1. File 类

1.1 概述

java.io.File 类是文件和目录路径名的抽象表示，主要用于文件和目录的创建、查找和删除等操作。

1.2 构造方法

- public File(String pathname) ：通过将给定的**路径名字符串**转换为抽象路径名来创建新的 File 实例。
- public File(String parent, String child) ：从**父路径名字符串和子路径名字符串**创建新的 File 实例。
- public File(File parent, String child)：从**父抽象路径名和子路径名字符串**创建新的 File 实例。
- 构造举例，代码如下：

```
// 文件路径名
String pathname = "D:\\aaa.txt";
File file1 = new File(pathname);

// 文件路径名
String pathname2 = "D:\\aaa\\bbb.txt";
File file2 = new File(pathname2);

// 通过父路径和子路径字符串
 String parent = "d:\\aaa";
 String child = "bbb.txt";
 File file3 = new File(parent, child);

// 通过父级 File 对象和子路径字符串
File parentDir = new File("d:\\aaa");
String child = "bbb.txt";
File file4 = new File(parentDir, child);
```

小贴士：

1. 一个 File 对象代表硬盘中实际存在的一个文件或者目录。

2. 无论该路径下是否存在文件或者目录，都不影响 File 对象的创建。

1.3 常用方法

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()：返回由此 File 表示的文件或目录的名称。
- public long length()：返回由此 File 表示的文件的长度。

  方法演示，代码如下：

```
public class FileGet {
    public static void main(String[] args) {
        File f = new File("d:/aaa/bbb.java");
        System.out.println("文件绝对路
径:"+f.getAbsolutePath());
        System.out.println("文件构造路径:"+f.getPath());
        System.out.println("文件名称:"+f.getName());
        System.out.println("文件长度:"+f.length()+"字节");

        File f2 = new File("d:/aaa");
        System.out.println("目录绝对路
径:"+f2.getAbsolutePath());
        System.out.println("目录构造路径:"+f2.getPath());
        System.out.println("目录名称:"+f2.getName());
        System.out.println("目录长度:"+f2.length());
    }
}
```

输出结果：

```
文件绝对路径:d:\aaa\bbb.java
文件构造路径:d:\aaa\bbb.java
文件名称:bbb.java
文件长度:636 字节

目录绝对路径:d:\aaa
目录构造路径:d:\aaa
目录名称:aaa
目录长度:4096
```

API 中说明：length()，表示文件的长度。但是 File 对象表示目录，则返回值未指定。

绝对路径和相对路径

- **绝对路径**：从盘符开始的路径，这是一个完整的路径。
- **相对路径**：相对于项目目录的路径，这是一个便捷的路径，开发中经常使用。

```
public class FilePath {
    public static void main(String[] args) {
        // D盘下的bbb.java文件
        File f = new File("D:\\bbb.java");
        System.out.println(f.getAbsolutePath());

                // 项目下的bbb.java文件
        File f2 = new File("bbb.java");
        System.out.println(f2.getAbsolutePath());
    }
}
```

输出结果：

```
D:\bbb.java
D:\idea_project_test4\bbb.java
```

判断功能的方法

- public boolean exists() ： 此 File 表示的文件或目录是否实际存在。
- public boolean isDirectory() ： 此 File 表示的是否为目录。
- public boolean isFile() ： 此 File 表示的是否为文件。

方法演示，代码如下：

```
public class FileIs {
    public static void main(String[] args) {
        File f = new File("d:\\aaa\\bbb.java");
        File f2 = new File("d:\\aaa");
        // 判断是否存在
        System.out.println("d:\\aaa\\bbb.java 是否存在:"+f.exists());
        System.out.println("d:\\aaa 是否存在:"+f2.exists());
        // 判断是文件还是目录
        System.out.println("d:\\aaa 文件?:"+f2.isFile());
```

```
        System.out.println("d:\\aaa 目录?:"+f2.isDirectory());
    }
}
```

输出结果：

```
d:\aaa\bbb.java 是否存在:true
d:\aaa 是否存在:true
d:\aaa 文件?:false
d:\aaa 目录?:true
```

创建删除功能的方法

- public boolean createNewFile() ：当且仅当具有该名称的文件尚不存在时，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。

方法演示，代码如下：

```
public class FileCreateDelete {
    public static void main(String[] args) throws IOException {
        // 文件的创建
        File f = new File("aaa.txt");
        System.out.println("是否存在:"+f.exists()); // false
        System.out.println("是否创建:"+f.createNewFile()); // true
        System.out.println("是否存在:"+f.exists()); // true

        // 目录的创建
        File f2= new File("newDir");
        System.out.println("是否存在:"+f2.exists());// false
        System.out.println("是否创建:"+f2.mkdir()); // true
        System.out.println("是否存在:"+f2.exists());// true

                // 创建多级目录
        File f3= new File("newDira\\newDirb");
        System.out.println(f3.mkdir());// false
        File f4= new File("newDira\\newDirb");
```

```
        System.out.println(f4.mkdirs());// true

        // 文件的删除
        System.out.println(f.delete());// true

        // 目录的删除
        System.out.println(f2.delete());// true
        System.out.println(f4.delete());// false
    }
}
```

API 中说明：delete 方法，如果此 File 表示目录，则目录必须为空才能删除。

1.4 目录的遍历

- public String[] list()：返回一个 String 数组，表示该 File 目录中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。

```
public class FileFor {
    public static void main(String[] args) {
        File dir = new File("d:\\java_code");

        //获取当前目录下的文件以及文件夹的名称。
                String[] names = dir.list();
                for(String name : names){
                        System.out.println(name);
                }
        //获取当前目录下的文件以及文件夹对象，只要拿到了文件对象，那么就可以获取更多信息
        File[] files = dir.listFiles();
        for (File file : files) {
            System.out.println(file);
        }
    }
}
```

小贴士：

调用 listFiles 方法的 File 对象，表示的必须是实际存在的目录，否则返回 null，无法进行遍历。

1.5 综合练习

练习 1：创建文件夹

在当前模块下的 aaa 文件夹中创建一个 a.txt 文件

代码实现：

```
public class Test1 {
    public static void main(String[] args) throws IOException {
        //需求：在当前模块下的 aaa 文件夹中创建一个 a.txt 文件

        //1.创建 a.txt 的父级路径
        File file = new File("myfile\\aaa");
        //2.创建父级路径
        //如果 aaa 是存在的，那么此时创建失败的。
        //如果 aaa 是不存在的，那么此时创建成功的。
        file.mkdirs();
        //3.拼接父级路径和子级路径
        File src = new File(file,"a.txt");
        boolean b = src.createNewFile();
        if(b){
            System.out.println("创建成功");
        }else{
            System.out.println("创建失败");
        }
    }
}
```

练习 2：查找文件（不考虑子文件夹）

定义一个方法找某一个文件夹中，是否有以 avi 结尾的电影（暂时不需要考虑子文件夹）

代码示例：

```
public class Test2 {
    public static void main(String[] args) {
        /*需求：
            定义一个方法找某一个文件夹中，是否有以 avi 结尾的电影。
                （暂时不需要考虑子文件夹）
        */
```

```
        File file = new File("D:\\aaa\\bbb");
        boolean b = haveAVI(file);
        System.out.println(b);
    }
    /*
    * 作用：用来找某一个文件夹中，是否有以 avi 结尾的电影
    * 形参：要查找的文件夹
    * 返回值：查找的结果  存在 true  不存在 false
    * */
    public static boolean haveAVI(File file){// D:||aaa
        //1.进入 aaa 文件夹，而且要获取里面所有的内容
        File[] files = file.listFiles();
        //2.遍历数组获取里面的每一个元素
        for (File f : files) {
            //f：依次表示 aaa 文件夹里面每一个文件或者文件夹的路径
            if(f.isFile() && f.getName().endsWith(".avi")){
                return true;
            }
        }
        //3.如果循环结束之后还没有找到，直接返回 false
        return false;
    }
}
```

练习 3：（考虑子文件夹）

找到电脑中所有以 avi 结尾的电影。（需要考虑子文件夹）

代码示例：

```
public class Test3 {
    public static void main(String[] args) {
        /* 需求：
        找到电脑中所有以 avi 结尾的电影。（需要考虑子文件夹）


        套路：
            1，进入文件夹
            2，遍历数组
            3，判断
            4，判断
```

```
        */

        findAVI();

    }

    public static void findAVI(){
        //获取本地所有的盘符
        File[] arr = File.listRoots();
        for (File f : arr) {
            findAVI(f);
        }
    }

    public static void findAVI(File src){//"C:||
        //1.进入文件夹 src
        File[] files = src.listFiles();
        //2.遍历数组,依次得到 src 里面每一个文件或者文件夹
        if(files != null){
            for (File file : files) {
                if(file.isFile()){
                    //3，判断，如果是文件，就可以执行题目的业务逻辑
                    String name = file.getName();
                    if(name.endsWith(".avi")){
                        System.out.println(file);
                    }
                }else{
                    //4，判断，如果是文件夹，就可以递归
                    //细节：再次调用本方法的时候，参数一定要是 src 的次一级路径
                    findAVI(file);
                }
            }
        }
    }
}
```

练习 4：删除多级文件夹

需求： 如果我们要删除一个有内容的文件夹 1.先删除文件夹里面所有的内容 2.再删除自己

代码示例：

```java
public class Test4 {
    public static void main(String[] args) {
        /*
            删除一个多级文件夹
            如果我们要删除一个有内容的文件夹
            1.先删除文件夹里面所有的内容
            2.再删除自己
        */

        File file = new File("D:\\aaa\\src");
        delete(file);

    }

    /*
    * 作用：删除 src 文件夹
    * 参数：要删除的文件夹
    * */
    public static void delete(File src){
        //1.先删除文件夹里面所有的内容
        //进入 src
        File[] files = src.listFiles();
        //遍历
        for (File file : files) {
            //判断,如果是文件，删除
            if(file.isFile()){
                file.delete();
            }else {
                //判断,如果是文件夹，就递归
                delete(file);
            }
        }
        //2.再删除自己
        src.delete();
    }
}
```

练习 5：统计大小

需求：统计一个文件夹的总大小

代码示例：

```java
public class Test5 {
    public static void main(String[] args) {
       /*需求：
            统计一个文件夹的总大小
        */


        File file = new File("D:\\aaa\\src");

        long len = getLen(file);
        System.out.println(len);//4919189
    }

    /*
    * 作用：
    *       统计一个文件夹的总大小
    * 参数：
    *       表示要统计的那个文件夹
    * 返回值：
    *       统计之后的结果
    *
    * 文件夹的总大小：
    *       说白了，文件夹里面所有文件的大小
    * */
    public static long getLen(File src){
        //1.定义变量进行累加
        long len = 0;
        //2.进入 src 文件夹
        File[] files = src.listFiles();
        //3.遍历数组
        for (File file : files) {
            //4.判断
            if(file.isFile()){
                //我们就把当前文件的大小累加到 len 当中
                len = len + file.length();
            }else{
                //判断，如果是文件夹就递归
                len = len + getLen(file);
            }
        }
        return len;
    }
```

```
}
```

练习 6：统计文件个数

需求：统计一个文件夹中每种文件的个数并打印。（考虑子文件夹） 打印格式如下：
txt:3 个 doc:4 个 jpg:6 个

代码示例：

```
public class Test6 {
    public static void main(String[] args) throws IOException {
        /*
            需求：统计一个文件夹中每种文件的个数并打印。（考虑子文件夹）
            打印格式如下：
            txt:3 个
            doc:4 个
            jpg:6 个
        */
        File file = new File("D:\\aaa\\src");
        HashMap<String, Integer> hm = getCount(file);
        System.out.println(hm);
    }

    /*
    * 作用：
    *       统计一个文件夹中每种文件的个数
    * 参数：
    *       要统计的那个文件夹
    * 返回值：
    *       用来统计 map 集合
    *       键：后缀名 值：次数
    *
    *       a.txt
    *       a.a.txt
    *       aaa（不需要统计的）
    *
    *
    * */
    public static HashMap<String,Integer> getCount(File src){
        //1.定义集合用来统计
        HashMap<String,Integer> hm = new HashMap<>();
        //2.进入 src 文件夹
```

```
File[] files = src.listFiles();
//3.遍历数组
for (File file : files) {
    //4.判断，如果是文件，统计
    if(file.isFile()){
        //a.txt
        String name = file.getName();
        String[] arr = name.split("\\.");
        if(arr.length >= 2){
            String endName = arr[arr.length - 1];
            if(hm.containsKey(endName)){
                //存在
                int count = hm.get(endName);
                count++;
                hm.put(endName,count);
            }else{
                //不存在
                hm.put(endName,1);
            }
        }
    }else{
        //5.判断，如果是文件夹，递归
        //sonMap 里面是子文件中每一种文件的个数
        HashMap<String, Integer> sonMap = getCount(file);
        //hm:  txt=1  jpg=2  doc=3
        //sonMap: txt=3 jpg=1
        //遍历 sonMap 把里面的值累加到 hm 当中
        Set<Map.Entry<String, Integer>> entries = sonMap.entrySet();
        for (Map.Entry<String, Integer> entry : entries) {
            String key = entry.getKey();
            int value = entry.getValue();
            if(hm.containsKey(key)){
                //存在
                int count = hm.get(key);
                count = count + value;
                hm.put(key,count);
            }else{
                //不存在
                hm.put(key,value);
            }
        }
    }
```

```
            }
            return hm;
        }
    }
```