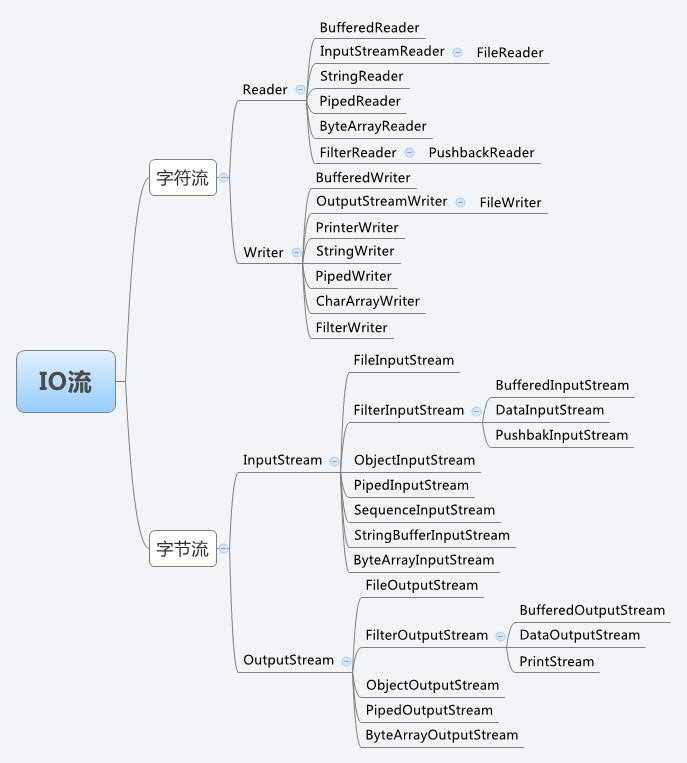
总体分为字节处理InputStream OutputStream

字符处理Reader Writer



1.字节

InputStream

-FileInputStream

-ByteArrayInputStream

-StringBufferInputStream

-FilterInputStream-DataInputStream

-BufferedInputStream

**抽象类InputStream**

abstract class InputStream{

//公共抽象方法read()：读取一个字节，灵魂函数

public abstract int read() throws IOException;

//使用了read()的具体方法

public int read(byte[] b){……};

public int read(byte[] b,int off,int len){……};

}

**子类FileInputStream、ByteArrayInputStream根据数据的来源划分，并实现具体的read()**

class FileInputStream extends InputStream{

public FileInputStream(String fileName){

……

}

}

class ByteArrayInputStream extends InputStream{

public ByteArrayInputStream(byte[] b){……}

}

**子类FilterInputStream包持有一个InputStream的具体对象，实现对该具体对象的功能装饰**

class FilterInputStream extends InputStream{

protected volatile InputStream in;

protected FilterInputStream(InputStream in){

this.in = in;

}

public int read(){

in.read();

}

}

过滤器(装饰器)FilterInputStream并没有修改所持有InputStream（实际传入的应该是具体对象）的read()方法，与原对象一模一样。例如，可以装饰FileInputStream对象，实现字节的读取。

**DataInputStream继承自FilterInputStream，可以提供从所持有InputStream对象中，java基本数据类型的读取功能。**

public class DataInputStream extends FilterInputStream{

DataInputStream(InputStream in){

super(in);

}

public int readInt(){….}

public char readChar(){….}

public byte readByte(){…}

}

**BufferedInputStream可以提供缓冲功能，将所持有InputStream对象的数据读取到内部缓冲当中的功能，对外提供的read()函数从内部缓冲中读取数据。**

public class BufferedInputStream extends FilterInputStream{

BufferedInputStream (InputStream in){

super(in);

}

内部实现一个byte数组作为所持有的InputStream对象读取数据的缓冲。

//提供自己实现的read()而不是用所持有InputStream具体对象的read函数。该read函数从byte数组中提取一个字节然后返回。当byte数组空，自动调用具体对象的read函数对byte数组进行填充。

public int read(){

}

}

总结1：DataInputStream分明可以直接继承InputStream然后实现readInt() readByte()等功能。增加了FilterInputStream是因为：还有BufferedInputStream等，也是持有一个InputStream具体对象然后对其进行功能扩展，对read()抽象方法进行新的实现。即有多个装饰者的出现。

总结2：使用装饰者模式的核心是装饰类持有一个原对象的引用。并在构造函数中传入具体的对象。然后实现原对象的方法（也可以不变），或者扩展新方法(readInt readByte)。

总结3：使用装饰者模式而不是继承可以防止类的膨胀。

继承：FileInputStream-

DataFileInputStream

BufferedFileInputStream

ByteArrayInputStream-

Data ByteArrayInputStream

Buffered ByteArrayInputStream

2.字符

Reader – InputStreamReader-FileReader

BufferedReader

StringReader

ByteArrayReader

**抽象类Reader**

public abstract class Reader{

//包含一个字段

protected Object lock;

//公共抽象方法read—核心函数

public abstract int read(char[] b,int off,int len);

//read()返回一个字符(的unicodem码)

public char read(){

使用read(buf,0,1);

}

**}**

**子类InputStreamReader**

public class InputStreamReader extends Reader{

StreamDecoder sd;

//构造函数1

public InputStreamReader(InputStream in){

super(in);

sd=StreamDecoder.forInputStreamReader(in,this,NULL);

}

//构造函数2，指定编码

public InputStreamReader(InputStream in,String charsetName){

super(in);

sd=StreamDecoder.forInputStreamReader(in,this,charsetName);

}

//使用StreamDecoder.read实现read(char[] buf,int off,int len)

public int read(char[] buf,int off,int len){

sd.read(buf,offset,len);

}

}

StreamDecoder负责解码：字节-🡪字符

StreamEncoder负责编码: 字符🡪字节

**子类BufferedReader持有一个Reader对象，是对具体对象的装饰。并提供缓存（类似于BufferedInputStream）例如可以装饰InputStreamReader对象**

public class BufferedReader extends Reader{

private Reader in;

public BufferedReader(Reader in){

this.in=in;

}

}