**《面向对象程序设计(Java)》**

**实验报告**

姓 名 石家晖

学 号 2019081238

学 院 软件工程学院

专 业 软件工程

年 级 2019级

指导教师 文立玉

**成都信息工程大学软件工程学院**

**2020年9月**

# 实验7：

1. **实验目的**

1. 掌握Java输入输出的结构，

2. 掌握Java流的概念和利用流进行文件操作的方法。

1. **实验要求**

1. 实验在2学时内完成。2

2. 1人1组独立完成。

1. **实验设备与环境**

JDK13, NEOVIM，Linux

1. **设计思路与具体实现**

1.

(1)设计思路

首先创建文件输入输出流reader和writer 分别接收第一个命令行参数和第二个命令行参数(文件路径)然后使用reader的read方法从输入流读取内容，用writer的write方法从输出流输出以实现文本的复制，注意处理IOexception

最后应关闭文件

(2)实验步骤

①新建类FileCopy

②新建FileReader reader/ FileWriter writer读取命令行参数中的路径

③设定标志变量flag以标志是否读到文件末尾和传出读取内容

④当未读到文件末尾(flag == -1)时，重复读取并且使用writer.write(flag)写入要复制到的文件中

⑤关闭文件处理异常

(3)关键代码  
try {  
 //创建文件输入输出流  
 reader = new FileReader(args[0]);  
 writer = new FileWriter(args[1]);  
 int flag = 0;  
 //从输入流读取内容使用输出流输出  
 while ((flag = reader.read()) != -1){  
 writer.write(flag);  
  
 }  
 System.*out*.println("文本复制成功!");  
  
  
 } catch(IOException e){  
 e.printStackTrace();  
  
 }finally {  
 if(reader != null){  
 try{  
 reader.close(); //关闭文件输入  
 }catch (IOException e) {  
 e.printStackTrace();  
 }  
 }

(4)程序测试过程

①在工程目录中新建read.txt 内容为：‘2019081238abcd’

②在工程目录中新建write.txt

③运行程序，检查read.txt内容是否复制到write.txt中

(5)程序运行结果(运行效果)

2020-11-22-230838_723x56_scrot

图7-1-1

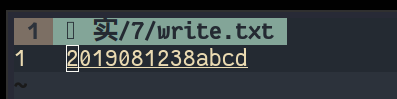
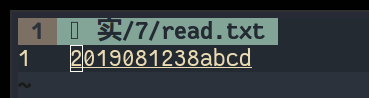


图7-1-2

2.

(1)设计思路

将任意两个文件合并到一个文件，要求采用java命令行方式在控制台按照“源文件1 源文件2 目标文件” 方式录入，注意多种异常处理。

首先定义输出文件流、读取文件流 ，使用File[]数组接收命令行参数路径的对应文件，并对每一个文件进行读取并写入换行操作。最后关闭文件处理异常。

(2)实验步骤

①首先定义输出流BufferedWriter bw 接收命令行参数中的目标文件路径

②定义文件对象数组list接收参数中的两个源文件路径

③定义文件输入流针对对象数组list中的每一个文件进行读取操作并写入 bw中然后使用bw.newline换行

1. 直至两个源文件都读取写入完成前，重复上述读写操作
2. 关闭文件处理异常。
3. 编译执行程序,观察实验结果

(3)关键代码

try {

BufferedWriter bw = new BufferedWriter(new FileWriter(args[2]));

//读取文件

File[] list = new File[2];

list[0] = new File(args[0]);

list[1] = new File(args[1]);

for (File file:list) {

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine())!=null){

bw.write(line);

bw.newLine(); //换行

}

br.close();

}

bw.close();

System.out.println("合并成功");

}catch (FileNotFoundException e){

System.out.println("未找到指定文件！");

e.printStackTrace();

}catch (IOException e){

e.printStackTrace();

}catch (Exception e){

e.printStackTrace();

}

(4)程序测试过程

①在工程目录中新建reader1.txt 内容为：‘1.这是第一个文件’

②在工程目录中新建reader2.txt 内容为 ‘2.这是第二个个文件’

③运行程序，检查reader内容是否合并到writer.txt中

(5)程序运行结果(运行效果)

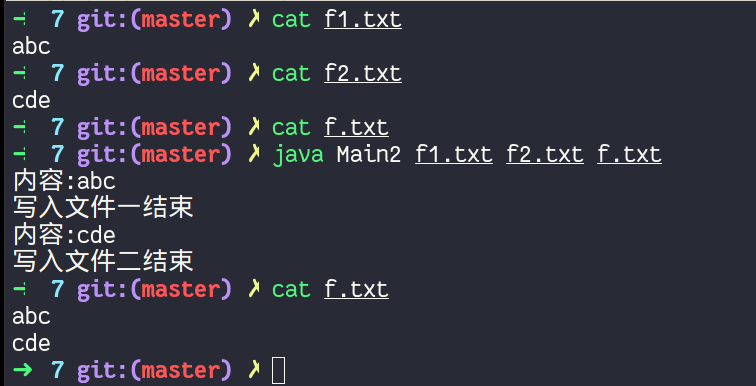


图7-2-2

图7-2-3

图7-2-4

图7-2-5

图7-2-6

3.

(1)设计思路

编写程序实现将一个文件内容追加到另一个文件内容后，如将D盘file文件夹下的f1.txt追加到E盘根目录下的f2.txt中。(必须异常处理)

首先使用BufferedReader和FileWriter创建输入输出流，输出流追加标示为true然后读取输入流追加到输出流，最后关闭文件处理异常。

(2)实验步骤

①新建工程和主类test7\_3

②定义文件输入流BufferedReader read 和 输出流FileWriter write以及String类型临时变量temp，注意输出流追加标记为true

③执行read.readline操作从输入流读取使用temp临时储存，将temp内容使用write.write()方法追加到输出文件后。

④处理FileNotFoundException/IOExcepetion等异常

⑤编译执行程序，测试检验程序运行结果。

(3)关键代码

BufferedReader read = new BufferedReader(new FileReader(new File("from.txt")));

FileWriter write = new FileWriter(new File("to.txt"),true);

String temp;

while((temp = read.readLine())!= null){

write.write(temp);

}

read.close();

write.close();

System.out.println("追加成功");

(4)程序测试过程

①在工程目录中新建read3.txt 内容为：‘追加的内容’

②在工程目录中新建write3.txt 内容为: ‘原文件’

③运行程序，检查read3.txt内容是否追加到write3.txt中

(5)程序运行结果(运行效果)

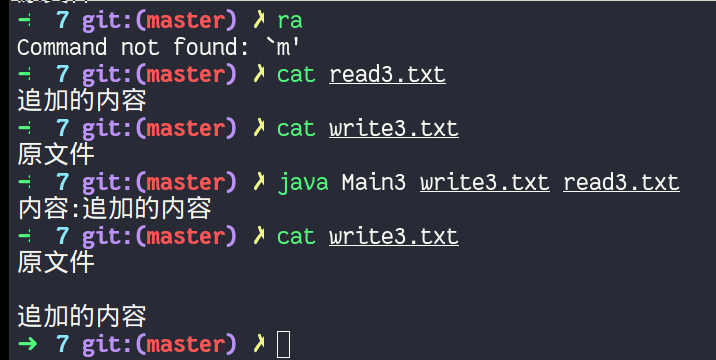


图7-3-1

图7-3-2

图7-3-3

1. **实验总结**

通过本次实验我熟练掌握了Java语言中流的概念和利用流进行对文件操作的方法，掌握了Java输入输出流的基本操作、基本类和基本方法以及这些类与方法的基本功能和基本应用场景，掌握了Java输入输出的结构，了解了输入输出和各类不同用途的Java IO类，了解了一些输入输出流的相关方法，以及强化了异常处理try catch语句的操作。总的说来，实验内容较为简单，但内涵深刻，收获良多。

1. **附录（程序代码）**

1.

import java.io.\*;

public class Main1 {

public static void main (String[] args) {

FileReader reader = null;

FileWriter writer = null;

try {

//创建文件输入输出流

reader = new FileReader (args[0]);

writer = new FileWriter (args[1]);

int flag = 0;

//从输入流读取内容使用输出流输出

while ( (flag = reader.read() ) != -1)

writer.write (flag);

System.out.println ("文本复制成功!");

} catch (IOException e) {

e.printStackTrace();

}

finally {

if (reader != null) {

try {

reader.close(); //关闭文件输入

} catch (IOException e) {

e.printStackTrace();

}

}

if (writer != null) {

try {

writer.close(); //关闭文件输入

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}2.

import java.io.\*;

public class Main2{

public static void main (String[] args) {

String in1Path = args[0];

String in2Path = args[1];

String toPath = args[2];

try {

BufferedReader br1 = new BufferedReader (new FileReader (in1Path) );

BufferedReader br2 = new BufferedReader (new FileReader (in2Path) );

BufferedWriter out = new BufferedWriter (new FileWriter (toPath) );

String s = null;

while ( (s = br1.readLine() ) != null) { //写入文件一

out.write (s);

out.newLine();

System.out.println ("内容:" + s);

}

br1.close();

System.out.println ("写入文件一结束");

while ( (s = br2.readLine() ) != null) { //写入文件二

out.write (s);

out.newLine();

System.out.println ("内容:" + s);

}

br2.close();

System.out.println ("写入文件二结束");

out.flush();

out.close();

} catch (IOException e) {

System.out.println (e.toString() );

}

}

}

3.

import java.io.\*;

public class Main3{

public static void main (String[] args) {

String txt1 = args[0];

String txt2 = args[1];

String s = null;

try {

BufferedWriter out = new BufferedWriter (new FileWriter (txt1, true) );

BufferedReader br1 = new BufferedReader (new FileReader (txt2) );

out.newLine();//提前换行，追加在后面

while ( (s = br1.readLine() ) != null) { //写入文件

out.write (s);

out.newLine();

System.out.println ("内容:" + s);

}

br1.close();

out.flush();

out.close();

} catch (IOException e) {

System.out.println (e.toString() );

}

}

}