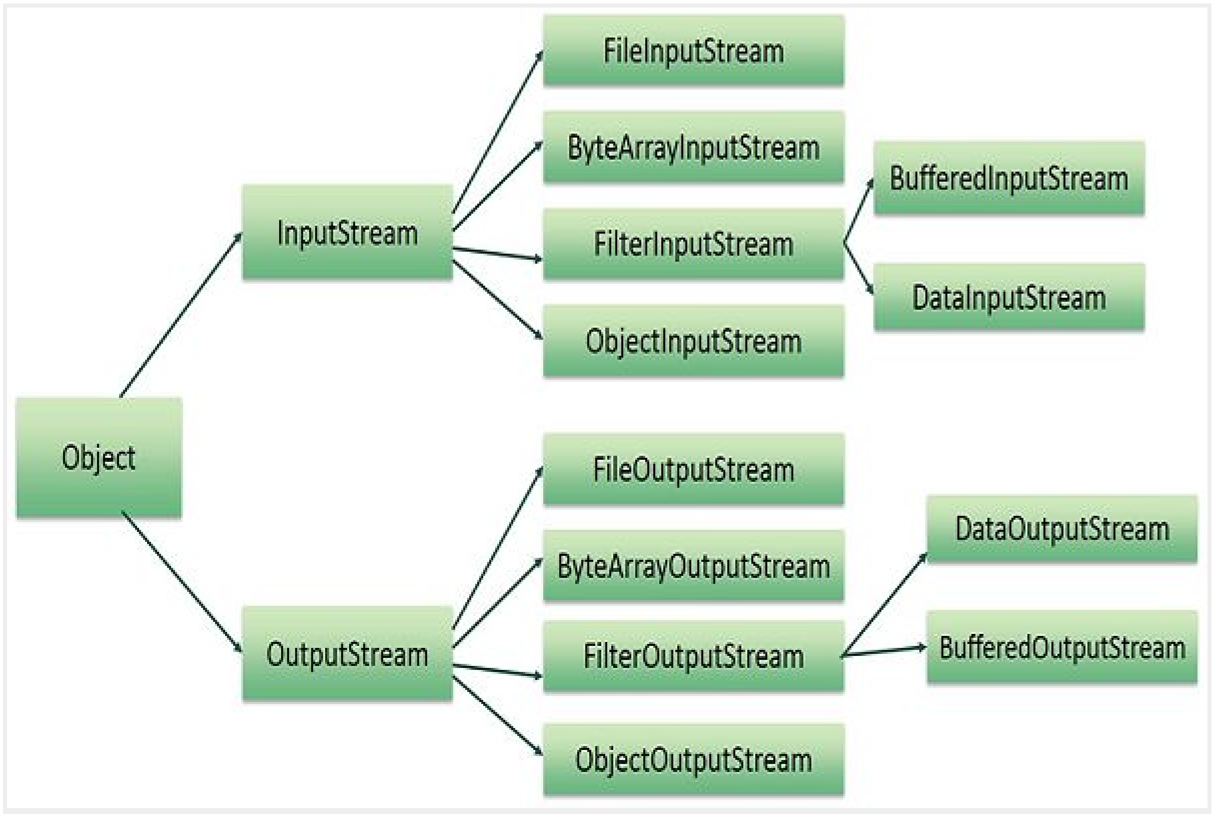

```
import java.io.*;
public class CopyFile {

   public static void main(String args[]) throws IOException {
      FileInputStream in = null;
      FileOutputStream out = null;

      try {
         in = new FileInputStream("input.txt");
         out = new FileOutputStream("output.txt");

         int c;
         while ((c = in.read()) != -1) {
            out.write(c);
         }
      }finally {
         if (in != null) {
            in.close();
         }
         if (out != null) {
                out.close();
            }
        }
    }
}
```

File handling in Java using FileWriter and FileReader

Java FileWriter and FileReader classes are used to write and read data from text files (they are Character Stream classes). It is recommended **not** to use the FileInputStream and FileOutputStream classes if you have to read and write any textual information as these are Byte stream classes.

FileWriter

FileWriter is useful to create a file writing characters into it.

- This class inherits from the OutputStream class.
- The constructors of this class assume that the default character encoding and the default byte-buffer size are acceptable. To specify these values yourself, construct an OutputStreamWriter on a FileOutputStream.
- FileWriter is meant for writing streams of characters. For writing streams of raw bytes, consider using a FileOutputStream.
- FileWriter creates the output file , if it is not present already.

Constructors:

- **FileWriter(File file) –** Constructs a FileWriter object given a File object.

- **FileWriter (File file, boolean append) –** constructs a FileWriter object given a File object.
- **FileWriter (FileDescriptor fd) –** constructs a FileWriter object associated with a file descriptor.
- **FileWriter (String fileName) –** constructs a FileWriter object given a file name.
- **FileWriter (String fileName, Boolean append) –** Constructs a FileWriter object given a file name with a Boolean indicating whether or not to append the data written.

Methods:

- **public void write (int c) throws IOException –** Writes a single character.
- **public void write (char [] stir) throws IOException –** Writes an array of characters.
- **public void write(String str)throws IOException –** Writes a string.
- **public void write(String str,int off,int len)throws IOException –** Writes a portion of a string. Here off is offset from which to start writing characters and len is number of character to write.
- **public void flush() throws IOException** flushes the stream
- **public void close() throws IOException** flushes the stream first and then closes the writer.

```
// Creating a text File using FileWriter
import java.io.FileWriter;
import java.io.IOException;
class CreateFile
{
	public static void main(String[] args) throws IOException
```

```
        {
                // Accept a string
                String str = "File Handling in Java using "+
                                " FileWriter and FileReader";

                // attach a file to FileWriter
                FileWriter fw=new FileWriter("output.txt");

                // read character wise from string and write
                // into FileWriter
                for (int i = 0; i < str.length(); i++)
                        fw.write(str.charAt(i));

                System.out.println("Writing successful");
                //close the file
                fw.close();
        }
}
```

FileReader

FileReader is useful to read data in the form of characters from a ‘text’ file.

- This class inherit from the InputStreamReader Class.
- The constructors of this class assume that the default character encoding and the default byte-buffer size are appropriate. To specify these values yourself, construct an InputStreamReader on a FileInputStream.
- FileReader is meant for reading streams of characters. For reading streams of raw bytes, consider using a FileInputStream.

Constructors:

- **FileReader(File file) –** Creates a FileReader , given the File to read from
- **FileReader(FileDescripter fd) –** Creates a new FileReader , given the FileDescripter to read from

- **FileReader(String fileName) –** Creates a new FileReader , given the name of the file to read from

Methods:

- **public int read () throws IOException –** Reads a single character. This method will block until a character is available, an I/O error occurs, or the end of the stream is reached.
- **public int read(char[] cbuff) throws IOException –** Reads characters into an array. This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.
- **public abstract int read(char[] buff, int off, int len) throws IOException –**Reads characters into a portion of an array. This method will block until some input is available, an I/O error occurs, or the end of the stream is reached.
  Parameters:
  cbuf – Destination buffer
  off – Offset at which to start storing characters
  len – Maximum number of characters to read
- **public void close() throws IOException** closes the reader.
- **public long skip(long n) throws IOException –**Skips characters. This method will block until some characters are available, an I/O error occurs, or the end of the stream is reached.
  Parameters:
  n – The number of characters to skip

Following program depicts how to read from the ‘text’ file using FileReader

```
// Reading data from a file using FileReader
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
class ReadFile
{
	public static void main(String[] args) throws
IOException
	{
		// variable declaration
		int ch;

		// check if File exists or not
		FileReader fr=null;
		try
		{
			fr = new FileReader("text");
		}
		catch (FileNotFoundException fe)
```

```
        {
            System.out.println("File not found");
        }

        // read from FileReader till the end of file
        while ((ch=fr.read())!=-1)
            System.out.print((char)ch);

        // close the file
        fr.close();
    }
}
```