10 IO流_字节流

JAVAEE高级

一：字节流

> 1、可以读取任意文件

1、读数据

```
@Test
    public void getVoid() throws IOException {
        FileInputStream fis = new FileInputStream("D:\\IO\\修改后.txt");//创建流对象
        int b;
        //fis.read()) 返回的是二进制
        while((b = fis.read()) != -1) {
            System.out.println(b);//码表对应的值
        }
        fis.close();
    }
```

①、read()方法读取的是一个字节、为什么返回是int、而不是byte？

> 1、因为字节输入流可以操作任意类型的文件、比如图片音频等
>
> 2、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
>
> 3、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
>
> 4、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
>
> 5、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

2、写数据

> 1、FileOutputStream在创建对象的时候是如果没有这个文件会帮我创建出来
>
> 2、如果有这个文件就会先将文件清空，在进行写出

```
    @Test
    public void getVoid() throws IOException {
        // 如果想续写(在后面进行追加)就在第二个参数传true
        FileOutputStream fos = new FileOutputStream("D:\\IO\\修改后.txt", true);
        fos.write(97);//虽然写出的是一个int数,但是到文件上的是一个字节
        fos.write(98);
        fos.close();
    }
```

二：案例

1、进行使用字节流，拷贝图片

> 1、如果图片非常大，那怎么来进行读的更快点呢？

```
    @Test
    public void getVoid() throws IOException {
         // 创建输入流对象(读)
        FileInputStream fis = new FileInputStream("D:\\IO\\微信图片.png");
        // 创建输出流对象(写)
        FileOutputStream fos = new FileOutputStream("D:\\IO\\微信图片copy.png");
        int b;
        // 在不断的读取每一个字节
        while ((b = fis.read()) != -1) {
            // 将每一个字节写出
            fos.write(b);
```

```
            }
        fis.close(); // 关流释放资源
        fos.close();
    }
```

三：案例优化

> 2、一次读的更多，并进行一次性的写入，使用数组的方式进行完成

```
@Test
    public void getVoid() throws IOException {
         // 创建输入流对象(读)
        FileInputStream fis = new FileInputStream("D:\\IO\\微信图片.png");
        // 创建输出流对象(写)
        FileOutputStream fos = new FileOutputStream("D:\\IO\\微信图片copy.png");
        int available = fis.available();//读文件的大小
        //创建一个与文件大小一致 的数组，可自定义
        byte[] ar1r = new byte[available];
        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();
    }
```

四：缓冲流

1、缓冲思想

> 1、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
>
> 2、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，所以提供了

字节缓冲区流

```
    @Test
    public void getVoid() throws IOException {
         // 创建输入流对象(读)
         //BufferedInputStream创建缓冲区对象,对输入流进行包装让其变得更加强大
        BufferedInputStream bis = new BufferedInputStream(new FileInputStream("D:\\IO\\微信图片.png"));
        // 创建输出流对象(写)
        //BufferedOutputStream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(new FileOutputStream("D:\\IO\\微信图片copy.png"));
        int b;
        while((b = bis.read()) != -1) {
            bos.write(b);
        }
        bis.close();
        bos.close();;
    }
```

2、原理

1、BufferedInputStream

1、BufferedInputStream内置了一个缓冲区(数组)

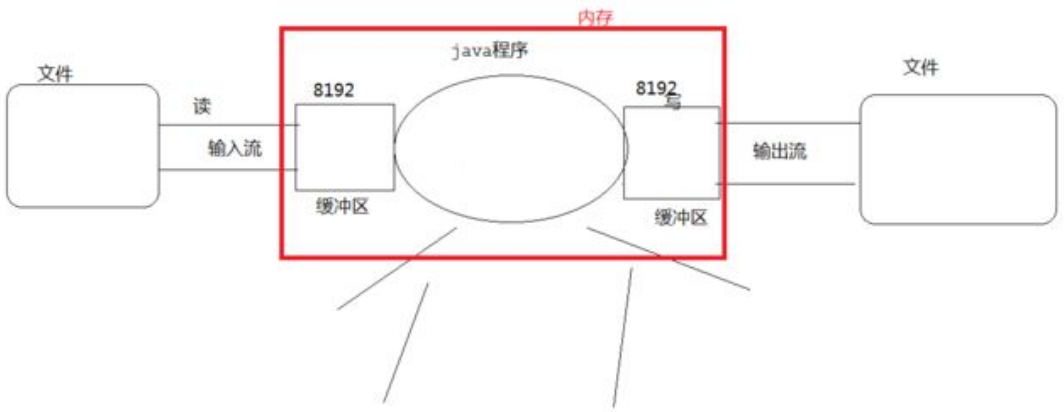
2、BufferedInputStream会一次性从文件中读取8192， 存在缓冲区中 返回给程序一个，程序再次读取时，就不用找文件了，直接从缓冲区中获取
，直到缓冲区中所有的都被使用过，，才重新从文件中读取8192个

2、BufferedOutputStream

1、BufferedOutputStream也内置了一个缓冲区(数组)

2、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

3、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



3、flush与close区别

> 1、flush()方法
>
> 用来刷新缓冲区的，刷新后可以再次写出
>
> 2、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

五：字节流读写中文

1、读

```
@Test
    public void getVoid() throws IOException {
         // 创建输入流对象(读)
        FileInputStream fis = new FileInputStream("D:\\IO\\修改后.txt");
        byte[] arr = new byte[1024];
        int len;
        while((len = fis.read(arr)) != -1) {
            System.out.println(new String(arr,0,len));
        }
        fis.close();
    }
```

2、写

```
    @Test
    public void getVoid() throws IOException {
         // 创建输出流对象(写)
        FileOutputStream fos = new FileOutputStream("D:\\IO\\修改后.txt",true);
        fos.write("我读书少,你不要骗我".getBytes());
        fos.write("\r\n".getBytes());//换行
        fos.close();
    }
```

3、jdk1.7以后关闭流的方法

```
public class TestJava {
    @Test
    public void getVoid() throws IOException {
        try (FileInputStream fis = new FileInputStream("D:\\IO\\修改后.txt");
                FileOutputStream fos = new FileOutputStream("D:\\IO\\dir2\\修改后1.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("关了");
    }
}
```

4、案例<加密图片>

> 1、将写出的字节异或上一个数、这个数就是密钥；
> 2、解密的时候再次异或就可以了

```
1.      @Test
2.      public void getVoid() throws IOException {
3.          BufferedInputStream bis = new BufferedInputStream(new FileInput
    Stream("D:\\IO\\微信图片.png"));
4.          BufferedOutputStream bos = new BufferedOutputStream(new FileOut
    putStream("D:\\IO\\微信图片加密后.png"));
5.          int b;
6.          while ((b = bis.read()) != -1) {
7.              bos.write(b ^ 123);
8.          }
9.          bis.close();
10.         bos.close();
11.     }
```