一.File类

-java.io.File类：文件和目录路径名的抽象表示形式。

Java把电脑中的文件和文件夹（目录）封装为了一个File类，我们可以使用File类对文件和文件夹进行操作。我们可以使用File类的方法：

创建一个文件/文件夹

删除文件/文件夹

获取文件/文件夹

判断文件/文件夹是否存在

对文件夹进行遍历

获取文件的大小

File类是一个与系统无关的类，任何的操作系统都可以使用这个类中的方法。

重点：记住这三个单词：

-file：文件

-directory：文件夹/目录

-path：路径

1.File类的静态成员变量



2.路径（path）:

绝对路径：是一个完整的路径

以盘符（c:,D:）开始的路径，比如：

C:\\a.txt

C:\\Users\\itcast\\IndeaProjects\\shuangyuan

相对路径：是一个简化的路径

相对指的是相对于当前项目的根目录(C:\\Users\\itcast\\IndeaProjects\\shuangyuan)

如果使用当前项目的根目录，路径可以简化书写

C:\\Users\\itcast\\IndeaProjects\\shuangyuan\\123.txt----🡪

简化为：123.txt（可以省略项目的根目录）

注意：

1、路径是不区分大小写

2、路径中的文件名称分隔符和windows使用反斜杠，反斜杠是转义字符，两个反斜杠代表一个普通的反斜杠。

3.File的构造方法

-File(String pathname):通过将给定路径名称字符串转换为抽象路径名来创建一个新File实例。

参数：

String pathname:字符串的路径名称

路径可以是以文件结束，也可以是以文件夹结尾

路径可以是相对路径，也可以是绝对路径

路径可以是存在，也可以是不存在

创建File对象，只是把字符串给封装为File对象，不考虑路径的真实情况。

-File(String parent,String child):根据parent路径名字符串和child路径名字符串创建一个新File实例。

参数：把路径分成了两部分

String parent:父路径

String child:子路径

好处：

父路径和子路径，可以单独书写，使用起来非常灵活，父路径和子路径都可以变化。

-File(File parent,String child):根据paren抽象路径名和child路径名字符串创建一个新File实例。

参数：把路径分为了两部分

File parent:父路径

String child：子路径

好处：

父路径和子路径，可以单独书写，使用起来非常灵活，父路径和子路径都可以变化。

父路径是File类型，可以使用File的方法对路径进行一些操作，再使用路径创建对象。



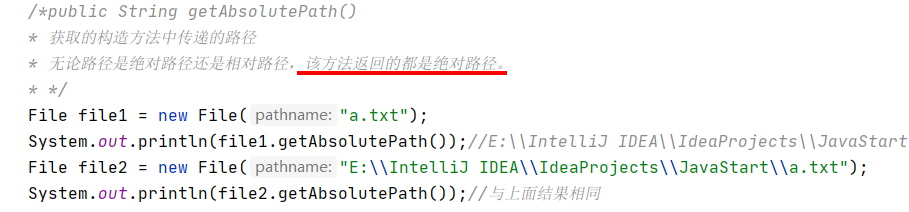
4.File中获取功能的方法

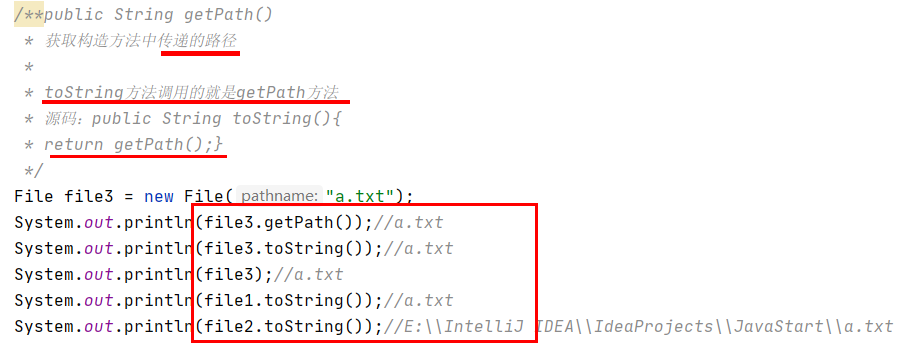
-public String getAbsolutePath();返回此File的绝对路径名字符串。

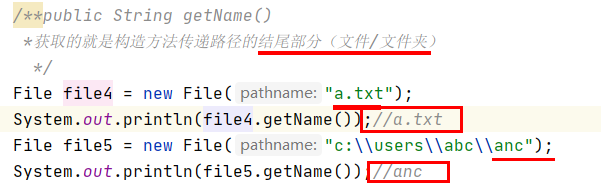
-public String getPath();将此File转换为路径名字符串。

-public String getName();返回此File表示的文件或目录的名称。

-public long length();返回此File表示的文件的长度。









5.File类中的判断功能的方法

-public boolean exists();此File表示的文件或者目录是否实际存在。

-public boolean isDirectory();此File表示的是否为目录。

-public boolean isFile();此Fil表示的是否为文件。



6.创建和删除功能的方法

-public boolean createNewFile();当且仅当具有该名称的文件上不存在时，创建一个新的空文件。

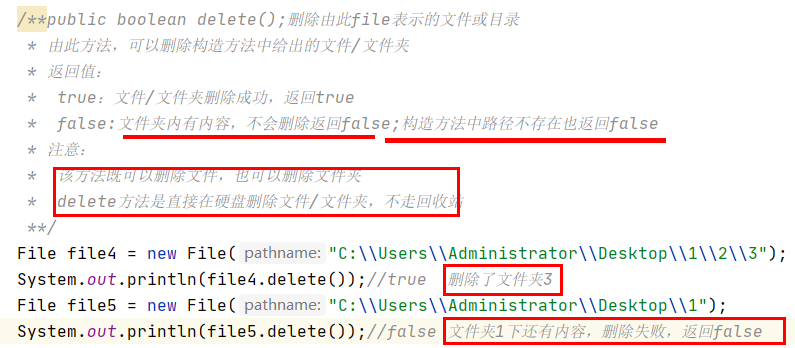
-public boolean delete();删除由此File表示的文件或目录。

-public boolean mkdir();创建由此File表示的目录。

-public boolean mkdirs();创建由此File表示的目录，包括任何必需但不存在的父目录。







7.File类遍历（文件夹）目录功能

-public String[] list();返回一个String数组，表示该File目录下的所有子文件或目录。

-public File[] listFiles();返回一个File数组，表示该File目录中的所有的子文件或目录。

注意：

如果构造方法中给出的目录的路径不存在，会抛出空指针异常。

如果构造方法中给出的路径不是一个目录，也会抛出空指针异常。



二.递归

1.概述：指在当前方法内调用自己的这种现象。

2.分类：

直接递归：自己调用自己。

-main(){

-a();

}

-a(){

-a();

}

间接递归：a方法调用b方法，b方法调用c方法,c方法调用a方法。

-b(){

-c();

}

-c(){

-b();

}

3.注意事项：

-递归一定要有条件限定，保证递归能够停止下来，否则会发生栈内存溢出。

-在递归中虽然有限定条件，但是递归次数不能太多，否则也会发生栈内存溢出。

-构造方法禁止递归。

4.递归的使用前提：

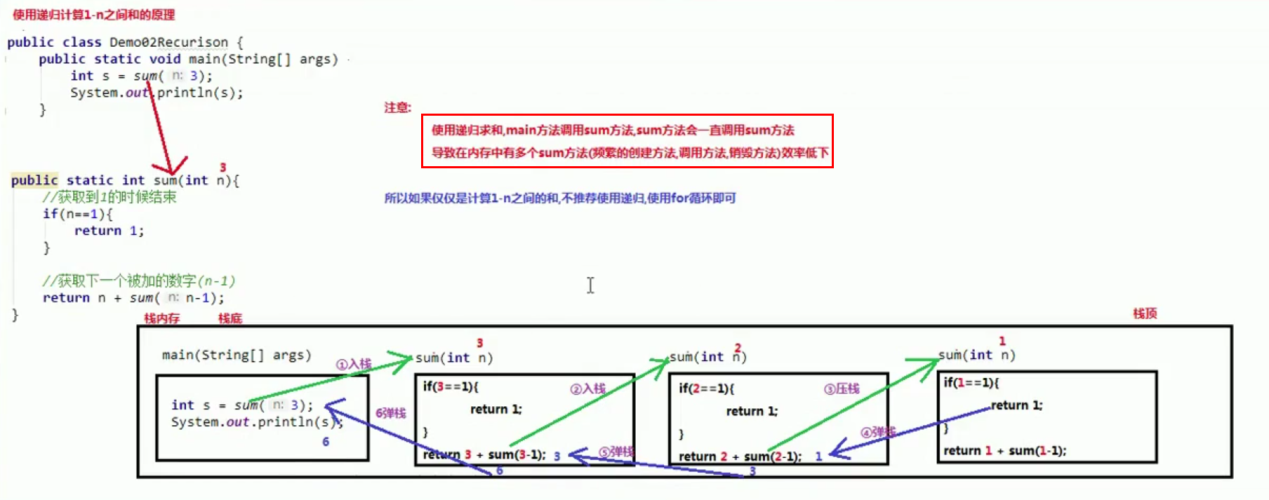
当调用方法的时候，方法的主体不变，每次调用的方法参数不同，可以使用递归。



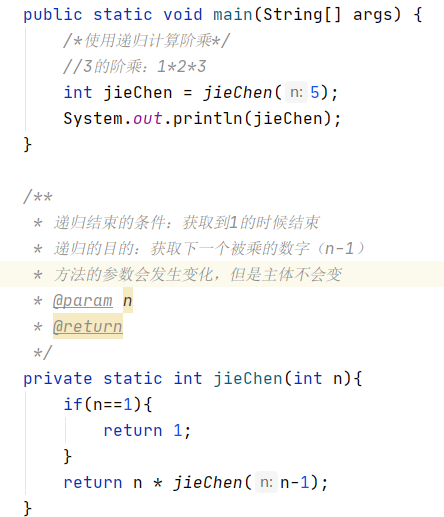




练习一：使用递归求1-n的和

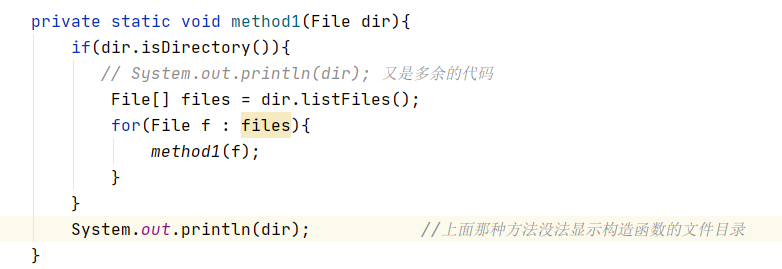


练习二：使用递归计算阶乘

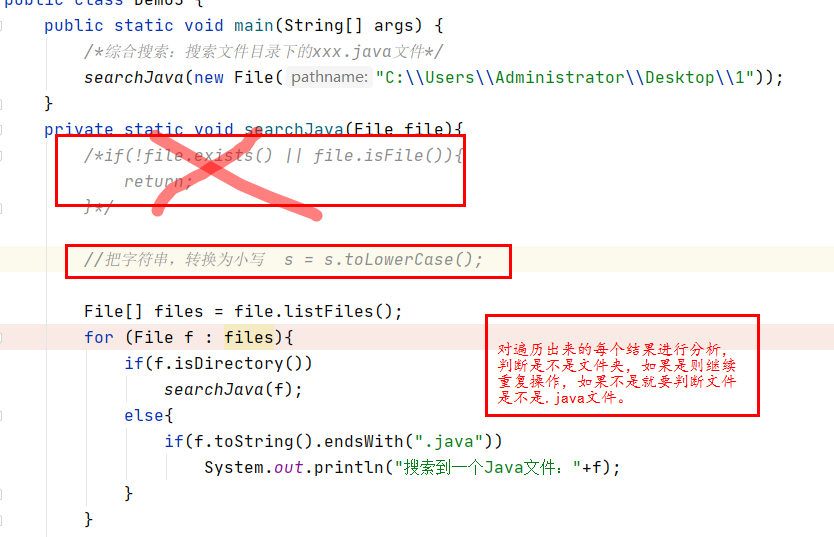


练习三：递归打印多级目录





练习四：综合搜索



过滤器

在File类中有两个和ListFiles重载的方法，方法的参数传递的就是过滤器。

-File[] listFiles(FileFilter filter)

-java.io.FileFilter接口：用于抽象路径名（File对象）的过滤器。

作用：用来过滤文件（File对象）

抽象方法：用来过滤文件的方法

-boolean accept(File pathname):测试指定抽象路径名是否包含在某个路径名列表中。

参数：File pathname:使用listFiles方法遍历目录，得到每一个文件对象。

-File[] listFiles(FilenameFilter filter)

-java.io.FilenameFilter接口：实现此接口的类实例可以用于过滤文件名。

抽象方法：用来过滤文件的方法

-boolean accept(File dir,String name):测试指定文件是否包含在某一文件列表中。

参数：

File dir:构造方法中传递的被遍历的目录

String name:使用ListFiles方法遍历目录，获取的每一个文件/文件夹的名称

注意：两个过滤器接口是没有实现类的，需要我们自己写实现类，重写过滤的方法accept，在方法中自己定义过滤的规则。



