【济南中心】JAVA 编程阶梯：基础篇之第二十三章

- **File 类递归练习（统计该文件夹大小）**

从键盘接收一个文件夹路径,统计该文件夹大小

```
/**
 * 从键盘接收一个文件夹路径,统计该文件夹大小
 *
 * @author Somnus
 *
 */
public class Demo {

	public static void main(String[] args) {
		System.out.println("输入需要查看文件大小的路径");
		Scanner sc = new Scanner(System.in);
		String file = sc.nextLine();
		sc.close();
		long folderSize = getFolderSize(new File(file));
		System.out.println("文件大小：" + folderSize + "字节");
	}

	/**
	 * 获取文件夹大小
	 *
	 * @param file
	 *                  File 实例
	 * @return long
	 */
	public static long getFolderSize(File file) {
		long size = 0;
		try {
			File[] fileList = file.listFiles();
			for (int i = 0; i < fileList.length; i++) {
				if (fileList.isDirectory()) {
					size = size +
getFolderSize(fileList);

				} else {
					size = size +
fileList.length();

				}
```

版权所有 翻印必究

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

- **File 类递归练习(删除该文件夹)**

从键盘接收一个文件夹路径,删除该文件夹

```
/**
 * 删除一个带有内容的目录
 *
 * @author Somnus
 *
 */
public class Demo {

    public static void main(String[] args) {
        /*
         * 思路：1，删除一个带有内容的目录原理；必须从里往外删。 2，到底有多级目录不确定，递归。
         */
        System.out.println("输入需要删除在文件目录");
        Scanner sc = new Scanner(System.in);
        String file = sc.nextLine();
        sc.close();
        File dir = new File(file);
        removeDir(dir);
    }

    /**
     * 删除一个目录。
     */
    public static void removeDir(File dir) {

        // 1,列出当前目录下的文件以及文件夹File[]
        File[] files = dir.listFiles();// 如果目录是系统级文件夹，java没有访问权限，那么会返回null数组。最好加入判断。
        if (files != null) {
            for (File file : files) {
                // 2,对遍历到的file对象判断是否是目录。
```

```
                                                        if (file.isDirectory()) {
                                                                removeDir(fi
le);
                                                        } else {
                                                                System.out.p
rintln(file + ":" + file.delete());// 删除文件。用打印语句验证是否删除成功，是否出现了误删操
作。
                                                        }
                                                }
                                        }
                                        System.out.println(dir + ":" + dir.delete());
                }
}
```

- **File 类递归练习**

从键盘接收两个文件夹路径,把其中一个文件夹中(包含内容)拷贝到另一个文件夹中

```
/**
 * 从键盘接收两个文件夹路径,把其中一个文件夹中(包含内容)拷贝到另一个文件夹中
 *
 * 复制文件夹,首先需要想到递归.因为文件夹是一个层次目录. 1.需要判断是文件夹还是文件.是文件
夹就递归,是文件就用流读写.
 * 2.文件夹的命名和文件的命名两者都需要在旧文件名称和新文件路径命名
 * .我们需要知道,当旧文件夹通过 listFiles 获得下一级文件时候,这时候,我们就要想到, 新文件夹和
旧文件夹的路径已经相差一级了.
 *
 *
 * */
public class Demo {
        private static Scanner sc;

        public static void main(String[] args) throws IOException {
                sc = new Scanner(System.in);
                System.out.println("请输入一个源文件夹路径");
                String str_oldFile = sc.nextLine();
                System.out.println("请输入一个目的文件夹路径");
                String str_newFile = sc.nextLine();
                sc.close();
                File oldFile = new File(str_oldFile);
                File newFile = new File(str_newFile);
                MyCopy(oldFile, newFile);
        }
```

```
public static void MyCopy(File oldFile, File newFile) throws IOException {
        if (!oldFile.exists()) // 判断源文件是不是存在,如果不存在,声明.
        {
                System.out.println("文件不存在");
        }
        // 新的目录,加上旧的文件名.就成了一个新的文件.但是此时还没有具体文件.
        File f = new File(newFile.getPath() + "\\" + oldFile.getName());
        System.out.println("f:" + f);
        f.mkdir(); // 创建新的文件夹 &&&重要的一部分.这里是建立一个具体的文件夹,此时新旧文件夹都相同了.
        File[] files = oldFile.listFiles(); // 列出旧文件夹里的所有文件.
        for (File file : files) // 遍历这个文件夹里面的文件..
        {
                if (file.isDirectory()) // 如果还是文件夹,就递归.
                {
                        MyCopy(file, f);
                } else // 否则就可以,用 IO 流来读写文件了.
                {
                        BufferedInputStream bis = new BufferedInputStream(
    new FileInputStream(file));
                        BufferedOutputStream bos = new BufferedOutputStream(
    new FileOutputStream(f.getPath() + "\\"
                                + file.getName()));
                        int b = 0;
                        while ((b = bis.read()) != -1) {
                                bos.write(b);
                        }
                        bis.close();
                        bos.close();
                        System.out.println(file.getName() + "  拷贝完成!");
                }
```

```
                }
        }
}
```

- **File 类递归练习(按层级打印)**

从键盘接收一个文件夹路径,把文件夹中的所有文件以及文件夹的名字按层级打印,

```
/**
 * 从键盘接受一个字符串,这个字符串是一个文件目录. 将该目录中所有的文件打印在控制台上.
 * 思路: 1.从键盘录入,得到一个字符串,
 * 2.通过该字符串,创建一个文件夹,
 * 3.将该文件夹进行遍历,如果是一个文件夹,那么就进行递归, 否则,就将文件名称打印在控制台.
 *
 * */
public class Demo {
        public static void main(String[] args) {
                SystemInFileName();
        }

        public static void SystemInFileName() {
                Scanner sc = new Scanner(System.in);
                while (true) {
                        System.out.println("输入文件夹路径，注意不能是盘符根目录");
                        String str = sc.nextLine();
                        File fi = new File(str);
                        if (!fi.exists() && !fi.isDirectory()) {
                                System.out.println("请输入正确的路径文件夹路径");
                        } else {
                                System.out.println("路径：" + str);
                                ListFile(fi);
                        }
                }
        }

        public static void ListFile(File fi) {

                File[] ff = fi.listFiles();
                for (File f : ff) {
                        if (f.isDirectory()) {
```

```
                                                System.out.println("文件夹:"
+ f.getName());
                                                ListFile(f);
                                        } else {
                                                System.out.println(f.getName
());
                                        }
                                }
                }
}
```

递归练习(斐波那契数列)

```
/**
 * 假设一对刚出生的小兔一个月后就能长成大兔，再过一个月就能生下一对小兔，并且此后每个月都生
一对小兔，一年内没有发生死亡
 * ， 问：一对刚出生的兔子，一年内繁殖成多少对兔子? 1 1 2 3 5 8 13
 *
 * @author Somnus
 *
 */
public class Demo {

        private static int getFibo(int i) {
                if (i == 1 || i == 2)
                        return 1;
                else
                        return getFibo(i - 1) + getFibo(i - 2);
        }

        public static void main(String[] args) {
                System.out.println("一年内繁殖的兔子数：");
                for (int j = 1; j <= 12; j++) {
                        System.out.print(getFibo(j) + "\t");
                        if (j % 5 == 0)
                                System.out.println();
                }
        }
}
```

- **递归练习(1000 的阶乘所有零和尾部零的个数)**

需求:求出 1000 的阶乘所有零和尾部零的个数,不用递归做

```
/**
```

```
 * 求出1000的阶乘所有零和尾部零的个数,不用递归做 因为1000的阶乘远远超出了int的取值范围所以要用到大整数
 *
 * @author Somnus
 *
 */
public class Demo {

	public static void main(String[] args) {
		demo1();// 求1000的阶乘中所有的零
		demo2();// 获取1000的阶乘尾部有多少个零
	}

	public static void demo1() {
		BigInteger bigger1 = new BigInteger("1");
		for (int i = 1; i <= 1000; i++) {
			BigInteger bigger2 = new BigInteger(i + "");
			bigger1 = bigger1.multiply(bigger2); // 将bigger1与bigger2相乘的结果赋值给bigger1
		}
		String str = bigger1.toString(); // 获取字符串表现形式
		int count = 0;
		for (int i = 0; i < str.length(); i++) {
			if ('0' == str.charAt(i)) { // 如果字符串中出现了0字符
				count++; // 计数器加1
			}
		}
		System.out.println(count);
	}

	public static void demo2() {
		BigInteger bigger1 = new BigInteger("1");
		for (int i = 1; i <= 1000; i++) {
			BigInteger bigger2 = new BigInteger(i + "");
			bigger1 = bigger1.multiply(bigger2);
		}
		String str = bigger1.toString();
		StringBuilder sb = new StringBuilder(str);
		str = sb.reverse().toString();

		int count = 0; // 定义计数器
		for (int i = 0; i < str.length(); i++) {
			if ('0' != str.charAt(i)) {
```

```
                                        break;
                                } else {
                                        count++;
                                }
                        }

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

- **递归练习(1000 的阶乘尾部零的个数)**

需求:求出 1000 的阶乘尾部零的个数,用递归做

```
/**
 * 求出 1000 的阶乘尾部零的个数,用递归做
 *
 * @author Somnus
 *
 */
public class Demo {

        public static void main(String[] args) {
                System.out.println(getLastCount(getSum(1000)));
        }

        public static BigInteger getSum(int n) {
                if (n == 1) {
                        return new BigInteger("1");
                } else {

                        BigInteger sum = new BigInteger(n + "");
                        return sum.multiply(getSum(n - 1));
                }
        }

        public static int getLastCount(BigInteger sum) {
                int count = 0;
                String s = sum.toString();
                StringBuffer sb = new StringBuffer(s);
                String s1 = sb.reverse().toString();
                for (int i = 0; i < s1.length(); i++) {
                        if ('0' == s1.charAt(i)) {
                                count++;
                        } else {
```

```
                                        break;
                                }
                        }
                        return count;
                }
}
```

- **集合练习(约瑟夫环)**

有 100 个人围成一个圈，从 1 开始报数，报到 14 的这个人就要退出。然后其他人重新开始，从 1 报数，到 14 退出。问：最后剩下的是 100 人中的第几个人？

```
/**
 * 分析： 有 100 个人围成一个圈，从 1 开始报数，报到 14 的这个人就要退出 然后其他人重新开始，从 1 报数，到 14 退出
 *
 * 思路:1.定义一个计数器每计数到 14 就删除掉报数的那个角标所对应的值,并把计数器值恢复初始值
 * 2.接着还是在被删除的角标位置开始循环，为了还是在该角标循环那么可以将现在循环到的角标-1 处理，它还会在下次循环中+1，那么下次的
 * 循环也就是在当前循环角标位置开始
 * 3.因为是围成一个圈报数，所以当循环到末尾时又要从 0 角标开始，那么当报数到最后一个角标时，先处理完有可能的删除动作然后判断
 * 当前角标是否为最后一个角标，是的话赋值为-1，它还会在下次循环中+1，那么下次的 循环也就是 0 角标角标位置又开始循环
 * 4.最后还要判断集合的长度是否为 1，因为要留最后一个人，这个人不能删，只要满足条件则结束循环并打印集合中仅有的一个元素
 * 那么该元素也就是 100 人当中的最后一个人
 */
public class Demo {

        public static void main(String[] args) {
                // 集合里只能存储 Integer 类型数据
                List<Integer> al = new ArrayList<Integer>();
                // 通过循环添加 100 个元素
                for (int i = 1; i <= 100; i++) {
                        al.add(i);
                }

                // 定义一个计数器，这里赋值为 1 是为了记录 0 角标从 1 开始报数
                int num = 1;
                for (int y = 0; y < al.size(); y++) {
                        // 当计数器满足了 14，即该角标就是报数为 14 的那个角标，然后删除该角标所对应的元素
                        if (num == 14) {
```

```
                    al.remove(y);
                    // 删除角标后计数器恢复到初始值
                    num = 1;
                    // 下一次内循环从当前角标开始
                    y = y - 1;
                } else {
                    // 如果没有报数到 14 则加 1 让下一个元素接着报数
                    num++;
                }
                // 有剩下的人还是围成一个圈报数，报到最后一个人则下一个报数的应该是第一个人
                // 那么报到最后一个人后，应该把循环角标再次调为 0，这里赋值-1 在上边会进行+1 处理
                if (y == (al.size() - 1))
                    y = -1;
                // 如果参与报数的人剩下一个了那么就不用再去参与循环报数了，他就是最后剩下的那个人
                if (al.size() == 1)
                    break;
            }
            System.out.println("最后剩下人的是 100 人中的第" + al.get(0) + "个人");
        }
}
```

识别二维码 关注黑马程序员视频库
免费获得更多 IT 资源

黑马程序员