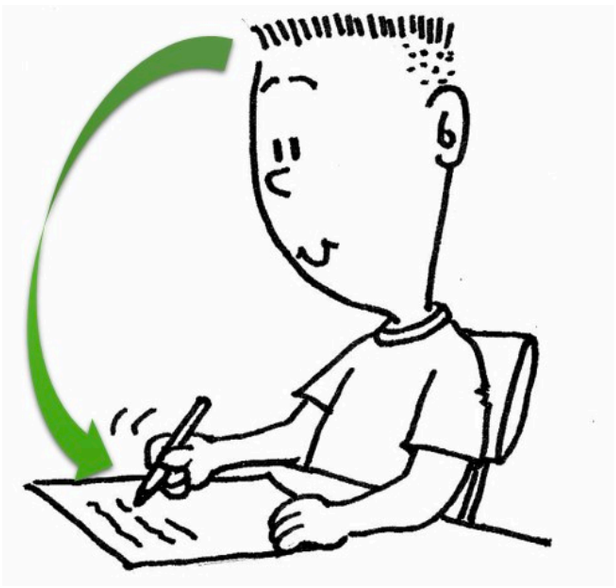
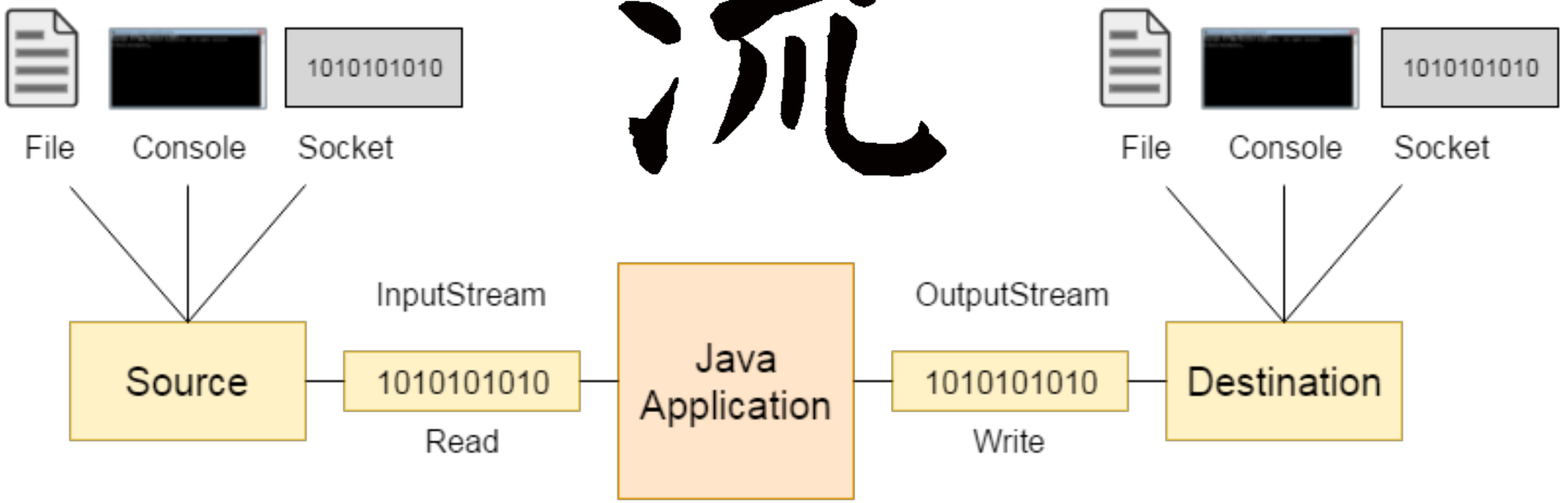
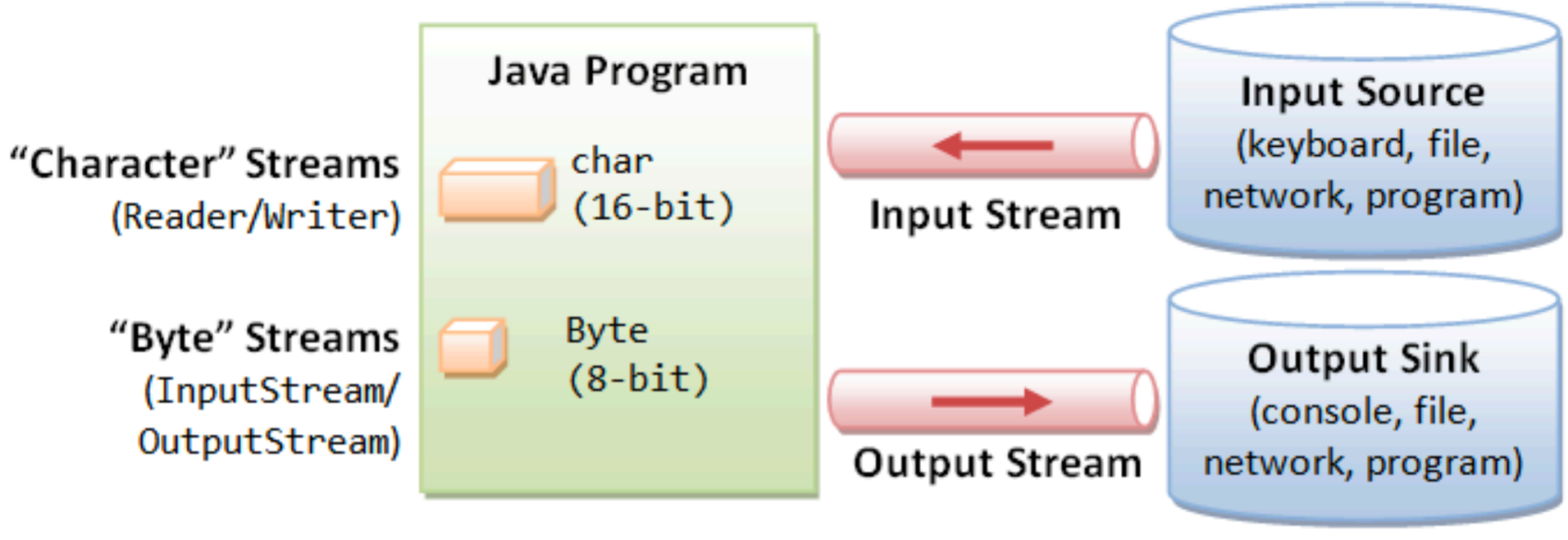
# 输入/输出

杨亮



# 流



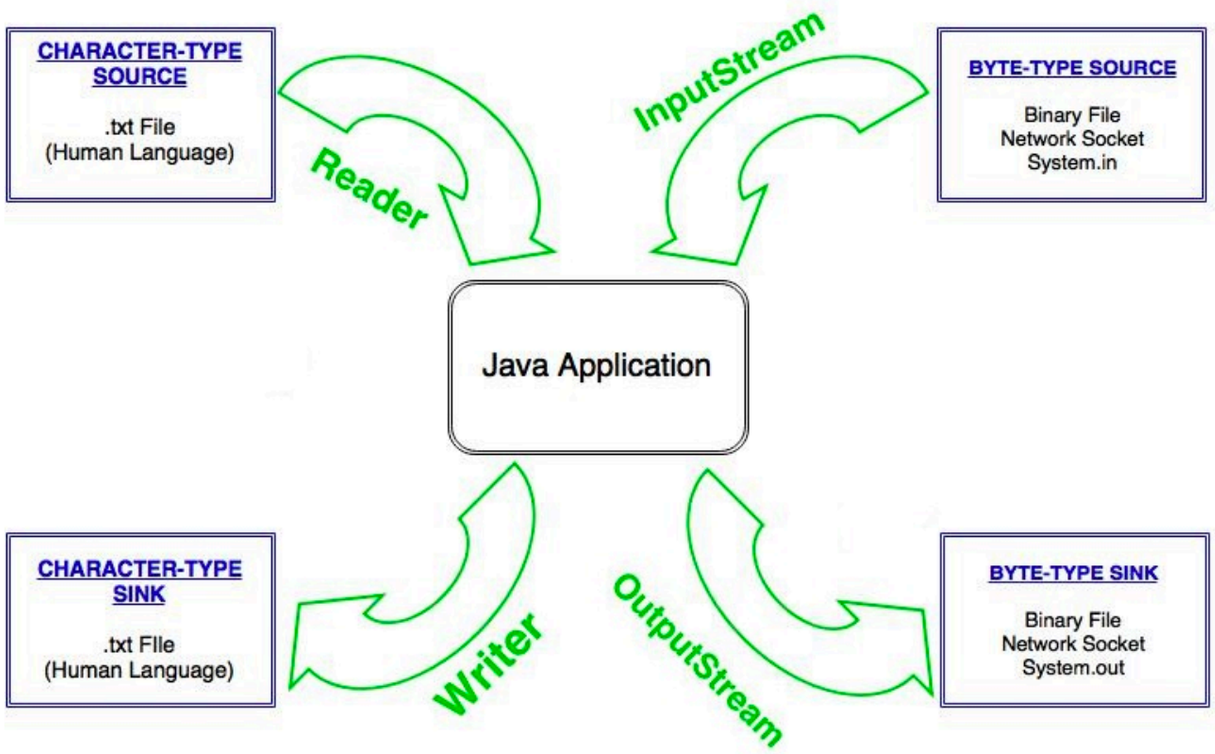

Internal Data Formats:

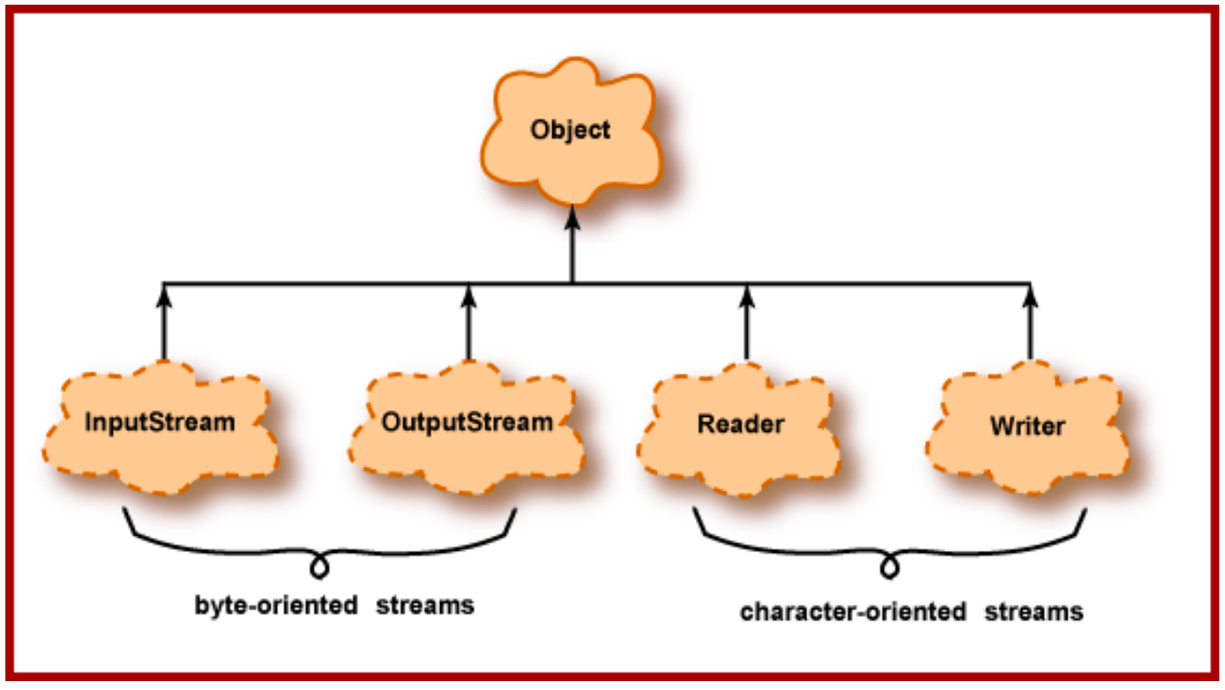
- Text (char): UCS-2
- int, float, double, etc.

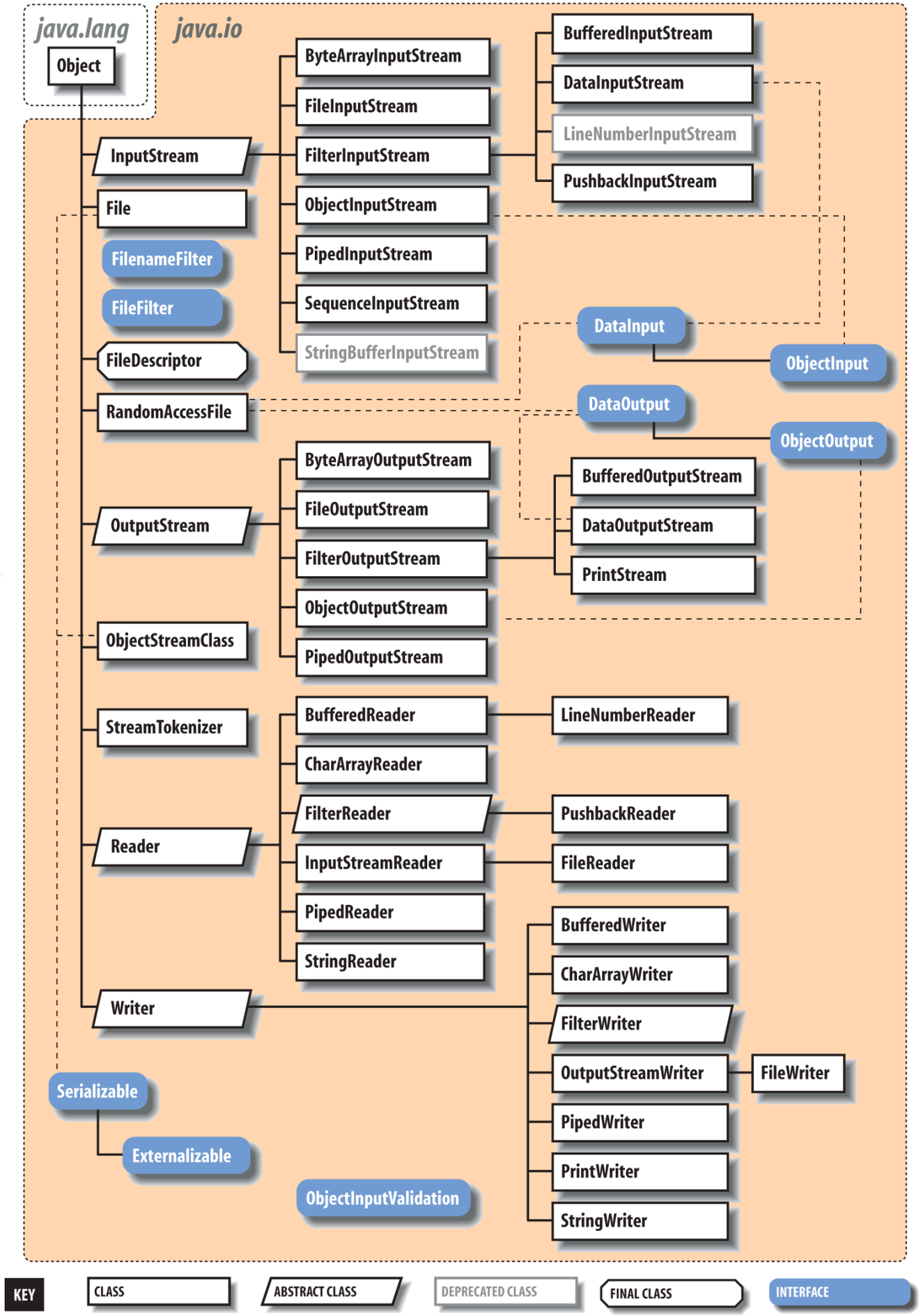
External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

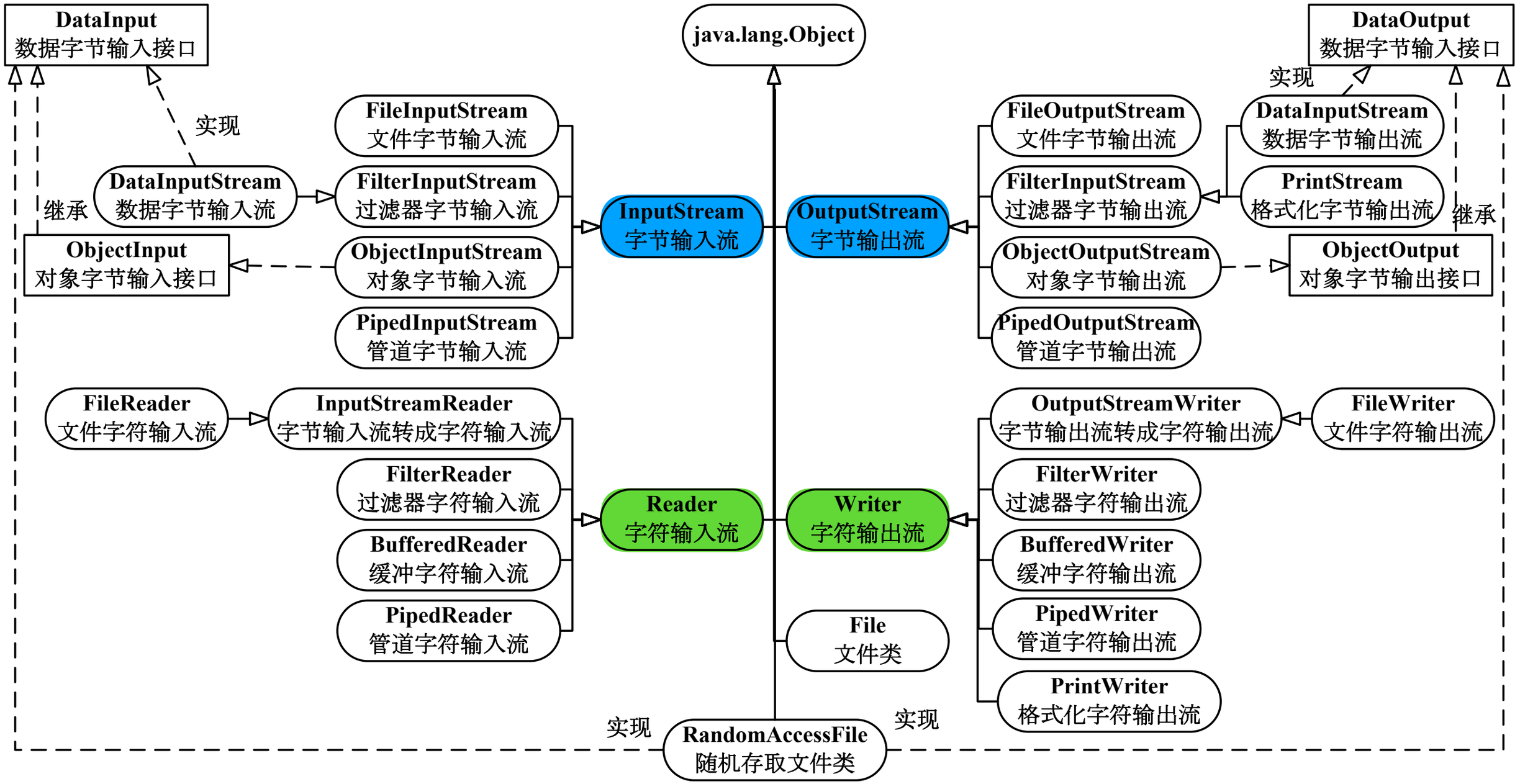
# 流的分类


CHARACTER-TYPE SOURCE
.txt File
(Human Language)
Reader
InputStream
BYTE-TYPE SOURCE
Binary File
Network Socket
System.in
Java Application
CHARACTER-TYPE SINK
.txt FIle
(Human Language)
Writer
OutputStream
BYTE-TYPE SINK
Binary File
Network Socket
System.out

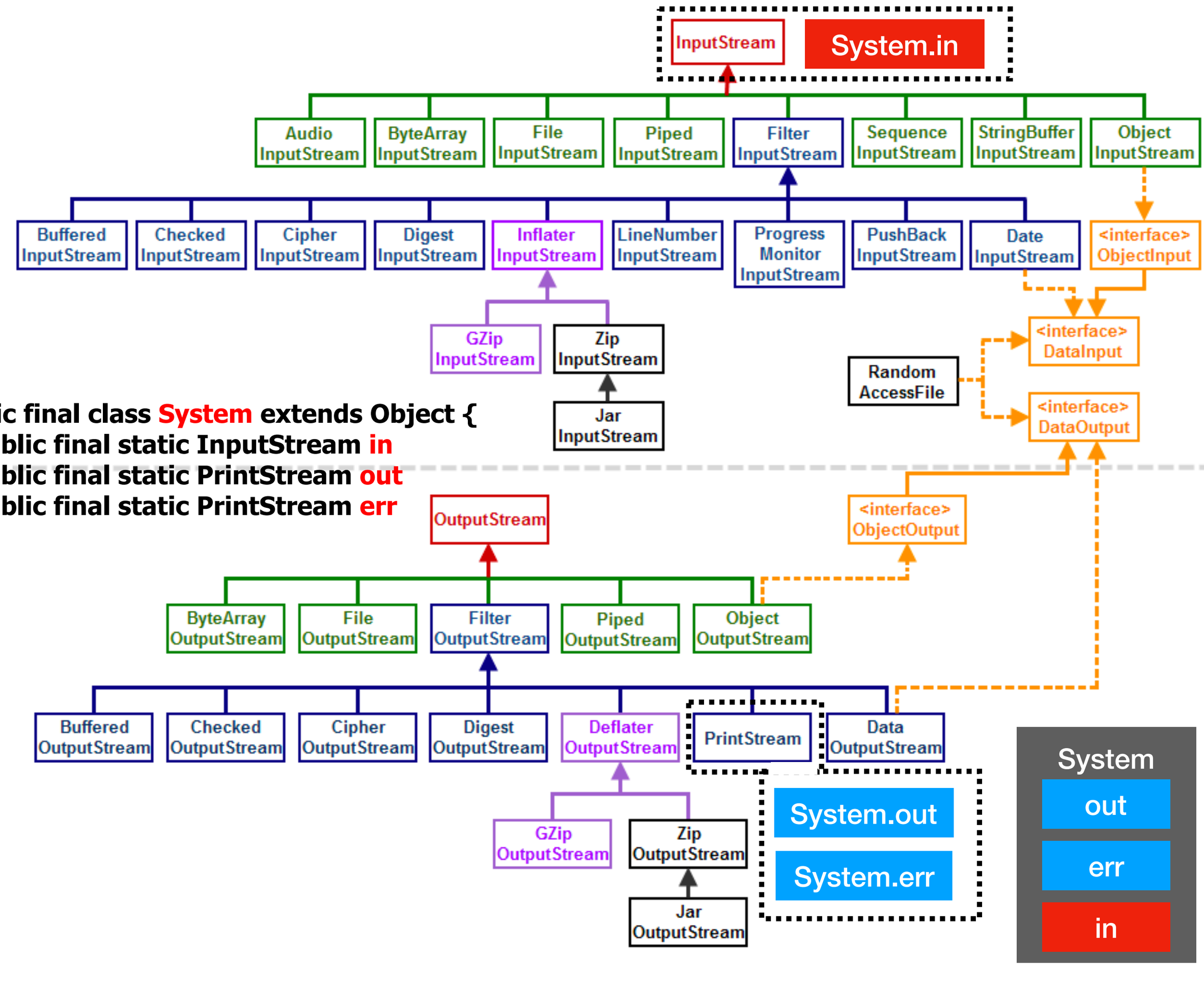

Object
InputStream
OutputStream
Reader
Writer
byte-oriented streams
character-oriented streams


java.lang
java.io
Object
InputStream
ByteArrayInputStream
FileInputStream
FilterInputStream
ObjectInputStream
PipedInputStream
SequenceInputStream
StringBufferInputStream
BufferedInputStream
DataInputStream
LineNumberInputStream
PushbackInputStream
File
FilenameFilter
FileFilter
FileDescriptor
RandomAccessFile
DataInput
ObjectInput
DataOutput
ObjectOutput
OutputStream
ByteArrayOutputStream
FileOutputStream
FilterOutputStream
ObjectOutputStream
PipedOutputStream
BufferedOutputStream
DataOutputStream
PrintStream
ObjectStreamClass
StreamTokenizer
Reader
BufferedReader
CharArrayReader
FilterReader
InputStreamReader
PipedReader
StringReader
LineNumberReader
PushbackReader
FileReader
Writer
BufferedWriter
CharArrayWriter
FilterWriter
OutputStreamWriter
PipedWriter
PrintWriter
StringWriter
FileWriter
Serializable
Externalizable
ObjectInputValidation
KEY
CLASS
ABSTRACT CLASS
DEPRECATED CLASS
FINAL CLASS
INTERFACE

# 输入输出相关类图


DataInput
数据字节输入接口
实现
DataInputStream
数据字节输入流
继承
ObjectInput
对象字节输入接口
FileInputStream
文件字节输入流
FilterInputStream
过滤器字节输入流
ObjectInputStream
对象字节输入流
PipedInputStream
管道字节输入流
java.lang.Object
InputStream
字节输入流
OutputStream
字节输出流
DataOutput
数据字节输入接口
实现
FileOutputStream
文件字节输出流
DataInputStream
数据字节输出流
FilterInputStream
过滤器字节输出流
PrintStream
格式化字节输出流
继承
ObjectOutputStream
对象字节输出流
ObjectOutput
对象字节输出接口
PipedOutputStream
管道字节输出流
FileReader
文件字符输入流
InputStreamReader
字节输入流转成字符输入流
FilterReader
过滤器字符输入流
BufferedReader
缓冲字符输入流
PipedReader
管道字符输入流
Reader
字符输入流
Writer
字符输出流
File
文件类
OutputStreamWriter
字节输出流转成字符输出流
FileWriter
文件字符输出流
FilterWriter
过滤器字符输出流
BufferedWriter
缓冲字符输出流
PipedWriter
管道字符输出流
PrintWriter
格式化字符输出流
实现
RandomAccessFile
随机存取文件类
实现

```
public final class System extends Object {
    public final static InputStream in
    public final static PrintStream out
    public final static PrintStream err
}
```



# 抽象类

**InputStream**

```
public abstract int read() throws IOException
public int read(byte[] bytes, int offset, int length) throws IOException
public int read(byte[] bytes) throws IOException
```

**OutputStream**

```
public void abstract void write(int unsignedByte) throws IOException
public void write(byte[] bytes, int offset, int length) throws IOException
public void write(byte[] bytes) throws IOException
```

**Reader**

```
public abstract int read() throws IOException
public int read(char[] chars, int offset, int length) throws IOException
public int read(char[] chars) throws IOException
```

**Writer**

```
public void abstract void write(int aChar) throws IOException
public void write(char[] chars, int offset, int length) throws IOException
public void write(char[] chars) throws IOException
```

# 文件流

FileInputStream

FileReader

```
public class FileInputStream extends InputStream {
    public FileInputStream(String filename)   throws  FileNotFoundException
    public FileInputStream(File file) throws   FileNotFoundException
}
```

FileOutputStream

FileWriter

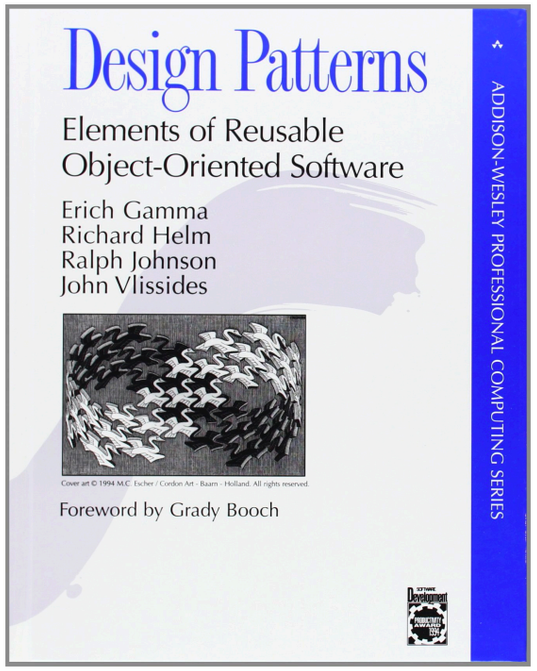
```
public class FileOutputStream extends OutputStream{
    public FileOutputStream(String filename)    throws FileNotFoundException
    public FileOutputStream(File file) throws   FileNotFoundException
    public FileOutputStream(String filename,    boolean append) throws
        FileNotFoundException
}
```

```java
import java.io.*;

public class fileStreamTest {
    public static void main(String args[]) {
        try {
            byte bWrite[] = { 11, 21, 3, 40, 5 };
            OutputStream os = new FileOutputStream("test.txt");
            for (int x = 0; x < bWrite.length; x++) {
                os.write(bWrite[x]); // writes the bytes
            }
            os.close();

            InputStream is = new FileInputStream("test.txt");
            int size = is.available();

            for (int i = 0; i < size; i++) {
                System.out.print((char) is.read() + "  ");
            }
            is.close();
        } catch (IOException e) {
            System.out.print("Exception");
        }
    }
}
```



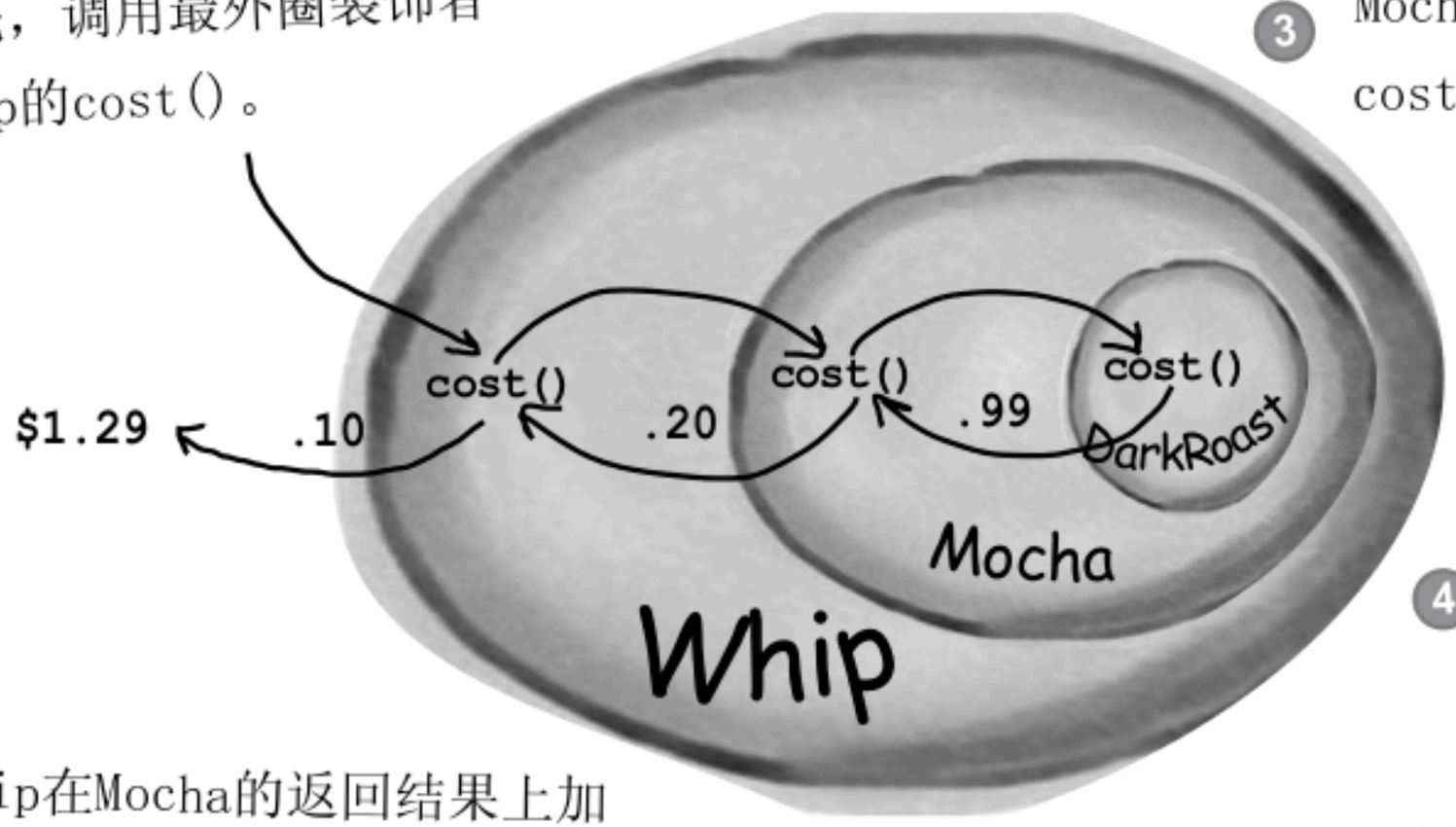

# 装饰者模式

知道这是如何办到的。

2 Whip调用Mocha的cost()。

1 首先，调用最外圈装饰者Whip的cost()。

3 Mocha调用DarkRoast的cost()。




4 DrakRoast返回它的价钱$0.99。

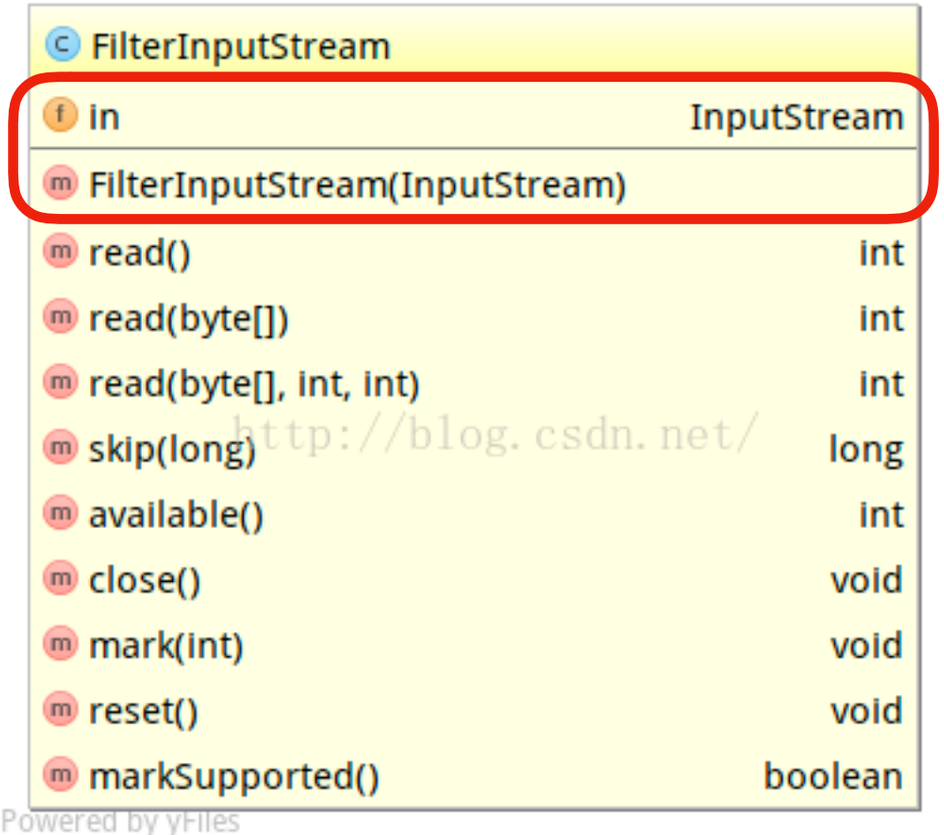
6 Whip在Mocha的返回结果上加上自己的价钱$0.10，然后返回最后结果$1.29。

5 Mocha在DarkRoast的结果上，加上自己的价钱$0.20，返回新的价钱$1.19。

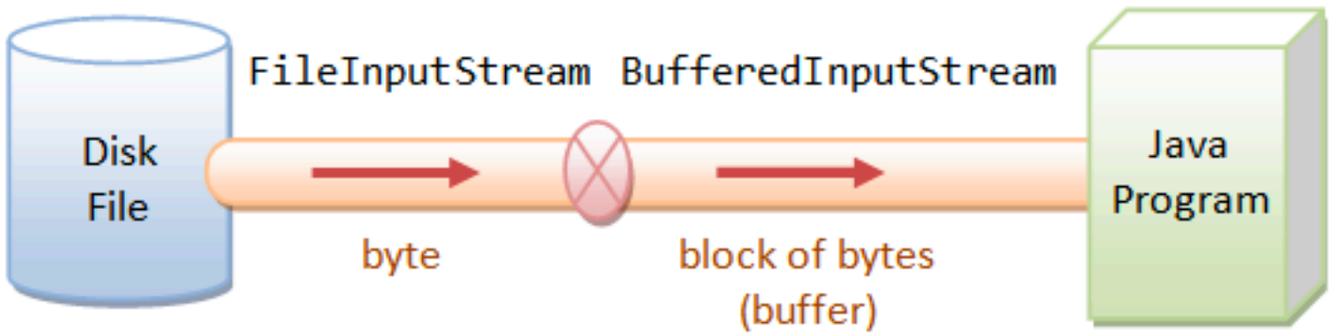
# 链式输入输出流

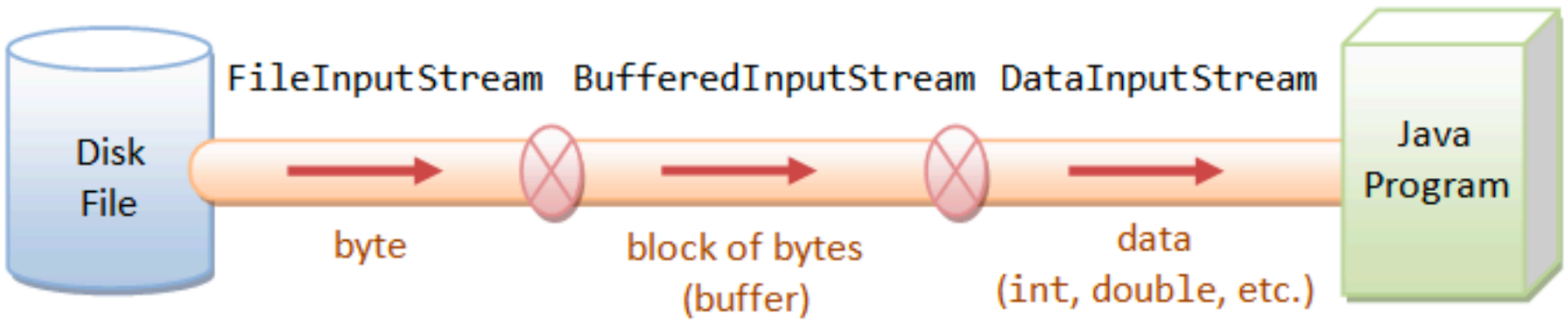
```
protected FilterInputStream(InputStream in) {
    this.in = in;
}
```

```
public int read() throws IOException {
    return in.read();
}
```



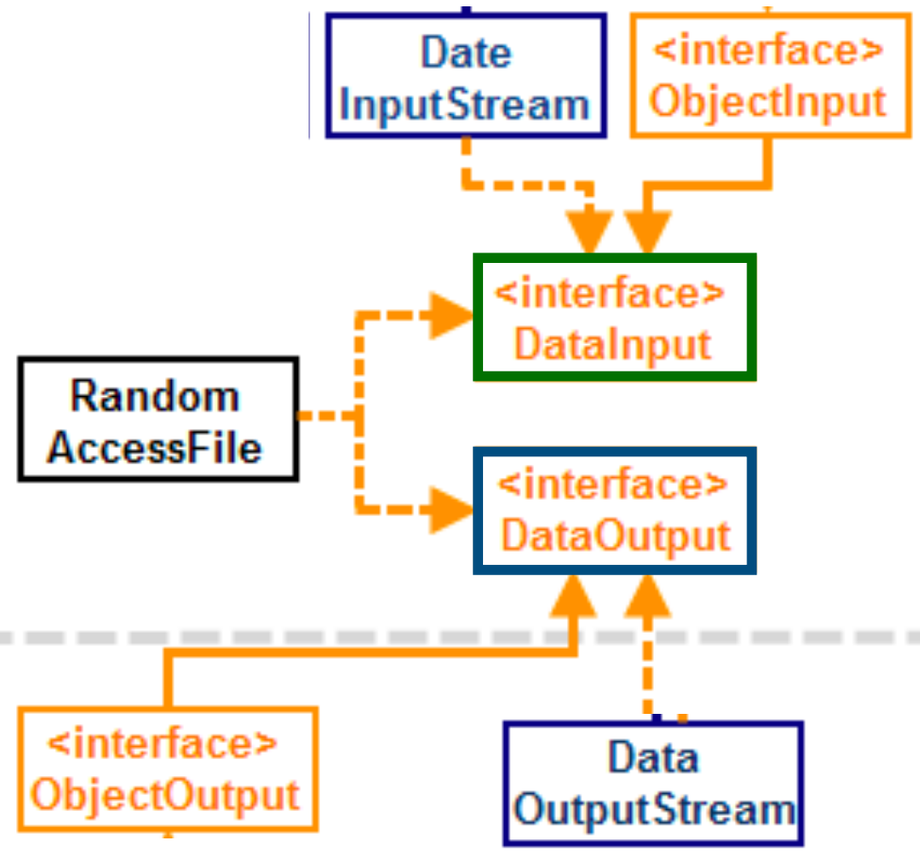




# DataXXputStream

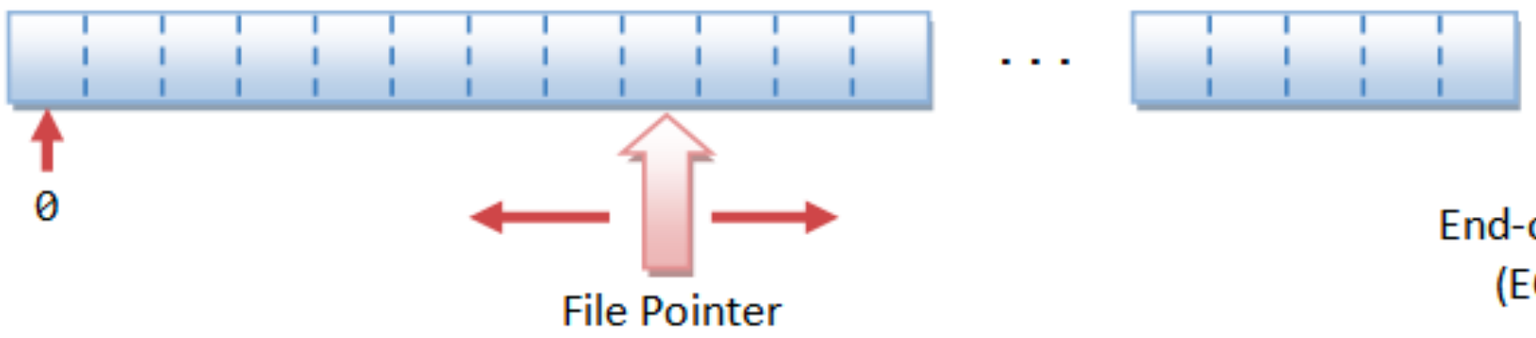
- public DataInputStream(InputStream in);
- readShort( )、 readByte( )、readInt( )、
- readLong( )、readFloat( )、readDouble( )、
- readChar( )、readBoolean( )




- public DataOutputStream(OutputStream out);
- writeShort( )、 writeByte( )、writeInt( )、
- writeLong( )、writeFloat( )、writeDouble()、
- writeChar( )、writeBoolean( )、writeChars( )

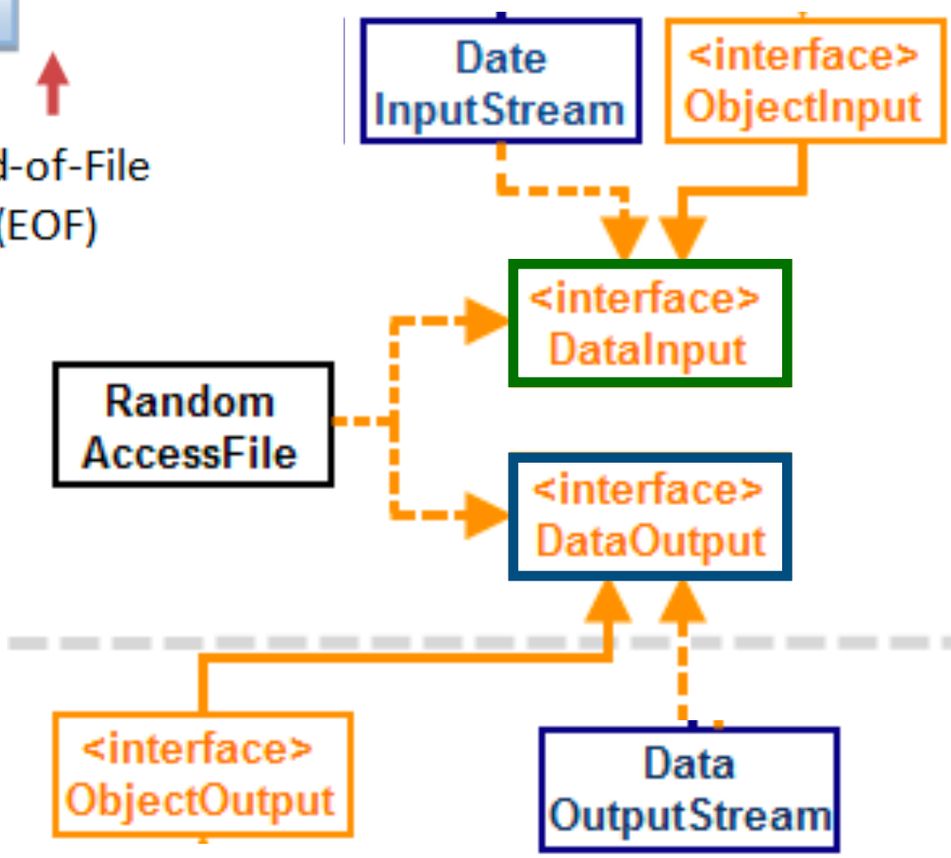
# RandomAccessFile

```
public void seek(long pos) throws IOException;
// Positions the file pointer for subsequent read/write operation.
public int skipBytes(int numBytes) throws IOException;
// Moves the file pointer forward by the specified number of bytes.
public long getFilePointer() throws IOException;
// Gets the position of the current file pointer, in bytes, from the beginning of
the file.
public long length() throws IOException;
// Returns the length of this file.
```

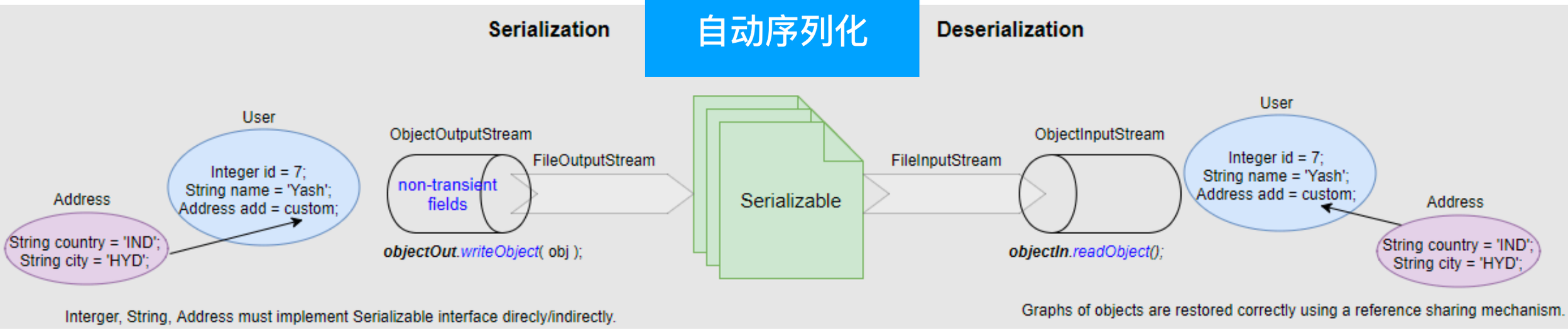



- public DataInputStream(InputStream in);
- readShort( )、 readByte( )、readInt( )、readLong( )、readFloat( )、readDouble( )、readChar( )、readBoolean( )

- public DataOutputStream(OutputStream out);
- writeShort( )、 writeByte( )、writeInt( )、writeLong( )、writeFloat( )、writeDouble()、writeChar( )、writeBoolean( )、writeChars( )



# ObjectXXputStream




```
ObjectOutputStream out =
   new ObjectOutputStream(
      new BufferedOutputStream(
         new FileOutputStream("object.ser")));
out.writeObject("The current Date and Time is "); // write a String object
out.writeObject(new Date());                      // write a Date object
out.flush();
out.close();
```

不序列化的属性

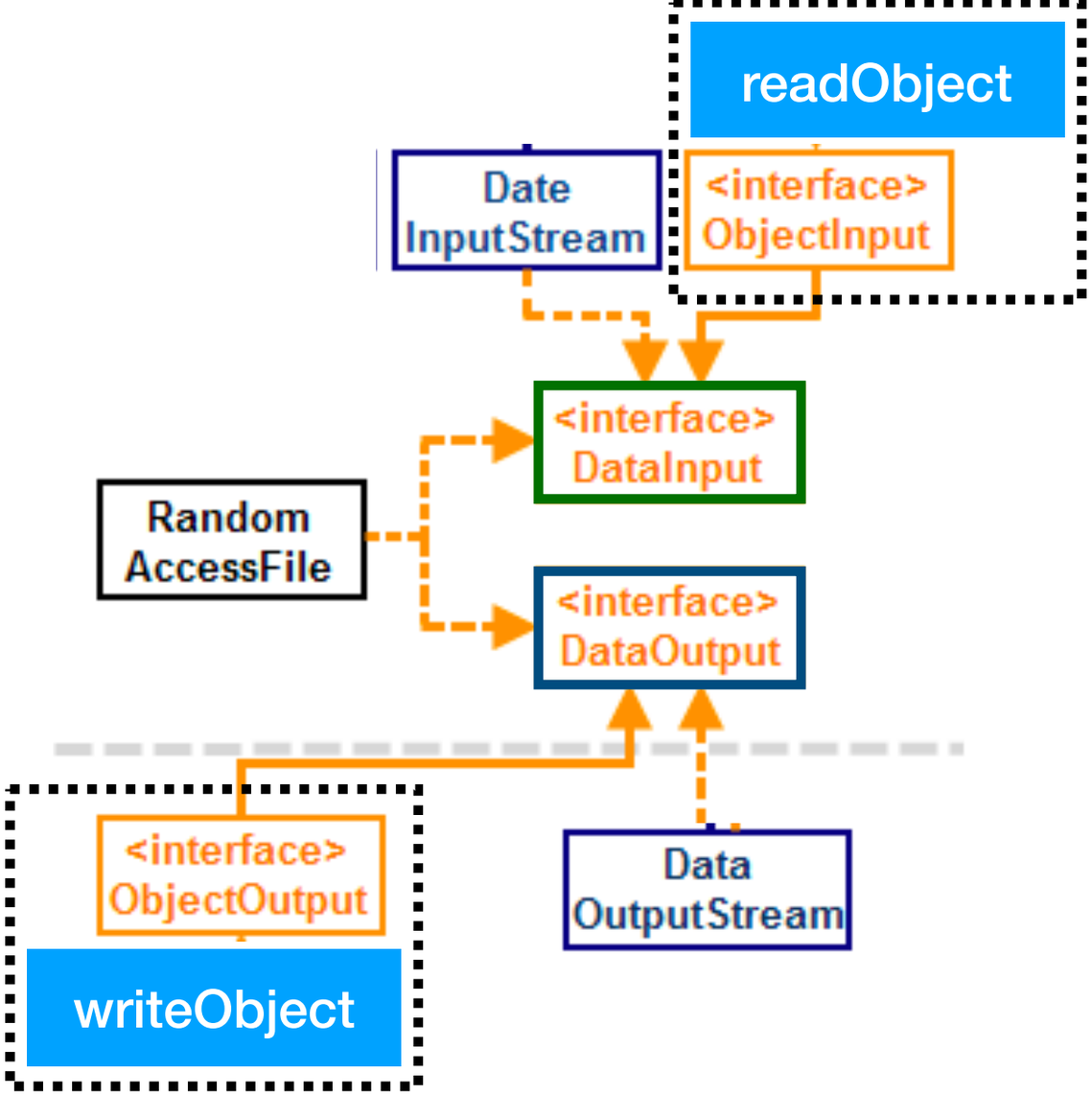
static

transient

```
public interface Serializable {};
```

```
public final Object readObject() throws IOException, ClassNotFoundException;
public final void writeObject(Object obj) throws IOException;
```

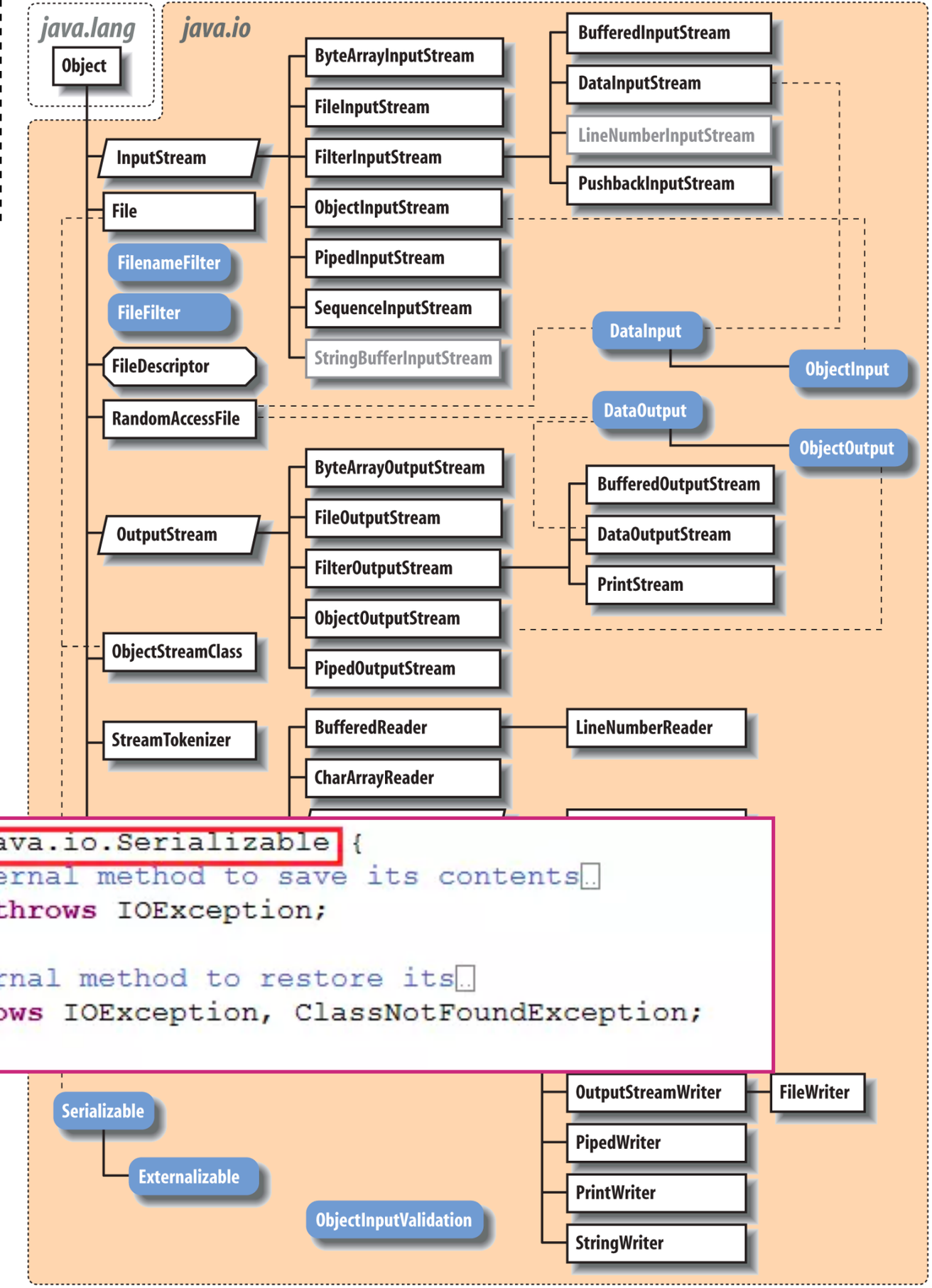
```
ObjectInputStream in =
   new ObjectInputStream(
      new BufferedInputStream(
         new FileInputStream("object.ser")));
String str = (String)in.readObject();
Date d = (Date)in.readObject(new Date());  // downcast
in.close();
```




```
66 public interface Externalizable extends java.io.Serializable {
68⊕     * The object implements the writeExternal method to save its contents
82     void writeExternal(ObjectOutput out) throws IOException;
83
85⊕     * The object implements the readExternal method to restore its
96     void readExternal(ObjectInput in) throws IOException, ClassNotFoundException;
97 }
```

```
public interface Serializable {};
```

手动序列化



# 文件 File

```
public interface Serializable {};
```

**public class File extends Object implements Serializable, Comparable<File>{**
**public File(String pathname)**
**public File(String parent, String child)**
**public File(File parent, String child)**
**}**

```
interface Comparable<T> {
abstract int  compareTo(T o);
}
```

```
public boolean exists()         // Tests if this file/directory exists.
public long length()            // Returns the length of this file.
public boolean isDirectory()    // Tests if this instance is a directory.
public boolean isFile()         // Tests if this instance is a file.
public boolean canRead()        // Tests if this file is readable.
public boolean canWrite()       // Tests if this file is writable.
public boolean delete()         // Deletes this file/directory.
public void deleteOnExit()      // Deletes this file/directory when the program terminates.
public boolean renameTo(File dest) // Renames this file.
public boolean mkdir()          // Makes (Creates) this directory.
```

```
public String[] list()      // List the contents of this directory in a String-array
public File[] listFiles()   // List the contents of this directory in a File-array
```

```
public String[] list()      // List the contents of this directory in a String-array
public File[] listFiles()   // List the contents of this directory in a File-array
```

```
public String[] list(FilenameFilter filter)
public File[] listFiles(FilenameFilter filter)
public File[] listFiles(FileFilter filter)
```

```
public interface FileFilter {
        public boolean accept(File pathname
    }
```

```
public interface FilenameFilter{
        public boolean accept(File dir,String filename)
    }
```

```
public class ListDirectoryWithFilter {
   public static void main(String[] args) {
      File dir = new File(".");   // current working directory
      if (dir.isDirectory()) {
         // List only files that meet the filtering criteria
         //  programmed in accept() method of FilenameFilter.
         String[] files = dir.list(new FilenameFilter() {
            public boolean accept(File dir, String file) {
               return file.endsWith(".java");
            }
         });  // an anonymous inner class as FilenameFilter
         for (String file : files) {
            System.out.println(file);
         }
      }
   }
```