# 文件流

## FileInputStream文件字节输入流

### 新增构造方法：

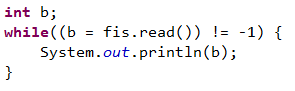
FileInputStream(String name);

FileInputStream(File file); **byte**[] b = **new** **byte**[5];

### 新增常用方法

protected void finalize()

确保在不再引用文件输入流时调用其 close 方法。



## FileOutputStream 文件字节输出流

### 新增构造方法：

FileOutputStream（String name）

FileOutputStream(File file)

### (3) 文件字节输入流 文本文件

FileReader **char**[] b = **new** **char**[20];

### (4) 文件字节输入流

FileWriter

# 文件拷贝

## FileInputStream，FileOutputStream

### 直接拷贝

**public** **static** **void** demo1() **throws** FileNotFoundException, IOException {

FileInputStream fis = **new** FileInputStream("双元.jpg"); //创建输入流对象,关联双元.jpg

FileOutputStream fos = **new** FileOutputStream("copy.jpg"); //创建输出流对象,关联copy.jpg

**int** b;

**while**((b = fis.read()) != -1) { //在不断的读取每一个字节

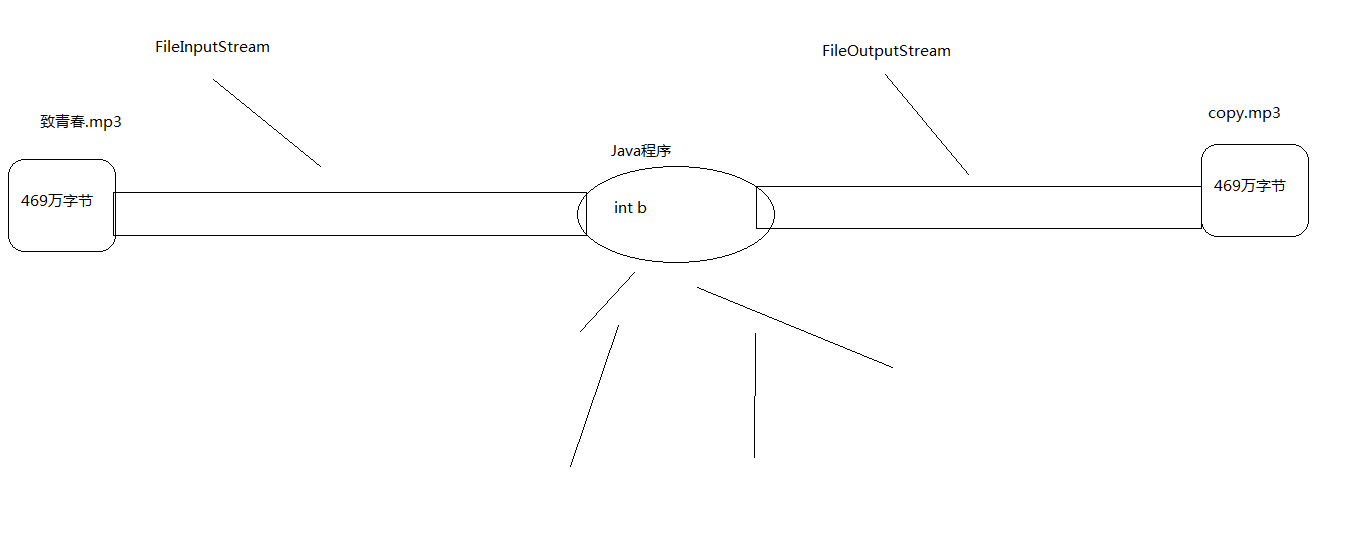
fos.write(b); //将每一个字节写出

}

fis.close(); //关流释放资源

fos.close();

}



### 定义大数组

**public** **static** **void** demo3() **throws** FileNotFoundException, IOException {

//第二种拷贝,不推荐使用,因为有可能会导致内存溢出

FileInputStream fis = **new** FileInputStream("致青春.mp3"); //创建输入流对象,关联致青春.mp3

FileOutputStream fos = **new** FileOutputStream("copy.mp3"); //创建输出流对象,关联copy.mp3

**int** len = fis.available();

System.*out*.println(len);

**byte**[] arr = **new** **byte**[fis.available()]; //创建与文件一样大小的字节数组

fis.read(arr); //将文件上的字节读取到内存中

fos.write(arr); //将字节数组中的字节数据写到文件上

fis.close();

fos.close();

}

### 定义小数组

**public** **static** **void** demo3() **throws** FileNotFoundException, IOException {

FileInputStream fis = **new** FileInputStream("致青春.mp3");

FileOutputStream fos = **new** FileOutputStream("copy.mp3");

**byte**[] arr = **new** **byte**[1024 \* 8];

**int** len;

**while**((len = fis.read(arr)) != -1) { //如果忘记加arr,返回的就不是读取的字节个数,而是字节的码表值

fos.write(arr,0,len);

}

fis.close();

fos.close();

}

## 缓冲流

**public** **static** **void** demo1() **throws** FileNotFoundException, IOException {

FileInputStream fis = **new** FileInputStream("致青春.mp3"); //创建输入流对象,关联致青春.mp3

FileOutputStream fos = **new** FileOutputStream("copy.mp3"); //创建输出流对象,关联copy.mp3

BufferedInputStream bis = **new** BufferedInputStream(fis); //创建缓冲区对象,对输入流进行包装让其变得更加强大

BufferedOutputStream bos = **new** BufferedOutputStream(fos);

**int** b;

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

bos.write(b);

}

bis.close();

bos.close();

}

# 缓冲流

## 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(装饰设计模式后面讲解)，所以提供了字节缓冲区流

把节点流包入缓冲流

## 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

## BufferedOutputStream类

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

BufferedInputStream bis = **new** BufferedInputStream(fis);

BufferedOutputStream bos = **new** BufferedOutputStream(fos);

//具体实现文本复制

**byte**[] b = **new** **byte**[20]; **char**[] b = **new** **char**[20];

**int** len ;

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

fos.write(b, 0, len );

}

### (2)BufferedReader和BufferedWriter类

读写文件的方法

readLine() BufferedReader流读取文本行

write(String s,int off,int len) BufferedWriter流把字符串s写到文件中

newLine(); BufferedWriter流向文件写入一个回行符

//具体实现文本复制

String str = **null**;

**while** ((str = br.readLine()) != **null**) {

bw.write(str);

bw.newLine(); //换行

bw.flush();

}

# 异常处理的标准代码

## jdk1.6以前

**public** **static** **void** demo1() **throws** FileNotFoundException, IOException {

FileInputStream fis = **null**;

FileOutputStream fos = **null**;

**try** {

fis = **new** FileInputStream("xxx.txt");

fos = **new** FileOutputStream("yyy.txt");

**int** b;

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

fos.write(b);

}

}**finally** {

**try**{

**if**(fis != **null**)

fis.close();

}**finally** { //try fianlly的嵌套目的是能关一个尽量关一个

**if**(fos != **null**)

fos.close();

}

}

}

## Jdk1.7

**public** **static** **void** demo2() **throws** IOException, FileNotFoundException {

**try**(

FileInputStream fis = **new** FileInputStream("xxx.txt");

FileOutputStream fos = **new** FileOutputStream("yyy.txt");

MyClose mc = **new** MyClose();

){

**int** b;

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

fos.write(b);

}

}

}

**class** MyClose **implements** AutoCloseable {

**public** **void** close() {

System.*out*.println("我关了");

}

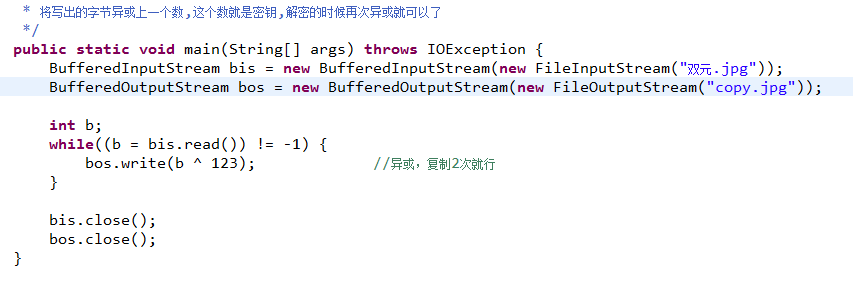
}

**在try()中创建的流对象必须实现了AutoCloseable这个接口,如果实现了,在try后面的{}(读写代码)执行后就会自动调用,流对象的close方法将流关掉**

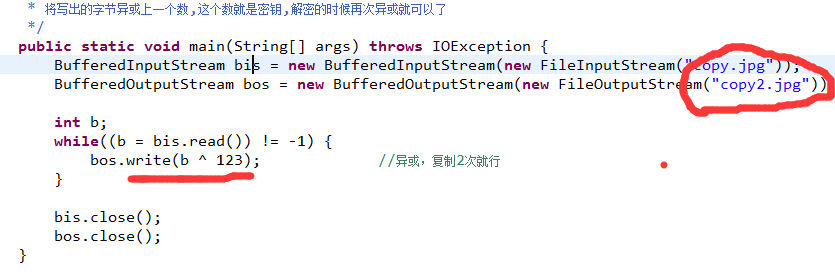
# 案例

## 文件加密

### 加密



### 解密



## 文件拷贝

**public** **class** Test2 {

/\*\*

\* 在控制台录入文件的路径,将文件拷贝到当前项目下

\*

\* 分析:

\*

\* 1,定义方法对键盘录入的路径进行判断,如果是文件就返回

\* 2,在主方法中接收该文件

\* 3,读和写该文件

\* **@throws** IOException

\*/

**public** **static** **void** main(String[] args) **throws** IOException {

File file = *getFile*(); //获取文件

BufferedInputStream bis = **new** BufferedInputStream(**new** FileInputStream(file));

BufferedOutputStream bos = **new** BufferedOutputStream(**new** FileOutputStream(file.getName()));

**int** b;

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

bos.write(b);

}

bis.close();

bos.close();

}

/\*

\* 定义一个方法获取键盘录入的文件路径,并封装成File对象返回

\* 1,返回值类型File

\* 2,参数列表无

\*/

**public** **static** File getFile() {

Scanner sc = **new** Scanner(System.*in*); //创建键盘录入对象

System.*out*.println("请输入一个文件的路径:");

**while**(**true**) {

String line = sc.nextLine(); //接收键盘录入的路径

File file = **new** File(line); //封装成File对象,并对其进行判断

**if**(!file.exists()) {

System.*out*.println("您录入的文件路径不存在,请重新录入:");

}**else** **if**(file.isDirectory()) {

System.*out*.println("您录入的是文件夹路径,请重新录入:");

}**else** {

**return** file;

}

}

}

}

## 录入数据拷贝到文件

**public** **class** Test3 {

/\*\*

\* 将键盘录入的数据拷贝到当前项目下的text.txt文件中,键盘录入数据当遇到quit时就退出

\*

\* 分析:

\* 1,创建键盘录入对象

\* 2,创建输出流对象,关联text.txt文件

\* 3,定义无限循环

\* 4,遇到quit退出循环

\* 5,如果不quit,就将内容写出

\* 6,关闭流

\* **@throws** IOException

\*/

**public** **static** **void** main(String[] args) **throws** IOException {

//1,创建键盘录入对象

Scanner sc = **new** Scanner(System.*in*);

//2,创建输出流对象,关联text.txt文件

FileOutputStream fos = **new** FileOutputStream("text.txt");

System.*out*.println("请输入数据:");

//3,定义无限循环

**while**(**true**) {

String line = sc.nextLine(); //将键盘录入的数据存储在line中

//4,遇到quit退出循环

**if**("quit".equals(line)) {

**break**;

}

//5,如果不quit,就将内容写出

fos.write(line.getBytes()); //字符串写出必须转换成字节数组

fos.write("\r\n".getBytes());

}

//6,关闭流

fos.close();

}

}