软件复用课程设计

Client-Server 应用程序

复用文档

PackerUtils构件

小组： Team10

成员： 谢志杰 1352975

计鹏玥 1352914

王思尧 1352896

谢明玥 1352937

徐锦程 1353012

[1． 构件简介 1](#_Toc32688)

[2． 接口说明 1](#_Toc28088)

[2.1 Packer.java 1](#_Toc30777)

[(1) Packer(String ipath, String opath) 1](#_Toc25174)

[(2) packupSuffix(String suffix) 1](#_Toc12499)

[(3) AddToZipFile(File file, ZipOutputStream zos) 2](#_Toc25039)

[(4) PackupTimer(String logFilename) 2](#_Toc28849)

[2.2 PackPerDay.java 3](#_Toc14591)

[(1) PackPerDay(String iPath, String oPath) 3](#_Toc12774)

[2.3 PackPerWeek.java 4](#_Toc32680)

[(1) PackPerWeek(String iPath, String oPath) 4](#_Toc9003)

[2.4 Unpacker.java 4](#_Toc12217)

[(1) unZip(String zipFile, String outputFolder) 4](#_Toc3196)

[3． Demo 5](#_Toc30684)

# 构件简介

|  |  |
| --- | --- |
|  | **对文件进行压缩和解压缩－ Zip&Unzip** |
| 构件介绍 | 这一构件的主要功能是根据制定时间周期对文件进行压缩和解压缩。 |
| 功能说明 | 用户可以使用该构件实现文件的打包相关处理。 |

# 接口说明

## 2.1 Packer.java

注意：应先导入PackerUtils包。

### Packer(String ipath, String opath)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| ipath | String | 文件读取路径 |
| opath | String | 文件保存路径 |

**作用描述：**

实例化Packer对象来对文件进行打包，无返回值，参数iPath对应文件读取路径，参数oPath对应文件输出路径。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，该方法没有返回值，参数ipath对应文件读取路径，参数opath对应文件输出路径，形如：

Packer packer = new Packer(“D://input”,”D://output”);

**注意事项：**

此方法是一个构造函数 ,对应文件读写路径必须存在且有效。

### packupSuffix(String suffix, boolean encryptIt)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| suffix | String | 文件后缀名称 |
| encryptIt | Boolean | 判断文件是否要进行加密 |

**作用描述：**

将打包后的文件保存为一定后缀名格式的文件，且判断是否要对该归档文件进行加密。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，该方法没有返回值，参数suffix表示打包后文件的后缀名，并决定是否要在归档时对该文件进行加密形如：

packer.packupSuffix(“.zip”, true);

**注意事项：**

A.压缩文件的后缀名合法。

B.文件已经进行了压缩。

### AddToZipFile(File file, ZipOutputStream zos)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| file | File | 文件 |
| zos | ZipOutputStream | 文件输出为压缩的zip格式 |

**作用描述：**

实例化Packer对象来对文件进行打包，无返回值，参数file对应要压缩的文件，参数zos对应压缩文件输出流。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，该方法没有返回值，参数ipath对应文件读取路径，参数opath对应文件输出路径，形如：

Packer.AddToZipFile(file, zos);

**注意事项：**

准备压缩的文件以及压缩后的文件路径均需要存在。

### PackupTimer(String logFilename)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| logFilename | String | 日志文件的名字 |

**作用描述：**

实例化PackTimer，从而对日志文件进行计时器绑定。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，该方法没有返回值，参数logFilename表示需要计时器绑定的日志文件名，形如：

PackerTimer packerTimer = new packupTimer(“logFile”);

## 2.2 PackPerDay.java

注意：应先导入PackerUtils包。

### (1) PackPerDay(String iPath, String oPath)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| iPath | String | 读取文件的路径 |
| oPath | String | 输出文件的路径 |

**作用描述：**

实例化用于构造PackPerDay对象进行每日生成文件的打包，无返回值，参数iPath对应文件读取路径，参数oPath对应文件输出路径。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，该方法没有返回值，参数iPath对应文件读取路径，参数oPath对应文件输出路径，形如：

PackPerDay packPerday = new PackPerDay(“D://input”,”D://output”);

**注意事项：**

此方法是一个构造函数 ,对应文件读取的路径下必须有可供读取的文件存在，以及文件读取路径和文件生成路径必须合法有效。

2.3 PackPerWeek.java注意：应先导入PackerUtils包。

### (1) PackPerWeek(String iPath, String oPath)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| iPath | String | 读取文件的路径 |
| oPath | String | 输出文件的路径 |

**作用描述：**

实例化用于构造PackPerWeek对象进行每周生成文件的打包，无返回值，参数iPath对应文件读取路径，参数oPath对应文件输出路径。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，该方法没有返回值，参数iPath对应文件读取路径，参数oPath对应文件输出路径，形如：

PackPerWeek packPerweek = new PackPerWeek(“D://input”,”D://output”);

**注意事项：**

此方法是一个构造函数 ,对应文件读取的路径下必须有可供读取的文件存在,以及文件读取路径和文件输出路径必须合法有效。

## 2.4 Unpacker.java

注意：应先导入PackerUtils包。

### (1) unZip(String zipFile, String outputFolder)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| zipFile | String | 压缩文件名 |
| outputFolder | String | 输出文件路径 |

**作用描述：**

用于解压压缩文件到指定路径下。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，该方法没有返回值，参数zipFile对应压缩文件的文件名，参数outputFolder对应文件输出路径，形如：

Unpacker unpacker = new Unpacker();

Unpacker.unZip(“log.zip”,”D:\\outputFolder”);

**注意事项：**

此方法对应压缩文件必须存在,以及文件输出路径必须合法有效。

## 2.5 DESEncryptor.java

注意：应先导入PackerUtils包。

### (1) DESEncryptor()

调用函数generateKey()来生成密钥。

**作用描述：**

调用函数generateKey()来生成密钥。

**使用方法：**

对于一个实例化的对象，进行构造时可以直接调用，该方法没有返回值，形如：

DESEncryptor desEncryptor = new DESEncryptor();

### (2) DESEncryptor(SecretKey ikey)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| ikey | SecretKey | 文件的密钥 |

**作用描述：**

将传入的密钥赋值给当前对象。

**使用方法：**

对于一个实例化的对象，进行构造时可以直接调用，传入的参数ikey作为文件的密钥，该方法没有返回值，形如：

DESEncryptor desEncryptor = new DESEncryptor(generateKey());

**注意事项：**

传入的ikey必须合法有效。

### (3) generateKey()

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| generateKey() | SecretKey | 生成的文件密钥 |

**作用描述：**

为文件生成密钥。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，没有参数，返回值代表生成的密钥，形如：

this.key = desEncryptor.generateKey();

**注意事项：**

生成的密钥必须复制给一个SecreteKey。

### (4) saveKey(SecretKey key, File file)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| key | SecretKey | 文件密钥 |
| file | File | 密钥对应的文件 |

**作用描述：**

将生成的密钥保存。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，参数分别代表生成的文件密钥以及密钥对应的文件，没有返回值，形如：

desEncryptor.saveKey(key, file);

**注意事项：**

Key和file必须合法有效。

### (5) loadKey(File file)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| file | File | 密钥对应的文件 |

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| SecretKeySpec(encoded, "DES") | SecretKey | 采用DES算法加密后的密钥 |

**作用描述：**

加载先前保存的文件密钥。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，参数分别代表密钥对应的文件，返回值为密钥，形如：

this.key = desEncryptor.loadKey(file);

**注意事项：**

Key必须已经保存过，否则报错，想要加载的密钥不存在。

### (6) encrypt(String file, String destFile)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| file | String | 将要加密的文件 |
| destFile | String | 加密后的文件 |

**作用描述：**

对文件file进行加密，并将加密后的文件保存为destFile。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，参数分别代表加密前后的文件名，无返回值，形如：

desEncryptor.encrypt(“c:/test/srcFile.txt”，”c:/加密后文件.txt”);

**注意事项：**

想要加密的文件file必须合法有效。

### (7) decrypt(String file, String dest)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| file | String | 将要解密的文件 |
| destFile | String | 解密后的文件 |

**作用描述：**

采用DES算法对文件file进行解密，并将解密后的文件保存为dest。

**使用方法：**

对于一个实例化的对象，可以直接调用该方法，参数分别代表解密前后的文件名，无返回值，形如：

desEncryptor.decrypt(“c:/加密后文件.txt”，”c:/ test/解密后文件.txt”);

**注意事项：**

想要解密的文件file必须合法有效。

# Demo

package main;  
  
import java.util.Timer;  
  
  
import File.GetPackageByTimer;  
import File.SaveToFile;  
  
public class Main {  
  
 public static void main(String[] args) {  
 // TODO Auto-generated method stub  
 Timer timer;  
 SaveToFile saveToFile = new SaveToFile("D:\\");  
   
 timer = new Timer();  
 timer.schedule(new GetPackageByTimer(saveToFile.getDirectoryPath()), 0,3000);  
   
 for(int i=0;i<300;i++)  
 {  
 saveToFile.write("write "+i);  
 try {  
 Thread.sleep(1000);  
 System.out.println(i);  
 } catch (InterruptedException e) {  
 // TODO Auto-generated catch block  
 e.printStackTrace();  
 }  
 }  
   
   
 }  
  
}