File Handling in Java

Introduction to File Handling in Java

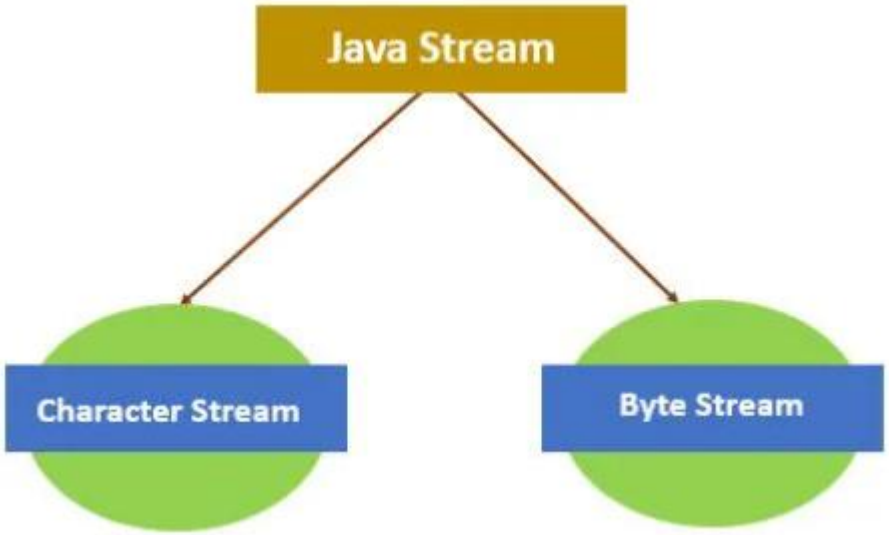
- File handling refers to working with the file in java. Reading files & writing into java files is known as file handling in java. The FIle is a container that can contain different types of information. The file can contain text, images, videos, tables, etc.

Types of Operation

- Different types of operation which can be performed on a file is given below:

How does File Handling work?

- In Java, **File handling takes place by** streaming concepts. Input/Output operations on a file perform through the streaming. Stream refers to a sequence of data.
- In java, Stream is of two types:



Continued..

- **Character Stream:** A stream that takes place with the characters is known as a character stream. Processing of the data takes place with the stream of characters in the files.
- **Byte Stream**: A stream that takes place with the byte data is known as a byte stream. Processing of the data takes place with the stream of bytes in the files.

File Handling Methods

- **createNewFile():** createNewFile method used to create an empty file. It returns the response as boolean.
- **getName():** This method is used to get the file name. It returns the string, i.e. name of the file in response.
- **getAbsolutePath():** It returns the absolute path of the file. The return type of this method is a string.
- **canRead():** This method used to check whether the file is readable or not. It returns a boolean value.
- **canWrite():** This method used to check whether the file is writable or not. It returns a boolean value.
- **delete():** This method used in deleting a file. It returns a boolean value.
- **exists():** This method used to check whether a file exists or not. It returns a boolean value.
- **length():** This method returns the size of the file in bytes. The return type of this method is long.
- **list():** This method returns an array of the files available in the directory. It returns an array of string values.
- **mkdir():** This method is used to create a directory. It returns a boolean value.

Example #1

- This example shows how different methods are used in the program to get the specified details. In this application, different methods are using to get information related to file such as:
  - Retrieving the absolute path of the file.
  - Checking whether the file is writable or not.
  - Checking whether the file is readable or not.
  - Retrieving file name.
  - Retrieving file size.
  - Retrieving list of files available to the specified directory etc.

Code:

Importing io package different classes.

```
import java.io.File;
import java.io.IOException;
public class FileHandlingExample2 {
public static void main(String[] args) {
// Creating an object of a file
File fileObj = new File("D:/Programs/fileHandlingOperations.txt");
if (fileObj.exists()) {
//retrieving the path of the specified file
System.out.println("\nSpecified file path: " + fileObj.getAbsolutePath());
//checking whether the file is writable or not
System.out.println("\nIs the file Writable: " + fileObj.canWrite());
//checking whether the file is Readable or not
System.out.println("\nIs the file Readable " + fileObj.canRead());
//retrieving file name
System.out.println("\nFile name: " + fileObj.getName());
//retrieving file size
System.out.println("\nFile size (in bytes) " + fileObj.length());
File fileDirObj = new File("D:/Programs/");
String[] fileList = fileDirObj.list();
//displaying here the list of files available in the directory
for (int i = 0; i < fileList.length; i++) {
System.out.print("\n" + fileList[i]);
}
System.out.println("\n");
}
else {
System.out.println("Specified file does not exist.");
}
}
}
```

Output:

In the above-given example, we can see how different methods are providing the information needed to perform the different checks related to files.

```
C:\Users\infor\jProjects\FileHandlingExamples>javac FileHandlingExample2.java

C:\Users\infor\jProjects\FileHandlingExamples>java FileHandlingExample2

Specified file path: D:\Programs\fileHandlingOperations.txt

Is the file Writeable: true

Is the file Readable true

File name: fileHandlingOperations.txt

File size (in bytes) 41

fileHandlingOperations.txt


C:\Users\infor\jProjects\FileHandlingExamples>
```

Example #2

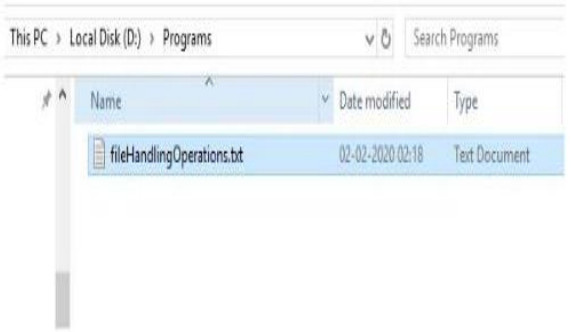
- In this example, we can see how different methods are being used in the program for different types of operation. exists() method using in the program to check if file exists are not, After that if..else.. condition is placed.
- In the If condition, it checks first whether the existing file is writable or not if the existing file remains writable, then code block under if section **used FileWriter class method** to write content into the existing file.

Importing io package different classes.

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
public class FileHandlingExample {
public static void main(String[] args) {
try {
File fileObj = new File("D:/Programs/fileHandlingOperations.txt");
if(fileObj.exists()){
System.out.println("File already exists.");
if(fileObj.canWrite()){
//creating object of FileWriter class to write things on file
FileWriter fwObj = new FileWriter("D:/Programs/fileHandlingOperations.txt");
// Writes this content into the specified file
fwObj.write("It is a basic example of writing in file!");
//closing the files once writing completed
fwObj.close();
System.out.println("\nContent has been written to the file.");
}else{
System.out.println("\nFile is not in writable mode.");
}
;
}else{
if (fileObj.createNewFile()) {
System.out.println("New File created: " + fileObj.getName());
}
}
}
catch (IOException ioError) {
System.out.println("An error occurred.");
ioError.printStackTrace();
}
}
}
```

Output:

In the above-given example, After compilation, running the program first time will create a file with the specified name in the program.



Running the program second time will write the content in the already existing file.