Unit 6 - File Handling

Streams

- A stream is a **sequence of data**. In Java a stream is composed of bytes.
- In java, 3 streams are created for us automatically.
  1. **System.out :** standard output stream
  2. **System.in :** standard input stream
  3. **System.err :** standard error stream

Byte Streams

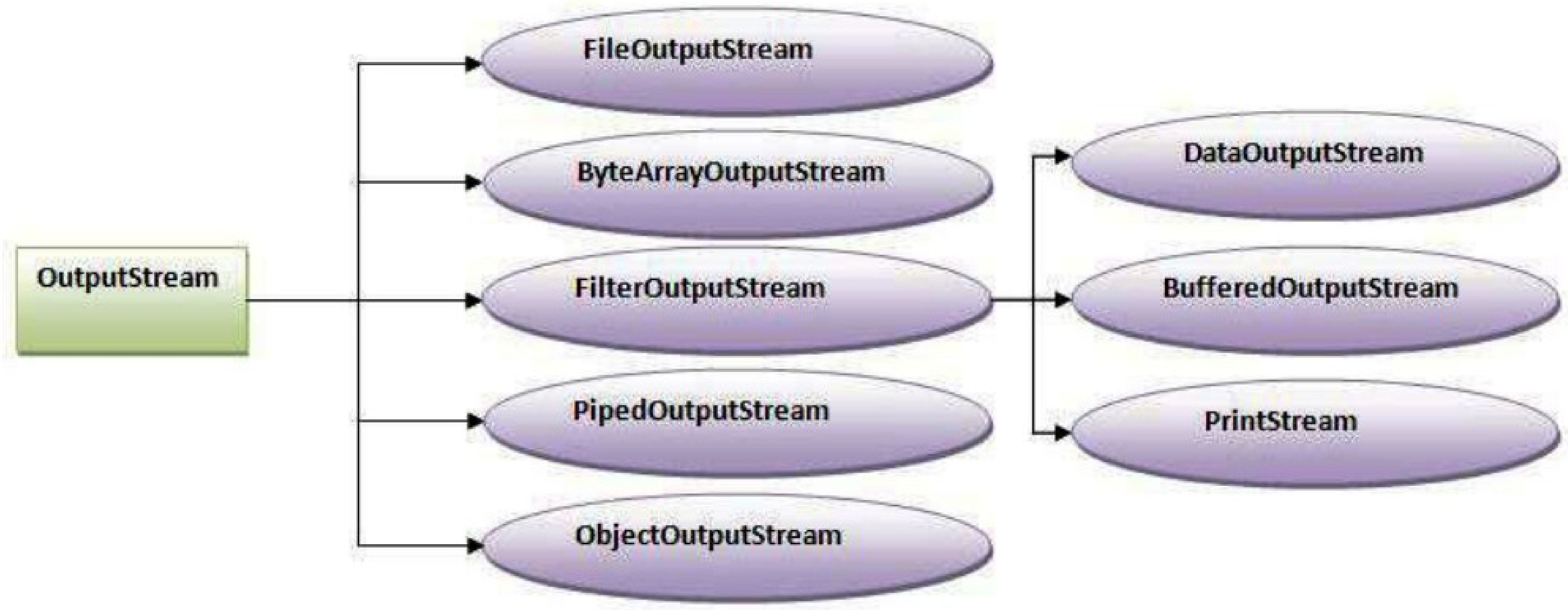
- Java byte streams are used to perform **input and output of 8-bits / 1 byte** at a time **from binary file.**
- **InputStream** and **OutputStream class** are most common used classes of byte stream**.**

Byte Streams classes

Stream class	Description
BufferedInputStream	Used for Buffered input stream.
BufferedOutputStream	Used for Buffered output stream.
DataInputStream	Contains method for reading java standard data type.
DataOutputStream	An output stream that contain method for writing java standard data type.
FileInputStream	Input stream that reads from a file.
FileOutputStream	Output stream that write to a file.
InputStream	Abstract class that describe stream input.
OutputStream	Abstract class that describe stream output.
PrintStream	Output stream that contain print() and println() method.

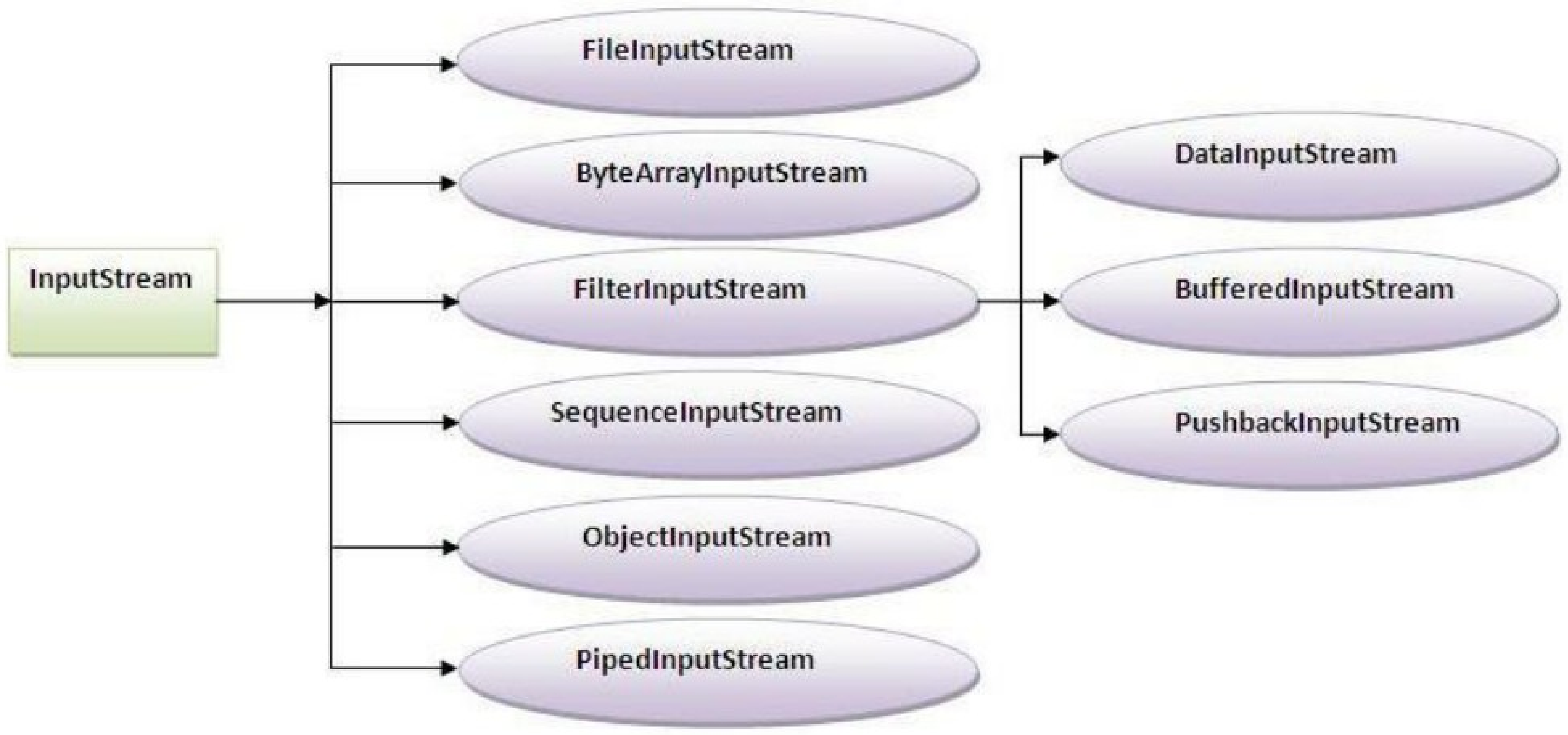
FileOutputStream

- FileOutputStream is used to **create a file** and **write data** into it.
- The OutputStream will only create a file, if it doesn't already exist, before opening it for output.
- There are two way for using file with **FileOutputStream**.
  1. **FileOutputStream f = new FileOutputStream("C:/java/hello.txt");**
     - ✓ Here, directly File path is specified while creating the FileOutputStream object.
  2. **File f = new File("C:/java/hello.txt");**
     FileOutputStream f1 = new FileOutputStream(f);
     - ✓ In the second method, first object of **File** is created and then **FileOutputStream** object is created and **File** object passed to the **FileOutputStream** as a argument.



FileInputStream

- FileInputStream is used for **reading data from the files.**
- While using FileInputStream class, file must already exist if file does not exist it will generate **error**.
- There are two way for using file with **FileInputStream**.
  1. **FileInputStream f1 = new FileInputStream("C:/java/hello.txt");**
     - ✓ Here, directly File path is specified while creating the FileInputStream object.
  2. **File f = new File("C:/java/hello.txt");**
     FileInputStream f1 = new FileInputStream(f);
     - ✓ In the second method, first object of **File** is created and then **FileInputStream** object is created and **File** object passed to the **FileInputStream** as a argument.



Example : Write a java program that writes the Hello.txt file into Hello1.txt using FileInputStream and FileOutputStream.

Solution: File_1.java

```
import java.io.*;
public  class  File_1
{
        public static void main(String[] args)
        {
                try
                {
                        FileInputStream in = new FileInputStream("C:/temp/Hello.txt ");
                        FileOutputStream out = new FileOutputStream("C:/temp/Hello1.txt ");
                        int c = 0;
                        while ((c = in.read())!=-1)
                        {
                                out.write(c);
                        }
                        in.close();
                        out.close();
                        System.out.println("Successfully Write");
                }
                catch (FileNotFoundException e)
                {
                        e.printStackTrace();
                }
                catch (IOException e)
                {
                        e.printStackTrace();
                }
        }
}
```

Output:

```
Successfully Write
```

Character Streams

- Character streams are used to perform **input and output of 16-bits / 2 bytes** at a time **from Character file.**
- **Reader** and **Writer** are most common used classes of character stream.

Character Stream classes

Stream class	Description
BufferedReader	Handles buffered input stream.
BufferedWriter	Handles buffered output stream.
FileReader	Input stream that reads from file.
FileWriter	Output stream that writes to file.
InputStreamReader	Input stream that translate byte to character.
OutputStreamReader	Output stream that translate character to byte.
Reader	Abstract class that define character stream input.
Writer	Abstract class that define character stream output.

FileWriter

- Java FileWriter class is used to **write character data to the file.**
- Two ways to create **FileWriter** Object.
    1. FileWriter f = new FileWriter(String fname); **//File path is specified in argument.**
    2. FileWriter f = new FileWriter(File f1); **//1st file is created and it is passed as argument.**

Method of FileWriter Class

void write(int ch)	Writing single character to the file.
void write(String s)	Writes the string into FileWriter.
void write(char[] c)	Writes char array into FileWriter.
void close()	Closes FileWriter.

FileReader

- Java FileReader class is used to **read character data from the file.**
- Two ways to create **FileReader** Object.
    1. FileReader f = new FileReader(String name); **//File path is specified in argument.**
    2. FileReader f = new FileReader(File f1); **//1st file is created and it is passed as argument.**

Method of FileReader Class

int read()	Returns a character in ASCII form. It returns -1 at the end of file.
int read(char[] ch)	To read data from the file into char array.
void close()	Closes FileReader.

Example :Write a java program that writes the Hello.txt file into Hello1.txt using FileReader and FileWriter.

Solution: File_2.java

```
import java.io.*;
public class File_2
{
        public static void main(String[] args) throws IOException
        {
                File f = new File("d:/temp/Hello.txt");
                File f1 = new File("d:/temp/Hello1.txt");
                FileReader fr = new FileReader(f);
                FileWriter fw = new FileWriter(f1);
                int c = 0;
                while ((c = fr.read())!=-1)
                {
                        fw.write(c);
                }
                fr.close();
                fw.close();
                System.out.println("Successfully Write");
        }
}
```

Output:

```
Successfully Write
```

BufferedReader

- BufferedReader class can be used to **read data line by line using of readLine() method**.

BufferedOutputStream

- BufferedOutputStream class uses an internal **buffer to store data.**
- It adds more efficiency than to write data directly into a stream. So, it makes the performance fast.

BufferedInputStream

- BufferedInputStream class is used to **read information from stream.** It internally uses buffer mechanism to make the performance fast.

Example : Write a program in java that take character as a input from console and write it in file develop this program using Bufered Classes.

Solution: File_3.java

```
import java.io.*;
public class File_3
{
        public static void main(String[] args)
        {
                InputStreamReader in = new InputStreamReader(System.in);
                BufferedReader br = new BufferedReader(in);
                File f = new File("d:/temp/Hello.txt");
                try
                {
                        OutputStream out = new FileOutputStream(f);
                        System.out.print("Type your text to write in file===");
                        String str;
                        str = br.readLine();
                        byte b[] = str.getBytes();
                        out.write(b);
                        System.out.println("File Successfully Write..");
                        br.close();
                        in.close();
                        out.close();
                }
                catch (IOException e)
                {
                        e.printStackTrace();
                }
        }
}
```

Output:

```
Type your text to write in file===i like an ice cream
File Successfully Write..
```