File类的使用

- 概述

File类在包java.io.File下、代表操作系统的文件对象（文件、文件夹）

File类提供了诸如：定位文件，获取文件本身的信息、删除文件、创建文件（文件夹）等功能

File对象可以定位文件和文件夹

File封装的对象仅仅是一个路径名，这个路径可以是存在的，也可以是不存在的

- **绝对路径和相对路径**

绝对路径是带盘符的，依赖当前系统
相对路径：不带盘符，默认直接到当前工程下的目录寻找文件
当前工程是空工程，就 模块名/src/...【如果你书写的代码正位于项目下，就不用模块名，直接src/】

File类创建对象

方法名称	说明
`public File(String pathname)`	根据文件路径创建文件对象
`public File(String parent, String child)`	从父路径名字符串和子路径名字符串创建文件对象
`public File(File  parent, String child)`	根据父路径对应文件对象和子路径名字符串创建文件对象

```
    @Test
    public void test1() {
        //D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\PPT\File、递归、IO.pdf
        //可以加杠
        //final File file = new File("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流(一)\\PPT\\",
"File、递归、IO.pdf");
        //可以不加杠
        final File file = new File("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流(一)\\PPT", "File、递
归、IO.pdf");
        //可以加杠 可以不加杠
        final File file1 = new File(new File("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流
(一)\\PPT\\"),
                                    "File、递归、IO.pdf");
        final File file2 = new File("src/logback.xml");
        final File file3 = new File("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流(一)\\PPT\\File、递
归、IO.pdf");
        System.out.println(file.getName());
        System.out.println(file1.getName());
        System.out.println(file2.getName());
        System.out.println(file3.getName());
    }
```

```
File、递归、IO.pdf
File、递归、IO.pdf
logback.xml
File、递归、IO.pdf
```

File类的判断文件类型、获取文件信息功能

方法名称	说明
`public boolean isDirectory()`	测试此抽象路径名表示的File是否为文件夹
`public boolean isFile()`	测试此抽象路径名表示的File是否为文件
`public boolean exists()`	测试此抽象路径名表示的File是否存在
`public String getAbsolutePath()`	返回此抽象路径名的绝对路径名字符串
`public String getPath()`	将此抽象路径名转换为路径名字符串
`public String getName()`	返回由此抽象路径名表示的文件或文件夹的名称
`public long lastModified()`	返回文件最后修改的时间毫秒值

```
    @Test
    public void tester1() {
        File file = new File("D:", "\\learn\\黑马程序员\\java基础\\《Java面试手册》.pdf");
        System.out.println("该路径是否存在文件：" + file.exists());
        System.out.println("文件？：" + file.isFile());
```

```
        System.out.println("文件夹？：" + file.isDirectory());
        System.out.println("绝对路径：" + file.getAbsolutePath());
        System.out.println("文件定义时使用的路径：" + file.getPath());
        System.out.println("文件名：" + file.getName());
        System.out.println("文件大小：" + file.length());
        System.out.println("最后修改时间：" + new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").format(file.lastModified()));
        System.out.println("-----------------------------------------------------------------------");
    }

    @Test
    public void testFile02() {
        File file = new File("src/Contacts2.xml");
        boolean exists = file.exists();
        boolean file1 = file.isFile();
        boolean directory = file.isDirectory();
        String name = file.getName();
        String absolutePath = file.getAbsolutePath();
        String path = file.getPath();
        long length = file.length();
        long l = file.lastModified();
        System.out.println("该路径是否存在文件：" + exists);
        System.out.println("文件？：" + file1);
        System.out.println("文件夹？：" + directory);
        System.out.println("绝对路径：" + absolutePath);
        System.out.println("文件定义时使用的路径：" + path);
        System.out.println("文件名：" + name);
        System.out.println("文件大小：" + length);
        System.out.println("最后修改时间：" + new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(l));
    }

    @Test
    public void testFile03() {
        File file = new File("src/ntacts2.xml");
        boolean exists = file.exists();
        boolean file1 = file.isFile();
        boolean directory = file.isDirectory();
        String name = file.getName();
        String absolutePath = file.getAbsolutePath();
        String path = file.getPath();
        long length = file.length();
        long l = file.lastModified();
        System.out.println("该路径是否存在文件：" + exists);
        System.out.println("文件？：" + file1);
        System.out.println("文件夹？：" + directory);
        System.out.println("绝对路径：" + absolutePath);
        System.out.println("文件定义时使用的路径：" + path);
        System.out.println("文件名：" + name);
        System.out.println("文件大小：" + length);
        System.out.println("最后修改时间：" + new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(l));
    }
```

```
该路径是否存在文件：true
文件？：true
文件夹？：false
绝对路径：D:\learn\黑马程序员\java基础\《Java面试手册》.pdf
文件定义时使用的路径：D:\learn\黑马程序员\java基础\《Java面试手册》.pdf
文件名：《Java面试手册》.pdf
文件大小：11086731
最后修改时间：2022-01-12 10:34:49
-----------------------------------------------------------------------
该路径是否存在文件：true
文件？：true
文件夹？：false
绝对路径：D:\learn\Idea\basis\src\Contacts2.xml
文件定义时使用的路径：src\Contacts2.xml
文件名：Contacts2.xml
文件大小：683
最后修改时间：2021-10-15 09:15:23
-----------------------------------------------------------------------
该路径是否存在文件：false
文件？：false
文件夹？：false
绝对路径：D:\learn\Idea\basis\src\ntacts2.xml
文件定义时使用的路径：src\ntacts2.xml
文件名：ntacts2.xml
文件大小：0
最后修改时间：1970-01-01 08:00:00
```

File类创建文件的功能

方法名称	说明
public boolean createNewFile()	创建一个新的空的文件
public boolean mkdir()	只能创建一级文件夹
public boolean mkdirs()	可以创建多级文件夹

File类删除文件的功能

方法名称	说明
public boolean delete()	删除由此抽象路径名表示的文件或空文件夹

```
        File file = new File("src/aaa.txt");
        System.out.println(file.createNewFile()); //当该文件存在时，false；否则true

        File file = new File("src/a/a");
        System.out.println(file.mkdir());//mkdir()是创建一级目录，当超过一级就会返回false；已经存在也会返回false

        File file = new File("src/c/c/c/c");
        System.out.println(file.mkdirs());//mkdirs()是创建多级目录，一级也可以；已经存在也会返回false

        File file = new File("src/c/a.txt");
        System.out.println(file.delete());//只能删除空文件夹或文件，文件夹非空返回false
```

File类的遍历功能

方法名称	说明
public String[] list()	获取当前目录下所有的"一级文件名称"到一个字符串数组中去返回。
public File[] listFiles()(常用)	获取当前目录下所有的"一级文件对象"到一个文件对象数组中去返回（重点）

```
@Test
public void testFile4() throws Exception {
    File file = new File("D:\\learn\\Idea\\basis\\src");
    String[] list = file.list();//获取该文件夹一级目录的所有文件名
    for (String s : list) {
        System.out.println(s);
    }
}
```

```
aaa.txt
array
bbb.txt
books.xml
branch
ccc.txt
collection
commonlyUsedApi
Contacts.xml
Contacts2.xml
file_io_demo
inherit
lib
logback
logback.xml
loop
method
operator_type
outputstreamwriter.txt
serializable
serializable.txt
static_demo
student.txt
xml_demo
```

```
    @Test
    public void test2() {
        //D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\资源
        File file = new File("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流(一)\\资源");
        File[] files = file.listFiles();//获取该文件夹一级目录的所有文件对象(当这个文件对象是空文件夹时，返回一个空数组)
        System.out.println(Arrays.toString(files));

        final File file2 = new File("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流(一)\\PPT");
        File[] files1 = file2.listFiles();//获取该文件夹一级目录的所有文件对象
        for (File file1 : files1) {
            System.out.println(file1.getAbsolutePath());
        }
    }
```

```
[]
D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\PPT\18、File、递归、IO一.pptx
D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\PPT\218、File、递归、IO一.pptx
D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\PPT\day08、阶段项目.pptx
D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\PPT\File、递归、IO.html
D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\PPT\File、递归、IO.md
D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\PPT\File、递归、IO.pdf
D:\learn\黑马程序员\java基础\day19、File、递归、IO流(一)\PPT\IO二.pdf
```

listFiles方法注意事项：

当调用者不存在时，返回null

当调用者是一个文件时，返回null

当调用者是一个空文件夹时，返回一个长度为0的数组

当调用者是一个有内容的文件夹时，将里面所有文件和文件夹的路径放在File数组中返回

当调用者是一个有隐藏文件的文件夹时，将里面所有文件和文件夹的路径放在File数组中返回，包含隐藏内容

当调用者是一个需要权限才能进入的文件夹时，返回null

方法递归

- 定义

方法直接调用自己或者间接调用自己的形式称为方法递归（recursion）

- 问题

递归如果没有控制好终止（边界），会出现递归死循环，导致栈内存溢出现象。

- 总结

递归算法三要素大体可以总结为：
　递归的公式： f(n) = f(n-1) * n;
　递归的终结点：f(1)
　递归的方向必须走向终结点
核心思想：
　递深归回

- 递归查找文件

```
    public static void main(String[] args) {
        long l1 = System.currentTimeMillis();
        search(new File("D:\\"), "JAVA");
        long l2 = System.currentTimeMillis();
        long l3 = l2 - l1;
        System.out.println(l3 / 1000 + "s");
    }


    public static void search(File target, String name) {
        //是一个文件夹
        if (target != null && target.isDirectory()) {
            //搜寻一级目录
            File[] listFiles = target.listFiles();

            //调用者存在，调用者不是文件，有权限，目录不为空
            if (listFiles != null && listFiles.length > 0) {
                for (File file : listFiles) {
                    if (file.isFile()) {
                        if (file.getName().contains(name)) {
                            System.out.println("所寻找的文件位于：" + file.getAbsolutePath());
                            //如果打破就是找到一个就不找，不打破就是找该文件夹的所有文件
                        }
                        //是文件夹
                    } else {
                        search(file, name);
                    }
                }
            }
              //null []

        }
    }
```

- 啤酒问题

```
需求：啤酒2元1瓶，4个盖子可以换一瓶，2个空瓶可以换一瓶，
请问10元钱可以喝多少瓶酒，剩余多少空瓶和盖子。
答案：15瓶 3盖子 1瓶子
```

```
    private static final int price = 2;
    private static int lastBottleCover;
    private static int lastNullBottle;
    private static int allBottle;

    public static void main(String[] args) {
        buy(10);
        System.out.println(allBottle);
        System.out.println(lastBottleCover);
        System.out.println(lastNullBottle);
    }

    public static void buy(int money) {
        int bottle = money / price;
        allBottle += bottle;
        int bottleCover = bottle + lastBottleCover;
        int nullBottle = bottle + lastNullBottle;
        int allMoney = bottleCover / 4 * price + nullBottle / 2 * price;
        lastNullBottle = nullBottle % 2;
        lastBottleCover = bottleCover % 4;
        if (allMoney >= price) {
            buy(allMoney);
        }
    }
```

字符集

- 定义

计算机底层只能存储010101二进制数

计算机对字符进行编号，编号的规则就是字符集

Unicode会先通过UTF-8，UTF-16，以及 UTF-32的编码成二进制后再存储到计算机，其中最为常见的就是UTF-8

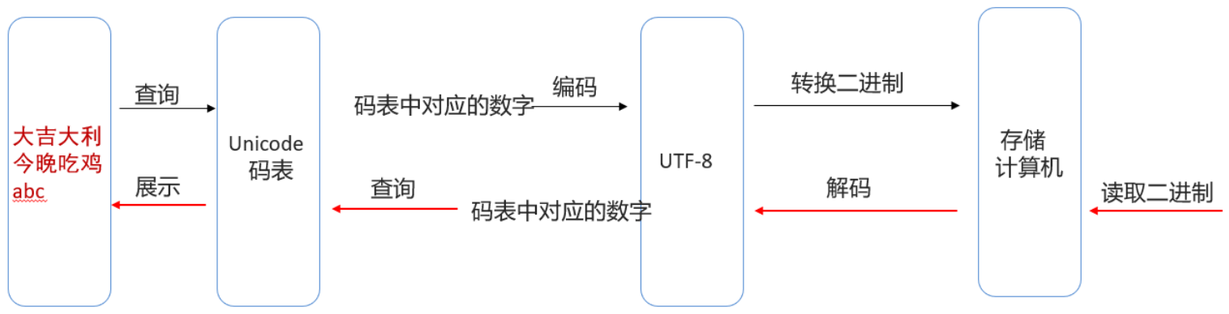
Unicode是万国码一个汉字两个字节，以UTF-8编码后一个中文 一般 以三个字节的形式存储

UTF-8也要兼容ASCII编码表

技术人员都应该使用UTF-8的字符集编码。
编码前和编码后的字符集需要一致，否则会出现中文乱码。
英文和数字等在任何国家的字符集中都占1个字节
GBK字符中一个中文字符占2个字节

UTF-8就是在互联网上使用最广的一种unicode的实现方式，这是为传输而设计的编码，并使编码无国界，这样就可以显示全世界上所有文化的字符了。UTF-8最大的一个特点，就是它是一种变长的编码方式。它可以使用1~4个字节表示一个符号，根据不同的符号而变化字节长度，当字符在ASCII码的范围时，就用一个字节表示，保留了ASCII字符一个字节的编码做为它的一部分，注意的是unicode一个中文字符占2个字节，而UTF-8一个中文字符占3个字节

汉字存储和展示过程解析



String编码

方法名称	说明
byte[] getBytes()	使用平台的默认字符集将该 String编码为一系列字节，将结果存储到新的字节数组中
byte[] getBytes(String charsetName)	使用指定的字符集将该 String编码为一系列字节，将结果存储到新的字节数组中

String解码

构造器	说明
String(byte[] bytes)	通过使用平台的默认字符集解码指定的字节数组来构造新的 String
String(byte[] bytes, String charsetName)	通过指定的字符集解码指定的字节数组来构造新的 String

---

```
        String name = "ab十六分公交卡";
        //以GBK的字符编码方式，将字符串转为字节数组
        byte[] bytes = name.getBytes("GBK");   //[97, 98, -54, -82, -63, -7, -73, -42, -71, -85, -67, -69,
-65, -88]

//        byte[] bytes = name.getBytes();//以平台默认的字符编码方式，将字符串转为字节数组
//[97, 98, -27, -115, -127, -27, -123, -83, -27, -120, -122, -27, -123, -84, -28, -70, -92, -27, -115, -95]

        String s=new String(bytes,"GBK");
        System.out.println(s);
```

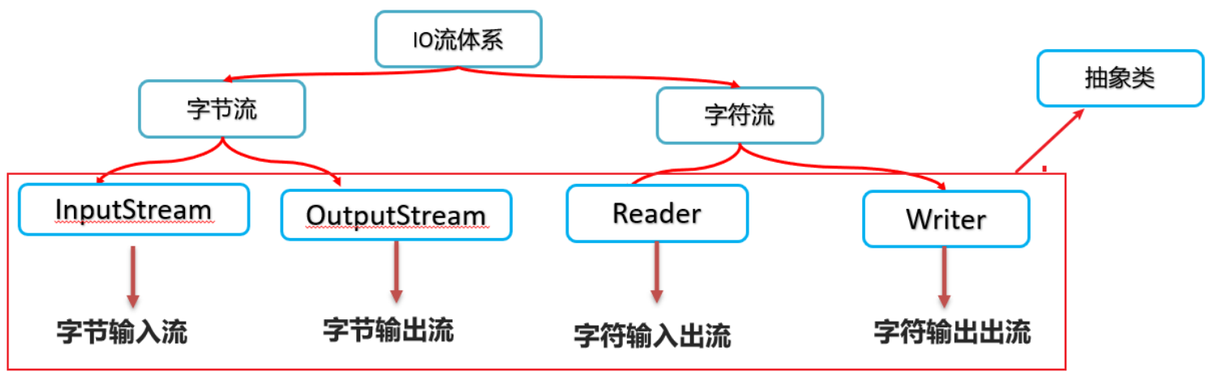
IO流的作用、分类

- IO流的分类

按流向分为：
输入流：流入内存
输出流：流出内存

按数据的最小单位：
字节流：可以读取任何数据，尤其适合音频视频图片等
字符流：仅适合读取文本文件

总结流的四大类：
字节输入流：以内存为基准，来自磁盘文件/网络中的数据以字节的形式读入到内存中去的流称为字节输入流。
字节输出流：以内存为基准，把内存中的数据以字节写出到磁盘文件或者网络中去的流称为字节输出流。
字符输入流：以内存为基准，来自磁盘文件/网络中的数据以字符的形式读入到内存中去的流称为字符输入流。
字符输出流：以内存为基准，把内存中的数据以字符写出到磁盘文件或者网络介质中去的流称为字符输出流



字符流、字节流

- 定义

内存到文件的一条 管道

文件字节输入流：FileInputStream

- 作用：以内存为基准，把磁盘文件中的数据以字节的形式读取到内存中去。

构造器	说明
public FileInputStream(File file)	创建字节输入流管道与源文件对象接通
public FileInputStream(String pathname)	创建字节输入流管道与源文件路径接通

方法名称	说明
public int read()	每次读取一个字节返回，如果字节已经没有可读的返回-1
public int read(byte[] buffer)	每次读取一个字节数组返回，如果字节已经没有可读的返回-1

- 低效，乱码

```
/**
*   //字节输入流 之 一次读取一滴
*/
public static void main(String[] args) throws Exception {
        InputStream is = new FileInputStream("src/aaa.txt");
        int i;//读到的字节存在这里
        while ((i = is.read()) != -1) {//每次读取一个字节，当没有数据时返回-1;性能低，乱码
            System.out.print((char)i);
        }
    }
```

```
/**
*   字节输入流 之 一次读取一桶
*/

    //  1绿2
    @Test
    public void testInStream() throws Exception {
        InputStream is = new FileInputStream("src/aaa.txt");
        byte[] buffer = new byte[3];//读到的数据存在这里
        int length;//这里记录独到的数据的字节长度
        while ((length = is.read(buffer)) != -1) {
            System.out.println(Arrays.toString(buffer));
            System.out.println(new String(buffer, 0, length));
        }
    }
```

```
[49, -25, -69]
1�
[-65, 50, -69]//-69是上一次的数据，如果没有被覆盖掉，会一直存在；所以要取多少，读多少
 �2
```

- 解决乱码（傻狗方式）

```
字节输入流 之 一次读取一桶,一桶有文件那么大
    @Test
    public void testInStream3() throws Exception {
        File file = new File("src/aaa.txt");
        InputStream is = new FileInputStream(file);
        byte[] bytes = new byte[(int) file.length()];//桶大小与文件大小一致
        int read = is.read(bytes);//一次读完
        System.out.println(new String(bytes, 0, read));//有多少读取多少
    }
```

- 小结

一次性读取完全部字节。
可以定义与文件一样大的字节数组读取
如果文件过大，定义的字节数组可能引起内存溢出

文件字节输出流：FileOutputStream

- 作用：以内存为基准，把内存中的数据以字节的形式写出到磁盘文件中去的流。

构造器	说明
`public FileOutputStream(File file)`	创建字节输出流管道与源文件对象接通
`public FileOutputStream(File file, boolean append)`	创建字节输出流管道与源文件对象接通，可追加数据
`public FileOutputStream(String filepath)`	创建字节输出流管道与源文件路径接通
`public FileOutputStream(String filepath, boolean append)`	创建字节输出流管道与源文件路径接通，可追加数据

流的关闭与刷新

方法	说明
flush()	刷新流，还可以继续写数据
close()	关闭流，释放资源，但是在关闭之前会先刷新流。一旦关闭，就不能再写数据

```
一次写一个字节
@Test
public void testStream1() throws Exception {
    FileOutputStream os = new FileOutputStream("D:\\learn\\Idea\\basis\\src\\bbb.txt", true);//追加数据
    os.write('a');//一次写一个字节，英文字母 byte，short，char，int可以相互转换
    os.write(48);//转为二进制的48，此时还在Ascii的范围中，会被转成0
    os.write('徐');//用utf-8编码会编成三个字节，但只能写一个字节
    os.flush();//刷新数据，管道还在链接着
    os.close();//关闭管道，并刷新管道
}
```

```
a0?
```

文件字节输出流（FileOutputStream）写数据出去的API

方法名称	说明
`public void write(int a)`	写一个字节出去
`public void write(byte[] buffer)`	写一个字节数组出去
`public void write(byte[] buffer , int pos , int len)`	写一个字节数组的一部分出去。

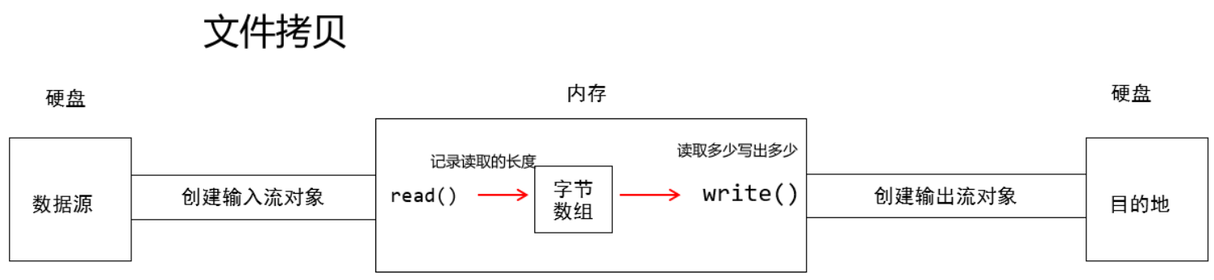
```
一次写一桶字节
    @Test
    public void testStream3() throws Exception {
        FileOutputStream os = new FileOutputStream("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流(一)\\资源\\ccc.txt");
        os.write("为天地立心，".getBytes());//一次写一个字节数组
        os.write("\r\n".getBytes());//回车+换行
        os.write("为生民立命，".getBytes());
        os.write("\r\n".getBytes());
        os.write("为往圣继绝学，".getBytes());
        os.write("\r\n".getBytes());
        os.write("为万世开太平.啊".getBytes(), 0, 19);//这是字节数组的起始位置，和长度
        os.write("\r\n".getBytes());
        os.flush();//刷新数据，管道还在链接着
        os.close();//关闭管道，并刷新管道
    }
```

```
为天地立心，
为生民立命，
为往圣继绝学，
为万世开太平.
```

- 回车与换行

\r与\n与\r\n的区别图解

机械打字机有回车和换行两个键作用分别是:
换行就是把滚筒卷一格，不改变水平位置。
回车就是把水平位置复位，不卷动滚筒



```
    @Test
    public void testStream4() throws Exception {
        FileInputStream is = new FileInputStream("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流(一)\\
                                                 PPT\\18、File、递归、IO一.pptx");
        FileOutputStream os = new FileOutputStream("D:\\learn\\黑马程序员\\java基础\\day19、File、递归、IO流
(一)\\
                                                   PPT\\218、File、递归、IO一.pptx");
        byte[] buffer = new byte[1024];
        int length;
        while ((length = is.read(buffer)) != -1) {
            os.write(buffer, 0, length);//注意：读取多少倒出多少
            os.flush();//刷新数据，管道还在链接着
        }
        is.close();//关闭管道
        os.close();//关闭管道，并刷新管道
    }
```

任何文件的底层都是字节，拷贝是一字不漏的转移字节，只要前后文件格式、编码一致没有任何问题

- （管道）流的关闭

```
//异常处理标准格式：try….catch…finally
    @Test
    public void test_1() {
        InputStream is = null;
        try {
            is = new FileInputStream("aaaaa");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("最终执行======");//无论是否发生异常都会执行
            try {       //麻烦
                if (is != null) {//防止空指针
                    is.close();//关闭会抛异常
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

finally：在异常处理时提供finally块来执行所有清除操作，比如说IO流中的释放资源
特点：被finally控制的语句最终一定会执行，除非JVM退出

```
//   try....catch...resource
    @Test
    public void test_2() {
        try (
                //把资源放在try括号内部，在使用完成资源后会自动释放资源，即使出现异常也会
                Myconnection myconnection = new Myconnection();
        ) {
            System.out.println(10 / 0);
            System.out.println("此处代码不会执行");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

class Myconnection implements Closeable{
    @Override
    public void close() throws IOException {
        System.out.println("关闭连接------------------------------");
    }
}
```

```
关闭连接------------------------------
java.lang.ArithmeticException: / by zero
进程已结束，退出代码为 0
```

- 小结

  try….catch…resource 只能放置资源对象，否则报错

资源对象：

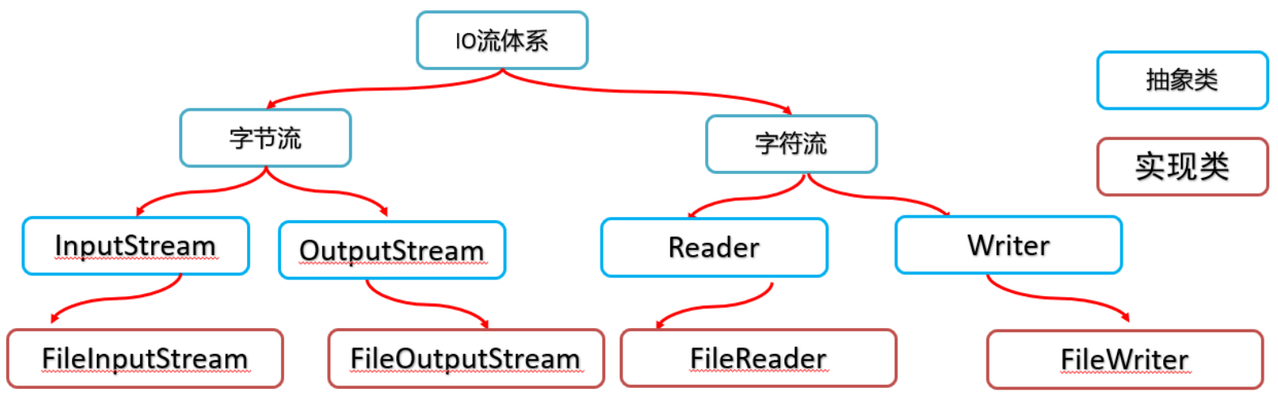
都是实现了Closeable/AutoCloseable接口的类对象

```
public abstract class InputStream implements Closeable {}
public abstract class OutputStream implements Closeable, Flushable{}
```

自动释放资源、代码简洁

- 字符流

字节流读取中文输出会存在乱码。或者内存溢出。
读取中文输出，字符流更合适，最小单位是按照单个字符读取的。



文件字符输入流：Reader

- 作用：以内存为基准，把磁盘文件中的数据以字符的形式读取到内存中去。

构造器	说明
public FileReader(File file)	创建字符输入流管道与源文件对象接通
public FileReader(String pathname)	创建字符输入流管道与源文件路径接通

方法名称	说明
public int read()	每次读取一个字符返回，如果字符已经没有可读的返回-1
public int read(char[] buffer)	每次读取一个字符数组，返回读取的字符个数，如果字符已经没有可读的返回-1

一次读取一个字符

```
    @Test
    public void test1() {
        try (
                Reader reader = new FileReader("src/ccc.txt");
        ) {
            int flag;//把一个字符放到这里
            while ((flag = reader.read()) != -1) {
                System.out.print((char) flag);//0---65534
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

读取中文字符不会出现乱码（如果代码文件编码一致）
性能较慢

一次读取一桶字符

```
    @Test
    public void test2() {
        try (
                Reader reader = new FileReader("src/ccc.txt");

        ) {
            int length;//记录字符数组长度
            char[] buffer = new char[3];//存取数据
            while ((length = reader.read(buffer)) != -1) {
                System.out.print(new String(buffer, 0, length));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

读取的性能得到了提升
读取中文字符输出不会乱码

文件字符输出流：FileWriter

- 作用：以内存为基准，把内存中的数据以字符的形式写出到磁盘文件中去的流。

构造器	说明
public FileWriter(File file)	创建字符输出流管道与源文件对象接通
public FileWriter(File file, boolean append)	创建字符输出流管道与源文件对象接通，可追加数据
public FileWriter(String filepath)	创建字符输出流管道与源文件路径接通
public FileWriter(String filepath, boolean append)	创建字符输出流管道与源文件路径接通，可追加数据

文件字符输出流（FileWriter）写数据出去的API

方法名称	说明
void write(int c)	写一个字符
void write(char[] cbuf)	写入一个字符数组
void write(char[] cbuf, int off, int len)	写入字符数组的一部分
void write(String str)	写一个字符串
void write(String str, int off, int len)	写一个字符串的一部分

```
    @Test
    public void test3() {
        try (Writer fw = new FileWriter("src/ccc.txt");) {
            fw.write(48);//每次写一个字符
            fw.write('a');
            fw.write('张');
            fw.write("\r\n");//写一个字符串
            fw.write("什么玩意的狗比东西", 5, 4);
            fw.write("\r\n");
            fw.write("无敌的女帝".toCharArray());//写一个字符数组
            fw.write("\r\n");//写一个字符串

            fw.write("无敌的女帝".toCharArray(), 0, 3);//写一个字符数组,有范围

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
0a张
狗比东西
无敌的女帝
无敌的
```