File:

To create files we use **File** Class, we can create **relative and absolute paths**

public **class java.io.File** implements **java.io.Serializable, java.lang.Comparable<java.io.File>**

```
//File Created which is relative to the path
File f1 = new File("cat.txt");
f1.createNewFile();
System.out.println("File Created");

//File Created which is relative to the path jvm will check with current existing directory
File f2 = new File("src/bat.txt");
f2.createNewFile();
System.out.println("File Created");

//File Created which is absolute to the path
File f3 = new File("E:\\Java Github\\CoreJavaPdf\\10 FileHandling\\dog.txt");
f3.createNewFile();
System.out.println("File Created");
```

Creating Directories and Sub Directories

```
//to create relative directory
File f1 = new File("Spring");
boolean mkdir = f1.mkdir();
System.out.println(mkdir);

//to create relative directory
File f2 = new File("CoreJava\\AdvancedJava\\Spring");
boolean mkdirs = f2.mkdirs();
System.out.println(mkdirs);
```

Stream

A stream is a sequence of data. In Java, **a stream is composed of bytes.**

java.io package contains all the classes required for **input and output operations.**

We can perform **file handling in Java by Java I/O API**.

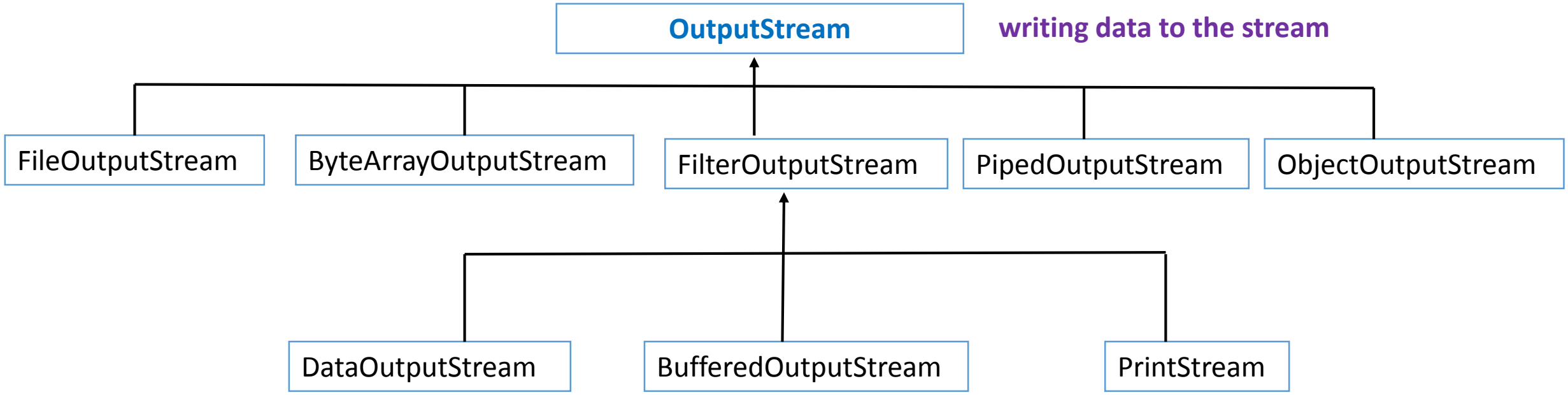
The **Stream** are divided in two ways they are **Byte Oriented Streams** and **Character Oriented Streams**

Types of Byte Oriented Streams

1.OutputStream (writing data to the stream)
public **abstract class java.io.OutputStream** implements **java.io.Closeable,java.io.Flushable**
2.InputStream (reading data to the stream)
public **abstract class java.io.InputStream** implements **java.io.Closeable**

Types Character Oriented Streams

1.Reader
public **abstract class java.io.Reader** implements **java.lang.Readable,java.io.Closeable**
2.Writer
public **abstract class java.io.Writer** implements **java.lang.Appendable,java.io.Closeable,java.io.Flushable**



public **abstract class java.io.OutputStream** implements **java.io.Closeable,java.io.Flushable**

OutputStream class is an **abstract class.**
It is the superclass of all classes representing an output stream of bytes.

FileWriter class is used to write **character oriented stream** to a file
Unlike **FileOutputStream** class, you don't need to **convert string to byte array** because it provides method to write String directly

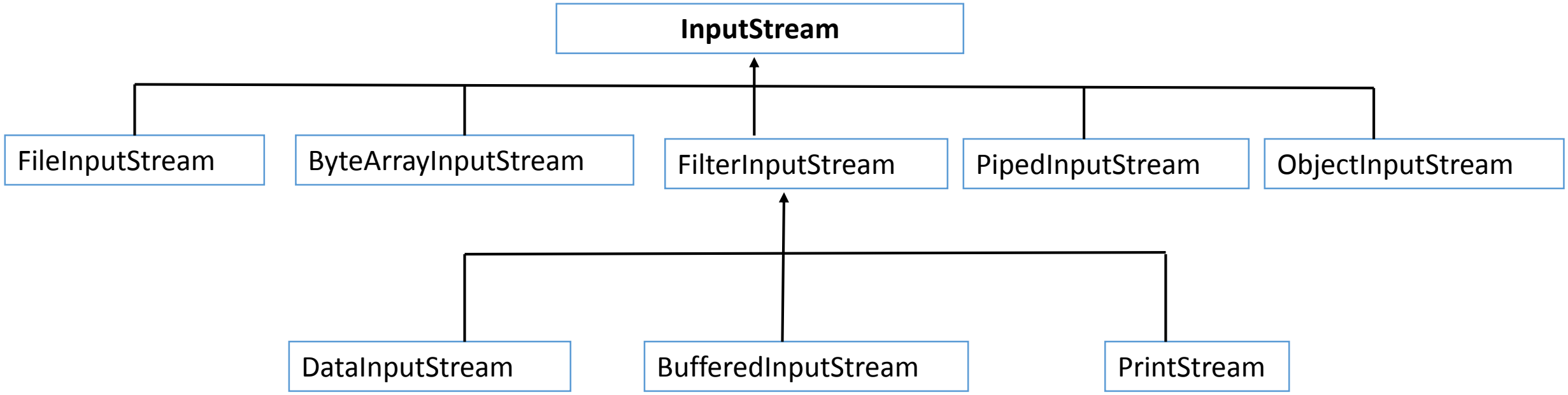
public class **java.io.FileWriter** extends **java.io.OutputStreamWriter**

```
FileWriter writer = new FileWriter("E:\\Java Github\\CoreJavaPdf\\10 FileHandling\\camel.txt");
writer.write("Hello Java and Hello Python");
writer.close();
System.out.println("File Created and Data Inserted");
```

FileOutputStream is an **OutputStream** for writing data to a **File** or to a **FileDescriptor.**
We can write byted oriented stream and character oriented stream, but for character oriented stream preferred to use **FileWriter**
public **class java.io.FileOutputStream** extends **java.io.OutputStream**

```
FileOutputStream fileOutputStream = new FileOutputStream("one.txt"); //data overides
//FileOutputStream fileOutputStream = new FileOutputStream("one.txt", true); // data appends
String s1 = "Java and Python and JavaScript";
byte[] bytes = s1.getBytes(); // Calling the getBytes() from String Class which return byte array
fileOutputStream.write(bytes); // write() is present in FileOutputStream class, which will write bytes data
System.out.println("New File Created and Data Inserted");
fileOutputStream.close();
```

New File Created and Data Inserted
Note: **writing data to the stream**


InputStream
FileInputStream
ByteArrayInputStream
FilterInputStream
PipedInputStream
ObjectInputStream
DataInputStream
BufferedInputStream
PrintStream

FileReader class is used to read data from the file. It is **character oriented stream**
It can be used to **reading text files**. Here read() reads **2 byte(16 bit)** at a time
It returns data in **byte format** like **FileInputStream class.** It is used when we are **reading text files, pdfs, or word documents**

public class **java.io.FileReader** extends **java.io.InputStreamReader**

```
FileReader reader = new FileReader("one.txt");
int i;
while ( (i = reader.read() ) != -1)
System.out.print((char) i);
reader.close();
```

reading data to the stream

Java and Python and JavaScript

read() method is present in **InputStreamReader** which will **return int type**

FileInputStream can be used to **read data** from files.It is used for reading **byte-oriented data**.
Here read() reads **1 byte (8 bit)** at a time. It is used when we are **reading audio, video or other multimedia files**

You can also read **character-stream data.** But, for reading **streams of characters**, it is recommended to use **FileReader** class.

public class java.io.**FileInputStream** extends java.io.**InputStream**

```
FileInputStream stream = new FileInputStream("one.txt");
int i;
while ( (i = stream.read() ) != -1) //read() method from FileInputStream
System.out.print((char) i);
stream.close();
}
```

reading data to the stream

Java and Python and JavaScript

available() method to check the number of available bytes in **FileInputStream**.
read() method 4 times to read 4 bytes from the **FileInputStream**
After reading the bytes again have checked the available bytes.

```
FileInputStream stream = new FileInputStream("one.txt");
// returns the number of available bytes
System.out.println("Bytes Before: " + stream.available());
// reads 4 bytes from the file
stream.read();
stream.read();
stream.read();
stream.read();
// returns the number of available bytes
System.out.println("Bytes After: " + stream.available());
stream.close();
```

reading data to the stream

```
Bytes Before: 30
Bytes After: 26
```

BufferReader class is used to read the text from **character oriented stream**
It can be used to read the data line by line by using **readLine()** method

```
System.out.println("Enter any number here: ");
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
String readLine = reader.readLine();
int parseInt = Integer.parseInt(readLine);
System.out.println("Entered value is:  " + parseInt);
```

The **InputStream** object **System.in** takes keyboard input, and put the data in **Java byte format**
An **InputStreamReader** object takes the **data, translate it from byte oriented stream to character oriented stream**
An **BufferReader object** buffer the character inputs, and allows the program to process the data

Here **in** is static variable defined in **Sytem** class
public static final InputStream *in = null;*

in represents a predefined **InputStream** object
public abstract class InputStream implements Closeable

readLine() method is used to read a line of data from BufferReader as String

read() method is able to read single character from BufferReader in the from of ASCII Value

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter Your Name: ");
String line = reader.readLine(); //readLine() is present in BufferedReader class
System.out.println("Enter Your Name: ");
int i = reader.read(); //read() is present in BufferedReader class
System.out.println(line);
System.out.println((char)i);
```

Note:
In case if we use InputStreamReader directly the performance of input operator is reduced, to improve performance then we use BufferReader

FileInputStream	FileReader	FileOutputStream-vs-Filewriter
Stream is Byte Based, it can be used to read bytes or write bytes.	**Reader is Character Based**, it can be used to read or write characters.	When we use Java to write something to a file, we can do it in the following two ways. One uses FileOutputStream, the other uses FileWriter. **FileOutputStream** is an OutputStream for writing data to a File or to a FileDescriptor. We can write **byted oriented stream** and character oriented stream, but for character oriented stream preferred to use FileWriter **public class java.io.FileOutputStream extends java.io.OutputStream** **FileWriter** class is used to write **character oriented stream** to a file Unlike FileOutputStream class, you don't need to convert string to byte array because it provides method to write String directly **public class java.io.FileWriter extends java.io.OutputStreamWriter**
Stream is used to binary input/output	FileReader is Character Based, it can be used to read characters.	
FileInputStream is used for reading binary files.	FileReader is used for reading text files	
FileInputStream and ObjectInputStream can be used for Serialization and DeSerialization, where serialized object can be persisted in file. In Serialization object is converted into byte stream and in deserialization it is converted back from byte to object.	FileReader is not used for Serialization and DeSerialization, as it reads characters not bytes.	
FileInputStream.read() reads 1 byte (8-bit) at a time.	FileReader.read() reads 2 bytes(16-bit) at a time, because char is 16-bit data type.	
FileInputStream must be used when we are reading audio, video or other multimedia files	FileReader must be used when we are reading text files, pdfs or word documents.	
public class java.io.**FileInputStream** extends java.io.**InputStream**	public class **java.io.FileReader** extends **java.io.InputStreamReader**	