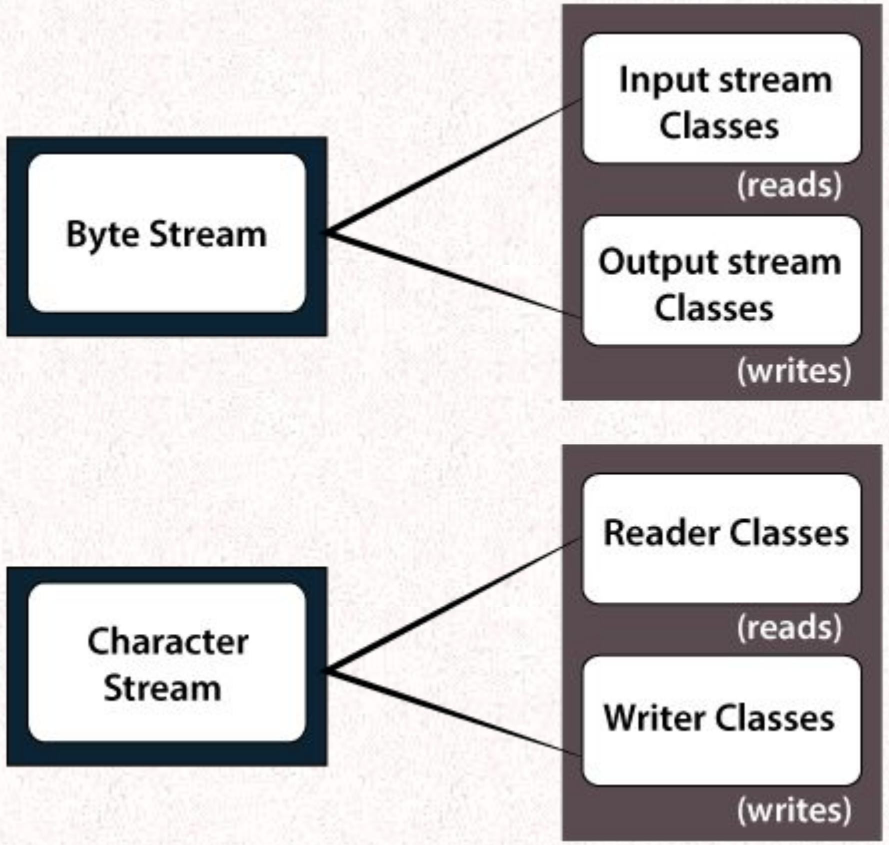
File Operation

SAM

In Java, a File is an abstract data type. A named location used to store related information is known as a File. There are several File Operations like creating a new File, getting information about File, writing into a File, reading from a File and deleting a File.



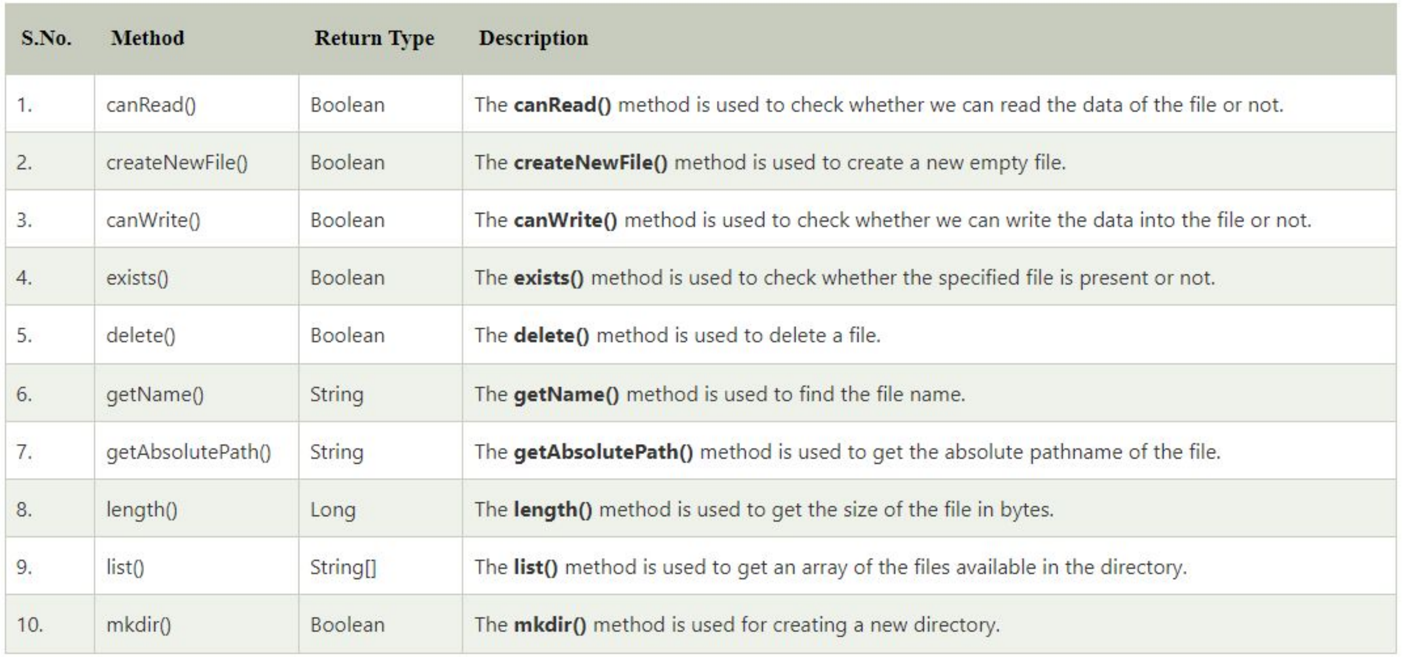
Brief classification of I/O streams

Byte Stream

Byte Stream is mainly involved with byte data. A file handling process with a byte stream is a process in which an input is provided and executed with the byte data.

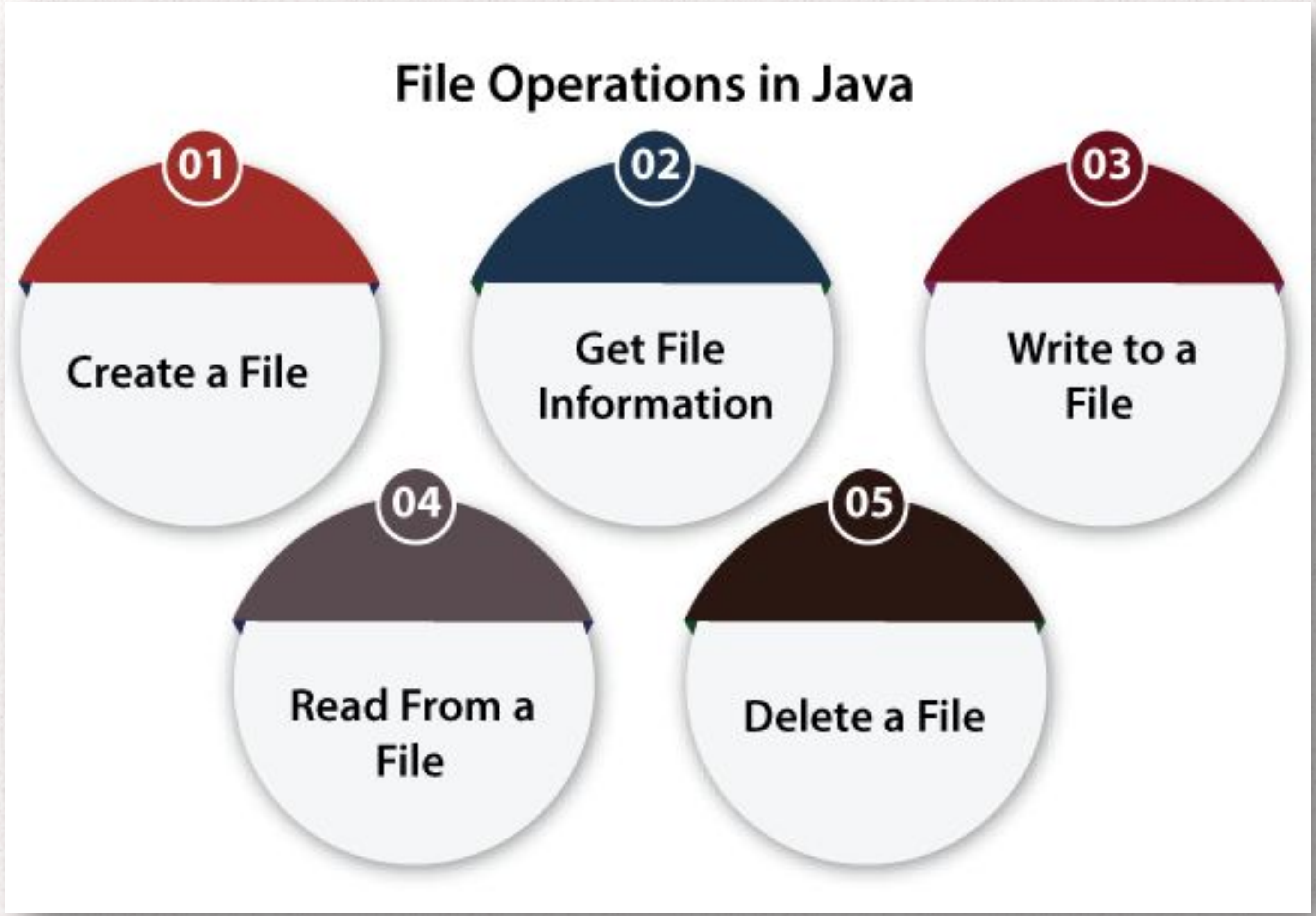
Character Stream

Character Stream is mainly involved with character data. A file handling process with a character stream is a process in which an input is provided and executed with the character data.



S.No.	Method	Return Type	Description
1.	canRead()	Boolean	The **canRead()** method is used to check whether we can read the data of the file or not.
2.	createNewFile()	Boolean	The **createNewFile()** method is used to create a new empty file.
3.	canWrite()	Boolean	The **canWrite()** method is used to check whether we can write the data into the file or not.
4.	exists()	Boolean	The **exists()** method is used to check whether the specified file is present or not.
5.	delete()	Boolean	The **delete()** method is used to delete a file.
6.	getName()	String	The **getName()** method is used to find the file name.
7.	getAbsolutePath()	String	The **getAbsolutePath()** method is used to get the absolute pathname of the file.
8.	length()	Long	The **length()** method is used to get the size of the file in bytes.
9.	list()	String[]	The **list()** method is used to get an array of the files available in the directory.
10.	mkdir()	Boolean	The **mkdir()** method is used for creating a new directory.

File Operations



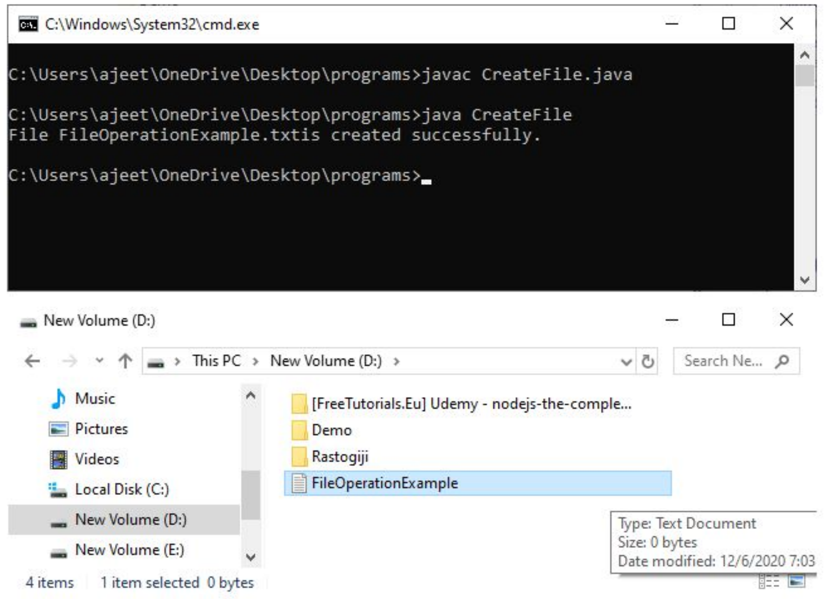
Jahangirnagar University
জাহাঙ্গীরনগর বিশ্ববিদ্যালয়

Create a File

Create a File operation is performed to create a new file. We use the createNewFile() method of file. The createNewFile() method returns true when it successfully creates a new file and returns false when the file already exists.

```
// Importing File class
import java.io.File;
// Importing the IOException class for handling errors
import java.io.IOException;
 class CreateFile {
        public static void main(String args[]) {
        try {
            // Creating an object of a file
            File f0 = new File("D:FileOperationExample.txt");
            if (f0.createNewFile()) {
                System.out.println("File " + f0.getName() + " is created
successfully.");
            } else {
                System.out.println("File is already exist in the directory.");
            }
          } catch (IOException exception) {
              System.out.println("An unexpected error is occurred.");
              exception.printStackTrace();
          }
     }
}
```

Output:



```
C:\Windows\System32\cmd.exe

C:\Users\ajeet\OneDrive\Desktop\programs>javac CreateFile.java

C:\Users\ajeet\OneDrive\Desktop\programs>java CreateFile
File FileOperationExample.txtis created successfully.

C:\Users\ajeet\OneDrive\Desktop\programs>_
```

New Volume (D:)
This PC › New Volume (D:) ›

Music
Pictures
Videos
Local Disk (C:)
New Volume (D:)
New Volume (E:)

[FreeTutorials.Eu] Udemy - nodejs-the-comple...
Demo
Rastogiji
FileOperationExample

Type: Text Document
Size: 0 bytes
Date modified: 12/6/2020 7:03

4 items   1 item selected 0 bytes

Get File Information

The operation is performed to get the file information. We use several methods to get the information about the file like name, absolute path, is readable, is writable and length.

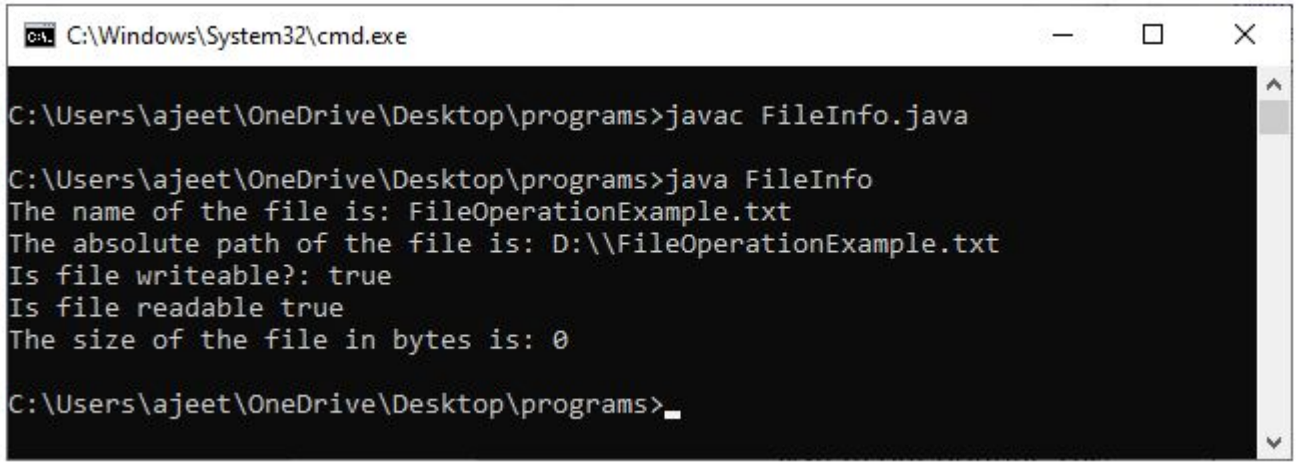
```
// Import the File class
import java.io.File;
class FileInfo {
    public static void main(String[] args) {
        // Creating file object
        File f0 = new File("D:FileOperationExample.txt");
        if (f0.exists()) {
            // Getting file name
            System.out.println("The name of the file is: " + f0.getName());

            // Getting path of the file
            System.out.println("The absolute path of the file is: " + f0.getAbsolutePath());

            // Checking whether the file is writable or not
            System.out.println("Is file writeable?: " + f0.canWrite());

            // Checking whether the file is readable or not
            System.out.println("Is file readable " + f0.canRead());

            // Getting the length of the file in bytes
            System.out.println("The size of the file in bytes is: " + f0.length());
        } else {
            System.out.println("The file does not exist.");
        }
    }
}
```



```
C:\Users\ajeet\OneDrive\Desktop\programs>javac FileInfo.java

C:\Users\ajeet\OneDrive\Desktop\programs>java FileInfo
The name of the file is: FileOperationExample.txt
The absolute path of the file is: D:\\FileOperationExample.txt
Is file writeable?: true
Is file readable true
The size of the file in bytes is: 0

C:\Users\ajeet\OneDrive\Desktop\programs>
```

Write to a File

The next operation which we can perform on a file is "writing into a file". In order to write data into a file, we will use the FileWriter class and its write() method together. We need to close the stream using the close() method to retrieve the allocated resources.

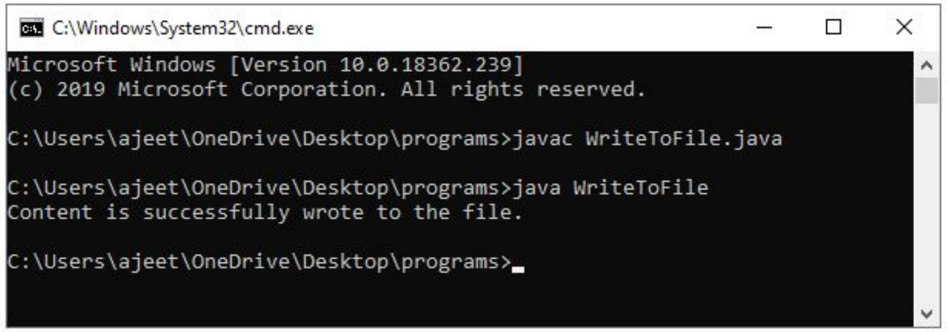
```
// Importing the FileWriter class
import java.io.FileWriter;

// Importing the IOException class for handling errors
import java.io.IOException;

class WriteToFile {
    public static void main(String[] args) {

    try {
        FileWriter fwrite = new FileWriter("D:FileOperationExample.txt");
        // writing the content into the FileOperationExample.txt file
        fwrite.write("A named location used to store related information is referred to
as a File.");

        // Closing the stream
        fwrite.close();
        System.out.println("Content is successfully wrote to the file.");
    } catch (IOException e) {
        System.out.println("Unexpected error occurred");
        e.printStackTrace();
        }
    }
}
```

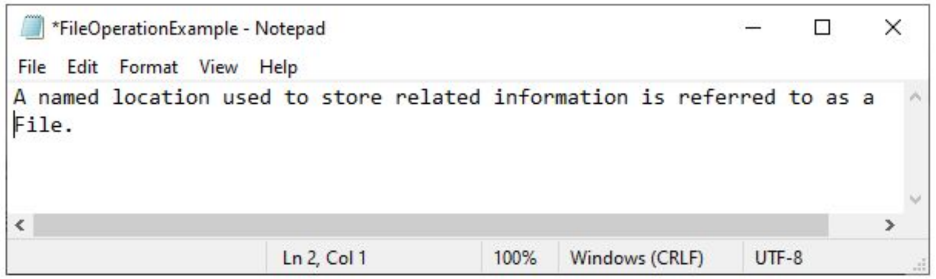


```
Microsoft Windows [Version 10.0.18362.239]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ajeet\OneDrive\Desktop\programs>javac WriteToFile.java

C:\Users\ajeet\OneDrive\Desktop\programs>java WriteToFile
Content is successfully wrote to the file.

C:\Users\ajeet\OneDrive\Desktop\programs>_
```



```
A named location used to store related information is referred to as a
File.
```

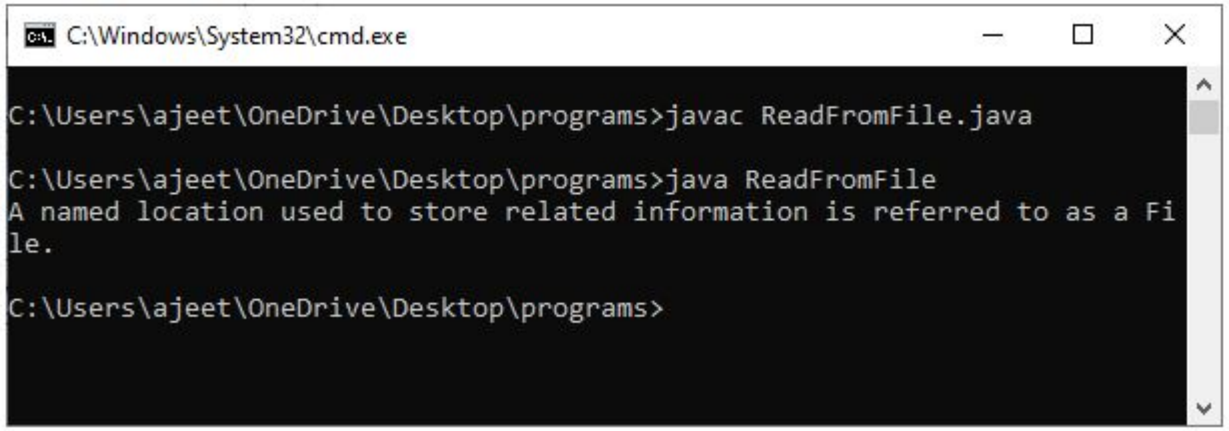
Read from a File

The next operation which we can perform on a file is "read from a file". In order to write data into a file, we will use the Scanner class. Here, we need to close the stream using the close() method. We will create an instance of the Scanner class and use the hasNextLine() method nextLine() method to get data from the file.

```
// Importing the File class
import java.io.File;
// Importing FileNotFoundException class for handling errors
import java.io.FileNotFoundException;
// Importing the Scanner class for reading text files
import java.util.Scanner;

class ReadFromFile {
    public static void main(String[] args) {
        try {
            // Create f1 object of the file to read data
            File f1 = new File("D:FileOperationExample.txt");
            Scanner dataReader = new Scanner(f1);
            while (dataReader.hasNextLine()) {
                String fileData = dataReader.nextLine();
                System.out.println(fileData);
            }
            dataReader.close();
        } catch (FileNotFoundException exception) {
            System.out.println("Unexcpected error occurred!");
            exception.printStackTrace();
        }
    }
}
```

Output:


C:\Windows\System32\cmd.exe

```
C:\Users\ajeet\OneDrive\Desktop\programs>javac ReadFromFile.java

C:\Users\ajeet\OneDrive\Desktop\programs>java ReadFromFile
A named location used to store related information is referred to as a Fi
le.

C:\Users\ajeet\OneDrive\Desktop\programs>
```

Resources

https://www.javatpoint.com/file-operations-in-java