I/O Streams

☑ Favorite	☐
↗ Tag	Java Network Programming

▼ Notes

▼ What is a stream?

I/O is about reading and writing data. I/O in Java is built on streams. A stream is a communication channel that a program has with the outside world. Java IO streams are flows of data you can either read from, or write to. It's called a stream because it's like a stream of water that continues to flow. Streams are typically connected to a data source, or data destination, like a file, devices, other programs, or a network connection.

▼ What is the purpose of `DataOutputStream` ?

To write primitive data types (such as `int`, `double`, `boolean`, etc.) as binary data to an underlying output stream

- Particularly useful when you need to serialize data, for example, when writing data to a file or transmitting it over a network.

▼ InputStreamWriter and OutputStreamWriter subclasses

1. **OutputStreamWriter Subclasses**:
   - `FileWriter`: A subclass of `OutputStreamWriter` that writes characters to a file using the default character encoding of the platform.
   - `BufferedWriter`: A buffered character-output stream that wraps another writer and improves performance by reducing the number of I/O operations.
   - `PrintWriter`: A character-output stream that provides convenient methods for writing common data types as text, along with automatic flushing.
2. **InputStreamReader Subclasses**:

- `FileReader`: A subclass of `InputStreamReader` that reads characters from a file using the default character encoding of the platform.
- `BufferedReader`: A buffered character-input stream that wraps another reader and improves performance by reducing the number of I/O operations.
- `LineNumberReader`: A buffered character-input stream that keeps track of line numbers as characters are read, allowing you to retrieve the current line number.
- `PushbackReader`: A character-input stream that allows characters to be pushed back into the stream, enabling you to "unread" characters that have been read.

▼ What is the difference between flush() and close()

The close method calls the flush() method. After you call the close() method you can't write() anything afterwards.

1. **flush():**
   - The `flush()` method is used to flush any buffered data from the stream to its destination without closing the stream itself.
   - When you call `flush()`, it ensures that any buffered data in the output stream is written to the underlying destination immediately.
   - Flushing the stream can be useful when you want to ensure that data is written out promptly without waiting for the buffer to fill up completely.
2. **close():**
   - The `close()` method is used to close the stream, releasing any system resources associated with it.
   - When you call `close()`, it not only flushes any buffered data but also closes the stream, making it unavailable for further writes or reads.
   - Closing the stream is typically done when you're finished using it and want to release any resources it's holding, such as file handles

or network connections.

- Once a stream is closed, further attempts to write to or read from it will result in an exception.

▼ Charset

Find what is the default charset. To do this we're gonna use the getEncoding() method. So I'm going to do System.out.println(out.getEncoding)). Run file. I get UTF8 as the default charset.The default charset of the Java Virtual Machine is that of the system is running on. UTF-8 can represent any character in the Unicode standard. And because the default charset is UTF-8 we can write "hello" in Japanese, for example.

▼ Code Examples

▼ Character Output Stream I

```
import java.io.*;

public class CharacterStream {

    public static void main(String [] args)
    {
        try
        {
            OutputStreamWriter out = new OutputStreamWrite
            InputStreamReader in = new InputStreamReader(r


            out.write("Hello CharacterStream!");

            out.flush();

            out.write("another string");

            out.flush();
            out.close();
```

```
        }
        catch(Exception e)
        {
            System.err.println(e.toString());
        }
    }
}
```

▼ Character Output Stream II

```
import java.io.*;

public class CharacterStream {

    public static void main(String [] args)
    {
        try
        {
            OutputStreamWriter out = new OutputStreamWrite
            InputStreamReader in = new InputStreamReader(r

            System.out.println(out.getEncoding());

            //out.write("reading using InputStreamReader"

            out.write("こんにちは")

            //out.flush();

            //out.close(); //calls the flush method

            //out.write("another string");
```

```
            out.flush();
            out.close();



        }
        catch(Exception e)
        {
            System.err.println(e.toString());
        }
    }
}
```

▼ Character Output Stream III

```
import java.io.*;

public class CharacterStream {

    public static void main(String [] args)
    {
        try
        {
            OutputStreamWriter out = new OutputStreamWrite
            InputStreamReader in = new InputStreamReader(r


            out.write("reading using InputStreamReader");

            //out.flush();

            //out.close(); //calls the flush method

            //out.write("another string");
```

```
            out.flush();
            out.close();

            int data = in.read();
             while(data != -1)
            {
                System.out.print((char)data);
                data = in.read();
            }

        }
        catch(Exception e)
        {
            System.err.println(e.toString());
        }
    }
}
```

▼ Buffered Stream

```
import java.io.*;

public class BufferStream
{
    public static void main(String args [])
    {
        try
        {
            BufferedReader reader = new BufferedReader(new
            BufferedWriter writer = new BufferedWriter(new

            String line = null;

            while((line = reader.readLine()) != null)
```

```
            {
                writer.write(line);
                writer.newLine();
            }
            writer.close();
            reader.close();
        }
        catch(Exception e)
        {
            System.err.println(e.toString());
        }
    }
}
```

▼ Print Stream

```
import java.io.*;

public class PrtStream
{
    public static void main(String [] args)
    {
        try
        {
            PrintStream out = new PrintStream("example8.tx
            //PrintStream out = new PrintStream(new File('

            int var1 = 10;

            System.out.println("The value of var1 is: " +

            out.println("The value of var1 is: " + var1);
            //out.close();

        }
```

```
        catch(FileNotFoundException e)
        {
            System.out.println(e.toString());
        }

    }
}
```

▼ System.in

```
import java.io.*;

public class SystemIn
{
    public static void main(String [] args)
    {
        System.out.print("Please enter the port number: "

        InputStreamReader in = new InputStreamReader(Syste

        /*
        int data = in.read();
        while(data != -1)
        {
            System.out.print((char)data);
            data = in.read();
        }
        */

        BufferedReader reader  = new BufferedReader(in);

        boolean isValid = false;
        int port = 0;

        while(!isValid)
```

```
        {
            try
            {
                String portString = reader.readLine();
                port = Integer.parseInt(portString);
                System.out.println("Port is accepted!");
                isValid = true;
            }
            catch(Exception e)
            {
                System.out.println("Please insert a numbe
                System.out.printf("Please enter the port 
            }
        }

        System.out.print("Please enter Server IP address:

        String ipAddress = "";

        try
        {
            ipAddress = reader.readLine();
        }
        catch(Exception e)
        {
            System.out.println("Cannot read the ip addres
        }

        System.out.println("");
        System.out.println("____________________________
        System.out.println("");

        System.out.println("Trying to connect to IP: " + 
```

```
    }
}
```