【济南中心】JAVA 编程阶梯：基础篇之第二十一章

IO 流字符流 FileReader

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.

FileReader 类的 read()方法可以按照字符大小读取

```
FileReader fr = new FileReader("text.txt");        //创建输入流对象,关联 aaa.txt
    int ch;
    while((ch = fr.read()) != -1) {                //将读到的字符赋值给 ch
      System.out.println((char)ch);                //将读到的字符强转后打印
    }
    fr.close();                                    //关流
```

IO 流字符流 FileWriter

FileWriter 类的 write()方法可以自动把字符转为字节写出

```
FileWriter fw = new FileWriter("text.txt");
                                fw.write("text");
                                fw.close();
```

IO 流字符流的拷贝

```
FileReader fr = new FileReader("a.txt");
                FileWriter fw = new FileWriter("b.txt");
                int ch;
                while((ch = fr.read()) != -1) {
                                fw.write(ch);
                }
                fr.close();
                fw.close();
```

IO 流使用指定的码表读写字符

FileReader 是使用默认码表读取文件，如果需要使用指定码表读取，那么可以使用 InputStreamReader(字节流,编码表)

FileWriter 是使用默认码表写出文件，如果需要使用指定码表写出，那么可以使用 OutputStreamWriter(字节流,编码表)

```
BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream("UTF-8.txt"), "UTF-8"));
                                                    //高效的用指定的编码表读
                        BufferedWriter bw =     new BufferedWriter(new OutputStreamWriter(new FileOutputStream("GBK.txt"), "GBK"));
                                                    //高效的用指定的编码表写
                        int ch;
                        while((ch = br.read()) != -1) {
                                    bw.write(ch);
                        }
                        br.close();
                        bw.close();
```

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

字符流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

IO 流自定义字符数组的拷贝

```
FileReader fr = new FileReader("aaa.txt");                                                    //创建字符输入流,关联 aaa.txt
                        FileWriter fw = new FileWriter("bbb.txt");                                    //创建字符输出流,关联 bbb.txt

                        int len;
```

```
                char[] arr = new
char[1024*8];
//创建字符数组
                while((len = fr.read(arr)) != -1)
{                                                       //将数据读到字符数组中
                        fw.write(arr, 0,
len);                                                                   //从字符
数组将数据写到文件上
                }
                fr.close();
                                        //关流释放资源
                fw.close();
```

IO 流带缓冲的字符流

BufferedReader 的 read()方法读取字符时会一次读取若干字符到缓冲区，然后逐个返回给程序，降低读取文件的次数，提高效率

BufferedWriter 的 write()方法写出字符时会先写到缓冲区，缓冲区写满时才会写到文件，降低写文件的次数，提高效率

```
BufferedReader br = new BufferedReader(new FileReader("aaa.txt"));              //创建字符输
入流对象,关联 aaa.txt
                BufferedWriter bw = new BufferedWriter(new
FileWriter("bbb.txt"));         //创建字符输出流对象,关联 bbb.txt

                int
ch;
                while((ch = br.read()) != -1) {                 //read 一次,
会先将缓冲区读满,从缓冲去中一个一个的返给临时变量 ch
                                bw.write(ch);
                //write 一次,是将数据装到字符数组,装满后再一起写出去
                }
                br.close();
                                        //关流
                bw.close();
```

IO 流 readLine()和 newLine()方法

BufferedReader 的 readLine()方法可以读取一行字符(不包含换行符号)

BufferedWriter 的 newLine()可以输出一个跨平台的换行符号"\r\n"

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

```
BufferedWriter bw = new BufferedWriter(new
FileWriter("bbb.txt"));
String line;
while((line = br.readLine()) != null) {
        bw.write(line);
        //bw.write("\r\n");
              //只支持 windows 系统
        bw.newLine();
                    //跨平台的
}
br.close();
bw.close();
```

IO 流 LineNumberReader

LineNumberReader是BufferedReader的子类, 具有相同的功能, 并且可以统计行号

调用 getLineNumber()方法可以获取当前行号

调用 setLineNumber()方法可以设置当前行号

```
LineNumberReader lnr = new LineNumberReader(new FileReader("aaa.txt"));
String line;
lnr.setLineNumber(100);

        //设置行号
while((line = lnr.readLine()) != null) {
        System.out.println(lnr.getLineNumber() + ":"
+ line);//获取行号

}

lnr.close();
```

装饰设计模式

```
interface Coder {
        public void code();
}

class Student implements Coder {

        @Override
        public void code() {
```

```
                                                        System.out.println("javase")
;
                                                        System.out.println("javaweb")
;
                                                }

                                }

                                class HeiMaStudent implements Coder {
                                                private Student
s;
    //获取到被包装的类的引用
                                                public ItcastStudent(Student s)
{                               //通过构造函数创建对象的时候,传入被包装的对象
                                                        this.s = s;
                                                }
                                                @Override
                                                public void code()
{                                                                       //对其原有功
能进行升级
                                                        s.code();
                                                        System.out.println("数据库
");
                                                        System.out.println("ssh");
                                                        System.out.println("安卓");
                                                        System.out.println(".....");
                                                }

                                }
```

IO 流递归遍历删除文件夹

```
class RemoveDir{
        public static void main(String[] args){
                File dir = new File("E:\\kankan");
                removeDir(dir);
        }

        public static void removeDir(File dir){ //删除 dir 目录下所有内容
                File[] files = dir.listFiles(); //列出所有文件及目录，返回 File[]

                for(int x=0; x<files.length; x++){
                        if(files[x].isDirectory()) //若是目录，继续递归调用
                                removeDir(files[x]);
                        else
```

```
                System.out.println(files[x].toString()+"::file::"+files[x].d
elete());
            }

            //继续删除空目录
            System.out.println(dir+"::dir::"+dir.delete());
        }
}
```