[Java中IO流，输入输出流概述与总结](http://www.cnblogs.com/biehongli/p/6074713.html)

**1：Java语言定义了许多类专门负责各种方式的输入或者输出，这些类都被放在java.io包中。其中，**

**所有输入流类都是抽象类InputStream(字节输入流)，或者抽象类Reader(字符输入流)的子类；**

**而所有输出流都是抽象类OutputStream(字节输出流)或者Writer(字符输出流)的子类。**

**【首先需要明白的是：流是干什么的？？？（为了永久性的保存数据）**

**根据数据流向的不同分为输入流和输出流；**

**根据处理数据类型的不同分为字符流和字节流；**

**】**

**【然后需要明白的是输入模式和输出模式是谁流向谁：**

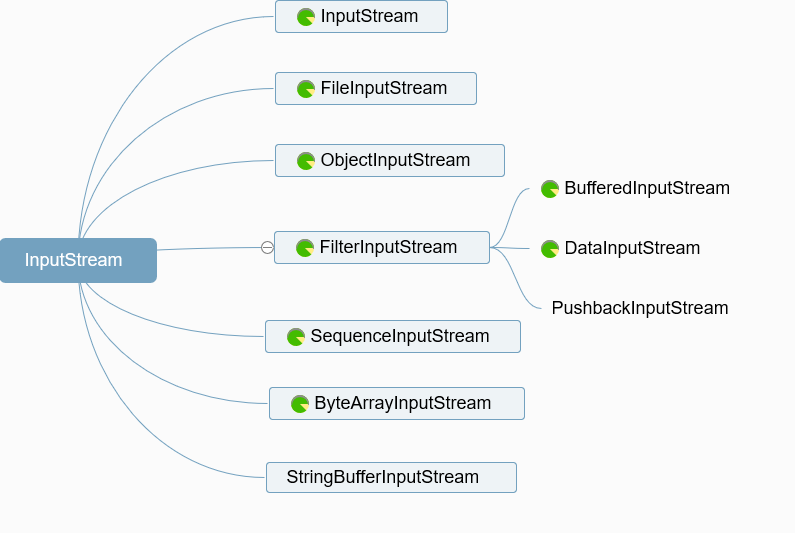
**InputStream(字节输入流)和Reader(字符输入流)通俗的理解都是读（read）的。**

**OutputStream(字节输出流)和Writer(字符输出流)通俗的理解都是写(writer)的。**

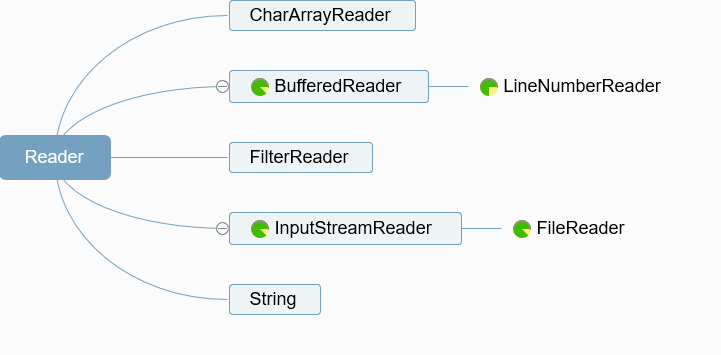
**】**

**最后下面搞清楚各种流的类型的该怎么用，谁包含谁，理清思路。**

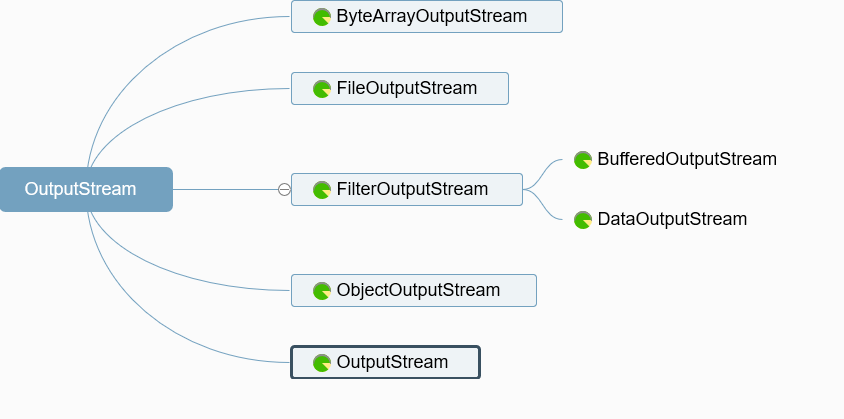
**2：InputStream类是字节输入流的抽象类，是所有字节输入流的父类，InputStream类具有层次结构如下图所示；**



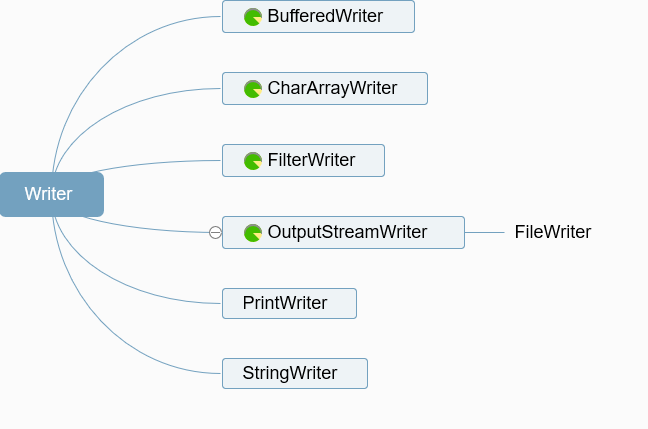
**3：java中的字符是Unicode编码的，是双字节的。InputStream是用来处理字节的，在处理字符文本时很不方便。Java为字符文本的输入提供了专门的一套类Reader。Reader类是字符输入流的抽象类，所有字符输入流的实现都是它的子类。**



**4:输出流OutputStream类是字节输入流的抽象类，此抽象类表示输出字节流的所有类的超类。**



**5:Writer类是字符输出流的抽象类，所有字符输出类的实现都是它的子类。**

****

**6：File类是IO包中唯一代表磁盘文件本身的对象。通过File来创建，删除，重命名文件。File类对象的主要作用就是用来获取文本本身的一些信息。如文本的所在的目录，文件的长度，读写权限等等。（有的需要记忆，比如isFile(),isDirectory(),exits();有的了解即可。使用的时候查看API）**

**详细如下：**

**File类(File类的概述和构造方法)**

**A:File类的概述**

**File更应该叫做一个路径**

**文件路径或者文件夹路径**

**路径分为绝对路径和相对路径**

**绝对路径是一个固定的路径,从盘符开始**

**相对路径相对于某个位置,在eclipse下是指当前项目下,在dos下**

**查看API指的是当前路径**

**文件和目录路径名的抽象表示形式**

**B:构造方法**

**File(String pathname)：根据一个路径得到File对象**

**File(String parent, String child):根据一个目录和一个子文件/目录得到File对象**

**File(File parent, String child):根据一个父File对象和一个子文件/目录得到File对象**

**File类(File类的创建功能)**

**A:创建功能**

**public boolean createNewFile():创建文件 如果存在这样的文件，就不创建了**

**public boolean mkdir():创建文件夹 如果存在这样的文件夹，就不创建了**

**public boolean mkdirs():创建文件夹,如果父文件夹不存在，会帮你创建出来**

**（使用createNewFile()文件创建的时候不加.txt或者其他后缀也是文件，不是文件夹；使用mkdir()创建文件夹的时候，如果起的名字是比如aaa.txt也是文件夹不是文件；）**

**注意事项：**

**如果你创建文件或者文件夹忘了写盘符路径，那么，默认在项目路径下。**

**File类(File类的重命名和删除功能)**

**A:重命名和删除功能**

**public boolean renameTo(File dest):把文件重命名为指定的文件路径**

**public boolean delete():删除文件或者文件夹**

**B:重命名注意事项**

**如果路径名相同，就是改名。**

**如果路径名不同，就是改名并剪切。**

**C:删除注意事项：**

**Java中的删除不走回收站。**

**要删除一个文件夹，请注意该文件夹内不能包含文件或者文件夹**

**File类(File类的判断功能)**

**A:判断功能**

**public boolean isDirectory():判断是否是目录**

**public boolean isFile():判断是否是文件**

**public boolean exists():判断是否存在**

**public boolean canRead():判断是否可读**

**public boolean canWrite():判断是否可写**

**public boolean isHidden():判断是否隐藏**

**File类(File类的获取功能)**

**A:获取功能**

**public String getAbsolutePath()：获取绝对路径**

**public String getPath():获取路径**

**public String getName():获取名称**

**public long length():获取长度。字节数**

**public long lastModified():获取最后一次的修改时间，毫秒值**

**public String[] list():获取指定目录下的所有文件或者文件夹的名称数组**

**public File[] listFiles():获取指定目录下的所有文件或者文件夹的File数组**

**File类(文件名称过滤器的概述及使用)**

**A:文件名称过滤器的概述**

**public String[] list(FilenameFilter filter)**

**public File[] listFiles(FileFilter filter)**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.File;

4

5 public class Test {

6

7 public static void main(String[] args) throws Exception{

8 // TODO Auto-generated method stub

9 File file=new File("aa.txt");//文件默认就创建在你创建的项目下面，刷新即可看到

10 System.out.println(file.exists());//判断文件是否存在

11 file.createNewFile();//创建文件，不是文件夹

12 System.out.println(file.exists());//再次判断是否存在

13 System.out.println(file.getName());//获取文件的名字

14 System.out.println(file.getAbsolutePath());//获取文件的绝对路径

15 System.out.println(file.getPath());//获取文件的相对路径

16 System.out.println(file.getParent());//获取文件的父路径

17 System.out.println(file.canRead());//文件是否可读

18 System.out.println(file.canWrite());//文件是否可写

19 System.out.println(file.length());//文件的长度

20 System.out.println(file.lastModified());//文件最后一次修改的时间

21 System.out.println(file.isDirectory());//判断文件是否是一个目录

22 System.out.println(file.isHidden());//文件是否隐藏

23 System.out.println(file.isFile());//判断文件是否存在

24 }

25

26 }

[复制代码](javascript:void(0);)

**public String[] list():获取指定目录下的所有文件或者文件夹的名称数组**

**public File[] listFiles():获取指定目录下的所有文件或者文件夹的File数组**

**list()获取某个目录下所有的文件或者文件夹：**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.File;

4

5 public class FileTest {

6

7 public static void main(String[] args){

8 File file=new File("D:/");//指定文件目录

9 String[] str=file.list();//获取指定目录下的所有文件或者文件夹的名称数组

10 for(String s : str){//加强for循环遍历输出

11 System.out.println(s);

12 }

13

14 }

15 }

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.File;

4

5 public class FileTest {

6

7 public static void main(String[] args){

8 File file=new File("D:/");//指定文件路径

9 File[] f=file.listFiles();//获取指定目录下的所有文件或者文件夹的File数组

10 for(File fi : f){//加强for循环遍历输出

11 System.out.println(fi);

12 }

13

14 }

15 }

[复制代码](javascript:void(0);)

**案例演示：**

**获取某种格式的文件比如获取某种后缀的图片，并输出文件名：**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.File;

4

5 public class FileTest {

6

7 public static void main(String[] args){

8 File file=new File("C:\\Users\\biehongli\\Pictures\\xuniji");

9 String[] str=file.list();

10

11 for(String s : str){

12 if(s.endsWith(".jpg") || s.endsWith(".png")){//如果后缀是这种格式的就输出

13 System.out.println(s);

14 }

15 }

16

17

18 }

19 }

[复制代码](javascript:void(0);)

**下面演示获取文件夹下面子目录里面的文件获取（并没有完全获取子目录的子目录等等，仅仅获取了子一级目录）：**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.File;

4

5 public class FileTest {

6

7 public static void main(String[] args){

8 File file=new File("C:\\Users\\biehongli\\Pictures\\Camera Roll");

9

10 File[] f=file.listFiles();

11

12 for(File fi : f){

13 if(fi.isDirectory()){//判断如果是一个目录

14 String[] s=fi.list();

15 for(String str : s){

16 if(str.endsWith(".jpg")){

17 System.out.println(str);

18 }

19 }

20 }

21 }

22 }

23 }

[复制代码](javascript:void(0);)

**A:文件名称过滤器的概述**

**public String[] list(FilenameFilter filter)**

**public File[] listFiles(FileFilter filter)**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.File;

4 import java.io.FilenameFilter;

5

6 public class FileTest {

7

8 public static void main(String[] args){

9 File file=new File("C:\\Users\\biehongli\\Pictures\\Camera Roll");

10

11 String[] str=file.list(new FilenameFilter() {//过滤器，匿名内部类

12

13 @Override

14 public boolean accept(File dir, String name) {

15 // TODO Auto-generated method stub

16 //System.out.println(dir);//获取文件的路径

17 //System.out.println(name);//获取文件的名字

18 File f=new File(dir,name);

19 return f.isFile() && f.getName().endsWith(".jpg");

20 }

21 });

22 for(String s : str){

23 System.out.println(s);

24 }

25

26 }

27 }

[复制代码](javascript:void(0);)

**7：下面以一些字节输入输出流具体的案例操作（操作的时候认清自己使用的是字节流还是字符流）：**

**注意：read()方法读取的是一个字节,为什么返回是int,而不是byte**

**字节输入流可以操作任意类型的文件,比如图片音频等,这些文件底层都是以二进制形式的存储的,如果每次读取都返回byte,有可能在读到中间的时候遇到111111111；那么这11111111是byte类型的-1,我们的程序是遇到-1就会停止不读了,后面的数据就读不到了,所以在读取的时候用int类型接收,如果11111111会在其前面补上；24个0凑足4个字节,那么byte类型的-1就变成int类型的255了这样可以保证整个数据读完,而结束标记的-1就是int类型**

**FileInputStream的单个字节读取：**

**FileOutputStream的单个字节写入：**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.FileInputStream;

4 import java.io.FileOutputStream;

5

6 public class FileTest {

7

8 public static void main(String[] args) throws Exception{

9 FileInputStream fis=new FileInputStream("aaa.txt");

10 FileOutputStream fos=new FileOutputStream("bbb.txt",true);

11 //FileOutputStream()后面加true指文件后面可追加

12

13 int a=fis.read();//read()一次读取一个字节

14 System.out.println(a);//读取的一个字节输出

15

16 fos.write(101);//write()一次写一个字节

17 fis.close();//一定记得关闭流，养成好习惯

18 fos.close();

19 }

20 }

[复制代码](javascript:void(0);)

**FileInputStream和FileOutputStream进行拷贝文本或者图片或者歌曲：**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.FileInputStream;

4 import java.io.FileOutputStream;

5

6 public class FileTest {

7

8 public static void main(String[] args) throws Exception{

9 FileInputStream fis=new FileInputStream("aaa.txt");

10 FileOutputStream fos=new FileOutputStream("bbb.txt");

11 //如果没有bbb.txt,会创建出一个

12

13 int b;

14 while((b=fis.read())!=-1){

15 fos.write(b);

16 }

17 fis.close();

18 fos.close();

19 }

20 }

[复制代码](javascript:void(0);)

**FileInputStream和FileOutputStream定义小数组进行读写操作：**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.FileInputStream;

4 import java.io.FileOutputStream;

5

6 public class FileTest {

7

8 public static void main(String[] args) throws Exception{

9 FileInputStream fis = new FileInputStream("aaa.txt");

10 FileOutputStream fos = new FileOutputStream("bbb.txt");

11 int len;

12 byte[] arr = new byte[1024 \* 8];//自定义字节数组

13

14 while((len = fis.read(arr)) != -1) {

15 //fos.write(arr);

16 fos.write(arr, 0, len);//写出字节数组写出有效个字节个数

17 }

18 //IO流(定义小数组)

19 //write(byte[] b)

20 //write(byte[] b, int off, int len)写出有效的字节个数

21

22 fis.close();

23 fos.close();

24 }

25 }

[复制代码](javascript:void(0);)

**IO流(BufferedInputStream和BufferOutputStream拷贝)**

**\* A:缓冲思想**

**\* 字节流一次读写一个数组的速度明显比一次读写一个字节的速度快很多，**

**\* 这是加入了数组这样的缓冲区效果，java本身在设计的时候，**

**\* 也考虑到了这样的设计思想，所以提供了字节缓冲区流**

**\* B.BufferedInputStream**

**\* BufferedInputStream内置了一个缓冲区(数组)**

**\* 从BufferedInputStream中读取一个字节时**

**\* BufferedInputStream会一次性从文件中读取8192个, 存在缓冲区中, 返回给程序一个**

**\* 程序再次读取时, 就不用找文件了, 直接从缓冲区中获取**

**\* 直到缓冲区中所有的都被使用过, 才重新从文件中读取8192个**

**\* C.BufferedOutputStream**

**\* BufferedOutputStream也内置了一个缓冲区(数组)**

**\* 程序向流中写出字节时, 不会直接写到文件, 先写到缓冲区中**

**\* 直到缓冲区写满, BufferedOutputStream才会把缓冲区中的数据一次性写到文件里**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.BufferedInputStream;

4 import java.io.BufferedOutputStream;

5 import java.io.FileInputStream;

6 import java.io.FileOutputStream;

7

8 public class FileTest {

9

10 public static void main(String[] args) throws Exception{

11 FileInputStream fis = new FileInputStream("aaa.txt");

12 FileOutputStream fos = new FileOutputStream("bbb.txt");

13

14 BufferedInputStream bis=new BufferedInputStream(fis);

15 //使用装饰模式，把fis装饰进去bis中。使用缓冲读取速度变快

16 BufferedOutputStream bos=new BufferedOutputStream(fos);

17

18 int b;

19 while((b=bis.read())!=-1){

20 bos.write(b);

21 }

22 bis.close();

23 bos.close();

24 }

25 }

[复制代码](javascript:void(0);)

**面试题**：**小数组的读写和带Buffered的读取哪个更快?**

**\* 定义小数组如果是8192个字节大小和Buffered比较的话**

**\* 定义小数组会略胜一筹,因为读和写操作的是同一个数组**

**\* 而Buffered操作的是两个数组**

**IO流(flush和close方法的区别)**

**flush()方法： 用来刷新缓冲区的,刷新后可以再次写出（字节缓冲流内置缓冲区，如果没有读取出来，可以使用flush()刷新来）**

**close()方法：用来关闭流释放资源的的,如果是带缓冲区的流对象的close()方法,不但会关闭流,还会再关闭流之前刷新缓冲区,关闭后不能再写出**

**8：字符流FileReader和FileWriter**

**字符流是什么**

**\* 字符流是可以直接读写字符的IO流**

**\* 字符流读取字符, 就要先读取到字节数据, 然后转为字符. 如果要写出字符, 需要把字符转为字节再写出.**

**IO流(什么情况下使用字符流)**

**\* 字符流也可以拷贝文本文件, 但不推荐使用. 因为读取时会把字节转为字符, 写出时还要把字符转回字节.**

**\* 程序需要读取一段文本, 或者需要写出一段文本的时候可以使用字符流**

**\* 读取的时候是按照字符的大小读取的,不会出现半个中文**

**\* 写出的时候可以直接将字符串写出,不用转换为字节数组**

**IO流(字符流是否可以拷贝非纯文本的文件)**

**\* 不可以拷贝非纯文本的文件**

**\* 因为在读的时候会将字节转换为字符,在转换过程中,可能找不到对应的字符,就会用?代替,写出的时候会将字符转换成字节写出去**

**\* 如果是?,直接写出,这样写出之后的文件就乱了,看不了了**

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.FileReader;

4 import java.io.FileWriter;

5

6 public class FileTest {

7

8 public static void main(String[] args) throws Exception{

9 //FileReader类的read()方法可以按照字符大小读取

10 FileReader fr=new FileReader("aaa.txt");

11 int b;

12 while((b=fr.read())!=-1){

13 System.out.println((char)b);//int类型转为字符型

14 }

15 fr.close();

16

17 //FileWriter类的write()方法可以自动把字符转为字节写出

18 FileWriter fw = new FileWriter("aaa.txt",true);

19 fw.write("aaa");

20 fw.close();

21

22 //字符流的拷贝

23 FileReader fr2 = new FileReader("aaa.txt");

24 FileWriter fw2 = new FileWriter("bbb.txt");

25

26 int ch;

27 while((ch = fr2.read()) != -1) {

28 fw2.write(ch);

29 }

30

31 fr2.close();

32 fw2.close();

33 }

34 }

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

1 package com.ningmeng;

2

3 import java.io.BufferedReader;

4 import java.io.BufferedWriter;

5 import java.io.FileReader;

6 import java.io.FileWriter;

7

8 public class FileTest {

9

10 public static void main(String[] args) throws Exception{

11 BufferedReader br=new BufferedReader(new FileReader("aaa.txt"));

12 BufferedWriter bw=new BufferedWriter(new FileWriter("bbb.txt"));

13 //**BufferedReader和BufferedWriter的使用：**

14 int b;

15 while((b=br.read())!=-1){

16 bw.write((char)b);

17 }

18 br.close();

19 bw.close();

20 }

21 }