What is File Handling in Java?

File handling in Java implies reading from and writing data to a file. The File class from the **java.io package**, allow to work with different formats of files. In order to use the File class, you need to create an object of the class and specify the filename or directory name.

For example:

```
// Import the File class
importjava.io.File

// Specify the filename
File obj =
newFile("filename.txt");
```

Java uses the concept of a stream to make I/O operations on a file. So let's now understand what is a Stream in Java.

What is a Stream?

In Java, Stream is a sequence of data which can be of two types.

1. Byte Stream

This mainly incorporates with byte data. When an input is provided and executed with byte data, then it is called the file handling process with a byte stream.

2. Character Stream

Character Stream is a stream which incorporates with characters. Processing of input data with character is called the file handling process with a character stream.

Now that you know what is a stream, let's dive deeper into this article on File Handling in Java and know the various methods that are useful for operations on the files like creating, reading and writing.

Java File Methods

Below table depicts the various methods that are used for performing operations on Java files.

Method	Type	Description
canRead()	Boolean	It tests whether the file is readable or not
canWrite()	Boolean	It tests whether the file is writable or not
createNewFile()	Boolean	This method creates an empty file
delete()	Boolean	Deletes a file
exists()	Boolean	It tests whether the file exists

getName()	String	Returns the name of the file
getAbsolutePath()	String	Returns the absolute pathname of the file
length()	Long	Returns the size of the file in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Creates a directory

File Operations in Java

Basically, you can perform four operations on a file. They are as follows:

a. Create a File
b. Get File Information
c. Write To a File
d. Read from a File

Now, let's get into the details of each of these operations

1. Create a File

In this case, to create a file you can use the *createNewFile()* method. This method returns **true** if the file was successfully created, and **false** if the file already exists.

Let's see an example of how to create a file in Java.

```
packageFileHandling;

 // Import the File class
importjava.io.File;

// Import the IOException class to handle errors
importjava.io.IOException;

publicclassCreateFile {
publicstaticvoidmain(String[] args) {
try{
// Creating an object of a file
File myObj = newFile("D:FileHandlingNewFilef1.txt");
if(myObj.createNewFile()) {
System.out.println("File created: "+ myObj.getName());
} else{
System.out.println("File already exists.");
}
} catch(IOException e) {
```

```
System.out.println("An error occurred.");
e.printStackTrace();
}
}
}
```

a file named ***NewFilef1*** gets created in the specified location. If there is an error, then it gets handled in the catch block.

Output:

1. **File Created: NewFile1.txt**

2. Get File information

```
packageFileHandling;
importjava.io.File; // Import the File class

publicclassFileInformation {
publicstaticvoidmain(String[] args) {
 // Creating an object of a file
File myObj = newFile("NewFilef1.txt");
if(myObj.exists()) {
 // Returning the file name
System.out.println("File name: "+ myObj.getName());

// Returning the path of the file
System.out.println("Absolute path: "+ myObj.getAbsolutePath());

// Displaying whether the file is writable
System.out.println("Writeable: "+ myObj.canWrite());

// Displaying whether the file is readable or not
System.out.println("Readable "+ myObj.canRead());

// Returning the length of the file in bytes
System.out.println("File size in bytes "+ myObj.length());
}
else{
System.out.println("The file does not exist.");
}
}
}
```

OutPut:

```
File name: NewFilef1.txt
Absolute path: D:FileHandlingNewFilef1.txt
Writable: true
Readable true
File size in bytes 52
```

3. Write to a File

The **FileWriter** class together with its **write()** method to write some text into the file.

```
package FileHandling;

// Import the FileWriter class
importjava.io.FileWriter;

// Import the IOException class to handle errors
importjava.io.IOException;

public class WriteToFile {
public static void main(String[] args) {
try{
FileWriter myWriter = newFileWriter("D:FileHandlingNewFilef1.txt");
 // Writes this content into the specified file
myWriter.write(Java is the prominent programming language of the
millenium!");

// Closing is necessary to retrieve the resources allocated
myWriter.close();
System.out.println("Successfully wrote to the file.");
}
catch(IOException e) {
System.out.println("An error occurred.");
e.printStackTrace();
}
}
}
```

Output:

```
Successfully wrote to the file
```

When you run the file, the above text, "*Java is the prominent programming language of the millennium!*" will be entered in the file that you have created. You can cross check it by opening the file in the specified location.

4. Read from a File

```
packageFileHandling;

// Import the File class
importjava.io.File;

// Import this class to handle errors
importjava.io.FileNotFoundException;

// Import the Scanner class to read text files
importjava.util.Scanner;

public class ReadFromFile {
public static void main(String[] args) {
try{
// Creating an object of the file for reading the data
File myObj = newFile("D:FileHandlingNewFilef1.txt");
Scanner myReader = newScanner(myObj);
while(myReader.hasNextLine()) {
String data = myReader.nextLine();
System.out.println(data);
}
myReader.close();
} catch(FileNotFoundException e) {
System.out.println("An error occurred.");
e.printStackTrace();
}
}
}
```

Output:

```
Java is the prominent programming language of the millennium!
```