Java进阶1 第5天

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【应用】File类**
2. 【应用】能够独立写出File类的常用构造方法
3. 【理解】能够阐述绝对路径与相对路径的概念
4. 【应用】能够独立测试File类的获取功能相关方法
5. 【应用】能够独立测试File类的创建删除功能相关方法
6. 【应用】能够独立测试File类的判断功能相关方法
7. 【应用】能够独立测试File类的listFiles方法
8. **【应用】递归**
9. 【理解】能够阐述递归的概念
10. 【应用】能够独立写出递归求阶乘案例
11. 【应用】能够独立写出递归打印多级文件夹路径

# File类

## File类的概念

打开API，搜索File类。

阅读其描述：File文件和目录路径名的抽象表示形式。

即，Java中把文件或者目录（文件夹）都封装成File对象。

也就是说如果我们要去操作硬盘上的文件，或者文件夹只要找到File这个类即可。

那么我们就要研究研究File这个类中都有哪些功能可以操作文件或者文件夹呢？

### 案例代码一:

|  |
| --- |
| **package** com.igeek\_01;  **import** java.io.File;  /\*\*  \* **@ClassName**: FileDemo  \* **@Description**: File文件类  \* **@date** 2017年11月20日 上午11:07:47  \* Company www.igeekhome.com  \*  \* File:  \* 文件可以持久化地存储数据  \* File的一个对象就代表一个文件或文件夹(自己简单地定义,方便记忆的说法)  \* 文档上说明File类代表文件或文件夹路径,但是我们可以通过路径找到对应的文件或文件夹.  \* 可以认为File类就代表文件或文件夹(通过路径找到)  \*  \* 构造方法:  \* public File(String pathname) 通过路径找到对应的文件或文件夹  \*/  **public** **class** FileDemo {  **public** **static** **void** main(String[] args) {  //查找文件  File file = **new** File("D:\\other\\test.txt");    //查找文件夹  File dir = **new** File("D:\\other");  }  } |

## 相对路径与绝对路径

绝对路径：从盘符开始到文件的路径

相对路径：从某个参照路径开始到指定文件所经过的路径

### 案例代码二:

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.File;  /\*\*  \* **@ClassName**: AbsoluteRelativeDemo  \* **@Description**: 相对路径与绝对路径  \* **@date** 2017年11月20日 下午1:38:10  \* Company www.igeekhome.com  \*  \* 路径分为绝对路径与相对路径:  \*  \* 绝对路径:从盘符开始到文件的全路径  \* 相对路径:从某个参照路径开始到指定文件所经过的路径  \* 在Eclipse中,所有的文件如果想使用相对路径,则其参照路径均为项目根目录  \* 如test.txt相对于javaadvanced1\_d05\_c04项目根目录经过了a/b/test.txt,则a/b/test.txt就是该文件的相对路径  \*  \* 在java当中对于路径的分隔符可以使用/或者\\,是通用的  \*/  **public** **class** AbsoluteRelativeDemo {  **public** **static** **void** main(String[] args) {  //通过绝对路径在Eclipse中的某个项目当中创建一个文件对象  //"D:/javaadvanced1\_workspace/javaadvanced1\_d05\_c04/a/b/test.txt"  File file = **new** File("D:\\javaadvanced1\_workspace\\javaadvanced1\_d05\_c04\\a\\b\\test.txt");    //通过相对路径在Eclipse中的某个项目当中创建一个文件对象  //"a/b/test.txt"  File file2 = **new** File("a\\b\\test.txt");    //获取文件大小,判断文件是否为test.txt  System.***out***.println(file.length());  System.***out***.println(file2.length());  }  } |

## File类中的方法

### File类中的构造方法

public File(String parent, String child)：parent指的是父级目录，child指的是子级目录

public File(File parent, String child) ：parent为File类型，为了使用File类中的方法

#### 案例代码三:

|  |
| --- |
| **package** com.igeek\_03\_01;  **import** java.io.File;  /\*\*  \* **@ClassName**: FileMethodDemo  \* **@Description**: File类中的构造方法  \* **@date** 2017年11月20日 下午1:51:03  \* Company www.igeekhome.com  \*  \* File类的方法:  \* 构造方法:  \* 1:直接给于路径(绝对路径,相对路径)  \* 2:路径分为父亲(该文件或文件夹所在的文件夹)与孩子(该文件或文件夹自己的名字)  \*  \*/  **public** **class** FileMethodDemo {  **public** **static** **void** main(String[] args) {    //通过父亲与孩子的构造方法创建test.txt对应的文件对象  File file = **new** File("a/b", "test.txt");  System.***out***.println(file.length());    //通过父亲与孩子的构造方法创建a文件夹下的b文件夹对应的文件对象  File dir = **new** File("a", "b");  }  } |

### File类的获取方法

public String getAbsolutePath() 获取绝对路径

public String getPath() 获取路径(用什么方式创建的对象,就返回什么方式的路径(绝对路径/相对路径))

public String getName() 获取文件/文件夹的名称

public String getParent() 返回所在文件夹路径(根据创建对象时是否为绝对路径/相对路径)

#### 案例代码四:

|  |
| --- |
| **package** com.igeek\_03\_02;  **import** java.io.File;  /\*\*  \* **@ClassName**: FileMethodDemo  \* **@Description**:File类的获取方法  \* **@date** 2017年11月20日 下午2:00:42  \* Company www.igeekhome.com  \*  \* File类常见方法  \* 1:获取文件/文件夹信息方法  \* public String getAbsolutePath() 获取绝对路径  \* public String getPath() 获取路径(用什么方式创建的对象,就返回什么方式的路径(绝对路径/相对路径))  \* public String getName() 获取文件/文件夹的名称  \* public String getParent() 返回所在文件夹路径(根据创建对象时是否为绝对路径/相对路径)  \*/  **public** **class** FileMethodDemo {  **public** **static** **void** main(String[] args) {  //创建文件对象  File file = **new** File("a/b/test.txt");  File file2 = **new** File("D:/javaadvanced1\_workspace/javaadvanced1\_d05\_c04/a/b/test.txt");  File dir = **new** File("a/b");    //获取绝对路径  System.***out***.println(file.getAbsolutePath());  System.***out***.println(file2.getAbsolutePath());  System.***out***.println("===========================");    //获取路径  System.***out***.println(file.getPath());  System.***out***.println(file2.getPath());  System.***out***.println("===========================");    //获取文件/文件夹的名称  System.***out***.println(file.getName());  System.***out***.println(dir.getName());  System.***out***.println("===========================");    //返回所在文件夹路径  System.***out***.println(file.getParent());  System.***out***.println(file2.getParent());  System.***out***.println(dir.getParent());  }  } |

### File类的创建与删除方法

public boolean delete() 删除文件

public boolean createNewFile() throws IOException 创建文件

在创建文件时,如果文件所在的文件夹不存在,则报错系统找不到指定的路径.创建文件时,必须确保文件夹已经存在

public boolean mkdir() 使用mkdir方法创建文件夹时,必须保证其所在文件夹已经存在,否则创建失败(不会报错)

public boolean mkdirs() 一次性创建多级目录

public boolean delete() 删除文件夹不能为非空(有东西),否则删除失败

#### 案例代码五:

|  |
| --- |
| **package** com.igeek\_03\_03;  **import** java.io.File;  **import** java.io.IOException;  /\*\*  \* **@ClassName**: FileMethodDemo  \* **@Description**:File类的创建与删除方法  \* **@date** 2017年11月20日 下午2:08:30  \* Company www.igeekhome.com  \*  \* 2:可以创建/删除该文件(Java删除时，不会使用windows的回收站)  \* public boolean delete() 删除文件  \* public boolean createNewFile() throws IOException 创建文件  \* 在创建文件时,如果文件所在的文件夹不存在,则报错系统找不到指定的路径.创建文件时,必须确保文件夹已经存在  \*  \* public boolean mkdir() 使用mkdir方法创建文件夹时,必须保证其所在文件夹已经存在,否则创建失败(不会报错)  \* public boolean mkdirs() 一次性创建多级目录  \*  \* public boolean delete() 删除文件夹不能为非空(有东西),否则删除失败  \*/  **public** **class** FileMethodDemo {  **public** **static** **void** main(String[] args) **throws** IOException {  File file = **new** File("a/b/test.txt");  //删除文件  System.***out***.println(file.getPath()+"的删除结果:"+file.delete());    //创建文件  System.***out***.println(file.getPath()+"的创建结果:"+file.createNewFile());    //在创建文件时,如果文件所在的文件夹不存在,则报错系统找不到指定的路径.创建文件时,必须确保文件夹已经存在  //File file2 = new File("c/d/test.txt");  //System.out.println(file2.createNewFile());  System.***out***.println("==========================");    //创建文件夹  File dir = **new** File("c/d");    //必须保证其所在文件夹已经存在,否则创建失败(不会报错)  System.***out***.println(dir.mkdir());  //一次性创建多级目录  System.***out***.println(dir.mkdirs());    File dir2 = **new** File("e1/e2");  //必须保证其所在文件夹已经存在,否则创建失败(不会报错)  System.***out***.println(dir2.mkdir());  System.***out***.println("==========================");    //删除文件夹  System.***out***.println(dir2.delete());    //删除文件夹不能为非空(有东西),否则删除失败  File dir3 = **new** File("a/b");  System.***out***.println(dir3.delete());  }  } |

### File类的判断方法

public boolean exists() 文件或者文件夹是否存在

public boolean isDirectory() 判断文件对象是否为文件夹

public boolean isFile() 判断文件对象是否为文件

#### 案例代码六:

|  |
| --- |
| **package** com.igeek\_03\_04;  **import** java.io.File;  /\*\*  \* **@ClassName**: FileMethodDemo  \* **@Description**: File类的判断方法  \* **@date** 2017年11月20日 下午2:18:30  \* Company www.igeekhome.com  \*  \* 3:判断该File对象是否存在或者判断该File对象代表一个文件还是代表一个文件夹  \* public boolean exists() 文件或者文件夹是否存在  \* public boolean isDirectory() 判断文件对象是否为文件夹  \* public boolean isFile() 判断文件对象是否为文件  \*/  **public** **class** FileMethodDemo {  **public** **static** **void** main(String[] args) {  File file = **new** File("a/b/test.txt");  File file2 = **new** File("a/b/c.txt");  File dir = **new** File("c/d");  File dir2 = **new** File("e1/e2");    //文件或者文件夹是否存在  System.***out***.println(file.exists());  System.***out***.println(file2.exists());  System.***out***.println(dir.exists());  System.***out***.println(dir2.exists());  System.***out***.println("================================");    // 判断文件对象是否为文件夹/文件  System.***out***.println(file.isDirectory());  System.***out***.println(file.isFile());  System.***out***.println(dir.isDirectory());  System.***out***.println(dir.isFile());  System.***out***.println("=================================");    //如果是不存在的文件或者文件夹,则判断是否为文件或文件夹时一律报false  System.***out***.println(file2.isFile());  System.***out***.println(dir2.isDirectory());  }  } |

### File类的listFiles方法

public File[] listFiles() 获取调用方法文件夹下的所有file对象(文件或文件夹)

#### 案例代码七:

|  |
| --- |
| **package** com.igeek\_03\_05;  **import** java.io.File;  /\*\*  \* **@ClassName**: FileListFilesDemo  \* **@Description**: File类的listFiles方法  \* **@date** 2017年11月20日 下午2:24:32  \* Company www.igeekhome.com  \*  \* public File[] listFiles() 获取调用方法文件夹下的所有file对象(文件或文件夹)  \*/  **public** **class** FileListFilesDemo {  **public** **static** **void** main(String[] args) {  //创建文件对象  File dir = **new** File("./");    //获取该文件夹下所有文件对象  File[] filesDirs = dir.listFiles();    **for** (File fileDir : filesDirs) {  System.***out***.println(fileDir);  }  }  } |

# 递归

## 递归概念

递归，指在当前方法内调用自己的这种现象

public void method(){

System.out.println(“递归的演示”);

//在当前方法内调用自己

method();

}

递归分为两种，直接递归和间接递归。

直接递归称为方法自身调用自己。

间接递归可以A方法调用B方法，B方法调用C方法，C方法调用A方法。

### 案例代码八:

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: RecursionDemo  \* **@Description**: 方法递归基础  \* **@date** 2017年11月20日 下午2:36:44  \* Company www.igeekhome.com  \*  \* 不同的方法之间可以相互调用,即A方法可以将B方法调用,内存当中就多了一个B方法,B方法结束,A继续执行  \*  \*/  **public** **class** RecursionDemo {  /\*\*  \* **@Fields** i : 成员位置定义变量,供不同的方法访问  \*/  **static** **int** *i* = 1;  **public** **static** **void** main(String[] args) {  *methodA*();  }    /\*\*  \* **@Title**: methodA  \* **@Description**: 第一个方法：打印i,i++,再调用其他方法,i++,打印  \*/  **public** **static** **void** methodA() {  System.***out***.println("方法A:"+*i*);  *i*++;  *methodB*();  *i*++;  System.***out***.println("方法A:"+*i*);  }    /\*\*  \* **@Title**: methodB  \* **@Description**: 第二个方法  \*/  **public** **static** **void** methodB() {  System.***out***.println("方法B:"+*i*);  *i*++;  *methodC*();  *i*++;  System.***out***.println("方法B:"+*i*);  }  /\*\*  \* **@Title**: methodC  \* **@Description**: 第三个方法  \*/  **private** **static** **void** methodC() {  System.***out***.println("方法C:"+*i*);  *i*++;  *methodD*();  *i*++;  System.***out***.println("方法C:"+*i*);  }  /\*\*  \* **@Title**: methodD  \* **@Description**: 第四个方法  \*/  **private** **static** **void** methodD() {  System.***out***.println("方法D:"+*i*);  }  } |

### 案例代码九:

|  |
| --- |
| **package** com.igeek\_01;  /\*\*  \* **@ClassName**: RecursionDemo2  \* **@Description**: 方法递归  \* **@date** 2017年11月20日 下午2:41:52  \* Company www.igeekhome.com  \*  \* 递归算法: 方法自身调用方法自身,必须有方法出口(可以结束方法的条件),递归次数不宜过多,会有stackoverflow(栈内存溢出)  \*/  **public** **class** RecursionDemo2 {    /\*\*  \* **@Fields** i : 成员位置定义变量,供不同的方法访问  \*/  **static** **int** *i* = 1;    **public** **static** **void** main(String[] args) {  *method*();  }    /\*\*  \* **@Title**: method  \* **@Description**: 定义了一个自己调用自己的递归方法  \*/  **public** **static** **void** method() {  System.***out***.println("方法开始:"+*i*);    //如果i已经加到4了,就将method方法结束  **if**(*i*==4) {  **return**;  }    *i*++;  *method*();  *i*++;  System.***out***.println("方法结束:"+*i*);  }  } |

## 递归案例

### 递归求阶乘

求N的阶乘,以5的阶乘为例

5\*4\*3\*2\*1

5! = 5 \* 4!;

4! = 4 \* 3!;

3! = 3 \* 2!;

2!= 2 \* 1!;

1! = 1;

#### 案例代码十:

|  |
| --- |
| **package** com.igeek\_02;  /\*\*  \* **@ClassName**: RecursionTest  \* **@Description**: 递归求阶乘  \* **@date** 2017年11月20日 下午3:24:15  \* Company www.igeekhome.com  \*  \* 递归可以将复杂问题简单化  \*  \* 求N的阶乘,以5的阶乘为例  \* 5\*4\*3\*2\*1  \*  \* 5! = 5 \* 4!;  \* 4! = 4 \* 3!;  \* 3! = 3 \* 2!;  \* 2!= 2 \* 1!;  \* 1! = 1; 单独唯一的逻辑  \*  \* 递归逻辑:  \* 当前这个数 \* (比当前这个数-1)的阶乘  \* 递归出口:  \* 如果求到了1的阶乘,就直接返回1,不再求阶乘了  \*/  **public** **class** RecursionTest {  **public** **static** **void** main(String[] args) {  **int** result=*method*(5);  System.***out***.println("result:"+result);  }    /\*\*  \* **@Title**: method  \* **@Description**: 求某个数的阶乘  \* **@param** n  \* **@return**  \*/  **public** **static** **int** method(**int** n) {    //方法出口,如果是求1的阶乘,则直接返回1  **if**(n == 1) {  **return** 1;  }    **return** n \* *method*(n-1);  }  } |

### 递归打印多级文件夹路径

打印一个文件夹内所有的文件路径

#### 案例代码十一:

|  |
| --- |
| **package** com.igeek\_02;  **import** java.io.File;  /\*\*  \* **@ClassName**: RecursionTest2  \* **@Description**: 递归打印多级文件夹路径  \* **@date** 2017年11月20日 下午3:28:24  \* Company www.igeekhome.com  \*  \* 打印一个文件夹内所有的文件夹和文件路径  \*  \* 1:定义方法,接受文件夹,打印该文件夹中所有的文件路径  \*  \* 2:获取该文件夹中的所有文件对象  \* 3:判断该文件对象是文件还是文件夹  \* 如果是文件,打印路径  \* 如果是文件夹,调用方法,打印该文件夹中所有的文件路径  \*/  **public** **class** RecursionTest2 {  **public** **static** **void** main(String[] args) {  File dir = **new** File("./");  *method*(dir);  }    /\*\*  \* **@Title**: method  \* **@Description**: 定义递归打印方法,打印该文件夹中所有的文件路径  \* **@param** dir  \*/  **public** **static** **void** method(File dir) {    //获取该文件夹中的所有文件对象  File[] filesDirs = dir.listFiles();  //依次获取所有的文件对象  **for** (File file : filesDirs) {  //判断该文件对象是文件还是文件夹  **if**(file.isFile()) {  //如果是文件,打印路径  System.***out***.println(file.getAbsolutePath());  }**else** {  //System.out.println(file.getAbsolutePath());  //如果是文件夹,调用方法,打印该文件夹中所有的文件路径  *method*(file);  }  }  }  } |

重点和总结

1、File类的基本方法及使用

2、相对路径和绝对路径

3、方法递归算法的概率及案例