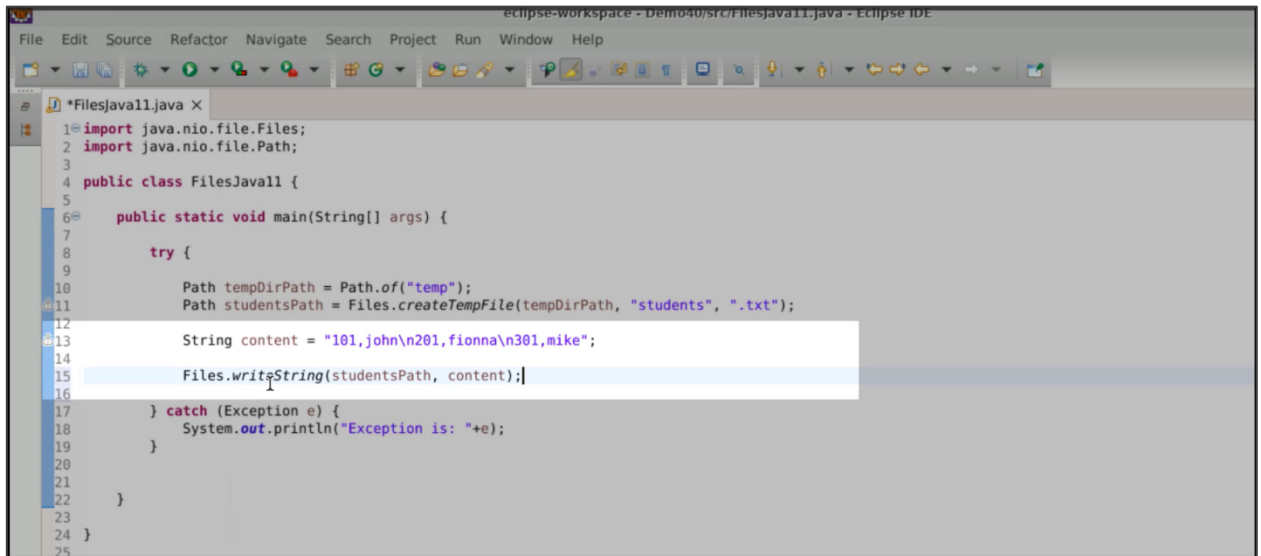
```
1 import java.nio.file.Files;
2 import java.nio.file.Path;
3
4 public class FilesJava11 {
5
6     public static void main(String[] args) {
7
8         try {
9
10            Path tempDirPath = Path.of("temp");
11            Path studentsPath = Files.createTempFile(tempDirPath, "students", ".txt");
12
13            String content = "101,John\n201,fionna\n301,mike";
14
15        } catch (Exception e) {
16            System.out.println("Exception is: "+e);
17        }
18    }
19
20 }
21
```

Step 6: Use this static method, to directly write the string and executing the code

- 6.1 To write the content directly onto the **students.txt** file, use the static method **writeString** on the **Files** class, providing the **studentPath** and the desired content as input

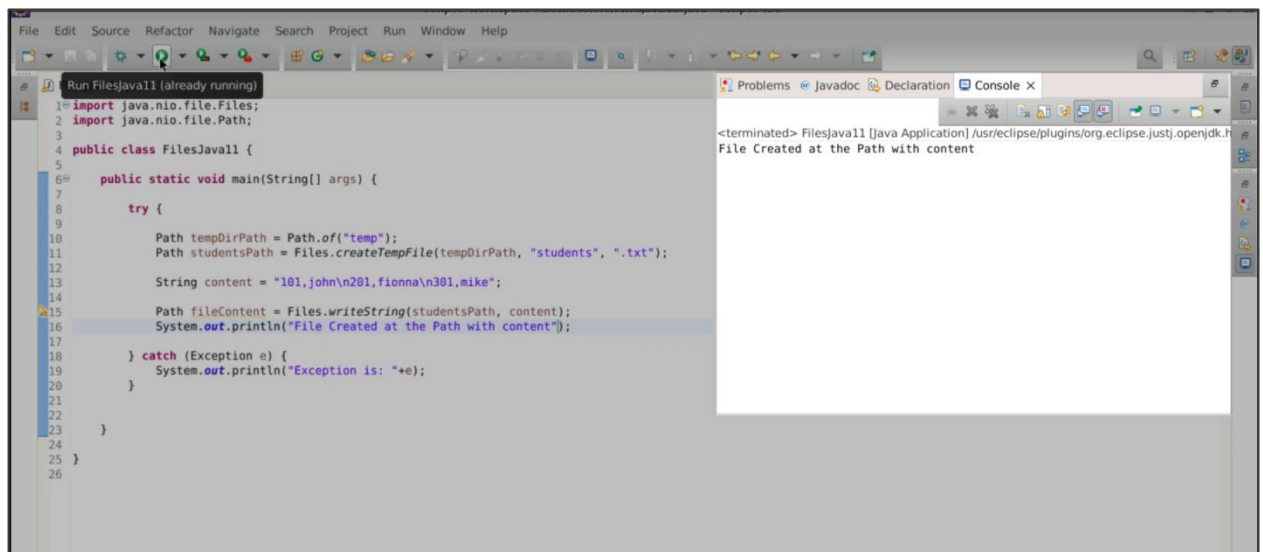


```
1 import java.nio.file.Files;
2 import java.nio.file.Path;
3
4 public class FilesJava11 {
5
6     public static void main(String[] args) {
7
8         try {
9
10            Path tempDirPath = Path.of("temp");
11            Path studentsPath = Files.createTempFile(tempDirPath, "students", ".txt");
12
13            String content = "101,John\n201,fionna\n301,mike";
14
15            Path fileContent = Files.writeString(studentsPath, content);
16            System.out.println("File Created at the Path with content");
17
18        } catch (Exception e) {
19            System.out.println("Exception is: "+e);
20        }
21    }
22
23 }
24
25 }
26
```



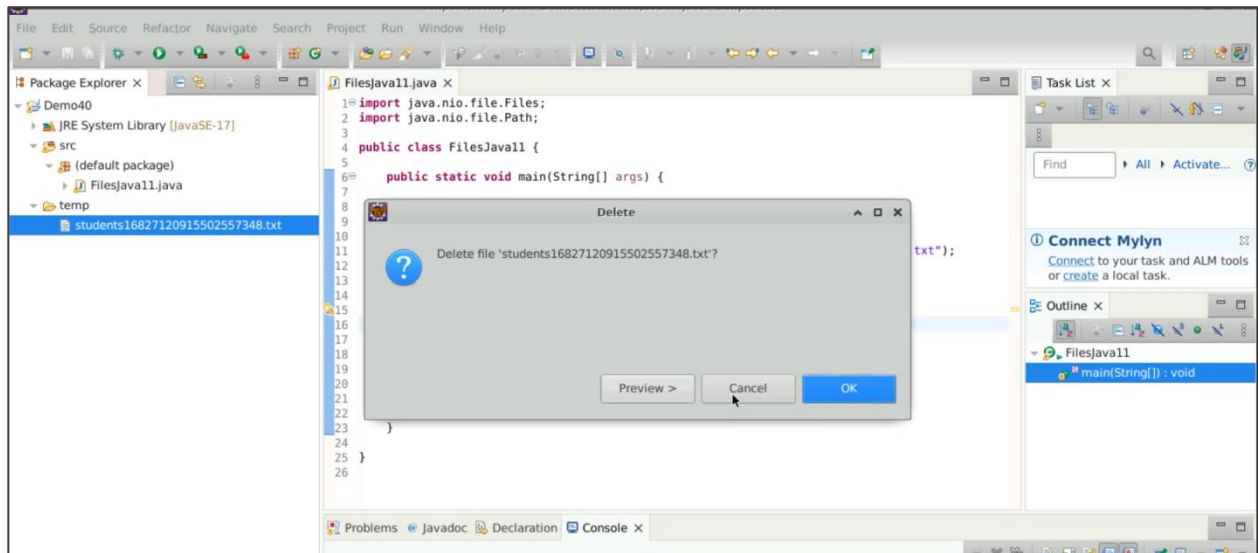
```
1 import java.nio.file.Files;
2 import java.nio.file.Path;
3
4 public class FilesJava11 {
5
6     public static void main(String[] args) {
7         try {
8             Path tempDirPath = Path.of("temp");
9             Path studentsPath = Files.createTempFile(tempDirPath, "students", ".txt");
10
11             String content = "101, john\n201, fionna\n301, mike";
12             Files.writeString(studentsPath, content);
13         } catch (Exception e) {
14             System.out.println("Exception is: " + e);
15         }
16     }
17 }
18
19
20
21
22
23
24
25
```

6.2 Let us run the code. When you run the program, it shows that **File Created at the Path with the content**

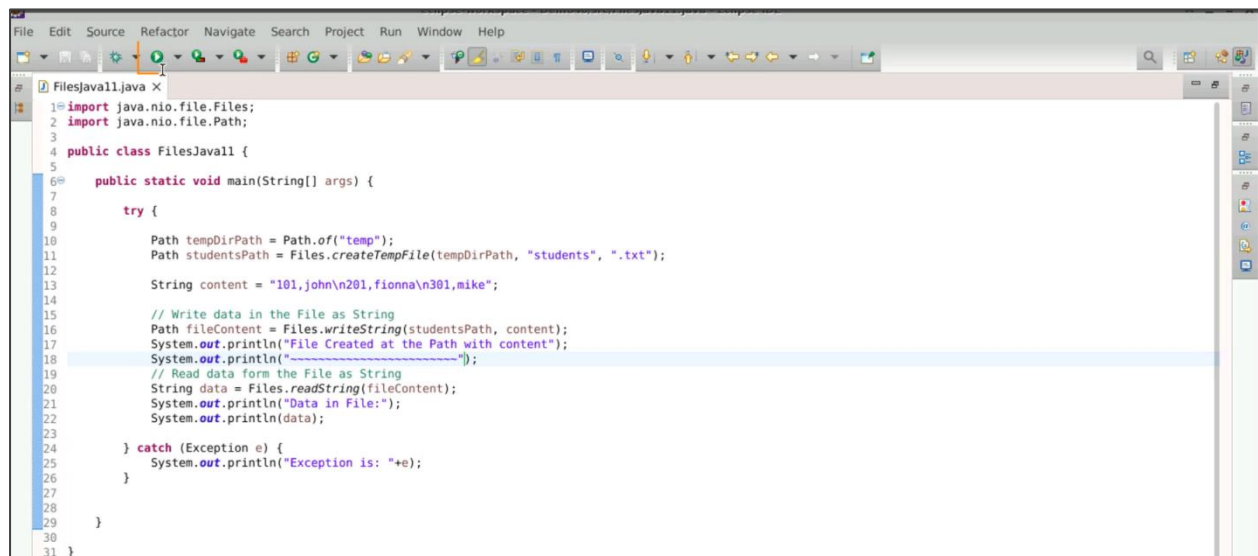


```
1 import java.nio.file.Files;
2 import java.nio.file.Path;
3
4 public class FilesJava11 {
5
6     public static void main(String[] args) {
7         try {
8             Path tempDirPath = Path.of("temp");
9             Path studentsPath = Files.createTempFile(tempDirPath, "students", ".txt");
10
11             String content = "101, john\n201, fionna\n301, mike";
12             Files.writeString(studentsPath, content);
13             System.out.println("File Created at the Path with content");
14         } catch (Exception e) {
15             System.out.println("Exception is: " + e);
16         }
17     }
18 }
19
20
21
22
23
24
25
26
```

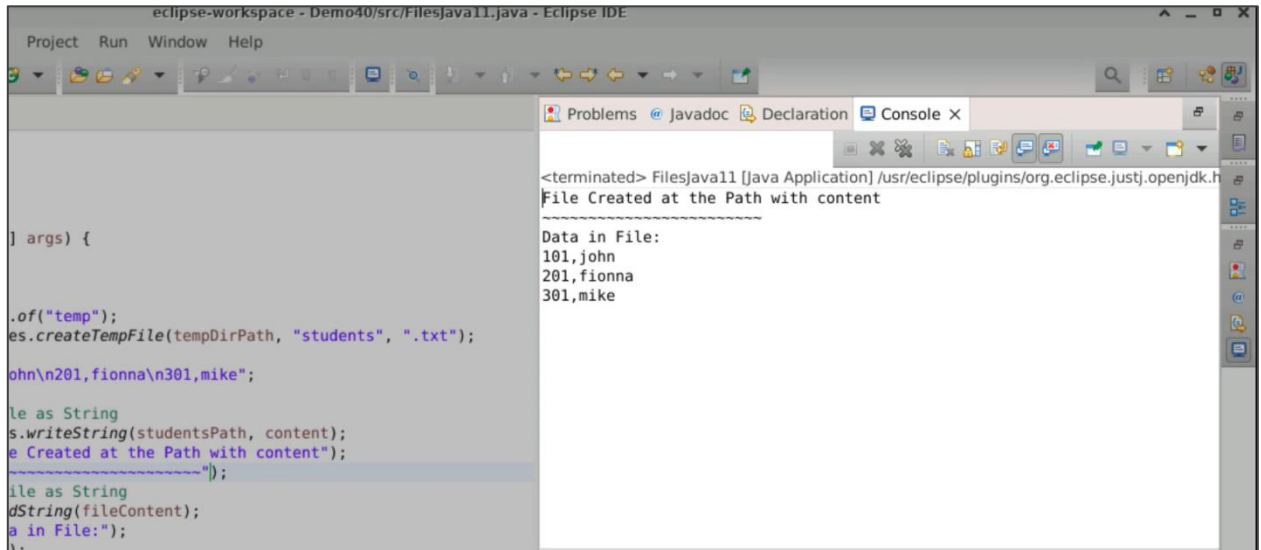

6.3 For simplicity, first remove this file. Now, there is no file. Write a string data, **files.readString** from this file content path



6.4 Execute the static method **readString** from Java 11 to read the contents of the file as a string, and print the retrieved data using **System.out.println**



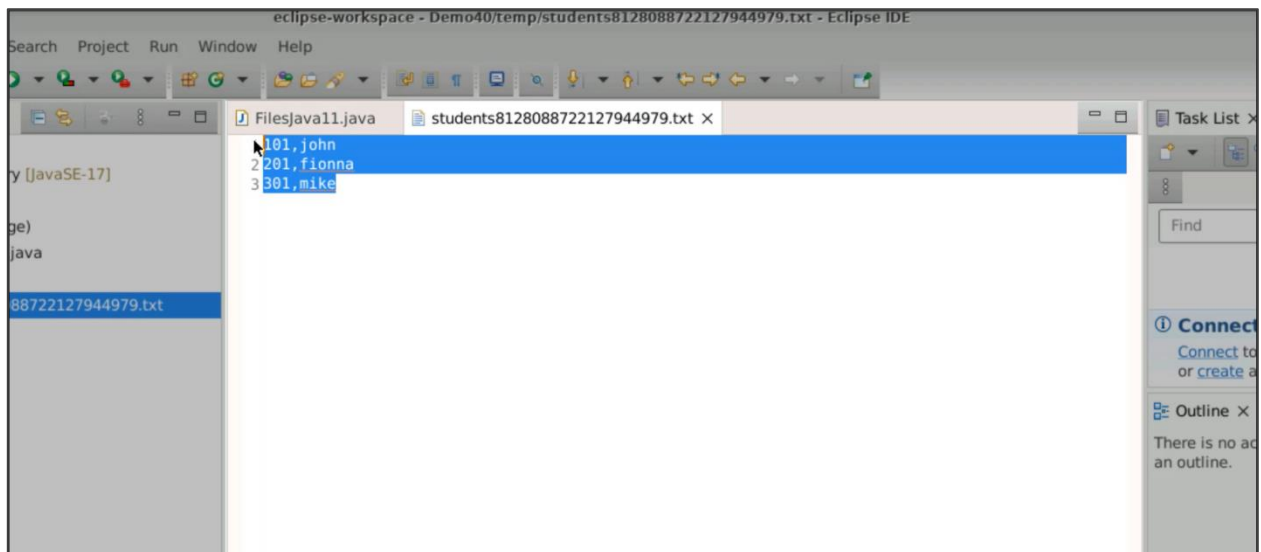
6.5 Let us run the code to see what happens when you save the file first and then try to read it. Add a small delimiter in between. As you can see, the data in the file is 101 John, 201 Fiona, and 301 Mike, which is the same data you wrote.



```
eclipse-workspace - Demo40/src/FilesJava11.java - Eclipse IDE
Project Run Window Help

<terminated> FilesJava11 [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.h
File Created at the Path with content
~~~~~
Data in File:
101,john
201,fionna
301,mike
```

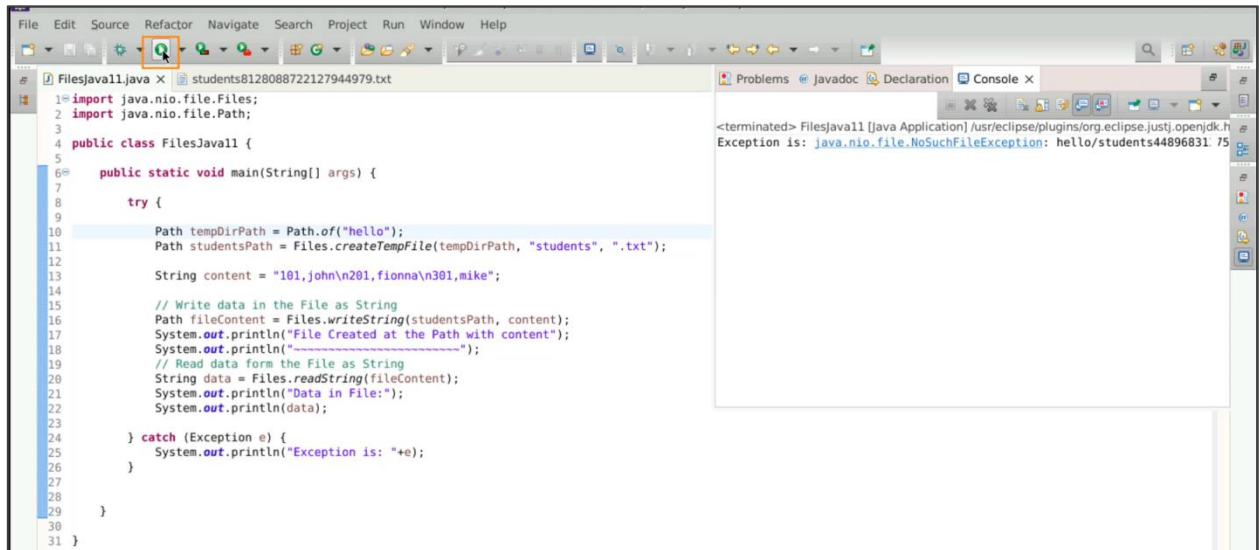
6.6 When you come back here and refresh on the temporary directory, you will see your file



```
eclipse-workspace - Demo40/temp/students8128088722127944979.txt - Eclipse IDE
Search Project Run Window Help

FilesJava11.java students8128088722127944979.txt x
1 101,john
2 201,fionna
3 301,mike
```

6.7 If you provide a non-existent directory path like **hello**, it will throw a **NoSuchFileException**. It is important to create the directory explicitly before working with it; otherwise, the directory cannot be automatically created.



The screenshot shows the Eclipse IDE with a Java file named `FilesJavall.java` open. The code defines a `main` method that attempts to create a file in a directory named `hello`. The code is as follows:

```
1 import java.nio.file.Files;
2 import java.nio.file.Path;
3
4 public class FilesJavall {
5
6     public static void main(String[] args) {
7
8         try {
9
10            Path tempDirPath = Path.of("hello");
11            Path studentsPath = Files.createTempFile(tempDirPath, "students", ".txt");
12
13            String content = "101,john\n201,fionna\n301,mike";
14
15            // Write data in the File as String
16            Path fileContent = Files.writeString(studentsPath, content);
17            System.out.println("File Created at the Path with content");
18            System.out.println("-----");
19            // Read data form the File as String
20            String data = Files.readString(fileContent);
21            System.out.println("Data in File:");
22            System.out.println(data);
23
24        } catch (Exception e) {
25            System.out.println("Exception is: "+e);
26        }
27
28    }
29
30 }
31 }
```

The console output shows the following error message:

```
<terminated> FilesJavall [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.h
Exception is: java.nio.file.NoSuchFileException: hello/students44896831.75
```

Remember this small prerequisite: The Java 11 feature **writeString** can write the content and **readString** can read the content. However, these methods are static and only work with the strings.

By following these steps, you have successfully utilized file methods in Java and efficiently handled file operations within a directory.