day08【File类、递归】

--File类

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。

绝对路径：以盘符开始的路径

“D:\aaa\bbb.java”

相对路径：

不以盘符开始的简化路径，IDEA项目

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。

参数：

String pathname：字符换的路径名称

路径可以使以文件结尾，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，只是把字符串路径封装为File对象，不考虑路径的真假情况

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。

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

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。

参数：把路径分成了两部分

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

注意:

一个File对象代表硬盘中实际存在的一个文件或者目录

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

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。

获取的构造方法中传递的路径，无论路径是绝对的还是相对的，此方法返回的都是绝对路径

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。

获取的就是构造方法传递路径的结尾部分

public long length() : 返回由此File表示的文件的长度。

获取的是构造方法指定的文件大小，以字节为单位

注意: 文件夹是没有大小概念的，不能获取文件夹的大小

如果构造方法中给出的路径不存在，那么length方法返回0

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。

用于判断构造方法中的路径是否存在

存在: true

不存在: false

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

用于判断构造方法中给定的路径是否以文件夹结尾

是: true

否: false

public boolean isFile() : 此File表示的是否为文件。

用于判断构造方法中给定的路径是否以文件结尾

是: true

否: false

注意: isDirectory()和isFile()使用前提, 路径必须是存在的, 否则都返回false, 两个方法是互斥的, 一个为true, 另一个必为false

```
package com.itheima.test02;

import java.io.File;

public class IsFile {
	public static void main(String[] args) {
		File f1 = new File("D:\\develop\\IdeaProjects\\JavaSE_2\\day08_code\\1.txt");
		File f2 = new File("D:\\develop\\IdeaProjects\\JavaSE_2\\day08_");
		System.out.println("exists():" + f1.exists());//exists():true
		System.out.println("exists():" + f2.exists());//exists():false

		if (f1.exists()) {
			System.out.println("exists():" + f1.exists());//exists():true
			System.out.println("isDirectory():" + f1.isDirectory());//isDirectory():false
```

```
                System.out.println("isFile():" +
f1.isFile());//isFile():true
            }
            if (f2.exists()) {
                System.out.println("isDirectory():" +
f2.isDirectory());//文件路径不存在，不打印任何值
                System.out.println("isFile():" +
f2.isFile());//文件路径不存在，不打印任何值
            }
        }
    }
```

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。

目录的遍历:

```
package com.itheima.test;

import java.io.File;
import java.lang.reflect.Field;

public class ListFile {
    public static void main(String[] args) {
        File file = new
File("D:\\develop\\IdeaProjects\\JavaSE_2\\day08_code");
        //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。
        String[] strings = file.list();
```

```
                for (String string : strings) {
                    System.out.println(string);
                /* 1.txt
                    day08_code.iml
                    day08【File类、递归】.java
                    src*/
                }
                //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。
                File[] files = file.listFiles();
                for (File f : files) {
                    System.out.println(f);

/*D:\develop\IdeaProjects\JavaSE_2\day08_code\1.txt

D:\develop\IdeaProjects\JavaSE_2\day08_code\day08_code.iml

D:\develop\IdeaProjects\JavaSE_2\day08_code\day08【File类、递归】.java

D:\develop\IdeaProjects\JavaSE_2\day08_code\src*/

                }
            }
        }
```

--递归

指在当前方法内调用自己的这种现象。

递归的分类:

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。

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

递归的使用前提:

当调用方法的时候，方法的主体不变，每次调用方法的参数不同，可以使用递归

注意：当一个方法调用其他方法的时候，被调用的方法没有执行完毕，当前方法会一直等待调用的方法执行完毕，才会继续执行

递归累加求和:

使用递归求和，main方法调用sum方法，sum方法会一直调用sum方法

导致在内存中有多个sum方法（频繁的创建方法，调用方法，销毁方法，效率低下）

所以如果仅仅计算1--n之间的和，不推荐使用递归，使用for循环即可

```
package com.itheima.test03;

//num的累和 = num + (num-1)的累和，所以可以把累和的操作定义成一个方法，递归调用。
//原理：压栈、出栈
public class DiGuiSum {
	public static void main(String[] args) {
		System.out.println(sum(5));
	}

	private static int sum(int num) {
		//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。

```
                if (num == 1) {
                        return 1;
                }
                int result = num + sum(num - 1);
                System.out.println(result);
                return result;
        }
}
```

递归求阶乘:

```
package com.itheima.test04;

public class DiGuiJC {
        public static void main(String[] args) {
                System.out.println(jC(5));
        }

        private static int jC(int n) {
                //设置递归的限制结束条件，否则会出现占内存溢出的异常
                if (n == 1) {
                        return 1;
                }
                int result = n * jC(n - 1);
                return result;
        }
}
```

递归打印多级目录:

```
package com.itheima.test05;

import java.io.File;
```

```
public class DiGuiDir {
	public static void main(String[] args) {
		//创建文件路径
		File dir = new
File("D:\\develop\\IdeaProjects\\JavaSE_2\\day08_code");
		printDir(dir);
	}

	private static void printDir(File dir) {
		//获取目录的子文件夹和目录
		File[] files = dir.listFiles();
		for (File file : files) {
			//System.out.println(file);
			//判断是否是文件夹
			if (file.isDirectory()){
				printDir(file);
			}else {
				System.out.println(file);
			}
		}
	}
}
```

文件过滤器优化：

boolean accept(File pathname) ：测试pathname是否应该包含在当前File目录中，符合则返回true。

接口作为参数，需要传递子类对象，重写其中方法。选择匿名内部类方式。

accept 方法，参数为File，表示当前File下所有的子文件和子目录。保留住则返回true，过滤掉则返回false

```
package com.itheima.test06;
```

```
import java.io.File;
import java.io.FileFilter;

public class FileFilterTest {
        public static void main(String[] args) {
                //创建文件路径
                File file = new
File("D:\\develop\\IdeaProjects\\JavaSE_2\\day08_code");
                getAllDir(file);
        }

        private static void getAllDir(File file) {
                //获取子文件或者子文件夹
                File[] files = file.listFiles(new FileFilter() {
                        @Override
                        public boolean accept(File pathname) {
                                if (pathname.isDirectory()) {
                                        return true;
                                }
                                /*if (pathname.getName().endsWith(".java")){
                                        System.out.println(pathname);
                                }
                                return true;*/
                                return pathname.getName().endsWith(".java");
                        }
                });
                //遍历文件数组
                for (File f : files) {
                        if (f.isDirectory()) {
                                getAllDir(f);
                        } else {
```

```
                    System.out.println(f);
                }
            }
        }
    }
```