第二阶段:

1.调试bug能力

2.用逻辑思维来思考问题和编码.

3.企业级开发(前两周把基本知识过滤,开始学习框架)

4.了解第二阶段课程

今天学习内容

1.File类:文件和目录路径名的抽象表示

2.绝对路径:带盘符的路径.

相对路径:相对其他文件或目录的一个路径叫相对路径.

如果当前文件或目录相对其他盘符的的文件或目录来说,此时相对路径==绝对路径.

如果当前文件或目录相对当前所有在的盘符的文件或目录来说,此时相对路径!=绝对路径.

3.file类常用方法:

```
public static void main(String[] args) {
    //获得文件对象
    File f1=new File("aa\\a.txt");
    System.out.println("文件名:"+f1.getName());
    System.out.println("绝对路径:"+f1.getAbsolutePath());


    System.out.println("相对路径:"+f1.getPath());
    System.out.println("是否可读:"+f1.canRead());
    System.out.println("是否可写:"+f1.canWrite());
    System.out.println("是否隐藏:"+f1.isHidden());
    System.out.println("文件的长度:"+f1.length());
    //得到文件最后一次修改时间
    Long time1=f1.lastModified();
    //创建格式化对象
    SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd
```

```
HH:mm:ss");
            //用格式化对象将时间进行格式化成指定格式的字符串
            String time2=sdf.format(new Date(time1));

            System.out.println("最后一次修改时间:"+time2);
        }

        public static void main(String[] args) throws IOException {
            //获得文件对象
            File f1=new File("cc\\bb\\b.txt");

            //获得当前文件对象父目录
            File parentFile=f1.getParentFile();
            //判断父目录是否存在
            if (parentFile.exists()) {//父目录存在
                System.out.println("父目录存在");
            } else {//父目录不存在
                //创建一级目录
                //parentFile.mkdir();
                //创建多级目录
                parentFile.mkdirs();
                System.out.println("父目录创建成功");

            }

            //判断当前文件是否存在
            if (f1.exists()) {//文件存在
                System.out.println("文件存在");
            } else {//文件不存在
                //创建文件
                f1.createNewFile();
                System.out.println("文件创建成功");
            }
```

```
    }

    public static void main(String[] args) throws IOException {
        //获得目录对象
        File f1=new File("cc");

        //判断当前目录是否存在
        if (f1.exists()) {//当前目录存在
            //获得当前目录所有子文件或子目录的字符串形式
            String[] childFiles=f1.list();
            //遍历子文件或子目录
            for (String child : childFiles) {
                //判断子文件或子目录是否以.txt结尾
                if (child.endsWith(".txt")) {
                    System.out.println(child);
                }
            }

        } else {
            System.out.println("当前目录不存在");
        }
    }
```

4.递归:方法自身调用自身.

4.1:可以将原问题拆分成若干子问题,子问题的解决方法与原问题的解决方法一样.

4.2:原问题的解决依赖于所有子问题的解决.

4.3:递归一定要有出口.

eg:

```
/**
 * 2.获得指定目录下所有以.txt结尾的文件(子文件,孙子文件,子子孙孙)
 * @author sx
 * @version 2020年3月16日
 */
public class FileTest4 {
```

```
    public static void main(String[] args) throws IOException {
        //获得目录对象
        File f1=new File("cc");
        if (f1.exists()) {
            //调用递归方法
            digun(f1);
        } else {
            System.out.println("目录不存在");
        }
    }

    /**
     * 判断当前目录的子文件是否以.txt结尾
     * @param f
     */
    public static void digun(File f) {
        //获得当前目录的子文件或子目录
        File[] files=f.listFiles();
        //遍历所有子文件和子目录
        for (File f2 : files) {
            if (f2.isDirectory()) {//当前遍历的是子目录
                digun(f2);//调用自身
            }else{//当前遍历的是子文件
                if (f2.getName().endsWith(".txt")) {
                    System.out.println(f2.getName());
                }
            }
        }
    }
}
```

5.过滤器:将需要数据留下来,不需要数据过滤掉.

```
    /**
```

```
 * 自定义文件过滤器类
 * @author sx
 * @version 2020年3月16日
 */
public class MyFilter implements FilenameFilter{
	/**
	 * 过滤方法
	 * @param dir 要过滤文件对象
	 * @param name要过滤文件名
	 */
	@Override
	public boolean accept(File dir, String name) {
		if (name.endsWith(".txt")) {
			return true;
		} else {
			return false;
		}
	}
}

public static void main(String[] args) throws IOException {
		//获得目录对象
		File f1=new File("cc");

		//判断当前目录是否存在
		if (f1.exists()) {//存在
			//创建一个过滤器对象
			FilenameFilter mf=new MyFilter();
			//用过滤器获得当前目录下满足条件以.txt结尾子文件
			String[] files=f1.list(mf);
			//遍历输出
			for (String f2 : files) {
				System.out.println(f2);
			}
```

```
            } else {
                System.out.println("目录不存在");
            }
    }
```