递归

2018年7月1日 14:58

递归：在方法中调用方法本身的现象

```
public void show(){
        show();
}
```

注意：

1.递归一定要有出口，否则就是死循环

2.递归的次数不能太多，否则会内存溢出

3.构造方法不能递归使用

解决问题的思想；

分解法

合并法

递归计算阶乘

2018年7月21日 13:21

```
/*
 * 求5的阶乘
 * 1.作递归要写一个方法
 * 2.出口条件
 * 3.规律
 */
public class DiGuiDemo {
	public static void main(String[] args) {
		int jc = 1;
		for (int x = 2; x <= 5; x++) {
			jc *= x;
		}
		System.out.println("5的阶乘为" + jc);

		System.out.println("5的阶乘为" + jieCheng(5));
	}

	public static int jieCheng(int n) {
		if (n == 1) {
			return 1;
		} else {
			return n * jieCheng(n - 1);
		}
	}
}
```

斐波那契数列

2018年7月21日 14:52

```
/*
 * 斐波那契数列
 * 方法:
 *                 1.数组实现
 *                 2.变量的变化实现
 *                 3.递归实现
 */
public class DiGuiTest {
        public static void main(String[] args) {
                int[] arr = new int[20];
                arr[0] = 1;
                arr[1] = 1;
                for (int x = 2; x < arr.length; x++) {
                        arr[x] = arr[x - 1] + arr[x - 2];
                }
                System.out.println(arr[19]);

                int a = 1;
                int b = 1;
                for (int x = 0; x < 18; x++) {
                        int temp = a;
                        a = b;
                        b = temp + b;
                }
                System.out.println(b);

                System.out.println(fib(20));
        }

        public static int fib(int n) {
                if (n == 1 || n == 2) {
                        return 1;
                } else {
                        return fib(n - 1) + fib(n - 2);
                }
        }
}
```

扫描文件

2018年7月21日 15:02

```
import java.io.File;

/*
 * 把桌面上文件夹中所有的java文件的绝对路径
 */
public class FilePathDemo {
	public static void main(String[] args) {
		File scrFolder = new File("C:\\Users\\TJtulong\\Desktop");

		//递归实现
		getAllFilePaths(scrFolder);
	}

	private static void getAllFilePaths(File scrFolder) {
		File[] fileArray =scrFolder.listFiles();

		for(File file:fileArray) {
			if(file.isDirectory()) {
				getAllFilePaths(file);
			}else {
				if(file.getName().endsWith(".java")) {
					System.out.println(file.getAbsolutePath());
				}
			}
		}
	}
}
```

IO流

2018年7月21日 15:02

IO流：用来进行设备间的数据传输问题

java中IO流的分类：

流向：输入流：读取数据
　　输出流：写出数据

数据类型：字节流
　　字符流：为了方便操作文本数据
如果操作的是文本数据，就用字符流（用记事本打来不乱码）
否则就用字节流（全能）

IO流构造方法

2018年7月21日 22:06

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * 如果没有明确说明按照哪种分类来说，默认按照数据类型分类，不是按流向
 * 超累（抽象类）
 * 字节输入流：InputStream
 * 字节输出流: OutPutStream JDK1.0
 * 字符输入流: Reader
 * 字符输出流: Writer
 *
 * 需求：向一个文本文件中输入"hello IO"
 * OutputStream是抽象类，不能实例化，需要找一个具体的子类FileOutputStream
 * 每种基类的子类都是以父类名作为后缀名
 *
 * 构造方法:
 *                  FileOutputStream(File file)
 *                  FileOutputStream(String name)
 */
public class FileOutputStreamDemo {
        public static void main(String[] args) throws IOException {
                //创建字节输出流对象(两种方式)
                //File file = new File("file.txt");
                //FileOutputStream fos = new FileOutputStream(file);

                FileOutputStream fos = new FileOutputStream("fos.txt");
                /*
                 * 创建字节输出流做了以下事情:
                 * 1.调用系统功能创建文件
                 * 2.创建fos对象
                 * 3.把fos对象指向文件
                 */

                //写数据
```

```
            fos.write("Hello,IO".getBytes());

            //释放资源
            fos.close();
            /*
             * 为什么一定要close
             * 1.让流对象变成垃圾，可以被垃圾回收器回收
             * 2.通知系统去释放该文件相关的资源
             */
            //fos.write("java".getBytes());报错
        }
}
```

IO的write方法

2018年7月21日 22:06

```
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * 字节流操作步骤:
 * 1.创建字节输出流对象
 * 2.调用write()方法
 * 3.释放资源
 *
 * public void write(int b):写一个字节
 * public void write(byte[] b):写一个字节数组
 * public void write(byte[] b, int off,int len)
 */
public class FileOutputStreamDemo2 {
	public static void main(String[] args) throws IOException {
		// 创建字节输出对象
		FileOutputStream fos = new FileOutputStream("fos2.txt");

		// 调用write()方法
		fos.write(97);// a -----底层是二进制数据---通过记事本打开---找97对应的字符值
		fos.write(57);// 9
		fos.write(55);// 7

		byte[] bys = { 97, 98, 99, 100, 101 };// abcde
		fos.write(bys);

		fos.write(bys, 0, 3);// abc

	}
}
```

IO的write方法

2018年7月21日 22:18

```
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * 如何实现数据的换行
 *                  写入换行符号\n
 *                  通过window记事本打开不可以
 *                  不同的系统针对不同的换行符号识别是不同的
 *                  windows：\r\n
 *                  linus:\r
 *                  Mac:\n
 *                  高级记事本软件可以识别任何换行符的
 *
 * 如何实现数据的追加写入：
 * 构造---->第二个方法参数为true
 */
public class FileOutputStreamDemo3 {
        public static void main(String[] args) throws IOException {
                //FileOutputStream fos = new FileOutputStream("fos3.txt");
                FileOutputStream fos = new FileOutputStream("fos3.txt",true);

                for (int x = 0; x < 10; x++) {
                        fos.write(("hello" + x + "\r\n").getBytes());// windows
                }

                fos.close();
        }
}
```

异常处理

2018年7月21日 22:44

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * 加入异常处理的字节输出流操作
 */
public class FileOutputStreamDemo4 {
	public static void main(String[] args) {
		/*
		 * FileOutputStream fos=null; try { fos = new FileOutputStream("fos3.txt"); }
		 * catch (FileNotFoundException e) { // TODO Auto-generated catch block
		 * e.printStackTrace(); }
		 *
		 * try { fos.write("java".getBytes()); } catch (IOException e) { // TODO
		 * Auto-generated catch block e.printStackTrace(); }
		 *
		 * try { fos.close(); } catch (IOException e) { // TODO Auto-generated catch
		 * block e.printStackTrace(); }
		 */

		// 分开做异常处理
		/*
		 * try { FileOutputStream fos = new FileOutputStream("fos3.txt");
		 * fos.write("java".getBytes()); fos.close(); } catch (FileNotFoundException e)
		 * { e.printStackTrace(); }catch(IOException e) { e.printStackTrace(); }
		 */

		// 改进：为了保证close()一定执行，放入finally
		FileOutputStream fos = null; // 为了让finally中有fos
		try {
			fos = new FileOutputStream("fos3.txt");
			fos.write("java".getBytes());
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		} catch (IOException e) {
```

```
            e.printStackTrace();
        } finally {
            // 如果fos不是null，才需要close()
            if (fos != null) {
                try {
                    fos.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }
}
```

读取数据

2018年7月21日 23:00

```
import java.io.FileInputStream;
import java.io.IOException;

/*
 * 字节输入流操作步骤:
 * 1.创建字节输入流对象
 * 2.调用read()方法获取数据
 * 3.close()
 *
 * 读取数据的方式:
 * 1.int read():一次读取一个字节
 *                 如果返回值为-1，则到达文件末尾
 * 2.int read(byte[] b):一次读取一个字节数组
 */
public class FileInputStreamDemo {
	public static void main(String[] args) throws IOException {
		FileInputStream fis = new FileInputStream("fos.txt");

		/*int by = fis.read();
		System.out.print((char) by);// H

		by = fis.read();
		System.out.print((char) by);// e

		// 用循环改进
		int by1 = fis.read();
		while (by1 != -1) {
			System.out.print((char) by1);
			by1 = fis.read();
		}*/

		//标准代码(无法读中文)
		int by = 0;
		while((by=fis.read())!=-1) {
			System.out.print((char) by);
		}
```

```
            fis.close();
        }
    }
```

复制文本文件

2018年7月21日 23:40

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * 复制文本文件
 *
 * 数据源：从哪里来
 * a.txt
 * 目的地：到哪里来
 * b.txt
 *
 * 复制中文没有问题
 * 在控制台输出一个字节转换为字符的结果
 * 通过IO流读取数据，写到文本文件，没有做任何转换
 * 在计算机中中文的存储分两个字节
 *                          第一个字节必为负数
 *                          第二个字节常见为负数，部分为正数，没有影响
 */
public class CopyFileDemo {
        public static void main(String[] args) throws IOException {
                FileInputStream fis= new FileInputStream("fos3.txt");
                FileOutputStream fos = new FileOutputStream("b.txt");

                int by = 0;
                while((by=fis.read())!=-1) {
                        fos.write(by);
                }

                fis.close();
                fos.close();
        }
}
```

复制图片

2018年7月22日 11:02

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * 拷贝图片
 */
public class CopyImageDemo {
	public static void main(String[] args) throws IOException {
		FileInputStream fis = new FileInputStream
				("C:\\Users\\TJtulong\\Desktop\\网络信息.png");
		FileOutputStream fos = new FileOutputStream("mn.png");

		int by =0;
		while((by=fis.read())!=-1) {
			fos.write(by);
		}

		fos.close();
		fis.close();
	}
}
```

读取字符数组

2018年7月22日 11:02

效率高

```
import java.io.FileInputStream;
import java.io.IOException;

/*
 * 一次读取一个字符数组：int read(byte[] b)
 * int为实际读取的字节个数
 * 如果读取到-1，说明没有数据了
 * 字节数组默认值为0
 */
public class FileInputStreamDemo2 {
	public static void main(String[] args) throws IOException {
		FileInputStream fis = new FileInputStream("fos3.txt");

		// 数组的长度一般是1024或1024的倍数（1k数据）
		byte[] bys = new byte[1024];
		int len = 0;
		while ((len = fis.read(bys)) != -1) {
			// System.out.println(len);
			System.out.print(new String(bys, 0, len));
		}

		fis.close();
	}
}
```

复制案例

2018年7月22日 11:35

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * 复制图片
 */
public class CopyFileDemo {
	public static void main(String[] args) throws IOException {
		FileInputStream fis = new FileInputStream("C:\\Users\\TJtulong\\Desktop
		\\网络信息.png");

		FileOutputStream fos = new FileOutputStream("dd.png");

		byte[] bys = new byte[1024];
		int len = 0;
		while ((len = fis.read(bys)) != -1) {
			fos.write(bys, 0, len);
		}

		fis.close();
		fos.close();
	}
}
```

BufferedOutputStream

2018年7月22日 12:22

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;

/*
 * Java在设计时专门提供了带缓冲区的字节类
 * 这种类被称为缓冲区类（高效类）
 * 读数据：BufferedIutputStream
 * 写数据：BufferedOutputStream
 *
 * 构造方法可以指定缓冲区的大小，但我们一般使用默认缓冲区大小
 * 为什么不传递一个具体文件，而是OutputStream对象
 * 原因：字节缓冲区仅仅提供缓冲区，为高效而设计，真正的操作还是需要基本的流对象实现
 */
public class BufferedOutputStreamDemo {
	public static void main(String[] args) throws IOException{
		//BufferedOutputStream(OutputStream out)
		/*FileOutputStream fos = new FileOutputStream("bos.txt");
		BufferedOutputStream bos = new BufferedOutputStream(fos);*/
		BufferedOutputStream bos = new BufferedOutputStream(
				new FileOutputStream("bos.txt"));

		bos.write("hello".getBytes());

		bos.close();
	}
}
```

BufferedInputStream

2018年7月22日 11:52

```
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.IOException;

public class BufferedInputStreamDemo {
	public static void main(String[] args) throws IOException {
		BufferedInputStream bis = new BufferedInputStream(
				new FileInputStream("b.txt"));

		/*int by = 0;
		while((by=bis.read())!=-1) {
			System.out.print((char)by);
		}
		System.out.println("---------");*/

		byte[] bys = new byte[1024];
		int len = 0;
		while((len=bis.read(bys))!=-1) {
			System.out.println(new String(bys,0,len));
		}

		bis.close();
	}
}
```