File Handling

File Stream Classes :-

- File Stream classes are used to read from and write to files in Java.
- They are part of the java.io package.
- File streams can handle byte data (binary) or character data (text).

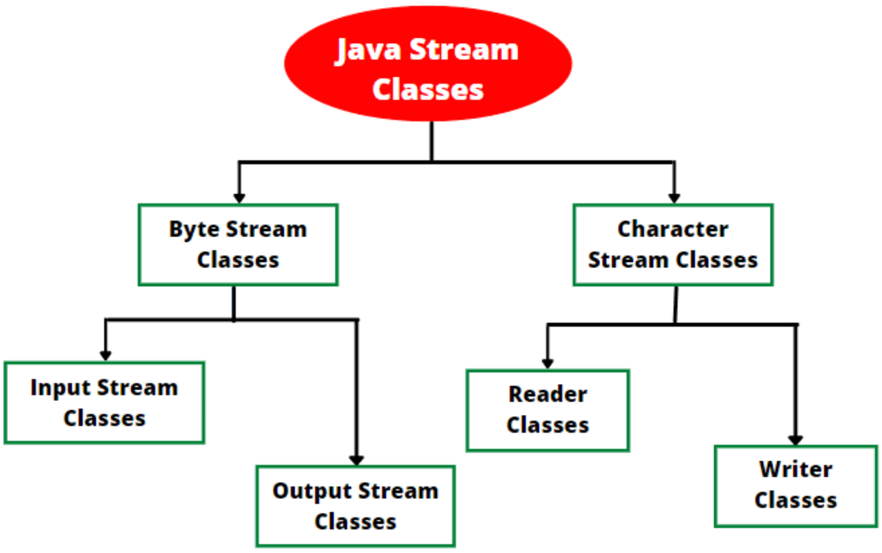

Fig: Classification of Java Stream Classes

1. Byte Stream Classes :-

- Handle input/output of 8-bit bytes.
- Used for binary data like images, audio, and video.
- Derived from InputStream (read) and OutputStream (write).

1. Input Stream Classes –

- InputStream is an abstract class used to read data as bytes from a source.
- It belongs to the java.io package.
- It is the **superclass** of all byte input stream classes.
- Data is read one byte at a time, and the stream returns -1 at the end of the file.

i. FileInputStream :-

- FileInputStream reads data from a file as bytes.
- It is used for binary data like images or audio.
- Data is read one byte at a time and the stream must be closed after use.

Syntax –

```
FileInputStream fis = new FileInputStream("filename.txt");
```

Example –

```java
package file_handling;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class file_stream_classes {

    public static void main(String[] args) throws IOException {
        File f=new File("C:\\Users\\sagar\\eclipse-workspace\\CoreJava\\newfile.txt");

        //For Reading the File
        FileInputStream fis=new FileInputStream(f);
        int i;
        while((i=fis.read())!=-1) //i=(70)!=-1
        {
            System.out.print((char)i);
        }
        fis.close();

    }

}
```

Output –

```
Fortune Cloud
```

2. Output Stream Classes –

- OutputStream is an abstract class used to write data as bytes to a destination.
- It belongs to the java.io package.
- It is the superclass of all byte output stream classes.
- Data is written one byte at a time, and streams must be closed after use.

1. FileOutputStream :-

- Writes bytes to a file**.**
- Used for binary data like images or audio.
- Data is written one byte at a time and stream must be closed after use.

Syntax –

FileOutputStream fos = new FileOutputStream("filename.txt");

Example –

```java
package file_handling;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class file_stream_classes {

    public static void main(String[] args) throws IOException {
        File f=new File("C:\\Users\\sagar\\eclipse-workspace\\CoreJava\\newfile.txt");

        //For Writing the File
        String name="Fortune Cloud";
        char ch='9';

        FileOutputStream fos=new FileOutputStream(f);
        fos.write(name.getBytes());
        fos.write(ch);

        fos.close();
        System.out.println("File Writted Successfully");


    }

}
```

Output –

2. Buffered Stream Classes :-

- Buffered Stream classes are used for efficient input and output in Java.
- They store data in an internal buffer to reduce the number of read/write operations.
- BufferedInputStream and BufferedOutputStream are used for byte streams.
- BufferedReader and BufferedWriter are used for character streams.

1. Reader Classes –

- Subclass of Reader used to read text efficiently.
- Reads line by line using an internal buffer.
- Often used with FileReader and must be closed after use.

i. BufferedReader :-

- Reads text from a file using a buffer.
- Improves efficiency by reducing read operations.
- Allows reading line by line and must be closed after use.

Syntax –

BufferedReader br = new BufferedReader(new file(f));

Example –

```java
package file_handling;

import java.io.BufferedReader;
import java.io.*;
import java.io.IOException;

public class buffer_classes {

    public static void main(String[] args) throws IOException{
        File f=new File("C:\\Users\\sagar\\eclipse-workspace\\CoreJava\\buffer.txt");

        //For Reading the File
        BufferedReader br = new BufferedReader(new FileReader(f));
        boolean line;

        while((line=br.readLine()!=null))
        {
            System.out.println(line);
        }
        br.close();
    }

}
```

Output –

```
True
```

2. Writer Classes :-

- Subclass of Writer used to write text efficiently.
- Uses an internal buffer to reduce write operations.
- Supports newLine(), and must flush and close after writing.

i. **BufferedWriter** :-

- Writes text to a file using a buffer.
- Improves efficiency by reducing write operations.
- Must flush and close the stream after writing

Syntax –

BufferedWriter bw = new BufferedWriter(new file(f));

Example –

```java
package file_handling;

import java.io.BufferedWriter;
import java.io.*;
import java.io.IOException;

public class buffer_classes {

    public static void main(String[] args) throws IOException{
        File f=new File("C:\\Users\\sagar\\eclipse-workspace\\CoreJava\\buffer.txt");

        //For Writing the File
        BufferedWriter bw = new BufferedWriter(new FileWriter(f));

        bw.write("Hello Java");
        bw.close();
        System.out.println("File Writted Successfully");

    }

}
```

Output –

```
File Writted Successfully
```

Difference Between BufferedReader & FileReader :-

Basis	BufferedReader	FileReader
Use	It is used to read characters from any type of character input stream (String, Files, etc.)	It can be used only for reading files
Buffer	Uses Buffer internally to read characters from	Doesn't use Buffer. Directly reads from the file by accessing the hard drive.
Speed	Faster	Slower
Efficiency	Much more efficient for reading files	Less efficient
Reading Lines	BufferedReader can be used to read a single character at a time as well as a line at a time.	It can read only one character at a time, can not read lines

Difference Between FileInputStream & FileReader :-

FileInputStream	FileReader
Stream is a byte-based object that can read and write bytes.	Reader is Character Based, it can be used to read or write characters.
FileInputStream is Byte Based, it can be used to read bytes.	FileReader is Character Based, it can be used to read characters.
Stream is used to binary input/output	Reader is used to character input/output
FileInputStream is used for reading binary files.	FileReader is used for reading text files in platform default encoding.
Serialization and DeSerialization can be done with FileInputStream and ObjectInputStream, and serialized objects can be saved to a file. Serialization converts an object to a byte stream, and deserialization converts it back to an object.	FileReader is not used for Serialization and DeSerialization, as it reads characters not bytes.
FileInputStream is descendant of InputStream class.	FileReader extends from Reader class
read() method of FileInputStream can read one byte at a time or multiple bytes in a byte array.	*read() method* of FileReader can read one character at a time or multiple characters into an array