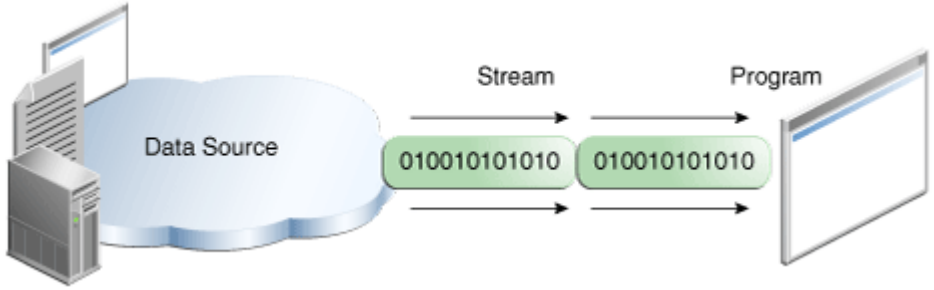
Funções de IO em Java

I/O Streams

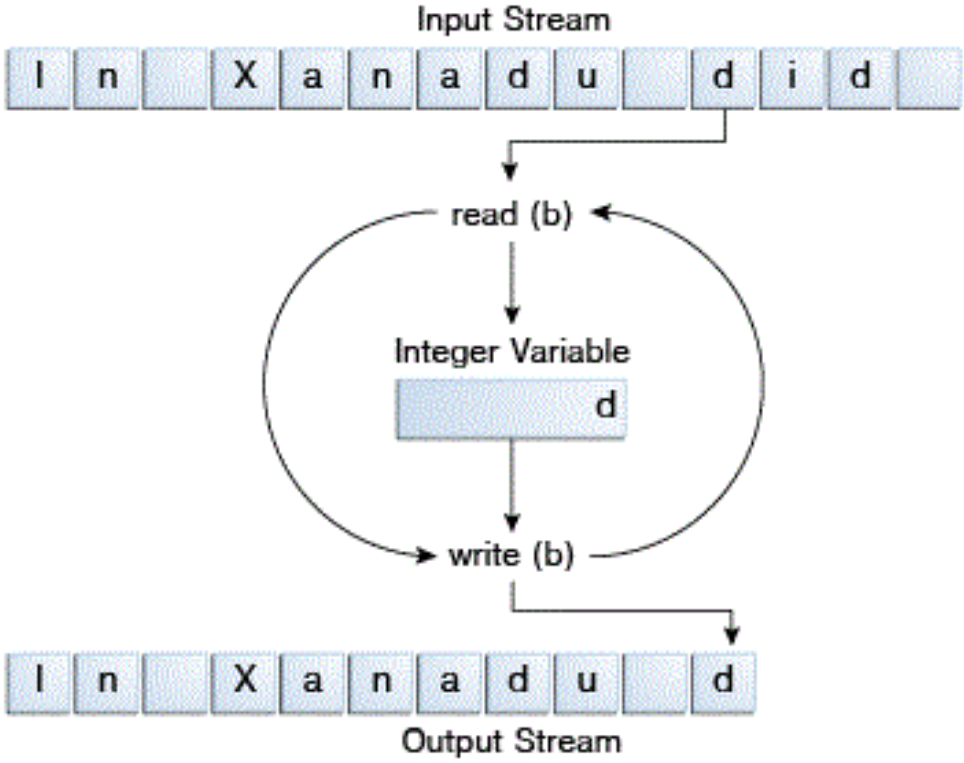
- Um fluxo de E/S pode representar uma origem ou um destino de dados. Os stream representam o fluxo de informação que fluem de uma fonte de dados para a aplicação (entrada) ou da aplicação para uma fonte de dados (destino).
- As fontes de dados podem ser matrizes de memórias, a arquivos ou dispositivos.
- Os fluxos permitem a manipulação de diferentes tipos de dados, de simples sequência de bytes, tipos primitivos, caracteres localizados a objetos.
- Alguns fluxos de dados simplesmente transmitem outros manipulam e transformam esses dados.
- A implementação interna dos fluxos tem pouca relevância para o desenvolvedor, essencialmente os programas usam os fluxos.



copybytes.java

aplicação de byte-stream

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class CopyBytes {
    public static void main(String[] args) throws IOException {
        FileInputStream in = null;
        FileOutputStream out = null;
        try {
            in = new FileInputStream("xanadu.txt");
            out = new FileOutputStream("outagain.txt");
            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
        } finally {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }
}
```



Character Streams

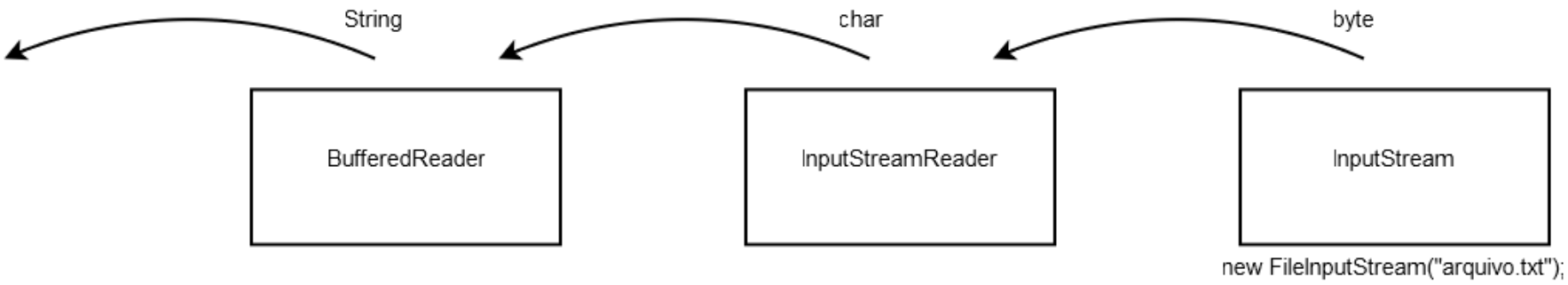
```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class CopyCharacters {
   public static void main(String[] args) throws IOException {
      FileReader inputStream = null;
      FileWriter outputStream = null;
      try {
         inputStream = new FileReader("xanadu.txt");
         outputStream = new FileWriter("characteroutput.txt");
         int c;
         while ((c = inputStream.read()) != -1) {
            outputStream.write(c);
         }
      } finally {
         if (inputStream != null) {  inputStream.close(); }
         if (outputStream != null) { outputStream.close(); }
      }
   }
}
```

Lendo arquivos “orientado a linhas”

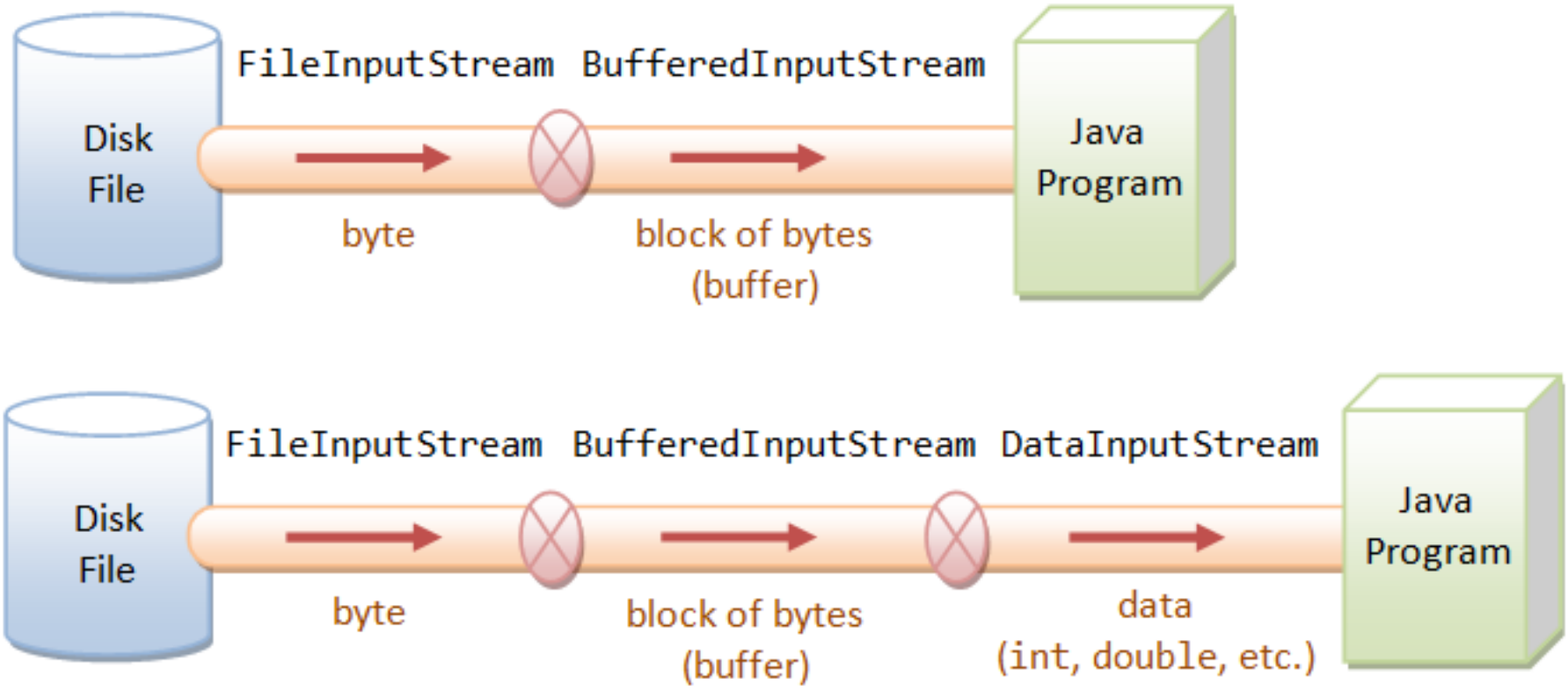
```
public class CopyLines {
   public static void main(String[] args) throws IOException {
      BufferedReader inputStream = null;
      PrintWriter outputStream = null;
      try {
         inputStream = new BufferedReader(new FileReader("xanadu.txt"));
         outputStream = new PrintWriter(new FileWriter("characteroutput.txt"));
         String l;
         while ((l = inputStream.readLine()) != null) {
            outputStream.println(l);
         }

      } finally {
         if (inputStream != null) { inputStream.close(); }
         if (outputStream != null) { outputStream.close();  }
      }
   }
}
```

Decorator Pattern

```
class TestaEntrada {
      public static void main(String[] args) throws IOException {
         InputStream is = new FileInputStream("arquivo.txt");
         InputStreamReader isr = new InputStreamReader(is);
         BufferedReader br = new BufferedReader(isr);
         String s = br.readLine();
      }
   }
```



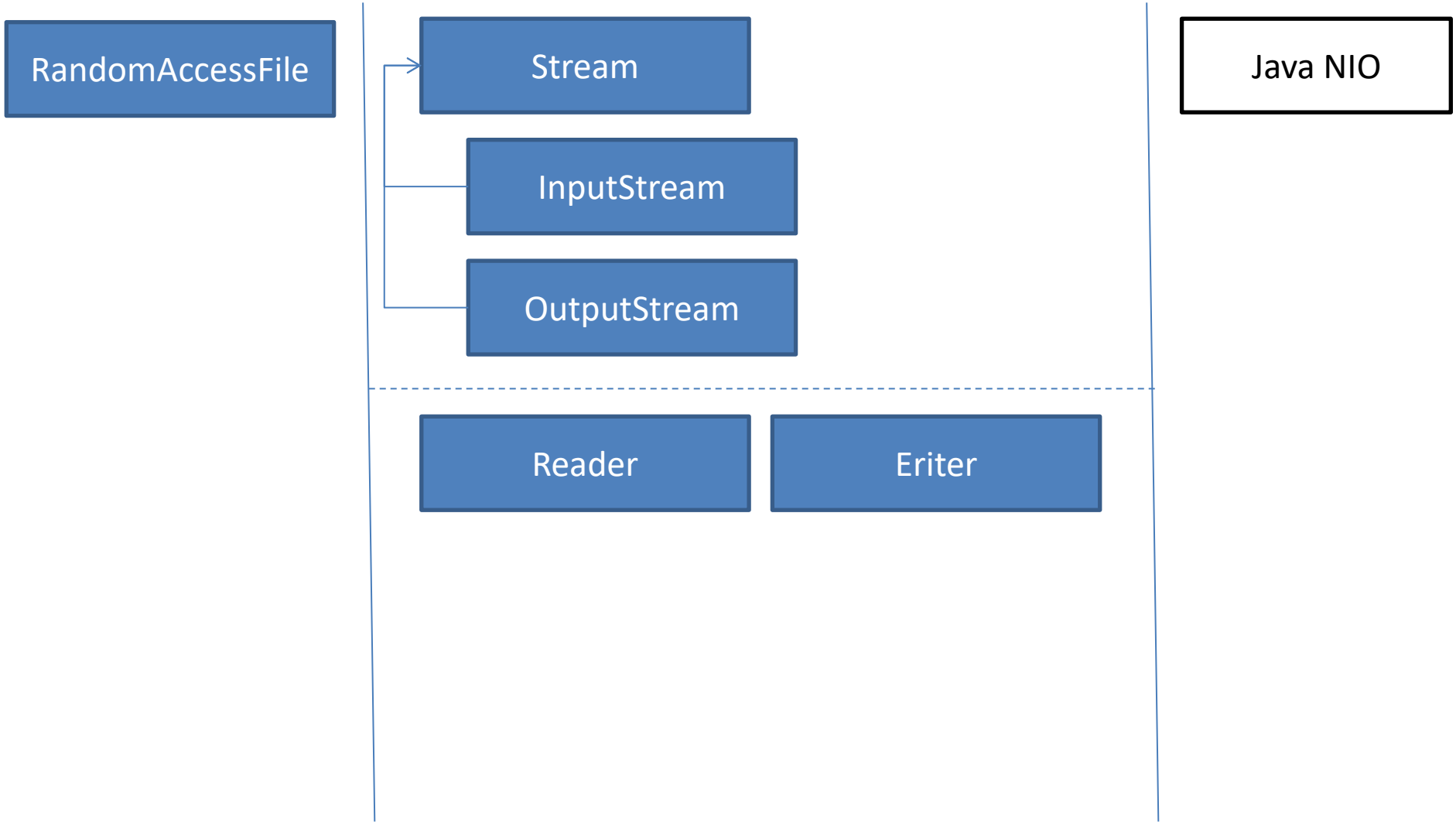
Decorator 2

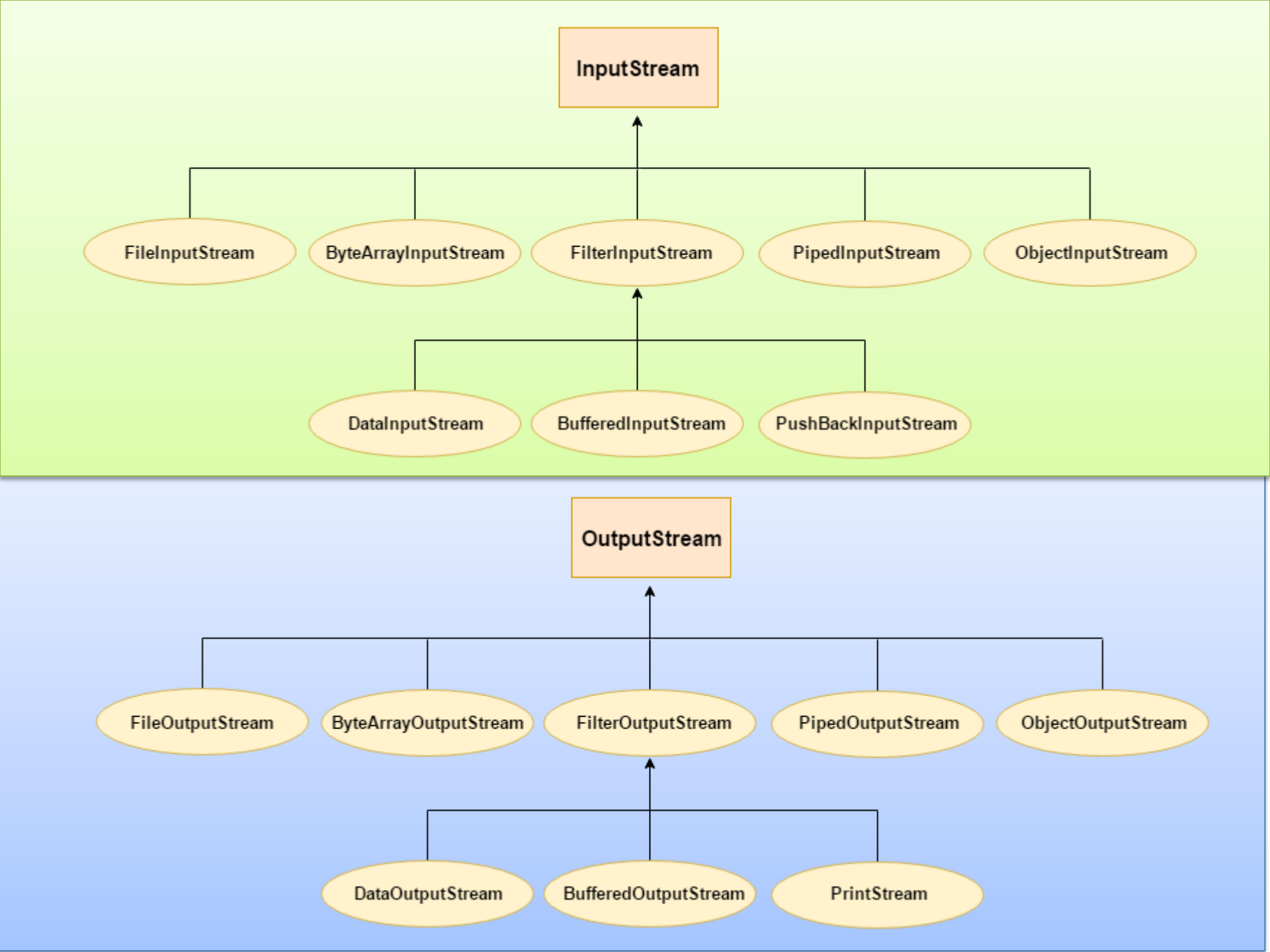

FileInputStream
BufferedInputStream
Disk File
Java Program
byte
block of bytes (buffer)
FileInputStream
BufferedInputStream
DataInputStream
Disk File
Java Program
byte
block of bytes (buffer)
data (int, double, etc.)

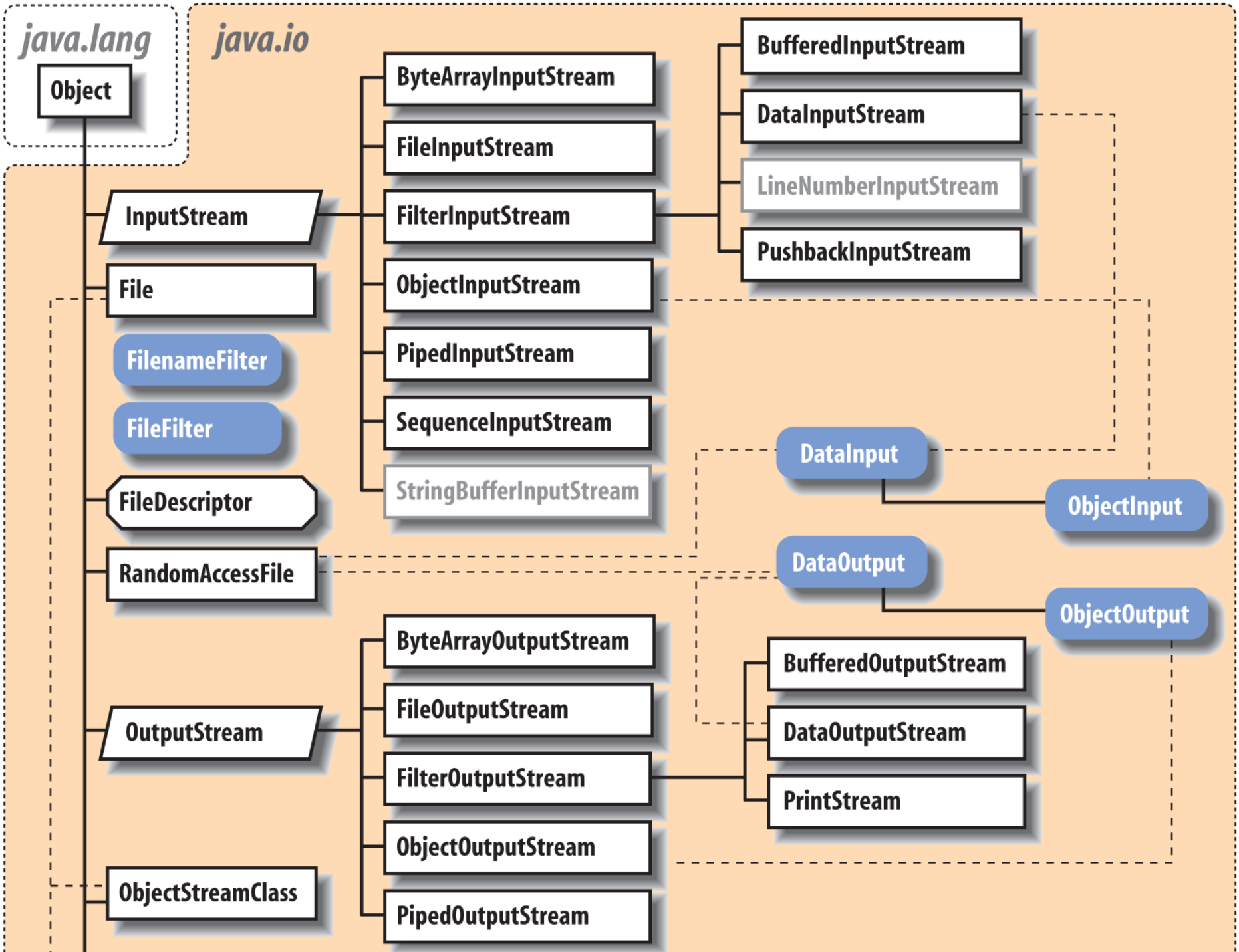
Random Access Files

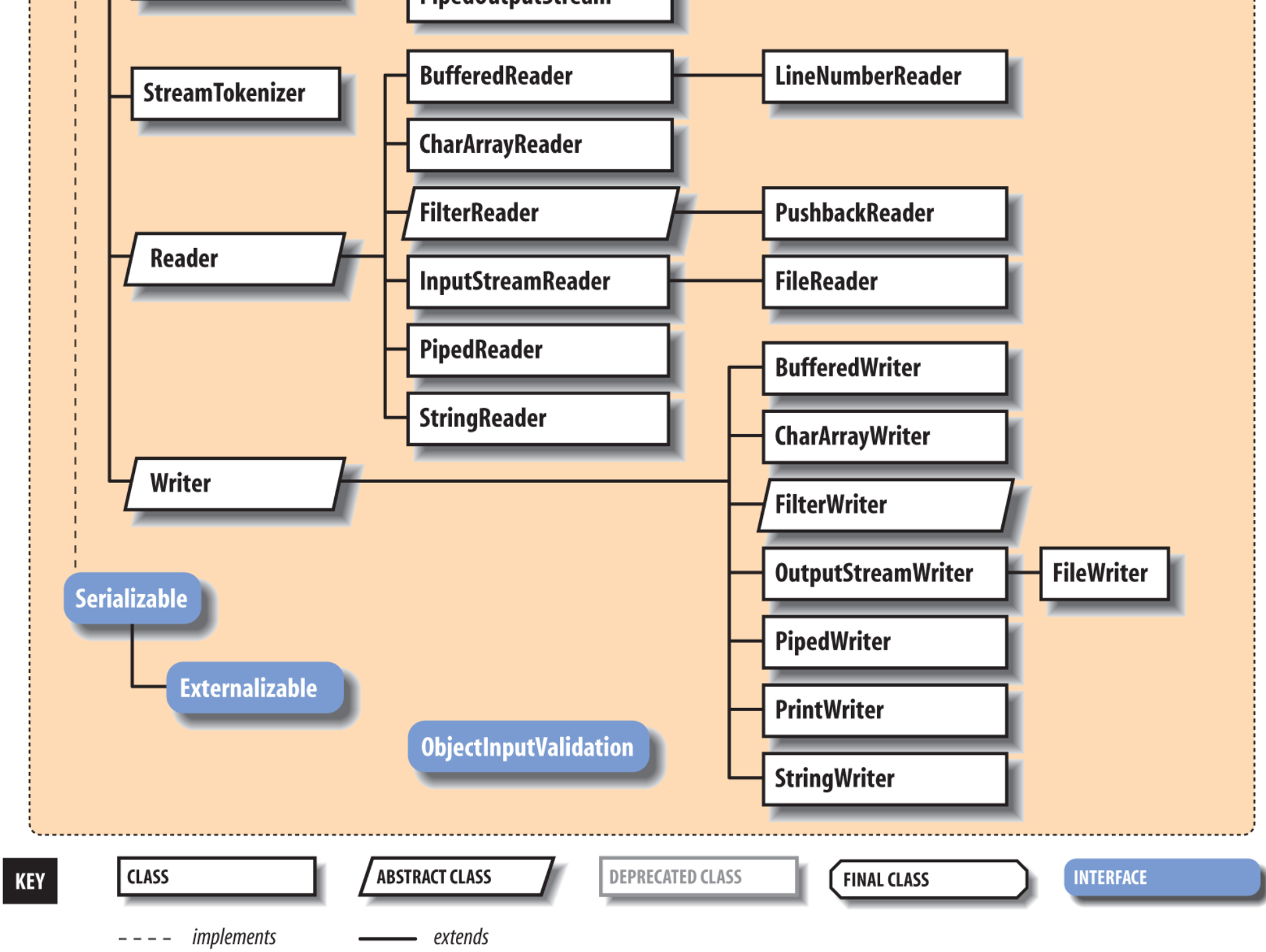
- position() – retorna a posição atual no arquivo.
- position(long) – define (set) a posição no arquivo.
- read(ByteBuffer) – Lê um conjunto de bytes
- write(ByteBuffer) – Escreve um conjunto de bytes.
- truncate(long) – redefine o tamanho do arquivo.

Visão Geral da API




InputStream
FileInputStream
ByteArrayInputStream
FilterInputStream
PipedInputStream
ObjectInputStream
DataInputStream
BufferedInputStream
PushBackInputStream
OutputStream
FileOutputStream
ByteArrayOutputStream
FilterOutputStream
PipedOutputStream
ObjectOutputStream
DataOutputStream
BufferedOutputStream
PrintStream


java.lang
Object
java.io
InputStream
File
FilenameFilter
FileFilter
FileDescriptor
RandomAccessFile
OutputStream
ObjectStreamClass
ByteArrayInputStream
FileInputStream
FilterInputStream
ObjectInputStream
PipedInputStream
SequenceInputStream
StringBufferInputStream
BufferedInputStream
DataInputStream
LineNumberInputStream
PushbackInputStream
DataInput
ObjectInput
DataOutput
ObjectOutput
ByteArrayOutputStream
FileOutputStream
FilterOutputStream
ObjectOutputStream
PipedOutputStream
BufferedOutputStream
DataOutputStream
PrintStream


StreamTokenizer
Reader
Writer
BufferedReader
CharArrayReader
FilterReader
InputStreamReader
PipedReader
StringReader
LineNumberReader
PushbackReader
FileReader
BufferedWriter
CharArrayWriter
FilterWriter
OutputStreamWriter
PipedWriter
PrintWriter
StringWriter
FileWriter
Serializable
Externalizable
ObjectInputValidation
KEY
CLASS
ABSTRACT CLASS
DEPRECATED CLASS
FINAL CLASS
INTERFACE
implements
extends

Referências

- CAELUM. Apostila - Capítulo 16 - Pacote java.io. https://www.caelum.com.br/apostila-java-orientacao-objetos/pacote-java-io/#inputstream-inputstreamreader-e-bufferedreader

- ORACLE. Basic I/O. https://docs.oracle.com/javase/tutorial/essential/io/

- ORACLE. https://docs.oracle.com/javase/8/docs/api/?java/io/File.html

- Java NIO: http://blog.caelum.com.br/evolucao-do-java-io-ao-ni/ http://tutorials.jenkov.com/java-nio/index.html

- **Chapter 12. Input/Output Facilities** http://quarkphysics.ca/ICS4U1/unit2-FileIO/OReilly_fileIO.html