Java Streams:

- Streams provide the basic concept for performing input/output operations in Java.
- The streams are the sequence of data that are read from the source and written to the destination in a sequential manner.
- Streams allow us to transfer the data between a program and an external I/O source such as a file, network connection, or other devices.
- The java.io is a package that provides classes for performing input and output operations including reading from and writing to various data sources and destinations.
- The java.io package is a fundamental part of java's I/O system.
- In Java, an input stream is used to read the data from the source and an output stream is used to write the data to the destination.
- Java I/O library provides two types of streams depending upon the data a stream can hold as:

    - Byte Stream
    - Character Stream

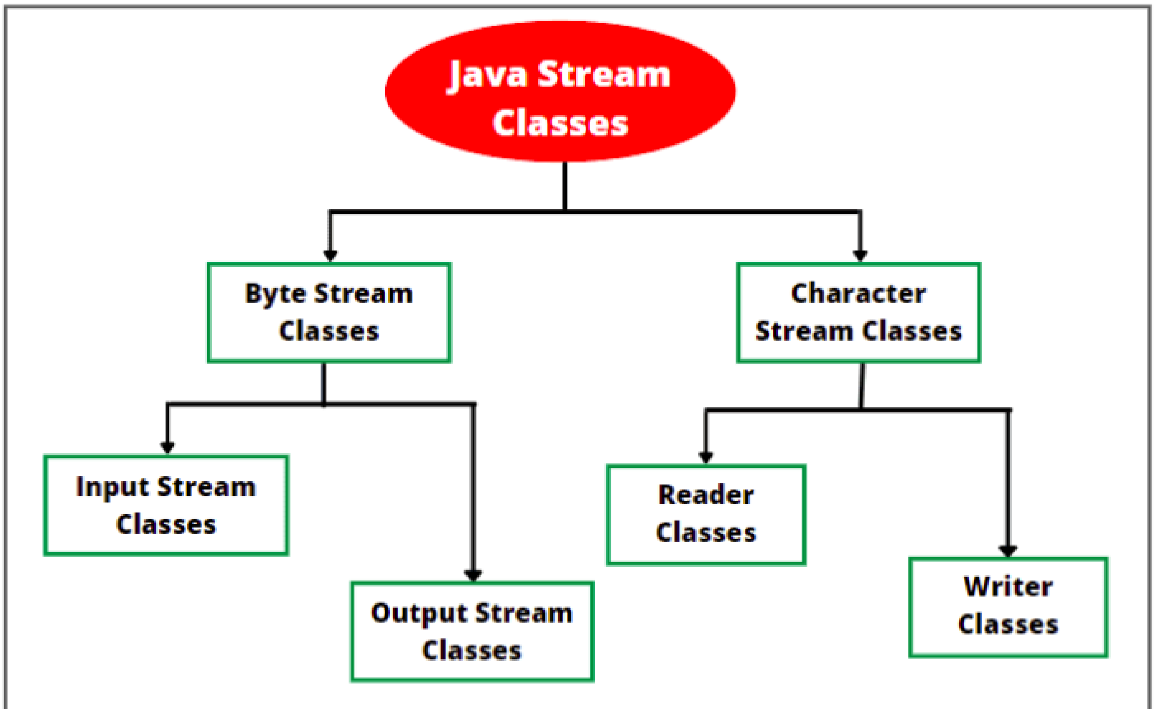


Fig: Classification of Java Stream Classes

1. Byte Streams(InputStream and OutputStream)
    - It allows us to read and write a single byte i.e. 8 bits of data.
    - The binary data include images, audio files, and other non-text files.
    - All byte stream classes are derived from base abstract classes as Input stream and output stream
        - Input Stream:
        - It is the abstract superclass of all byte-based input streams.
            - Some common subclasses are FileOutputStream, ByteArrayOutputStream and BufferedInputStream

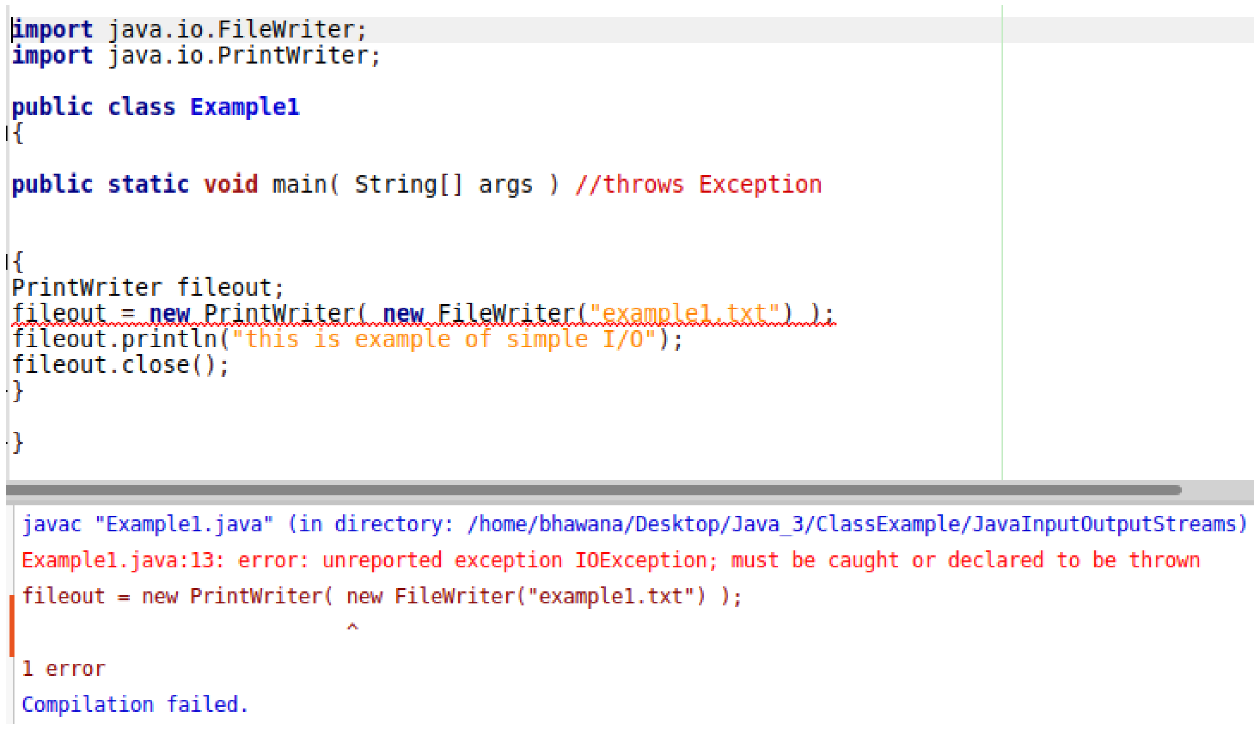
    - It provides methods like read(), read(byte[] buffer), and available() to read data from the source.

  - Output Stream:
    - It is the abstract superclass of all byte-based output streams.
    - Common subclasses include FileOutputStream, ByteArrayOutputStream, and BufferedOutputStream.
    - It provides methods like write(int b), write(byte[] buffer), and flush() to write data to the destination.

2. Character Stream:
   - It allows us to read and write a single character of data.
   - Data includes text-based data like strings and text files.
   - All character stream classes are derived from base abstract classes such as Reader and Writer.
   - Reader:
     - It is the abstract superclass for all input character streams.
     - Common subclasses include FileReader, InputStreamReader, and BufferedReader.
     - It provides methods like read(), read(char[] buffer), and ready() to read character data from the source.
   - Writer:
     - It is the abstract superclass for all output character streams.
     - Common subclasses include FileWriter, OutputStreamWriter, and BufferedWriter.

- It provides methods like write(int c), write(char[] buffer), and flush() to write character data to the destination.

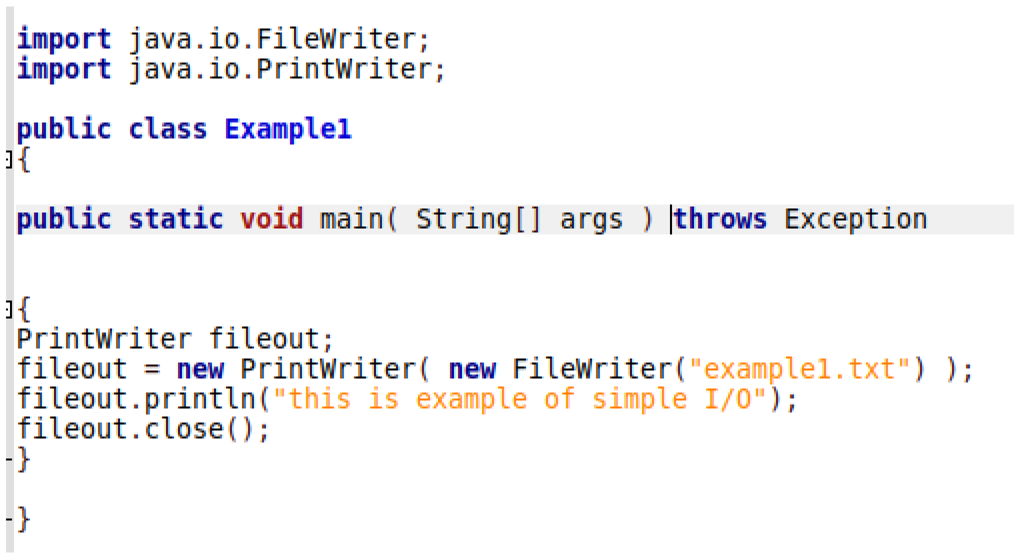
Example 1:



```
import java.io.FileWriter;
import java.io.PrintWriter;

public class Example1
{

public static void main( String[] args ) //throws Exception


{
PrintWriter fileout;
fileout = new PrintWriter( new FileWriter("example1.txt") );
fileout.println("this is example of simple I/O");
fileout.close();
}

}
```

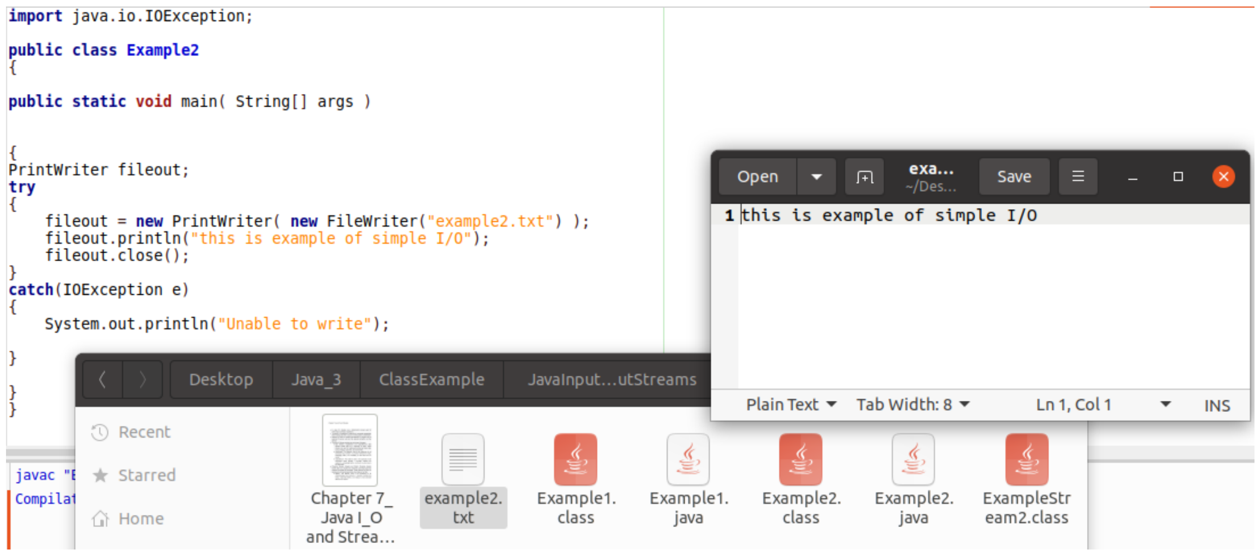
```
javac "Example1.java" (in directory: /home/bhawana/Desktop/Java_3/ClassExample/JavaInputOutputStreams)
Example1.java:13: error: unreported exception IOException; must be caught or declared to be thrown
fileout = new PrintWriter( new FileWriter("example1.txt") );
                           ^

1 error
Compilation failed.
```

So, while working with files, exception handling must be performed as:



```
import java.io.FileWriter;
import java.io.PrintWriter;

public class Example1
{

public static void main( String[] args ) throws Exception


{
PrintWriter fileout;
fileout = new PrintWriter( new FileWriter("example1.txt") );
fileout.println("this is example of simple I/O");
fileout.close();
}

}
```

Example 2:



```
import java.io.IOException;

public class Example2
{

public static void main( String[] args )


{
PrintWriter fileout;
try
{
    fileout = new PrintWriter( new FileWriter("example2.txt") );
    fileout.println("this is example of simple I/O");
    fileout.close();
}
catch(IOException e)
{
    System.out.println("Unable to write");

}

}
}
```

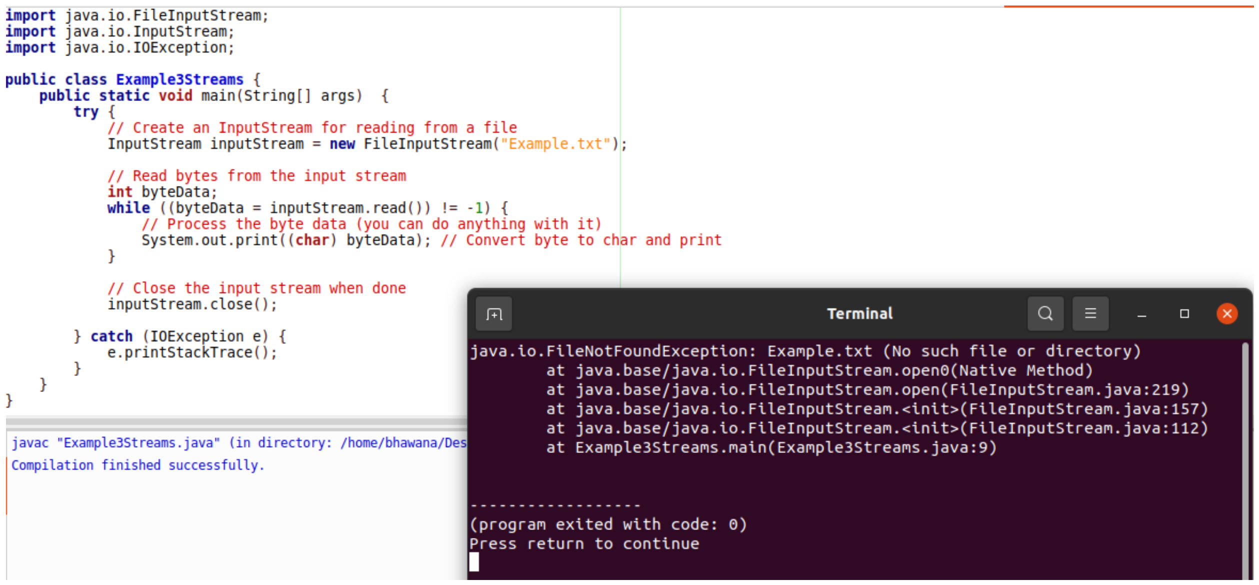
```
javac "E
Compilat
```

```
1 this is example of simple I/O
```

InputStream classes:

- Input Stream classes is used for reading the data from various input sources like files, or any other input stream source.
- It allows to read bytes of the data in the sequential manner.

Example:

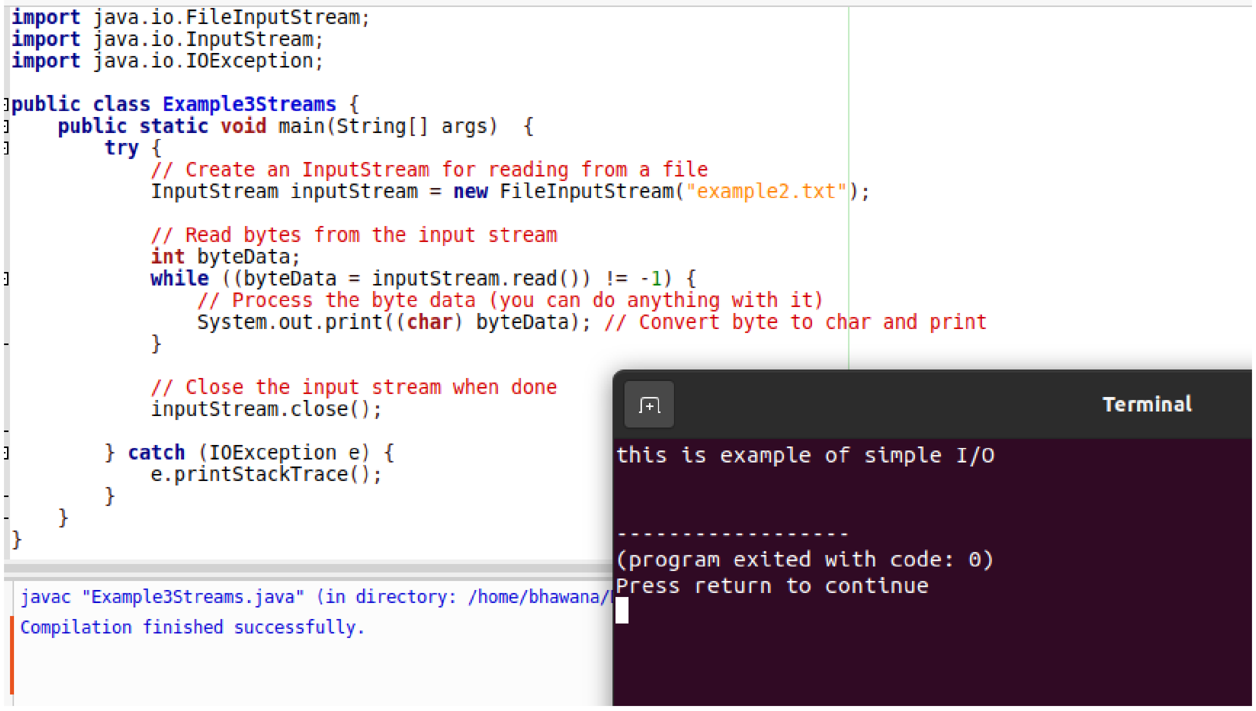


```java
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.IOException;

public class Example3Streams {
    public static void main(String[] args)  {
        try {
            // Create an InputStream for reading from a file
            InputStream inputStream = new FileInputStream("Example.txt");

            // Read bytes from the input stream
            int byteData;
            while ((byteData = inputStream.read()) != -1) {
                // Process the byte data (you can do anything with it)
                System.out.print((char) byteData); // Convert byte to char and print
            }

            // Close the input stream when done
            inputStream.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
javac "Example3Streams.java" (in directory: /home/bhawana/Des
Compilation finished successfully.
```

```
java.io.FileNotFoundException: Example.txt (No such file or directory)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:219)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:112)
        at Example3Streams.main(Example3Streams.java:9)


------------------
(program exited with code: 0)
Press return to continue
```

For solving:



```
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.IOException;

public class Example3Streams {
    public static void main(String[] args)  {
        try {
            // Create an InputStream for reading from a file
            InputStream inputStream = new FileInputStream("example2.txt");

            // Read bytes from the input stream
            int byteData;
            while ((byteData = inputStream.read()) != -1) {
                // Process the byte data (you can do anything with it)
                System.out.print((char) byteData); // Convert byte to char and print
            }

            // Close the input stream when done
            inputStream.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
javac "Example3Streams.java" (in directory: /home/bhawana/
Compilation finished successfully.
```

Terminal

```
this is example of simple I/O


------------------
(program exited with code: 0)
Press return to continue
```

Here's a brief example of how to use Java I/O streams to read and write data from/to a file:

```
import java.io.*;

public class FileStreamExample {
    public static void main(String[] args) {
        try {
            // Writing to a file
            String data = "Hello, this is a test!";
                                    OutputStream    outputStream    =    new
FileOutputStream("output.txt");
            outputStream.write(data.getBytes());
            outputStream.close();
            // Reading from a file
            InputStream inputStream = new FileInputStream("output.txt");
            byte[] buffer = new byte[1024];
            int bytesRead = inputStream.read(buffer);
            String readData = new String(buffer, 0, bytesRead);
            System.out.println("Data read from the file: " + readData);
            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

In this example, we use byte streams to write the string "Hello, this is a test!" to a file named "output.txt" and then read the data back from the file using byte streams as well. Remember to handle exceptions appropriately when working with I/O operations.

Example 2:

```
import java.io.*;

public class FileStreamExample {
    public static void main(String[] args) {
        try {
            // Writing to a file using FileOutputStream
FileOutputStream                    outputStream                    =new
FileOutputStream("example.txt");
```

```
                String dataToWrite = "Hello, this is an example of Java I/O
streams.";
            byte[] bytes = dataToWrite.getBytes();
            outputStream.write(bytes);
            outputStream.close();

            // Reading from a file using FileInputStream
    FileInputStream inputStream = new FileInputStream("example.txt");
            int data;
            while ((data = inputStream.read()) != -1) {
                System.out.print((char) data);
            }
            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```