****

**Java课程上机报告**



**题 目： 上机作业4**

**学 号：**

**姓 名：**

**教 师：**

**日期：2019/11/24**

**一、上机内容或题目：**

**8-2**

实现一个程序，该程序的输入是一个目录字符串和一个文件扩展名字符串，程序递归地搜索该目录及其各级子目录，在这些目录中查找所有扩展名与指定扩展名相同的文件，将这些文件的相对路径名记录下来并向控制台输出。

**8-7**

实现两个程序，第一个程序读入一个文件plain.txt，将读入的每个字节的数值加5，然后输出到另一个文件cipher.txt中；第二个程序从cipher.txt中读取每个字节，将每个字节的数值减5，再输出到文件decrypt.txt中。比较plain.txt与decrypt.txt的内容。

**10-6**

实现日志管理类Logger，实现向唯一的日志文件写入日志日志的功能，并使用Singleton设计模式，同时各个线程排他性地使用日志管理对象的日志写入功能，每隔一段随机时间就执行写入日志动作。

**二、上机步骤及实验结果：**

（抓取截图，并文字说明）

**8-2：**

**思路**：先获取某个路径下的所有文件以及文件夹，放在subFiles数组中，对数组中每个元素遍历，判断每个元素是否是目录，如果是目录则递归，如果是文件，则利用endswith()函数判断文件扩展名是不是符合自己的要求，符合则输出。

**代码：**

package p11\_24;

import java.io. \*;

import java.util.Scanner;

public class OutputRelativePath {

public static void main(String[] args) throws IOException {

// TODO Auto-generated method stub

System.out.println("请输入路径：");

Scanner out1 = new Scanner(System.in);

String path = out1.next();

//String path = "C:\\Users\\natur\\eclipse-workspace";

//String str = ".java" ;

System.out.println("请输入文件扩展名：");

Scanner out2 = new Scanner(System.in);

String str = out2.next();

matchFS(path,str);

}

public static void matchFS(String path,String str) throws IOException{

//递归搜索该目录及其各级子目录

File test = new File(path);

File[] subFiles = test.listFiles(); //获取路径下的所有的文件或文件夹对象

if(null != subFiles) {

for(File subFile:subFiles) {

if(subFile.isDirectory()) //当前路径是一个目录

matchFS(subFile.getAbsolutePath(),str);//递归查找目录

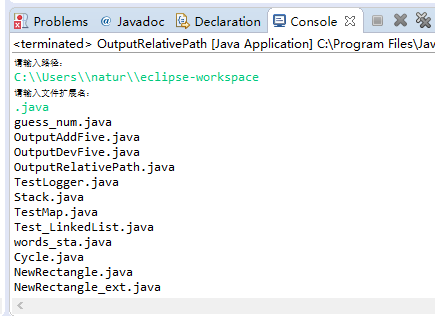
else if(subFile.isFile() && subFile.getName().endsWith(str))

System.out.println(subFile.getName());

}

}

}

}

**8-7：**

**思路：**将plain.txt文件中的内容按字节读出，默认读出是ASCII值，判断该字符是否为换行符，如果是换行符，则直接写入cipher.txt，如果不是则ASCII值加5写入cipher.txt(第二个程序改下输入输出文件，ASCII值减5)

**代码：**

**第一个：**

package p11\_24;

import java.io.\*;

public class OutputAddFive {

public static void main(String[] args) throws IOException {

// TODO Auto-generated method stub

FileInputStream in = new FileInputStream("A:\\test\\plain.txt");

FileOutputStream out = new FileOutputStream("A:\\test\\cipher.txt");

int c;

int s=0;

while((c=in.read())!=-1) {

if(((char)c)!='\r') {

s=c+5;

out.write(s);

}

else

out.write((char)c);

}

in.close();

out.close();

}

}

**第二个：**

package p11\_24;

import java.io.\*;

public class OutputDevFive {

public static void main(String[] args) throws IOException {

// TODO Auto-generated method stub

FileInputStream in = new FileInputStream("A:\\test\\cipher.txt");

FileOutputStream out = new FileOutputStream("A:\\test\\decrypt.txt");

int c;

int s=0;

while((c=in.read())!=-1) {

if(((char)c)!='\r') {

s=c-5;

out.write(s);

}

else

out.write((char)c);

}

in.close();

out.close();

}

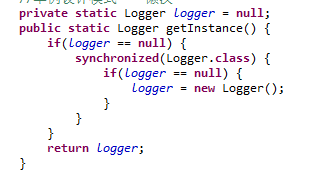
}



加减5之后的结果与明文相同，就是换行符在加5的时候有点奇怪

**10-6：**

**思路：** 实现类Logger具有写日志的功能，先是字符流形式读入文件，则向文件写入时，应该利用getBytes()函数逐比特写入文件。单例模式分为饿汉和懒汉两种模式，考虑多线程，安全，重复等问题，采用懒汉模式。



通过继承Thread类的方法创建线程，所要获取的是线程的id，并且利用Random函数随机的时间执行写入日志动作。

**代码：**

**package** p11\_24;

**import** java.lang.Thread;

**import** java.util.Random;

**import** java.io.\*;

**public** **class** TestLogger{

**public** **static** **void** main(String[] args) {

Logger logger = Logger.*getInstance*();

LogWrite[] t = **new** LogWrite[13];//线程数为13

**for**(**int** i = 0;i<10;i++) {

t[i] = **new** LogWrite();//数组逐位创建

}

**for**(**int** i = 0;i<10;i++) {

t[i].start(); //逐位运行

}

**try** {

Thread.*sleep*(1000);

}

**catch**(Exception e) {

e.printStackTrace();

}

**for**(**int** i =0;i<10;i++) {

t[i].stopRunning();

}

logger.terminate();

}

}

**class** Logger{

**private** **static** FileOutputStream *out*; //字符流写入

**private** **static** File *file*;

**private** Logger() {

*file* = **new** File("A:\\test\\log.txt");

**try** {

*out* = **new** FileOutputStream("A:\\test\\log.txt");

}

**catch**(Exception e) {}

}

**public** **void** LogWrite(String str) {

**try** {

*out*.write(str.getBytes()); //逐比特读入

}

**catch**(Exception e) {

}

}

**public** **void** terminate() {

**try** {

*out*.close();

}

**catch**(Exception e) {

e.printStackTrace();

}

}

//单例设计模式----懒汉

**private** **static** Logger *logger* = **null**;

**public** **static** Logger getInstance() {

**if**(*logger* == **null**) {

**synchronized**(Logger.**class**) {

**if**(*logger* == **null**) {

*logger* = **new** Logger();

}

}

}

**return** *logger*;

}

}

**class** LogWrite **extends** Thread{ //通过继承Thread类的方法创建线程

**private** **boolean** quit = **false**;

**private** **static** **int** *idcnt* = 1;

**private** **final** **int** threadid = *idcnt*++; //线程id

**public** **void** run() {

**while**(!quit) {

Logger logger = Logger.*getInstance*();

Random r = **new** Random();//随机数

String str = "thread".concat(String.*valueOf*(threadid)).concat("\n");

**try** {

Thread.*sleep*(r.nextInt(1000));

}

**catch**(Exception e) {

e.printStackTrace();

}

**synchronized**(logger) {

logger.LogWrite(str);

}

}

}

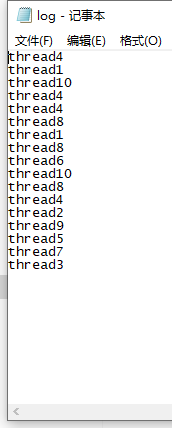
**public** **void** stopRunning()

{

quit = **true**;

}

}



**三、上机总结**

（自己完成本次上机后的心得）

1. 熟练了递归的操作
2. 清楚了利用read()函数读入的是ASCII码值，但是显示还是字符
3. 掌握了通过继承Thread类的方法创建线程，还有注意静态方法在多个类中的使用。
4. 当文件是字符流形式写入是，不能用write()函数，而要用getBytes()函数，逐比特写入文件