JAVA FILE HANDLING

BY,

NAME: - PRIYANSHYU MEHER

JAVA PROGRAMMING INTERN(5TH JAN BATCH 2024)

MOB-8456953405

What is File in a Java :-

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

Stream

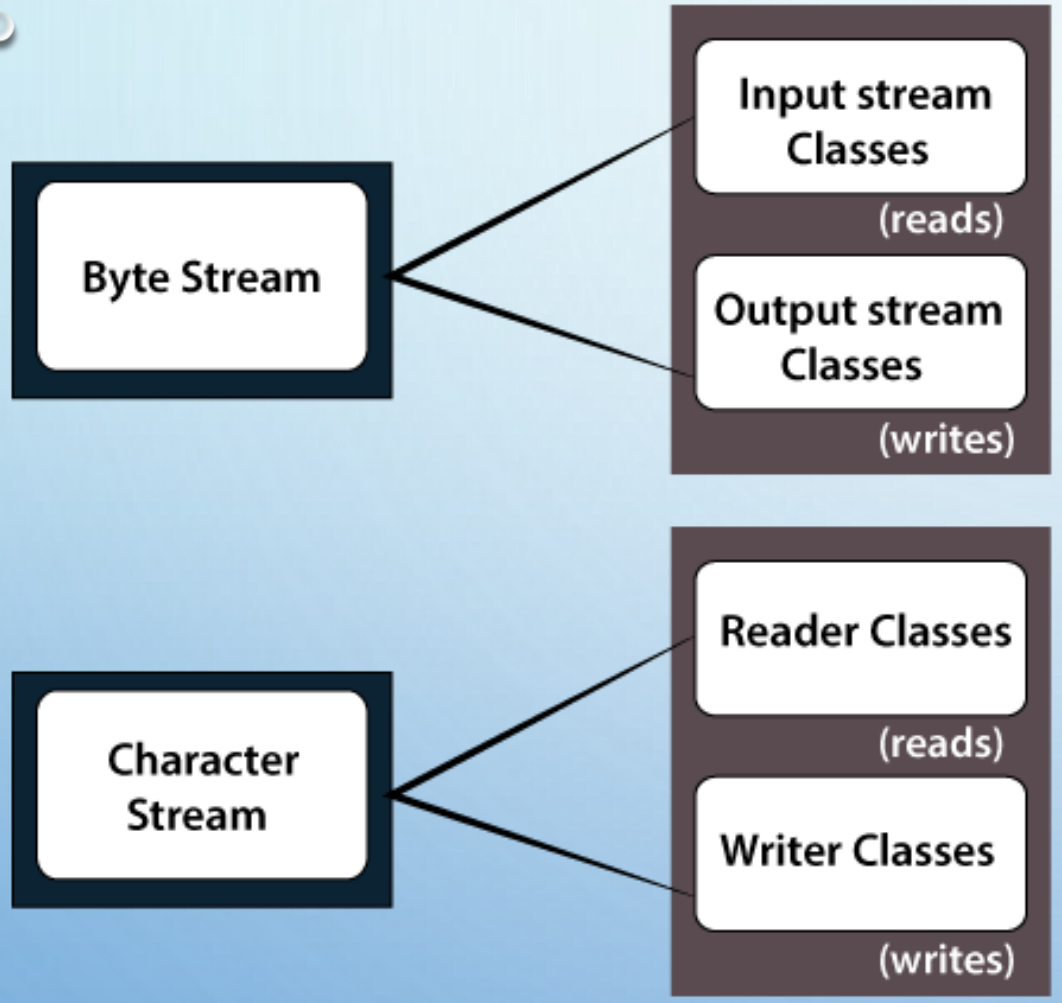
A series of data is referred to as **a stream**. In java, **Stream** is classified into two types, i.e., **Byte Stream** and **Character Stream**.

Byte Stream

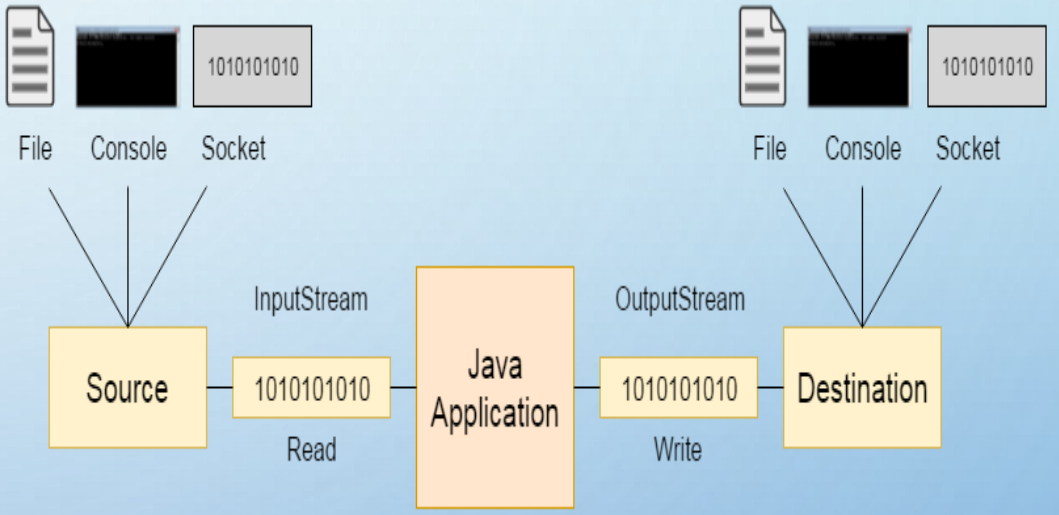
Byte Stream is mainly involved with byte data. A file handling process with a byte stream is a process in which an input is provided and executed with the byte data.

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



Brief classification of I/O streams



File Operations

We can perform the following operation on a file:

- Create a File
- Get File Information
- Write to a File
- Read from a File
- Delete a File

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.

```
1.import java.io.File;
2. class CreateFile {
3.          public static void main(String args[]) {
4.          try {

5.                File f0 = new File("D:FileOperationExample.txt");
6.                if (f0.createNewFile()) {
7.                        System.out.println("File " + f0.getName() + " is created successfully.");
8.                } else {
9.                        System.out.println("File is already exist in the directory.");
10.                 }
11.               } catch (IOException exception) {
12.                       System.out.println("An unexpected error is occurred.");
13.                       exception.printStackTrace();
14.           }
15.     }
16.}
```

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.

```
1.import java.io.File;
2.class FileInfo {
3.    public static void main(String[] args) {
4.        File f0 = new File("D:FileOperationExample.txt");
5.        if (f0.exists()) {
6.            System.out.println("The name of the file is: " + f0.getName());
7.
8.            System.out.println("The absolute path of the file is: " + f0.getAbsolutePath());
9.
10.             System.out.println("Is file writeable?: " + f0.canWrite());
11.
12.                     System.out.println("Is file readable " + f0.canRead());
13.
14.                 System.out.println("The size of the file in bytes is: " + f0.length());
15.       } else {
16.           System.out.println("The file does not exist.");
17.       }
```

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.

```
1.import java.io.FileWriter;
2.import java.io.IOException;
3.
4. class WriteToFile {
5.    public static void main(String[] args) {
6.
7.    try {
8.        FileWriter fwrite = new FileWriter("D:FileOperationExample.txt");
9.        fwrite.write("A named location used to store related information is referred to as a File.");
10.         fwrite.close();
11.         System.out.println("Content is successfully wrote to the file.");
12.    } catch (IOException e) {
13.         System.out.println("Unexpected error occurred");
14.         e.printStackTrace();
15.         }
16.    }
17.}
```

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.

```
1.mport java.io.*;
2.import java.util.Scanner;
3.class ReadFromFile {
4.    public static void main(String[] args) {
5.        try {
6.            File f1 = new File("D:FileOperationExample.txt");
7.            Scanner dataReader = new Scanner(f1);
8.            while (dataReader.hasNextLine()) {
9.                String fileData = dataReader.nextLine();
10.                System.out.println(fileData);
11.            }
12.            dataReader.close();
13.        } catch (FileNotFoundException exception) {
14.            System.out.println("Unexcpected error occurred!");
15.            exception.printStackTrace();
16.        }
17.    }
```

Delete a File

The next operation which we can perform on a file is **"deleting a file"**. In order to delete a file, we will use the **delete()** method of the file. We don't need to close the stream using the **close()** method because for deleting a file, we neither use the FileWriter class nor the Scanner class

```
1.import java.io.File;
2.class DeleteFile {
3.  public static void main(String[] args) {
4.    File f0 = new File("D:FileOperationExample.txt");
5.    if (f0.delete()) {
6.      System.out.println(f0.getName()+ " file is deleted successfully.");
7.    } else {
8.      System.out.println("Unexpected error found in deletion of the file.");
9.    }
10.  }
11.}
```

THANK YOU