中南大学计算机学院

《Java语言与系统设计》

实验报告

计算机学院

|  |  |
| --- | --- |
| 课程名称： | Java语言与系统设计 |
| 实验名称： | No.1中缀转后缀求值 |
| 指导教师： | 张伟 |
| 学生姓名： | 张子洋 |
| 实验日期： | 23-12-04 |
| 实验地点： | 信息楼403 |
| 实验成绩： |  |

目录

[《Java语言与系统设计》 1](#_Toc14270)

[实验报告 1](#_Toc26798)

[一、实验目的 3](#_Toc5697)

[二、实验环境 4](#_Toc12941)

[三、实验内容与实验要求 5](#_Toc31756)

[四、实验过程与分析 6](#_Toc23578)

[五、实验结果总结 7](#_Toc32578)

[六、附录（源代码） 8](#_Toc8449)

[《Java语言与系统设计》 15](#_Toc23186)

[实验报告 15](#_Toc20512)

[一、实验目的 17](#_Toc19621)

[二、实验环境 18](#_Toc13240)

[三、实验内容与实验要求 19](#_Toc19466)

[四、实验过程与分析 20](#_Toc12327)

[五、实验结果总结 21](#_Toc31622)

[六、附录（源代码） 22](#_Toc30069)

[《Java语言与系统设计》 25](#_Toc27126)

[实验报告 25](#_Toc16950)

[一、实验目的 27](#_Toc27278)

[二、实验环境 28](#_Toc29225)

[三、实验内容与实验要求 29](#_Toc11363)

[四、实验过程与分析 30](#_Toc20352)

[五、实验结果总结 31](#_Toc25082)

[六、附录（源代码） 32](#_Toc9098)

一、实验目的

中缀表达式到后缀表达式的转换以及求值，这个实验有助于学生理解表达式的不同表示方式以及如何使用栈数据结构来进行表达式的处理。

目的：

1. 理解中缀表达式和后缀表达式的概念；
2. 掌握中缀表达式到后缀表达式的转换算法；
3. 学习使用栈进行表达式求值：通过实现中缀表达式到后缀表达式的转换，了解如何使用栈这种数据结构来辅助这一过程。后缀表达式的求值也通常涉及到使用栈；
4. 加深对操作符优先级的理解：在中缀表达式转换为后缀表达式的过程中，需要处理不同操作符的优先级，这有助于加深对运算符的理解；

5. 熟悉基本的表达式求值：学生需要实现后缀表达式的求值算法，这要求他们对基本的数学运算和栈的使用有一定的熟练度。

二、实验环境

Java 开发工具包JDK17版本

Eclipse IDE for Java Developers

Windows 11 系统

三、实验内容与实验要求

把中缀表达式转变为后缀表达式，并求值。

四、实验过程与分析

实现带括号，多位数，小数，正负数的四则和取余运算,以及三角函数，反三角函数运算。

先通过deal方法对输入的中缀表达式进行操作，包括在每个符号前后插入空格（insert），以方便后续的分割（split）。使用 MiddleToEnd 方法将中缀表达式转换为后缀表达式，使用两个栈（Opstack 和 list）进行操作。将处理后的后缀结果存到arrylist动态数组中，然后输出后缀表达式。

使用Positive\_negative方法在输入中的加法和减法运算符前添加0，以处理正负数的情况（加0后的结果不会输出）。然后使用calculate方法对后缀表达式进行求值，通过一个数据栈（dataStack）来存储中间结果。

在 main 方法中，通过 Scanner 获取用户输入的中缀表达式，并创建 A 类的实例来进行处理和计算。代码使用StringBuilder类来处理字符串，通过HashMap存储运算符的优先级，简化运算符比较的逻辑。

中缀转后缀操作：

// 1.如果遇到操作数时，将其压入s2；

// 2.如果运算符时，比较其与s1栈顶运算符的优先级；

//  （1）如果s1为空，或栈顶运算符为左括号“（”，则直接入栈；

//  （2）否则，若优先级比栈顶运算符高，也将运算符压入s1；

//  （3）否则，将s1栈顶运算符弹出并压入s2中，再转到4-（1）与s1中新的栈顶运算符相比较；

// 3.遇到括号时：

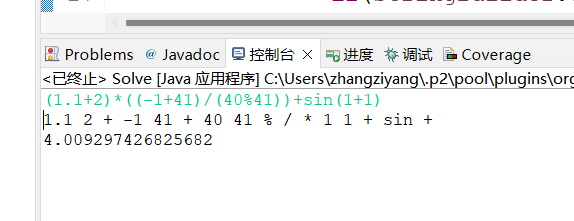
//  （1）如果是左括号“(”,则直接压入s1；

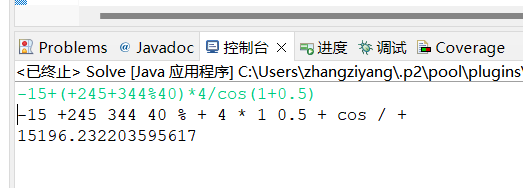
//  （2）如果是右括号“）”，则依次弹出s1栈顶的运算符，并压入s2,直到遇到左括号，此时将这一对括号丢弃；

// 重复步骤1至3，直到表达式的最右边；

// 4.将s1中剩余的运算符依次弹出并压入s2；

运行结果：





五、实验结果总结

总结：

代码实现的功能较多，采用字符串和判断语句，实现的较为完整。

改进意见：

1. 添加更多的算术函数，以提高代码的灵活性和适用性。
2. 考虑对输入表达式进行错误处理，如空格匹配检查等。
3. 增加注释，以提高代码的竞争力和可维护性。

4.calculate方法可以将if else语句换成switch语句，更整洁。

六、附录（源代码）

package test\_1;

import java.util.Stack;

import java.util.Scanner;

import java.util.ArrayList;

import java.util.HashMap;

//——————————————————————————————————————

//这个代码写的时间较长，实现带括号，多位数，小数，正负数的四则和取余运算,以及三角函数，反三角函数运算

//不足是运算符较少，可以再扩展一些运算符

//-1+(+2+3)\*4 (1+2)\*((39+1)/40) (1.1+2)\*((-1+41)/(40%41))+sin(1+1)

class A{

Stack <String>Opstack =new Stack<>();

ArrayList<String> list=new ArrayList<>();//Arraylist来存放后缀，char和string不兼容

String str;

String []strs;

Stack<String> dataStack = new Stack<>();

//预处理。初始化strings，通过stringbuilder在字符串中每个符号的前后插入空格（如果是"("只在后面加空格,")"加在前面）

//通过split方法以空格为标识符拆分字符串，并存入strings

public void deal(){//( 1.1 + 2 ) \* ( ( -1 + 41 ) / ( 40 % 41 ) ) + sin ( 1 + 1 )

int i=0;

StringBuilder stringBuilder=new StringBuilder(str);

while(i< stringBuilder.length()){

if(!Character.isDigit(stringBuilder.charAt(i))){

if(stringBuilder.charAt(i)=='s'||stringBuilder.charAt(i)=='c'||stringBuilder.charAt(i)=='t'||stringBuilder.charAt(i)=='a'){

while(true) {

if(stringBuilder.charAt(i)>='a'&&stringBuilder.charAt(i)<='z')

i++;

else {

stringBuilder.insert(i,' ');

i++;

if(stringBuilder.charAt(i)=='(') {

stringBuilder.insert(i+1,' ');

i++;

}

break;

}

}

} else if(stringBuilder.charAt(i)=='+'||stringBuilder.charAt(i)=='-') {

if(i==0) {

//stringBuilder.insert(i+1,' ');

//i++;

}

else if(i>=2&&stringBuilder.charAt(i-2)=='('){

//stringBuilder.insert(i+1,' ');

//i++;

}

else {

stringBuilder.insert(i, ' ');

stringBuilder.insert(i + 2, ' ');

i = i + 2;

}

} else if(stringBuilder.charAt(i)=='('){

stringBuilder.insert(i+1,' ');

i++;

}else if (stringBuilder.charAt(i)==')'){

stringBuilder.insert(i,' ');// 啊啊啊啊啊啊啊啊啊 检查了很长世间

i++;

}else if(stringBuilder.charAt(i)=='.') { }

else {

stringBuilder.insert(i, ' ');

stringBuilder.insert(i + 2, ' ');

i = i + 2;

}

}

i++;

}

//System.out.println(stringBuilder);

str =stringBuilder.toString();//返回该对象的字符串表示（默认表现形式：类的名称+@+把一个hashcode的值转成16进制的数字）。

strs=str.split(" ");//空格为分隔符

}

// 1.如果遇到操作数时，将其压入s2；

// 2.如果运算符时，比较其与s1栈顶运算符的优先级；

//  （1）如果s1为空，或栈顶运算符为左括号“（”，则直接入栈；

//  （2）否则，若优先级比栈顶运算符高，也将运算符压入s1；

//  （3）否则，将s1栈顶运算符弹出并压入s2中，再转到4-（1）与s1中新的栈顶运算符相比较；

// 3.遇到括号时：

//  （1）如果是左括号“(”,则直接压入s1；

//  （2）如果是右括号“）”，则依次弹出s1栈顶的运算符，并压入s2,直到遇到左括号，此时将这一对括号丢弃；

// 重复步骤1至3，直到表达式的最右边；

// 4.将s1中剩余的运算符依次弹出并压入s2；

public void MiddleToEnd() {

HashMap<String, Integer> characterHashMap = new HashMap<String ,Integer>();

characterHashMap.put("+",0);

characterHashMap.put("-",0);

characterHashMap.put("%",1);

characterHashMap.put("\*",1);

characterHashMap.put("/",1);

characterHashMap.put("sin",2);

characterHashMap.put("cos",2);

characterHashMap.put("tan",2);

characterHashMap.put("arctan",2);

characterHashMap.put("arcsin",2);

characterHashMap.put("arccos",2);

characterHashMap.put("(",4);

characterHashMap.put(")",4);

//优先级

for(String c:strs){

if(Character.isDigit(c.charAt(0))) {

list.add(c);

}

else if(c.equals("(")) {

Opstack.push(c);

}

else if(c.equals(")")) {

String judge;

judge=Opstack.pop();

while(true) {

list.add(judge);//(1+1)/2

judge=Opstack.pop();

if(judge.equals("("))

break;

}

}

else {//运算符

if(c.length()>=2&&Character.isDigit(c.charAt(1))){

if(c.charAt(0)=='-'||c.charAt(0)=='+'){

list.add(c);

}

}//正负数

else {

while(true) {

if(Opstack.isEmpty()||Opstack.peek().equals("(")){

Opstack.push(c);

break;

}

else if(characterHashMap.get(c) > characterHashMap.get(Opstack.peek())){

Opstack.push(c);

break;

}else {

list.add(Opstack.pop());

}

}

}

}

}

//将 op中剩余的运算符依次弹出并压入 list

while(!Opstack.isEmpty()){

String n=Opstack.pop();

list.add(n);

}

}

//遍历我们的集合（存放后缀表达式的）将数据暂时放入栈中方便我们操作，然后在遍历过程中进行判断，如果是数据就直接放入栈中，

//如果是运算符就从栈中取出两个数据进行运算，运算结果又放入栈中，直到栈中只存在一个数据时，就是最后的运算结果。

public double calculate() {

//Stack <Double> Calstack =new Stack<>();//用于计算

for (String s:list) {

if (Character.isDigit(s.charAt(0))){

dataStack.push(s);

}else {//运算符

double num1;

double num2;

double res = 0;

if (s.equals("+")){

num1= Double.parseDouble(dataStack.pop());//字符串转为 Double.parseDouble()

num2= Double.parseDouble(dataStack.pop());

res = num1 + num2;

}else if (s.equals("-")) {

num1= Double.parseDouble(dataStack.pop());

num2= Double.parseDouble(dataStack.pop());

res = num2 - num1;

}else if (s.equals("\*")) {

num1= Double.parseDouble(dataStack.pop());

num2= Double.parseDouble(dataStack.pop());

res = num1 \* num2;

}else if (s.equals("/")) {

num1= Double.parseDouble(dataStack.pop());

num2= Double.parseDouble(dataStack.pop());

res = num2 / num1;

}else if (s.equals("%")) {

num1= Double.parseDouble(dataStack.pop());

num2= Double.parseDouble(dataStack.pop());

res = num2 % num1;

}else if(s.equals("sin")) {

num1= Double.parseDouble(dataStack.pop());

res=Math.sin(num1);

}else if(s.equals("cos")) {

num1= Double.parseDouble(dataStack.pop());

res=Math.cos(num1);

}else if(s.equals("tan")) {

num1= Double.parseDouble(dataStack.pop());

res=Math.tan(num1);

}else if(s.equals("arcsin")) {

num1= Double.parseDouble(dataStack.pop());

res=Math.asin(num1);

}else if(s.equals("arccos")) {

num1= Double.parseDouble(dataStack.pop());

res=Math.acos(num1);

}else if(s.equals("arctan")) {

num1= Double.parseDouble(dataStack.pop());

res=Math.atan(num1);

}

else{

throw new RuntimeException("运算符异常！");

}

dataStack.push("" + res);

}

}

//返回结果

return Double.parseDouble(dataStack.pop());

}

public void Positive\_negative() {

StringBuilder pn=new StringBuilder(str);

int i=0;

while(i< pn.length()){

if(pn.charAt(i)=='+'||pn.charAt(i)=='-') {

if(i==0||pn.charAt(i-1)=='(') {

pn.insert(i,"0");

i++;

}

}

i++;

}

str=pn.toString();//返回该对象的字符串表示（默认表现形式：类的名称+@+把一个hashcode的值转成16进制的数字）。

//System.out.println(str);

}//正负只需要加个0就行了，0不能输出来

}

public class Solve {

public static void main(String []args) {

Scanner in =new Scanner(System.in);

A a=new A();

A b=new A();

a.str = in.nextLine();

b.str=a.str;

a.deal();

a.MiddleToEnd();

for (int i=0;i<a.list.size();i++) {

System.out.print(a.list.get(i)+" ");

}

System.out.println();

b.Positive\_negative();

b.deal();

b.MiddleToEnd();

// for (int i=0;i<b.list.size();i++) {

// System.out.print(b.list.get(i)+" ");

// }

//System.out.println();

double x=b.calculate();

System.out.println(x);

in.close();

}

}

中南大学计算机学院

《Java语言与系统设计》

实验报告

计算机学院

|  |  |
| --- | --- |
| 课程名称： | Java语言与系统设计 |
| 实验名称： | No.2比较重复率 |
| 指导教师： | 张伟 |
| 学生姓名： | 张子洋 |
| 实验日期： | 23-12-07 |
| 实验地点： | 信息楼403 |
| 实验成绩： |  |

目录

[《Java语言与系统设计》 1](#_Toc12285)

[实验报告 1](#_Toc32061)

[一、实验目的 3](#_Toc32341)

[二、实验环境 4](#_Toc12341)

[三、实验内容与实验要求 5](#_Toc27662)

[四、实验过程与分析 6](#_Toc12614)

[五、实验结果总结 7](#_Toc28613)

[六、附录（源代码） 8](#_Toc24710)

[《Java语言与系统设计》 15](#_Toc17785)

[实验报告 15](#_Toc32048)

[一、实验目的 17](#_Toc15817)

[二、实验环境 18](#_Toc12912)

[三、实验内容与实验要求 19](#_Toc10263)

[四、实验过程与分析 20](#_Toc10107)

[五、实验结果总结 21](#_Toc14006)

[六、附录（源代码） 22](#_Toc5508)

[《Java语言与系统设计》 25](#_Toc30091)

[实验报告 25](#_Toc14931)

[一、实验目的 27](#_Toc29827)

[二、实验环境 28](#_Toc14045)

[三、实验内容与实验要求 29](#_Toc32429)

[四、实验过程与分析 30](#_Toc7954)

[五、实验结果总结 31](#_Toc10518)

[六、附录（源代码） 32](#_Toc4453)

一、实验目的

1. 熟悉文件操作：帮助学生熟悉Java中的文件操作，包括读取文本文件的内容、比较文件中的文本等。
2. 字符串处理：文本比较涉及字符串的处理和匹配，可以促使学生深入理解Java中的字符串处理方法，如字符串比较、替换、分割等。
3. 掌握文本相似性算法：学习和实现文本相似性算法，例如基于编辑距离的算法、基于余弦相似度的算法等。这有助于理解不同算法的原理和性能。

4. 应用文本处理库：使用Java中的文本处理库，例如正则表达式、字符串匹配算法等，以提高文本处理的效率和灵活性。

通过这些手段，让学生可以在实际问题中应用Java编程技能，深入了解文本处理和相似性比较的算法原理，提高解决实际问题的能力。

二、实验环境

Java 开发工具包JDK17版本

Eclipse IDE for Java Developers

Windows 11 系统

三、实验内容与实验要求

比较两个文本文件的重复率。

四、实验过程与分析

文件读取：

使用readFileContent方法读取文件内容，将文件内容使用StringBuilder存储，这有助于更高效地处理大文件。

文本处理：

使用 splitToWords 方法将文本转为字符串数组。该方法在处理中文时，通过插入空格的方式将中文字符与其他字符分隔开。用编码范围来判断字符和中文。

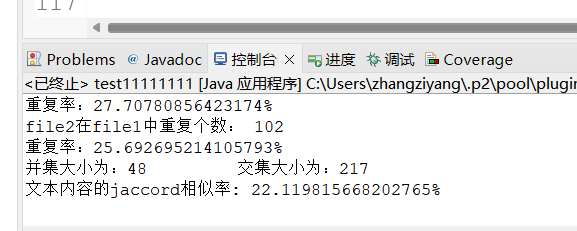
相似尺寸计算：

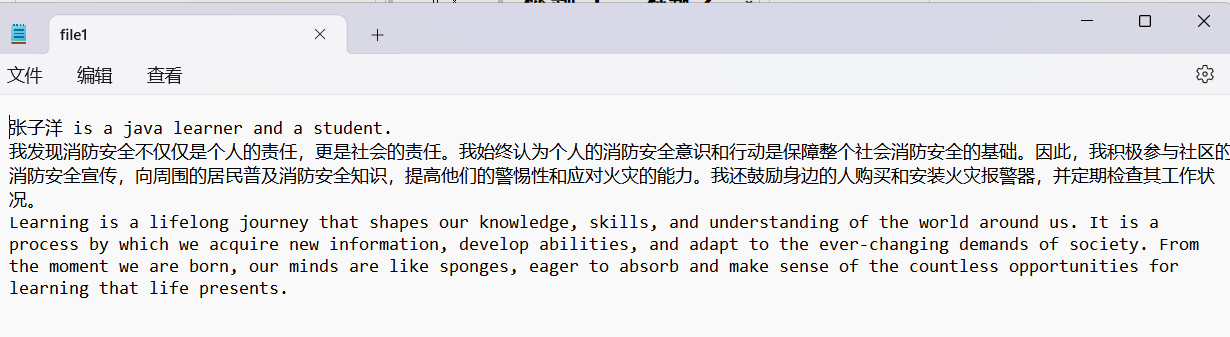
1.使用for循环嵌套实现统计重复单词和汉字的个数

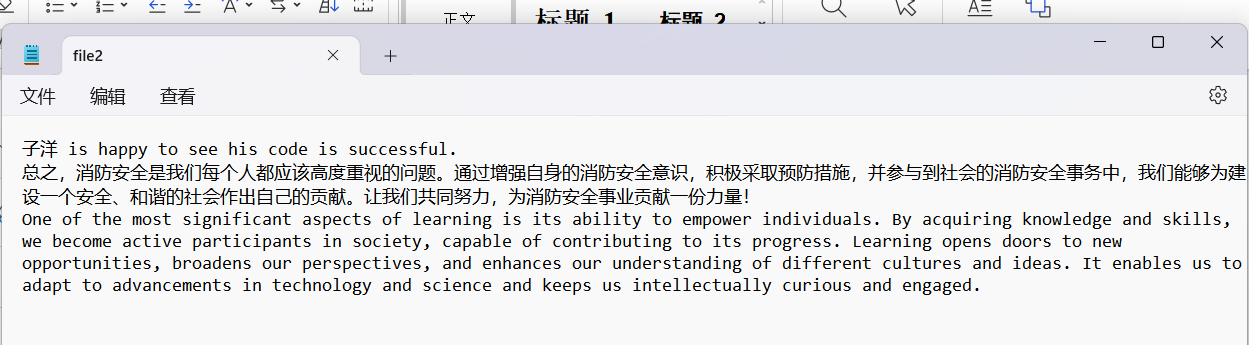
2.使用HashSet存储文本中的词汇，计算交集和并集的大小，并通过使用Jaccard相似算法计算两个的文本的相似度。

输出：

用两种方法输出文本文件中的重复词汇数量以及两个文件的重复率。







五、实验结果总结

总结：

基本实现两个文本重复率比较，但还有不少的改进空间。

改进：

中文分词：对中文的处理方式并不完善。虽然在splitToWords方法中插入了空格，但在中文处理上可能还存在一些问题。使用专业的中文分词工具，可以提高对中文文本的处理效果。

优化Jaccard相似性计算：考虑使用Java集合框架的更多方法，以简化Jaccard类似性的计算逻辑。

提高通用性：在处理特殊符号时，应当考虑到文本的清理。将文件读取、文本处理和相似度计算等功能分割维护成更小的，以提高代码的判断性和可读性。

异常处理：在文件读取时，可以添加一些异常处理机制，以应对文件不存在等异常情况。

代码注释：添加详细的代码注释，解释算法和逻辑的实现细节，提高代码的可用性。

六、附录（源代码）

package test;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashSet;

import java.util.Set;

import java.util.Arrays;

//———————————————————————————————————————————————————————————————————————————

//代码的不足：首先是极端案例，代码中所有的符号都不会被记入重复，除了中英文，大小写也已经实现，

//比如若文章引用了公式，不会计入重复率

//然后是两种计算方法不够好，jaccord相似度很笼统，而那两个for循环不太好，只是机械的记录数据

public class test11111111 {

//1.转为字符串

public static StringBuilder readFileContent(File file) {

StringBuilder sb = new StringBuilder();

try (BufferedReader br = new BufferedReader(new FileReader(file))) {

String line;

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

sb.append(line);//通过readLine读取字符串；append拼接一下

}

} catch (IOException e) {

System.out.println("Error: " + e.getMessage());

}

return sb;

}

public static double calculateSimilarity(StringBuilder content1, StringBuilder content2) {

//转成字符串数组

String [] str1=splitToWords(content1);

String [] str2=splitToWords(content2);

Set<String> set1 = new HashSet<>(Arrays.asList(str1));

Set<String> set2 = new HashSet<>(Arrays.asList(str2));

Set<String> intersection = new HashSet<>(set1);

intersection.retainAll(set2); //retainAll 交集

Set<String> union = new HashSet<>(set1);

union.addAll(set2); //addAll 并集

double num1=0,num2=0;

for(int i=0;i<str1.length;i++) {

for(int j=0;j<str2.length;j++) {

if(str1[i].equalsIgnoreCase(str2[j])) {

num1++;

//System.out.print(str1[i]+" ");

break;

}//IgnoreCase可以实现忽略大小写

}

}

//System.out.println();

double y=str1.length+str2.length;

double x=0;

x=num1/y;

System.out.println("file1在file2中重复个数： "+(int)num1);

System.out.println("重复率："+x\*100+"%");

for(int i=0;i<str2.length;i++) {

for(int j=0;j<str1.length;j++) {

if(str2[i].equalsIgnoreCase(str1[j])) {

//System.out.print(str1[j]+" ");

num2++;

break;

}

}

}

//System.out.println();

System.out.println("file2在file1中重复个数： "+(int)num2);

x=num2/(str1.length+str2.length);

System.out.println("重复率："+x\*100+"%");

System.out.println("并集大小为："+intersection.size()+" 交集大小为："+union.size());

return (double) intersection.size() / union.size();

}

public static boolean isChinese(char c) {

//return c >= 0x4E00 && c <= 0x9FA5;

String b = Character.toString(c);

return b.matches("[\u4E00-\u9FA5]+");

}

public static String[] splitToWords(StringBuilder content) {

int i=0;

while(i<content.length()) {

char c = content.charAt(i);

if (isChinese(c)) {

//System.out.println(c);

content.insert(i+1,' ');

i++;

}

i++;

}

String sb=content.toString() ;

//String[] words = sb.split("\\s+|，|。|；|,|?|？|\\.|\\\\");

// '.'是正则表达式中的一个元字符，代表着任意一个字符（除换行符），所以上一行代码写出来会报错

String words[]=sb.split("\\s+|[\\p{Punct}\\s]+|[\\pP]+");

//[\\p{Punct}\\s]+ 表示所有半角字符 [\\pP]+ 全角

return words;

}

public static void main(String[] args) {

File file1 = new File("file1.txt");

File file2 = new File("file2.txt");

StringBuilder content1=readFileContent(file1);

StringBuilder content2=readFileContent(file2);

double similarity = calculateSimilarity(content1, content2)\*100;

System.out.println("文本内容的jaccord相似率: " + similarity+"%");

}

}

中南大学计算机学院

《Java语言与系统设计》

实验报告

计算机学院

|  |  |
| --- | --- |
| 课程名称： | Java语言与系统设计 |
| 实验名称： | No.3多消费-生产者模拟 |
| 指导教师： | 张伟 |
| 学生姓名： | 张子洋 |
| 实验日期： | 2023.12.15 |
| 实验地点： | 信息楼403 |
| 实验成绩： |  |

目录

[《Java语言与系统设计》 1](#_Toc24401)

[实验报告 1](#_Toc5030)

[一、实验目的 3](#_Toc31447)

[二、实验环境 4](#_Toc13481)

[三、实验内容与实验要求 5](#_Toc1331)

[四、实验过程与分析 6](#_Toc29956)

[五、实验结果总结 7](#_Toc12019)

[六、附录（源代码） 8](#_Toc7154)

[《Java语言与系统设计》 15](#_Toc8126)

[实验报告 15](#_Toc8816)

[一、实验目的 17](#_Toc25802)

[二、实验环境 18](#_Toc8285)

[三、实验内容与实验要求 19](#_Toc32752)

[四、实验过程与分析 20](#_Toc7103)

[五、实验结果总结 21](#_Toc1133)

[六、附录（源代码） 22](#_Toc4868)

[《Java语言与系统设计》 25](#_Toc22156)

[实验报告 25](#_Toc8825)

[一、实验目的 27](#_Toc26190)

[二、实验环境 28](#_Toc13677)

[三、实验内容与实验要求 29](#_Toc1500)

[四、实验过程与分析 30](#_Toc17620)

[五、实验结果总结 31](#_Toc31250)

[六、附录（源代码） 32](#_Toc15131)

一、实验目的

1.学习多线程编程：了解多线程编程的基本概念和机制，包括线程的创建、同步、互斥等，以及Java中多线程编程的相关类和方法。

2.理解生产者-消费者问题：通过实际模拟生产者-消费者问题，深入理解这一典型的多线程同步问题，学会如何有效地协调多个线程的工作。

3.熟悉信号量的使用：学习信号量的概念和用法，了解如何通过信号量来控制线程的数量，以及在多线程环境下如何使用信号量进行同步。

4.掌握UI编程：通过在Java中使用UI组件，如Swing，学会创建用户界面，实现图形化展示多消费者-生产者问题的模拟。

5.实践多线程同步机制：掌握在多线程环境下使用锁、条件变量等同步机制，以保证生产者和消费者之间正确良好的工作，不会出现死锁等问题。

6.了解线程间通信：学会在多线程程序中实现线程间的通信，包括生产者通知消费者有新数据可用，消费者通知生产者有足够的数据空间搁置数据等。

二、实验环境

Java 开发工具包JDK17版本

Eclipse IDE for Java Developers

Windows 11 系统

三、实验内容与实验要求

运用UI、信号量、多线程等技术，实现多消费者-生产者问题的模拟。

四、实验过程与分析

多线程模拟生产消费：

使用了Producer和Consumer两个实现了Runnable接口的类来模拟生产者和消费者。这是一种经典的多线程同步问题的解决方案。

使用JTextArea和JScrollPane：

通过JTextArea实现文本区域，使用JScrollPane创建滚动面板，方便查看多行文本。

按钮和事件监听：

使用JButton实现开始和结束按钮，通过添加ActionListener实现按钮的点击事件监听。

界面布局：

使用BorderLayout布局管理器进行简单的界面布局。

线程启动和停止：

通过按钮触发，启动生产者和消费者线程，通过设置 desk.flag 为 false 来停止生产者和消费者线程的执行。



五、实验结果总结

总结：

这个代码实现了简单的生产消费模拟，使用了滚动面板，Jframe窗口和文本框，由button按钮控制进行简单的文本显示。多线程采用Runnable 接口实现，缓冲区由int数组实现，整体代码很简单，功能也简单，实现的一般。

改进：

使用Executor框架：

考虑使用Executor框架来管理线程池，以更灵活地提高管理线程的启动和关闭，效率更高。

参数可配置：

通过界面设置文本框，通过外部输入传参，使得生产者、消费者和缓冲区大小等参数可以动态设置，提高代码的通用性。

同步控制：

在多线程环境中，对于共享资源的访问需要进行同步处理。虽然代码中使用了synchronized关键字，但在更复杂的场景中可能需要更详细的同步控制，可以使用更精细的锁定机制、条件变量等。

界面优化：

在界面上增加更多的交互元素，提高用户体验，例如显示当前生产者和消费者的状态、生产和消费速率等，图像也更直观。

异常处理：

增加对可能的异常的处理，对Interrupted Exception进行更进一步的处理。

代码注释和命名：

添加更多的注释，解释代码的逻辑和实现细节，提高可读性。

六、附录（源代码）

**package** experiment\_3;

**import** java.awt.BorderLayout;

**import** javax.swing.JFrame;

**import** javax.swing.JTextArea;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.JButton;

**import** javax.swing.JScrollPane;

//———————————————————————————————————————————————————————————————————————————

//这个代码实现了简单的生产消费模拟，使用了滚动面板，Jframe窗口和文本框，由button按钮控制进行简单的文本显示

//多线程采用Runnable 接口实现，缓冲区由int数组实现，整体代码很简单，功能也简单，相对来说实现的一般

**class** Desk{

**int** []buffer=**new** **int**[10];

**int** capacity=9;

**int** num=0;

**int** value=0;

**boolean** flag=**true**;

JTextArea textArea = **new** JTextArea();

**public** **void** setcapacity(**int** n) {

capacity=n;

}

**public** **synchronized** **void** p() {

**while**(num==capacity) {

**try** {

**this**.wait();

}**catch**(InterruptedException e) {

e.printStackTrace();

}

}

value=num;

buffer[value]=value;

//System.out.println("生产者"+Thread.currentThread().getName()+"生产了：" + value);

num++;

// 通知等待的消费者线程

**this**.notify();

}

**public** **synchronized** **void** c(){

**while**(num==0) {

**try** {

**this**.wait();

}**catch**(InterruptedException e) {

e.printStackTrace();

}

}

value= buffer[num];

//desk.buffer.add(num-1);

num--;

//System.out.println("消费者"+Thread.currentThread().getName()+"消费了：" + value);

// 通知等待的消费者线程

**this**.notify();

}

}

**class** Producer **implements** Runnable {

Desk desk;

**public** Producer(Desk desk) {

**this**.desk=desk;

}

@Override

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

**while**(desk.flag) {

**try** {

**synchronized** (desk) {

desk.p();

String a=("生产者"+Thread.*currentThread*().getName()+"生产了：" + desk.value);

desk.textArea.append(a+"\n");

}

// 模拟生产数据需要时间

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**class** Consumer **implements** Runnable {

Desk desk;

**public** Consumer(Desk desk) {

**this**.desk=desk;

}

@Override

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

**while** (desk.flag) {

**try** {

**synchronized** (desk) {

desk.c();

String a=("消费者"+Thread.*currentThread*().getName()+"消费了：" + desk.value);

desk.textArea.append(a+"\n");

// 缓冲区非空，消费数据并从队列中移除

//int value = buffer.removeFirst();

// 通知等待的生产者线程

//desk.notifyAll();

}

// 模拟消费数据需要时间

Thread.*sleep*(800);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**public** **class** the\_last **extends** JFrame {

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

JFrame frame=**new** JFrame();

frame.setTitle("生产消费模拟");//设置标题

frame.setSize(500,400);//设置宽高

frame.setLocationRelativeTo(**null**);//水平垂直居中 设置相对位置

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);//关闭窗口时退出程序 设置默认关闭操作

frame.setResizable(**true**);//窗口拖拽

frame.setVisible(**true**);//窗口可视化的

Desk desk=**new** Desk();

Producer producer = **new** Producer(desk);

Producer producer2 = **new** Producer(desk);

Consumer consumer = **new** Consumer(desk);

Consumer consumer2 = **new** Consumer(desk);

Thread producerThread1 = **new** Thread(producer);

Thread producerThread2 = **new** Thread(producer2);

Thread consumerThread3 = **new** Thread(consumer2);

Thread consumerThread4 = **new** Thread(consumer);

Thread consumerThread5 = **new** Thread(consumer);

// JPanel panel=new JPanel(new BorderLayout());

//frame.add(panel); //创建一个面板

//desk.textArea.setBounds(50, 50, 50, 50);//设置文本域大小

//frame.add(desk.textArea);

//frame.add(new JScrollPane(desk.textArea));

JScrollPane scrollPane = **new** JScrollPane(desk.textArea);// 创建滚动面板，将文本区域添加进去

frame.add(scrollPane);

scrollPane.add(desk.textArea);

scrollPane.setViewportView(desk.textArea);

//创建一个按钮,并且将这个按钮添加到面板上面

JButton startbutton=**new** JButton("开始");

JButton stopbutton =**new** JButton("结束");

frame.add(startbutton,BorderLayout.***NORTH***);

frame.add(stopbutton,BorderLayout.***SOUTH***);

startbutton.setBounds(50,280,60,60);

stopbutton.setBounds(350,280,60,60);

startbutton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

producerThread1.start();

producerThread2.start();

consumerThread3.start();

consumerThread4.start();

consumerThread5.start();

// JTextField textField= new JTextField();

// textField.setColumns(10);

}

});

stopbutton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

desk.flag=**false**;

}

});

}

}