序号\_\_\_\_\_

**华南理工大学广州学院**

**2019-2020学年度第1学期**

**课程名称：Java面向对象程序设计大作业**

**题　　目：关系和相关计算**

**专业班级：计算机科学与技术五班**

**年　　级：2018级**

**姓　　名：杨千旺**

**学　　号：**201810089182

|  |  |  |  |
| --- | --- | --- | --- |
| 一级指标 | 二级指标 | 分值 | 评分及成绩 |
| 得分 |
| 作品完成度  （占50%） | 1.工作量达标，功能点完整，独立完成 | 15 |  |
| 2.设计合理性，逻辑条理性 | 10 |  |
| 3.功能测试 | 10 |  |
| 4.数据处理水平 | 10 |  |
| 5.特色鲜明，有设计亮点 | 5 |  |
| 文档撰写（30%） | 1.文字描述规范，语言通顺，结构完整 | 10 |  |
| 2.思路描述清晰，功能测试结果完整 | 10 |  |
| 3.结果分析合理 | 10 |  |
| 答辩成绩 （占20%） | 1.能够对设计进行合理说明 | 10 |  |
| 2.能较好的回答答辩所提问题，解释合理清晰 | 10 |  |
| 合计（百分制） | | |  |
| 总 评 | □优 □良 □中 □及格 □不及格 | | 签名 |

目录

[一、 概述 4](#_Toc21715940)

[选题原因 4](#_Toc21715941)

[题目理解和分析 4](#_Toc21715942)

[二、程序概要设计 5](#_Toc21715943)

[总功能模块 6](#_Toc21715944)

[程序总流程图 6](#_Toc21715945)

[方法列表 6](#_Toc21715946)

[二、 程序详细设计 10](#_Toc21715947)

[1、 GUI设置 11](#_Toc21715948)

[2、 关于系统 11](#_Toc21715949)

[3、 关系运算 11](#_Toc21715950)

[交集运算 intersection(String formula1,String formula2) 12](#_Toc21715951)

[并集运算unionSet(String formula1,String formula2) 12](#_Toc21715952)

[逆运算opposite(String formula1,String formula2) 12](#_Toc21715953)

[复合运算recombination(String formula1,String formula2) 13](#_Toc21715954)

[4、映射判断JudgeMapImpl(String mapDomainX,String mapDomainY,String formula) 14](#_Toc21715955)

[5、关系与性质判断 14](#_Toc21715956)

[1、 自反性reflexxivity(String set,String formula) 15](#_Toc21715957)

[2、 反自反性antiReflexivity(String set,String formula) 16](#_Toc21715958)

[3、 对称性symmetry(String set,String formula) 16](#_Toc21715959)

[4、 反对称性性antiSymmetry(String set,String formula) 16](#_Toc21715960)

[5、 传递性transitivity(String set,String formula) 17](#_Toc21715961)

[等价关系equivalence(String set,String formula) 18](#_Toc21715962)

[偏序关系partialOrder(String set,String formula) 18](#_Toc21715963)

[6、闭包求取 18](#_Toc21715964)

[自反闭包getReflexive(String set ,String set) 18](#_Toc21715965)

[对称闭包getSymmetry(String set,String formula) 19](#_Toc21715966)

[传递闭包getTransitivity(String set,String formula) 19](#_Toc21715967)

[7、等价关系对集合进行划分 19](#_Toc21715968)

[代码实现divide(String set,String formula) 19](#_Toc21715969)

[8、偏序关系中极值求取 20](#_Toc21715970)

[代码实现getExtremum(String set,String formula) 20](#_Toc21715971)

[9、工具类util包 21](#_Toc21715972)

[JudgeInput 输入数据的判断 21](#_Toc21715973)

[StringCutting 处理输入的字符串 22](#_Toc21715974)

[程序测试 23](#_Toc21715975)

[小结 26](#_Toc21715976)

[参考文献 27](#_Toc21715977)

# 概述

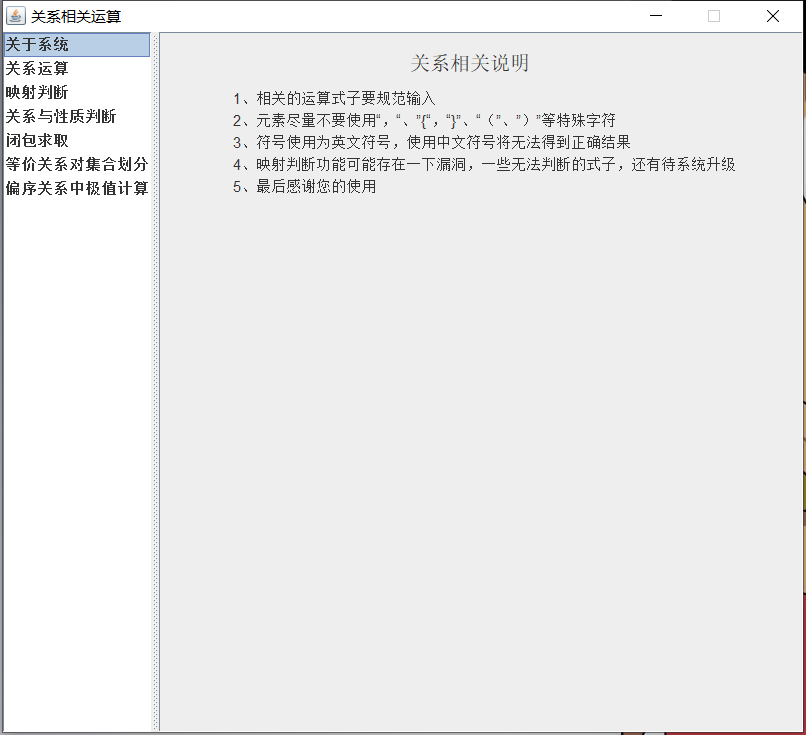
## 选题原因

选择“关系和相关计算”是有两方面的原因，一是为了复习离散数学，更深入理解离散的相关运算、映射关系、闭包求取、等价划分、偏序关系中极值计算；二是想锻炼自己的思维能力。

## 题目理解和分析

要求实现交集运算、并集运算、逆运算、复合运算；自反性、反自反性、对称性、反对称性、传递性五大性质的判断；单射、双射、满射三种映射的判断；偏序关系、等价关系、相容关系三种关系的判定；自反闭包、对称闭包、传递闭包三种闭包的求取；通过等价关系对集合进行划分；偏序关系中极值的计算。完成GUI应用程序的设计。

设计合适的类来描述关系、相容关系、等价关系、偏序关系、映射；要慎用静态方法。在程序设计中体现继承和多态，选择集合类或关系矩阵完成类的实现；偏序关系中极值求取可作为选做。

 基于题目要求，程序可分为关系运算模块、映射判断模块、性质于关系判断模块、闭包求取模块、通过偏序关系对集合进行划分模块、偏序关系中极值计算模块六大模块。为了方便开发还设有一个工具层的模块，用于对输入字符串的各种操作。序设计中以类作为划分，每个类中的方法实现该功能模块中的一个小功能，如关系运算模块中的交集运算，交集运算就是关系运算模块中的一个子功能，交集运算的功能实现封装在一个方法中。

基于本此系统开发的需要，程序开发环境为jdk8、编译器为eclipse的SE版。

# 二、程序概要设计

程序设计分为gui、logicalProcess、logicalProcessImpl、util三个包。gui用于完成图形界面的设计；logicalProcess为接口包，在写逻辑实现类前写出各类要完成的任务，在开发过程中起到总体设计的作用；logicalProcessImpl是logicalProcess的实现包，实现logicalProcess包中的各个接口，完成最要逻辑任务；util为工具包，里面的方法都为静态方法，有利于程序的频繁调用，主要实现了对输入的字符串检验是否复合要求、对关系式和集合进行切割，使其易于进行程序设计。

GUI图形界面的代码封装在GUI类中，该类实现了这个应用程序的所以界面显示，和各个Swing组件的监听。

Util工具包，包含了JudgeInput类、StringCutting类。JudgeInput类对输入的字符串进行检测，是否符合关系式或集合的形式、对关系式中的X和Y做判断是否属于定义域；StringCutting类对输入的关系式和集合进行切割，使其利于程序设计。

应用程序功能大致分为“关于系统”、“关系运算”、“映射判断”、“关系与性质判断”、“闭包求取”、“等价关系对集合划分”、“偏序关系中极值计算”六大部分。

关于系统，功能是展示系统的一些使用注意事项，使用户更便捷使用系统。代码实现是在GUI类中。

关系运算， 逻辑代码写在OperationImpl类中，实现了集合的交集运算、并集运算；关系的逆运算、复合运算的实现。

映射判断，逻辑代码写在JudgeMapImpl类中，实现了对关系的满射、双射、单射的判断。

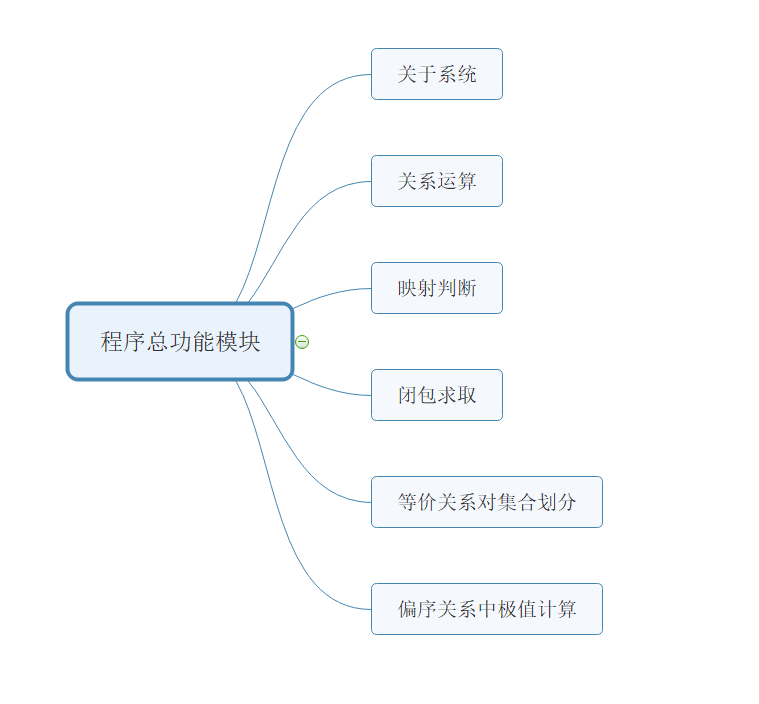
关系与性质的判断，逻辑代码分别写着JudgeRelationImpl类和JudgeNatureImpl类中，其中JudgeRelationImpl继承了JudgeNatureImpl。JudgeNatureImpl实现了五大性质的判断、JudgeRelationImpl实现了等价关系、等价关系、偏序关系的判断。

闭包求取，逻辑代码写在ClosureImpl类中，实现了自反闭包、对称闭包、传递闭包的求取。

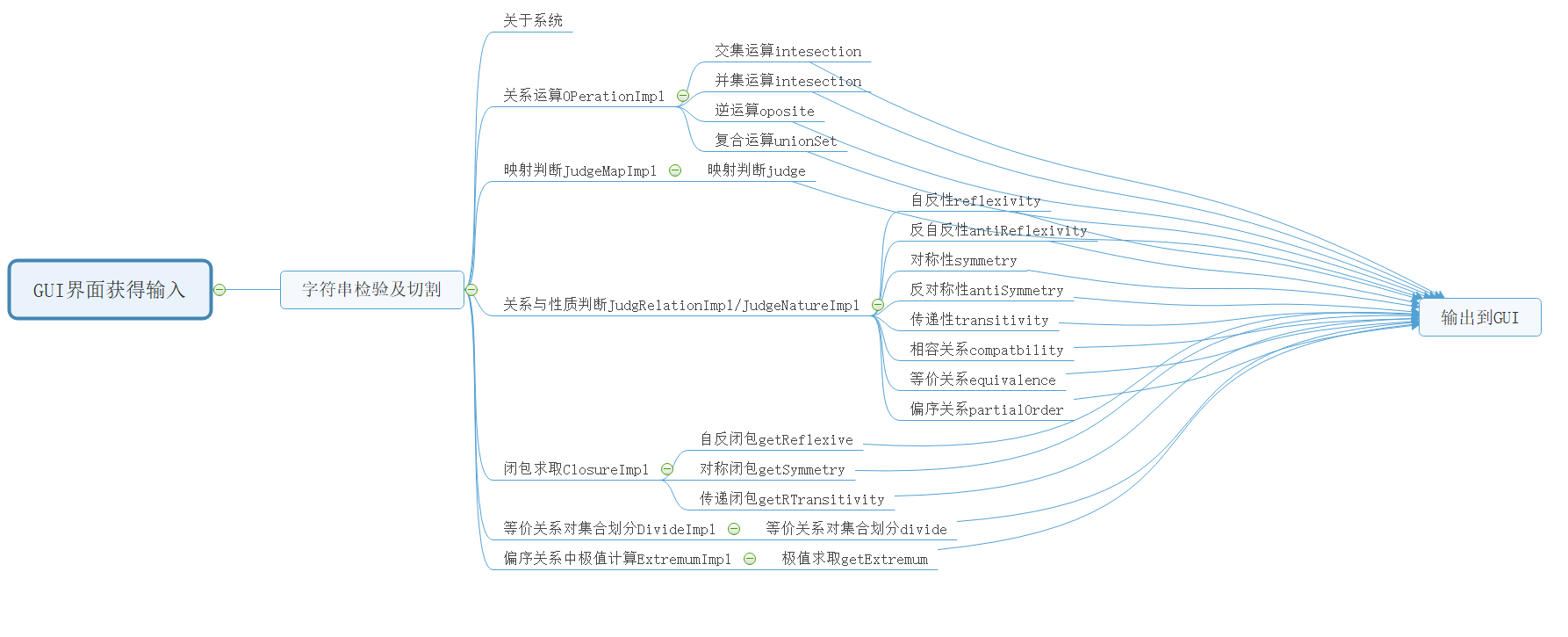
等价关系对集合划分，逻辑代码写在DivideImpl类中，DivideImpl类继承了JudgeNatureImpl类，实现了通过等价关系对集合进行划分功能。

偏序关系中极值求取，逻辑代码写在ExtrmumImpl类中，ExtrmumImpl类继承了JudgeNatureImpl类，实现偏序关系中极值求取。

## 总功能模块



## 程序总流程图



## 方法列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| gui | GUI | JPanel | getOperationPanel() | 获得关运算面板 |
| JPanel | getMapPanel() | 获得映射面板 |
| JPanel | getPanelAboutSystem() | 获得关于 系统面板 |
| JPanel | getNatureAndRelationPanel() | 获得性质与关系判断面板 |
| JPanel | getDividePanel() | 获得划分面板 |
| JPanel | getClosure() | 获得闭包面板 |
| JPanel | getExtPanel() | 获得极值面板 |
| void | actionPerformed(ActionEvent envet) | 动作监听 |
| void | valueChanged(ListSelectionEvent e) | 分割面板监听 |
| void | main(String[] arr) | Main函数 |
| logicalProcessImpl | ClosureImpl | StringBuilder | getReflexive(String set,String formula) | 自反闭包 |
| StringBuilder | getSymmetry(String set,String formula) | 获取对称闭包 |
| StringBuilder | getTransitivity(String set,String formula) | 获取传递闭包 |
| DivideImpl | StringBuilder | divide(String set,String formula) | 偏序关系对集合进行划分 |
| ExtremumImpl | String | getExtremum(String set,String formula) | 偏序关系极值求取 |
| JudgeMapImpl | String | judge(String mapDomainX,String mapDomainY,String mapFormula) | 判断关系式的映射关系 |
| JudgeNatureImpl | boolean | reflexivity(String set,String formula) | 判断自反性 |
| boolean | antiReflexivity(String set,String formula) | 判断反自反性 |
| boolean | symmetry(String set,String formula) | 判断对称性 |
| boolean | antiSymmetry(String set,String formula) | 判断反对称性 |
| boolean | transitivity(String set,String formula) | 判断传递性 |
| JudgeRelationImpl | String | compatbility(String set,String formula) | 相容关系 |
| String | equivalence | 等价关系 |
| String | partialOrder | 偏序关系 |
| OperationImpl | StringBuffer | intesection (String formula1,String formula2) | 交集运算 |
| StringBuffer | unionSet (String formula1,String formula2) | 并集运算 |
| StringBuffer | opposite (String formula1,String formula2) | 逆运算 |
| StringBuffer | recombination (String formula1,String formula2) | 复合运算 |
| util | JudgeInput | boolean | judgeFormula(String formula) | 判断关系形式 |
| boolean | judgeSet(String set) | 判断集合形式 |
| boolean | IncludeSet(String[]set,String[]formula) | 判断关系式中是否存在X或Y不在定义中 |
| StringCutting | String[] | setoNum(String formula) | 把集合切成元素 |
| String[] | betweenToNun(String formula) | 把关系对切成元素 |
| String[] | cutRelation(String formula) | 把关系 式切成关系对 |

# 程序详细设计

## GUI设置

**private** String[] function = {"关于系统","关系运算","映射判断","关系与性质判断","闭包求取","等价关系对集合划分","偏序关系中极值计算"};

list = **new** JList(function); list.setSelectionMode(ListSelectionModel.***SINGLE\_SELECTION***);

//list.setSelectedIndex(0);

list.addListSelectionListener(**this**);

list.setFont(**new** Font("粗体",Font.***BOLD***,15));

**------------------------------------------------------------**

**public** **void** valueChanged(ListSelectionEvent e) {

JList list = (JList) e.getSource();

splitPane.setRightComponent(panelList.get(list.getSelectedIndex()));

}

------------------------------------------------------------

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

GUI图形界面是采用分割面板把功能选择和具体功能实现分割开来。各个功能模块设有自己独有的面板在分割板的右侧。ValueChanged方法为分割面板的监听器。actionPerformed方法为动作监听器，监听界面按钮的动作。

每个功能模块以方法为单位，获取它的面板。

## 关于系统

使文本格式化输出，获取JTextPane面板，用于格式输出文本，StyledDtocument 把属性集、文本内容结合起来，SimpleAttributeSet设置文本属性。

## 关系运算

关系运算模块包含四个子功能，分别是集合的交集运算、并集运算，关系的逆运算、复合运算。每个子功能都包含对关系式或集合的检验功能，检查是否符合关系式或集合的要求。

JudgeInput为util工具包中的一个工具类，其方法judgeSet、judgeFormula和IncludeSet分别用于检验集合、关系式和验证关系式中是否存在X或Y不在定义中，此方法不仅用于关系运算模块，在下面的各个功能模块中都有应用。

StringCuttiong是util包中的一个工具类，用于对切割输入的字符串和对字符串数组进行查重。setToNum用于把集合切成一个个元素；cutRelation用于切割关系式，把关系式切割成关系对；betwenToNum用于把关系对切割成元素；repetition用于字符数据查重，返回值为int型，上面方法的具体实现会在下面讲解util包时一一分说。

以上的切割字符串、字符数组查重和集合或关系式的检验在其他功能模块中都有存在，在以下的解释中就不一一说明了。

### 交集运算 intersection(String formula1,String formula2)

子功能交集运算，实现了两集合取交集的功能。 难点在于取到两个关系式中重复的元素。我的解决办法时利用Set的特性，去重。

交集功能的实现依赖于HashSet类的使用，Set容器的特点是不能保存重复数据，当数据重复是Set的add方法便会返回false，而且保存的是无序的数。此算法利用了不能保存重复数据这一特点，建立一个HashSet类，把第一个集合全都装在里面，利用for循环遍历另一个集合，用add方法加入hashSet类，单返回false是证明此元素在集合一中已存在，所以为交集元素，把它加入到重复结果的StringBuffer类中。

特殊样例;

//当集合都为空时，返回{}

**if**(formula1.equals("{}")&&formula2.equals("{}")) {

result.append("{}");

**return** result;}

### 并集运算unionSet(String formula1,String formula2)

子功能并集运算，实现两给集合的并集运算 。难点在关系式一的X值与关系式二的Y值怎么联系、复合结果有重复。我的解决办法是采用两层for循环，使关系式一和关系式二产生联系；用Set的特点取重。

这个与交集运算一样利用了Set的特点，不同的是在关系式二做循环遍历时，能被Set集合类add的元素加入到结果字符串中。

### 逆运算opposite(String formula1,String formula2)

逆运算功能，实现了关系式逆运算。逆运算只需要一条关系式，但输入框关系输入框有两个、怎么把X值和Y值互换位置。我解决办法是，输入框问题在事件监听中做处理，一开始将两个参数设计一个值，当在输入框有输入时赋值给参数，到逻辑层做判断，是否有被重新赋值，用于区分那个参数需要处理。X和Y互换位置，先把关系式切割成元素，再互调位置

因为逆运算和集合运算在同一个功能模块，所以输入框有两给，但逆运算只需要一条关系式就可以完成，所以在GUI类中的按钮监听器就设定了输入框可以输入一条或两条的设代码的设计思路是：在监听器中先给formualA和formualB赋值为“”，用个体getText()获取字符串分别赋值给formualA和formualB，在finally中判断是否两哥都为空，如果是输出“输入存在空值！”，把formualA和formualB传入 opposite方法。如下面代码：

**-----------GUI监听器中---------------------------------------**

**else** **if**(bt == opposite) {//逆运算

String formualA = **new** String("");

String formualB = **new** String("");

**try** {

formualA=operationA.getText();

}**catch**(Exception e) {}

**try** {

formualB=operationB.getText();

}**catch**(Exception e) {}

**finally** {

**if**(formualA.equals("") && formualB.equals("")) {

operationResult.setText("输入存在空值！");}

**else** {

strBuf = operation.opposite(formualA, formualB);

operationResult.setText(strBuf.toString());}}}

---------------OperationImpl类中----------------------------

**if**(formula1 != **null** && !formula1.equals(""))

完成逆运算的设计思路是：把关系式切成关系对数组，然后对关系对数组进行遍历，把每个关系对以逗号为分割，切成两部分，再将两部分互调，放到结果集中。

### 复合运算recombination(String formula1,String formula2)

复合运算，实现了两个关系式的符合运算。

程序设计思路是：双重循环外面以关系式一循环，切割关系对中的后半Y与内层循环的关系二的关系对的X部分做匹配，如果成功，把关系一的那个关系对的X部分连接关系二中的关系对Y部分，再add进Set集判重，把不重复的加到结果集中。

## 4、映射判断JudgeMapImpl(String mapDomainX,String mapDomainY,String formula)

映射判断功能，实现了判断映射关系的功能。难点在于映射种类多样，难找到其中的规律。我的解决办法是应用定义域、值域、关系式的长度，在其中找关系。

映射判断功能的实现思路是：判断定义域的长度和关系式的长度，若不相同，表示映射关系不成立。接着判断定义域的X是否被重复利用，若重复利用，映射关系不成立。

判断Y值是否全部来自于输入的值域。

//检验Y是否全都在Y域中

Set<String> setY = **new** HashSet<String>();

**for**(i=0;i<lengthF;i++) {

String[] strY = StringCutting.*betweenToNun*(formula[i]);

//System.out.println("strY[1]="+strY[1]);

**if**(!setY1.contains(strY[1])) {

**return** "关系式中存在Y值不在值域中";

}

setY.add(strY[1]);

}

利用长度判断映射类型，但还有一部分无法判断，仍需努力。如下面代码。

**if**(setY.size() < lengthY ) {

**return** "单射";

}

**else** **if**(setY.size() == lengthY && lengthY == lengthX) {

**return** "双射";

}

**else** **if**(setY.size() == lengthY && lengthY < lengthX) {

**return** "满射";

}

**else** {

**return** "因技术有限，暂无法判断此式！";

}

## 5、关系与性质判断

关系与性质判断功能，实现了对关系式进行关系类型判断和性质判断的功能。分别写了两个类实现。输入形式是否正确在事件监听的时候完成。

### 自反性reflexxivity(String set,String formula)

自反性功能，实现了对关系式自反性的判断。自反的定义是，设R是集合A中的一个

关系，如果对于集合A中每一个啊，均有（a，a）属于关系R。难点是怎么判断集合中的每个元素在关系式中都存在这样的关系。解决办法：设双层循环，以集合为外层，遍历关系式，检验每个元素是否在关系式中均有该关系。

设judge = false 把集合中的元素送到内层循环，如果检验到存在（a，a）关系，把judge赋为true，结束当前循环。若元素未有（a，a）关系，程序跳进if语句返回false。如下面代码所示：

//循环setX查看是否每个X在pair上都有x,x

**for**(i=0;i<lengthX;i++) {

judge = **false**;

**for**(j=0;j<lengthF;j++) {

String[] strF = StringCutting.*betweenToNun*(pair[j]);

**if**(setX[i].equals(strF[0]) && setX[i].equals(strF[1])){

judge = **true**;

**break**;}}

**if**(!judge) {**return** **false**;}}

在GUI的监听器中获得返回值输出结果。但还存在一些特殊样例样单独做处理。

如下面代码所示：

judge = judgeNature.antiReflexivity(setText,formulaText);

**if**(judge){

judgeResult.setText("反自反性成立！");}

**else** {

judgeResult.setText("反自反性不成立！");}

特殊样例：

//当非空集的空关系

**if**( formula.equals("{}") && !set.equals("{}")) {

**return** **false**;}

//当空集的空关系

**if**(formula.equals("{}") && set.equals("{}")) {

**return** **true**;}

### 反自反性antiReflexivity(String set,String formula)

反自反性功能，实现了判断关系式反自反性的功能。 定义刚好与自反性想反。难点是判断每个元素在关系式中都不存在（a，a）。解决方法是循环遍历判是否每个元素都不存在（a，a）的关系。

当存在返回false，在GUI类的事件监听器中做判断输出结果。

存在一些特殊样例： 关系式为空或关系式为空且集合也为空

**if**(formula.equals("{}")||set.equals("{}")&&formula.equals("{}")) {

**return** **true**;

}

### 对称性symmetry(String set,String formula)

对称性功能，实现了判断关系式对称性的功能。问题难点是:把关系式转化成关系二维数组。解决办法是：利用双层for循环外出时关系式，内层是集合。遍历到获得二维关系数组

把遍历到的元素下标作为二维关系数组的下标，得到二维关系数组。

//获得关系二维表

**for**(i=0;i<lengthF;i++) {

String[] strF = StringCutting.*betweenToNun*(pair[i]);

**for**(j=0;j<lengthX;j++) {

**if**(strF[0].equals(setX[j])) {

row = j;}

**if**(strF[1].equals(setX[j])) {

list = j;}}

array[row][list] = 1;}

### 反对称性性antiSymmetry(String set,String formula)

反对称性功能，实现了判断关系式是否具备反对称性。功能实现和上面的对称性差不多。先获得关系二维数组，在对数组进行遍历，到遍历到数组元素值为1且下表不相等时，下标交换，判断是否是一，是则返回false，否返回true。返回值送到GUI的事件监听器类中判断输出。

特殊样例：

//当关系式为空时或关系式为空且关系式也为空，返回true

**if**(formula.equals("{}")||set.equals("{}")&&formula.equals("{}")) {

**return** **true**;

}

### 传递性transitivity(String set,String formula)

传递性功能，实现了判断关系式是否具有传递性的性质。功能实现的难点是把关系式转成二维关系数组，利用数组求取闭包。难点解决：把关系式转化成二维关系数组，方法与前面一样，不过多了把数组再复制多一份的步骤。利用一个数组做warshall算法获得传递闭包关系二维数组。传递闭包关系二维数组与另一个数组比较是否相等。不相等返回false，否在返回true。

大致过程：获得关系二维数组，求取闭包，闭包与关系二维数组比较，二维数组相等，返回true，不相等返回false。

//Warshall算法，求闭包；

**for**(j=0;j<lengthX;j++) {

**for**(i=0;i<lengthX;i++) {

**if**(array[i][j] !=0) {

**for**(k=0;k<lengthX;k++) {

array[i][k] = array[i][k]|array[j][k]; }}}}

//循环判断数组array与array1是否相等，不相等false

**for**(i=0;i<lengthX;i++) {

**for**(j=0;j<lengthX;j++) {

**if**(array[i][j] != array1[i][j]) {//闭包与副本比较

**return** **false**; }}}**return** **true**;

特殊样例：

//当关系为空或集合为空且关系为空时成立

**if**(formula.equals("{}")||set.equals("{}")&&formula.equals("{}")) {

**return** **true**;

}

#### 相容关系compatbility(String set,String formula)

相容关系功能，实现了判断关系式是否具有相容关系的功能。相容性具有自反性、对称性。判断这里个性质可调任reflexivity方法和symmetry方法实现。

### 等价关系equivalence(String set,String formula)

等价关系功能，实现了判断关系式是否具有等价关系的功能。等价关系具备自反性、对称性、传递性。实现该程序调用了reflexivity、symmetry、transitivity方法。

### 偏序关系partialOrder(String set,String formula)

偏序关系功能，实现了判断关系式是否具备偏序关系的功能。偏序关系具备自反性、反对称性、传递性。实现该程序调用了reflexiviy、antiSymmetry、transitivity方法。

## 6、闭包求取

闭包求取功能模块实现在ClosureImpl类中，该类实现了三种闭包的求取，分别是自反闭包、对称闭包、传递闭包。这三种闭包的求取每个分别用一个方法封装。

### 自反闭包getReflexive(String set ,String set)

自反闭包功能，实现了从输入的关系式转换成满足自反性的性质。实现的步骤是：把关系式转换成二维关系数组，把对角线上的元素都赋值为1，然后再把二维关系数组转化成关系式，与StringBuilder的形式返回到GUI的事件监听方法，输出结果。

特殊样例：

//当空集空关系返回空集

**if**(set.equals("{}") && formula.equals("{}")) {

result.append("{}");

**return** result;}

//当非空集的空关系

**if**(!set.equals("{}") && formula.equals("{}")) {

String[] setX = StringCutting.*setToNum*(set);

**int** lengthX = StringCutting.*repetition*(setX);

result.append("{");

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

result.append("("+setX[i]+","+setX[i]+"),");}

**int** index = result.length() - 1;

result.deleteCharAt(index);

result.append("}");

**return** result;}

### 对称闭包getSymmetry(String set,String formula)

对称闭包功能，实现了从关系式获取关系式的对称闭包。实现的步骤是：从关系式获取二维关系数组，遍历二维关系数组，当二维关系数组元素为1时，把下表相反的元素赋值为1直到遍历结束。

特殊样例：

//当空关系返回空集

**if**(formula.equals("{}")) {

result.append("{}");

**return** result;

}

### 传递闭包getTransitivity(String set,String formula)

传递闭包功能，实现了从输入的关系式获取关系式的传递闭包。实现步骤是：从关系式获取二维关系数组，对二维关系数组使用warshall算法获得该关系式的传递闭包二维关系数组，然后再把二维关系数组转化成关系式。

特殊样例：

//当空关系返回空集

**if**(formula.equals("{}")) {

result.append("{}");

**return** result;}

## 7、等价关系对集合进行划分

等价关系对集合的划分，通过从GUI界面传进的集合和关系式，通过等价关系对集合进行划分。划分后在结果框输出，每个子集占输出结果的一行。

### 代码实现divide(String set,String formula)

实现了以偏序关系对集合进行划分的功能。难点是通过对关系式对集合进行划分。解决办法是：找到划分的规律，从规律入手。根据偏序关系进行划分，划分出来的每个子集与自己做笛卡尔乘的并集又可得到偏序关系式，根据笛卡尔乘的规律，子集的每个元素都与其他的元素有（X，Y）关系。从上规律，使用双层循环，集合作为外循环，遍历关系式，若符合条件的元素记录起来，到最后输出。

ArrayList<String> listX = **new** ArrayList<String>();

ArrayList<StringBuilder> listR = **new** ArrayList<StringBuilder>();

**for**(i=0;i<lengthX;i++) {

listX.add(setX[i]);}

// 解释

//{1,2,3,4}->{1,2,3}、{4} ->{1,2,3}X{1,2,3}U{4}X{4}

//{(1,1),(1,2),(1,3)(1,4),(2,1),(2,2),(2,3),(2,4),(4,1)~

//做笛卡尔乘，每个子集的元素都与同一子集的元素有关系，以第一个元素作为X为遍历关系式，若为（X，Y）则Y元素为同一子集。

//一个划分集合的任一个元素均与同个划分集合能的元素有关系，以listR为参考循环，当X在serF中出现，把Y记录出来，并在listR中删除Y；

**for**(i=0;i<listX.size();i++) {

StringBuilder strBui = **new** StringBuilder(listX.get(i)+",");

**for**(j=0;j<lengthF;j++) {

String[] strF = StringCutting.*betweenToNun*(pair[j]);

**if**(listX.get(i).equals(strF[0]) && !listX.get(i).equals(strF[1])) {

strBui.append(strF[1]+",");

listX.remove(strF[1]); }}listR.add(strBui);}

特殊样例：

//当关系为空时，划分为集合本身

**if**(formula.equals("{}")) {

result.append(set);

**return** result;}

## 8、偏序关系中极值求取

求取偏序关系的极值，GUI图形界面接收集合和偏序关系，传到ExtremumImpl类中进行处理，返回结果在GUI图形界面的结果输出框显示。

### 代码实现getExtremum(String set,String formula)

实现求取偏序关系中的极值。难点是怎么求取极值、极值的规律。解决办法是：通过观察偏序关系的哈希图，从中发现极大值的元素没有被其他的元素盖住，极小值的元素没有盖住其他元素。根据这样规律转换成关系式的形式，如关系式形式{（X，Y）}，极大值不会出现在X的位置、极小值不会出现在Y的位置。根据以上规律设置程序。

设计双层循环，使集合元素作为外层循环遍历关系式，判断该元素出现的位置，如果遍历完关系式，元素都没有出现在X位置，那么元素便是极大值；反之没有出现过在Y位置，那么该元素便是极大值。判断利用定义两个Boolean型的变量，分别是max、min，均赋值为true，当元素出现在X位置，那么max被赋值为false，当元素出现在Y位置，那么min被赋值为true。把min和max放到if语句中判断该元素是否加入到结果集、加入到哪个结果集。

特殊样例：当关系式为空时，集合的所有元素既是极大值也是极小值。输出结果最大值和最小值都是集合所有元素

StringBuilder strMax = **new** StringBuilder("极大值：");

StringBuilder strMin = **new** StringBuilder("极小值：");

**for**(i=0;i<lengthX;i++) {max = **true**;min = **true**;

**for**(j=0;j<lengthF;j++) {

String[] strF = StringCutting.*betweenToNun*(pair[j]);

**if**(strF[0].equals(setX[i]) && !strF[0].equals(strF[1])) {

min = **false**;}**if**(strF[1].equals(setX[i]) && !strF[1].equals(strF[0])) {max = **false**;}}

**if**(min) {strMin.append(setX[i]+",");}

**if**(max) {strMax.append(setX[i]+",");} }

## 9、工具类util包

Util包一共写了两个类，分别是JudgeInput和StringCutting。JudgeInput用于判断输入的形式是否正确并符合要求，StringCutting用于对输入的数据做相应的处理，使它能符合各种运算。工具包里的方法都是静态方法，因为在里面的方法在多次都需要用到，如果不是静态方法需要反复new出一个新的对象，不便于使用。

### JudgeInput 输入数据的判断

JudgeInput类包含三个方法，分别实现了判断关系式形式是否正确、集合形式是否正确、关系式中的元素是否均来自集合。

#### Judge Formula(String formula)

作用是使关系式符合如{(1,2),(2,3),(3,4),(4,1),(5,1),(6,1)}这样的形式。设计思路：先做基本判断字符串的两边是否符合要求，符合要求后进入双层循环，循环从下标为2开始到字符串长度-1处截止，当循环遇到‘（’，判断它前两字符是否符合要求，若符合要求进入第二层循环，判断括号内是否只有一个‘，’并符合要求若符合和要求重新设置第二次循环的开始位置为i+2。

但当到最后一个关系对时没有‘（’符号，所有要另作处理。设置当i=字符串长度-2时进入处理使用，处理办法和上面第二层循环一样，从写一次。

还要注意如{(1,2),(2,3),(3,4))}这种情况，使用到第一层for循环后面加上判断如：

**else** **if**(formula.charAt(i) == ')' && formula.charAt(i-1) == ')') {// {(1,2),(2,3),(3,4))}

**return** **false**;

}

#### judgeSet(String set)

作用判断集合形式是否正确。设计思路：判断集合两边是否为‘{’、‘}’、第二个和倒二个字符不可以是‘，’，设置循环从下标为2开始遍历字符串到下标为字符串长度减二结束判断是否有相连的两个‘，’存在。

#### IncludeSet(String[] set,String[] formula)

作用是判断关系式中的元素是否都来自集合。设计思路是：利用Set集合类的特性，先把集合元素都add进Set集合类，设置循环遍历关系对，把关系对的X和Y都add进Set，如果返回true则证明关系式里存在元素不来自集合中。

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

setX.add(set[i]);}

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

String[] str = StringCutting.*betweenToNun*(formula[i]);

**if**(setX.add(str[1]) || setX.add(str[0])) {

**return** **false**;}}

### StringCutting 处理输入的字符串

StringCutting类中包含四个静态方法，作用是切割字符串和去重的作用。切割字符串主要使用了一些正则表达式结合String的split方法，去重主要使用了Set的特点。

# 程序测试

测试样例：

//交集:

{1,21,3,4}

{1,21}

结果：{1,21}

//并集

{1,21,3,4}

{1,21,8}

结果：{1,21,3,4,8}

//逆关系运算：

{(1,2),(1,3)}

结果：{(2,1),(3,1)}

//复合运算

{(1,2),(2,3)}

{(2,3),(2,4),(3,4)}

{(1,3),(1,4),(2,4)}

//单射：

集合：{1,2,3,4}

集合：{1,2,3,4,5}

关系式：{(1,2),(2,3),(3,4),(4,5)}

//满射

集合：{1,2,3,4,5}

集合：{1,2,3,4}

关系式：{(1,2),(2,3),(3,4),(4,1),(5,1)}

//双射

集合：{1,2,3,4,5}

集合：{6,7,8,9,10}

关系式：{(1,6)(2,7)(3,8)(4,9)(5,10)}

//双射

集合：{1,2,3,4,5}

集合：{7,8,9,6,10}

关系式：{(1,6),(2,7),(3,8),(4,9),(5,10)}

//错误映射

集合：{1,2,3,4,5}

集合：{1,2,3,4}

关系式：{(1,2),(2,3),(3,4),(4,1),(5,1),(6,1)}

//既是对称也是返对称

集合：{a,b,d}

关系式：{(a,a),(b,b),(d,d)}

等价关系划分例子:

集合：{1,2,3,4,5}

关系：{(1,1),(1,2),(2,1),(2,2),(3,3),(4,4),(4,5),(5,4),(5,5)}

结果：{1,2}

{3}

{4,5}

集合：{1,2,3}

关系：{(1,2),(2,1),(1,1),(2,2),(3,3)}

结果：{1,2}

{3}

集合：{4,2,3,1}

关系：{(1,1),(2,2),(3,3),(4,4),(1,2),(2,3),(1,3),(2,1),(3,2),(3,1)}

结果：{4}

{2,3,1}

偏序关系极值:

集合：{2,3,4,5,6,7,8,9,10,12}

关系式：

{(2,4),(2,6),(2,8),(2,10),(2,12),(3,6),(3,9),(3,12),(4,8),(4,12),(5,10),(6,12),(2,2),(3,3),(4,4),(5,5),(6,6),(7,7),(8,8),(9,9),(10,10),(12,12)}

极大值：2,3,5,7

极小值：7,8,9,10,12

闭包求取

集合：{4,2,3,1}

关系：{(1,1),(2,2),(3,3),(4,4),(1,2),(2,3),(1,3),(2,1),(3,2),(3,1)}

结果：

自反：{(4,4),(2,2),(2,3),(2,1),(3,2),(3,3),(3,1),(1,2),(1,3),(1,1)}

对称：{(4,4),(2,2),(2,3),(2,1),(3,2),(3,3),(3,1),(1,2),(1,3),(1,1)}

传递：{(4,4),(2,2),(2,3),(2,1),(3,2),(3,3),(3,1),(1,2),(1,3),(1,1)}

集合：{1,2,3,4,5}

关系：{(1,2),(2,3),(3,4),(4,1),(5,1)}

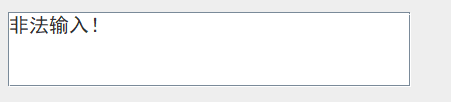
结果：

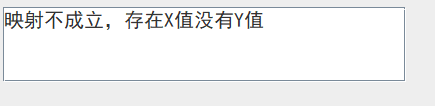
自反：{(1,1),(1,2),(2,2),(2,3),(3,3),(3,4),(4,1),(4,4),(5,1),(5,5)}

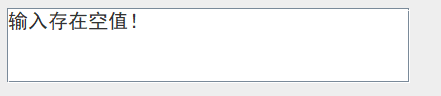
对称：{(1,2),(1,4),(1,5),(2,3),(3,4),(4,1),(5,1)}

传递：{(1,1),(1,2),(1,3),(1,4),(2,1),(2,2),(2,3),(2,4),(3,1),(3,2),(3,3),(3,4),(4,1),(4,2),(4,3),(4,4),(5,1),(5,2),(5,3),(5,4)}

当错误输入时：







# 小结

虽然题目需求都满足了，但还是存在着不少的问题，需要在以后的学习中改进自己，做出更好的程序。此次的不足主要在于代码的重复率比较高、功能实现大多都用了双层循环，时间复杂度比较高、程序中许多去重都用了Set类增加了程序的空间开销、页面布局也欠妥、在开发前没用充分做好需求分析和总体的代码实现规划。优点在于功能完成度高，题目的所有要求基本完成，附加题也都做了，在容错方面也做了比较多的功夫。

在程序开发过程中遇到的困难还是蛮多的，如把关系式转换成二维关系数组、极值求取找中找极值的规律、映射的判断规律、等价关系划分集合。这些都是以前没有尝试过的东西，需要我花费大量的时间和精力，不过结果还是好的，我基本都把问题解决了。

在这次的java大作业中我的最大收获是知道数据达到一定的数量便可在数据身上找到它的规律所在，如在做等价关系划分集合时，我在看它的定理时知道它是什么，但不知要怎么实现，当我写了多条通过等价关系划分集合的例子时找到了它的规律；偏序关系中极值求取也时通过这要的方法找到规律实现功能的。所以有些时候不要局限于定理所描述的，在理解定理时多收集一些数据可能就会在其中找到规律。还有就是让我意识到在制作一个系统前，要多花点时间对实现流程多加分析，否则当项目完成到一半的时候发现之前写的一些代码现在需要用了，但难以实现调用，这就是在做之前缺乏规划所带来的结果。

# 参考文献

1. 温武 钟沃坚.离散数学及应用.第二版.出版社：华南理工大学出版社
2. 王薇.Java程序设计与实践教程.出版社：清华大学出版社