



哈爾濱工業大學  
HARBIN INSTITUTE OF TECHNOLOGY

**2019 年春季学期**  
**计算机学院《软件构造》课程**

**Lab 3 实验报告**

姓名	张景润
学号	1172510217
班号	1703002
电子邮件	2584363094@qq.com
手机号码	18530272728

## 目录

1 实验目标概述	1
2 实验环境配置	1
3 实验过程	2
3.1 待开发的三个应用场景	2
3.2 基于语法的图数据输入	2
3.2.1 行星系统正则表达式	3
3.2.2 原子系统正则表达式	3
3.2.3 社交系统正则表达式	3
3.3 面向复用的设计: <code>CircularOrbit&lt;L,E&gt;</code>	3
3.4 面向复用的设计: <code>Track</code>	4
3.5 面向复用的设计: <code>L</code>	4
3.5.1 <code>Stellar</code>	4
3.5.2 <code>CentralObject</code>	5
3.6 面向复用的设计: <code>PhysicalObject</code>	5
3.6.1 <code>PhysicalObject</code> 类 (父类)	5
3.6.2 <code>StellarSystemObject</code> 类 (子类)	5
3.6.3 <code>Friend</code> 类 (子类)	6
3.7 可复用 API 设计	7
3.8 设计模式应用	7
3.8.1 迭代器设计模式	7
3.8.2 备忘录设计模式	8
3.9 应用设计开发与 GUI	9
3.9.1 <code>StellarSystem</code>	11
3.9.2 <code>AtomStructure</code>	11
3.9.3 <code>SocialNetworkCircle</code>	12
3.10 application 类与我的 GUI	13
3.10.1 App 类-用户类	13
3.10.2 <code>StellarSystemGUI</code> 类	13
3.10.3 <code>AtomStructureGUI</code> 类	15

3.10.4 SocialNetworkSystemGUI 类 .....	16
3.11 Difference 类 .....	17
3.12 新变化 .....	18
3.12.1 StellarSystem .....	18
3.12.2 AtomStructure .....	20
3.12.3 SocialNetworkCircle .....	23
3.13 Git 仓库结构 .....	23
4 实验进度记录 .....	24
5 实验过程中遇到的困难与解决途径 .....	25
6 实验过程中收获的经验、教训、感想 .....	25
6.1 实验过程中收获的经验教训 .....	25
6.2 针对以下方面的感受 .....	25

## 1 实验目标概述

本次实验覆盖课程第 3、5、6 章的内容，目标是编写具有可复用性和可维护性的软件，主要使用以下软件构造技术：

子类型、泛型、多态、重写、重载

继承、代理、组合

常见的 OO 设计模式

语法驱动的编程、正则表达式

基于状态的编程

API 设计、API 复用

本次实验给定了五个具体应用（径赛方案编排、太阳系行星模拟、原子结构可视化、个人移动 App 生态系统、个人社交系统），学生不是直接针对五个应用分别编程实现，而是通过 ADT 和泛型等抽象技术，开发一套可复用的 ADT 及其实现，充分考虑这些应用之间的相似性和差异性，使 ADT 有更大程度的复用（可复用性）和更容易面向各种变化（可维护性）。

## 2 实验环境配置

- 在本地建立 git 仓库

```
echo "# Lab3_1172510217" >> README.md
```

```
git init
```

```
git add README.md
```

```
git commit -m "1-commit"
```

```
git status
```

```
git remote add origin https://github.com/ComputerScienceHIT/Lab3-1172510217.git
```

```
git push -u origin master
```

```
git status
```

- 张景润实验 URL 地址

<https://github.com/ComputerScienceHIT/Lab3-1172510217>

## 3 实验过程

### 3.1 待开发的三个应用场景

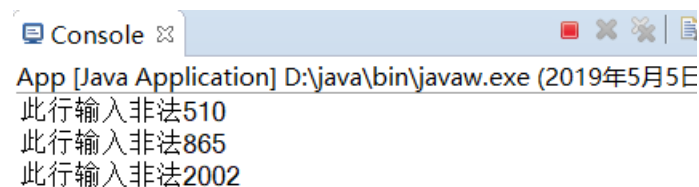
- Stellar System
- AtomStructure System
- Social Network System

#### 分析异同

- 不同点：行星系统中心物体存在，且参数与其他三个不同；轨道上的物体参数分别不相同；原子系统可以跃迁、而社交系统有关系；
- 相同点：均有轨道，物体存在于轨道上；轨道均可用一个 `double` 参数表示；许多功能类似，比如增删轨道、增删物体、获取轨道数目、计算熵值、得到差异

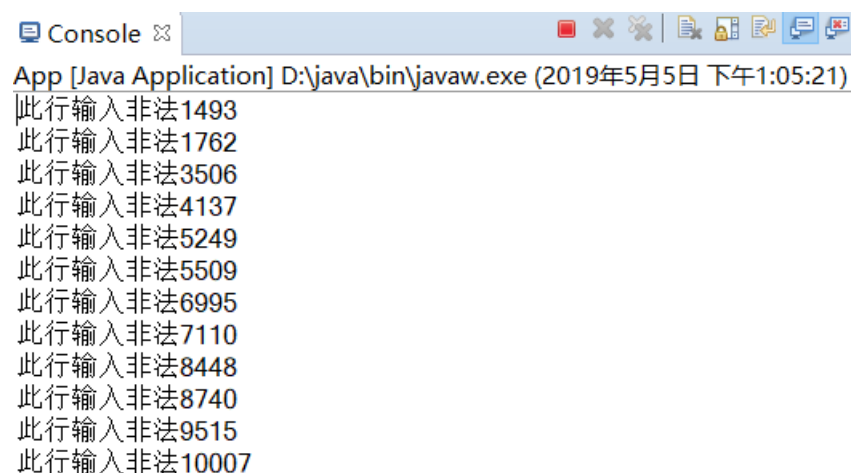
### 3.2 基于语法的图数据输入

设计的正则表示式可以匹配正确输入，并终端标识非法输入行，截图如下。  
同时正则表达式设计的正确性可由 GUI 显示正确来验证



```
App [Java Application] D:\java\bin\javaw.exe (2019年5月5日)
此行输入非法510
此行输入非法865
此行输入非法2002
```

上面是行星系统中文件的非法数据行



```
App [Java Application] D:\java\bin\javaw.exe (2019年5月5日 下午1:05:21)
此行输入非法1493
此行输入非法1762
此行输入非法3506
此行输入非法4137
此行输入非法5249
此行输入非法5509
此行输入非法6995
此行输入非法7110
此行输入非法8448
此行输入非法8740
此行输入非法9515
此行输入非法10007
```

上面是行星系统大文件的非法数据行

### 3.2.1 行星系统正则表达式

- 运用 Pattern 类以及 Matcher 类的方法；
- 设计正则表达式如下（十分全面！）

```
1, Pattern pattern1 = Pattern.compile("([a-z]|[A-Z])([a-z]|[A-Z]|[0-9])*(CW)|(CCW)"); // 名称的格式
2, Pattern pattern2 = Pattern.compile("([1-9][0-9]{1,3}([.][0-9]+)?|1-9([.][0-9]+)?([Ee][+-]?[0-9]+)?|[0])"); // 数值的格式
```

->设计提取数值的正则表达式格式时，充分考虑了科学记数法的要求：大于 10000 的数字按科学记数法表示。并将非法数据行标出。

### 3.2.2 原子系统正则表达式

- 运用 Pattern 类以及 Matcher 类的方法；
- 设计正则表达式如下（十分全面！）

```
1, String regex1 = "[ ]*[A-Z][a-z]?[ ]*"; // 匹配原子核
2, String regex2 = "[ ]*[0-9]+[ ]*"; // 匹配电子数
```

### 3.2.3 社交系统正则表达式

- 运用 Pattern 类以及 Matcher 类的方法；
- 设计正则表达式如下（十分全面！）

```
Pattern pattern1 = Pattern.compile("([A-Z]|[a-z]|[0-9])+"); // 取出用户名
Pattern pattern2 = Pattern.compile("([0][.][0-9]{0,2}[1-9])|([1]([.][0]{0,3})?)"); // 取出亲密度
```

正则表达式保证亲密度在区间(0-1]。

## 3.3 面向复用的设计：CircularOrbit<L,E>

设计思想：寻找共有功能

由于每个系统均存在：增删轨道、增善舞提、增加中心物体、得到中心物体  
共有功能如下

```
public boolean addTrack(Track<E> track); // 增加轨道
public double getSystemEntropy(); // 得到熵值
public int getTracksNumber(); // 得到轨道数目
public int getTrackObjectsNumber(int i); // 得到特定轨道序号i上的物体数目
public Track<E> getTrack(int i); // 得到特定轨道且其序号为i
public boolean deleteTrack(Track<E> track); // 删除轨道
public boolean addCentralPoint(L l); // 增加中心点
public boolean addTrackObject(Track<E> track, E e); // 在已有轨道上增加物体
public L getCentralPoint(); // 得到中心物体
```

```
public Iterator<E> iterator();//遍历的迭代器
```

### 3.4 面向复用的设计：Track

设计思想：寻找共同点-轨道半径、轨道上具有物体

设计 double 类型的轨道半径，行星的轨道半径和社交关系的关系网层次以及原子系统的层数都可以用此参数来表示

#### ● 设计共有属性

```
private double radius;// the radius of the track or the order of the tracks
private List<E> physicalObjects = new ArrayList<>();// save all the physical
objects on the same track
```

#### ● 设计共有方法

public Track(double radius)	构造器，轨道必有的属性是半径
public double getRadius()	得到半径
public void setRadius(double radius)	设置半径
public boolean add(E e)	增加轨道物体
public boolean remove(E e)	删除轨道物体
public boolean contains(E e)	判断物体是否在轨道上
public int getNumberOfObjects()	得到轨道物体数目
@Override public String toString()	重写 toString 方法
public List<E> getTrackObjects()	得到所有的轨道物体
public int getPhysicalObjectIndex(E e)	得到某物体所在 list 中的序号

### 3.5 面向复用的设计：L

#### 3.5.1 Stellar

该类为行星系统中心物体类且继承自 CentralObject 类

特有属性

```
private double radius = -1;//恒星半径
```

```
private double mess = -1;//恒星质量
```

特有方法

public double getRadius()	得到半径
public void setRadius	设置半径
public double getMess	得到质量
public void setMess	设置质量
@Override public String toString()	重写 toString 方法

### 3.5.2 CentralObject

原子系统中心物体运用此类，且此类被行星系统中心物体继承  
属性

`private String name`//中心物体的名字

方法

<code>public String getName()</code>	得到名字
<code>public void setName(String name)</code>	设置名字
<code>@Override public String toString()</code>	重写 toString 方法

而社交系统中心物体类运用的是下面的轨道物体类 Friend

## 3.6 面向复用的设计：PhysicalObject

### 3.6.1 PhysicalObject 类（父类）

该类被下面两个类继承且该类是原子系统所采用的物体类。

属性：`private double trackRadius = -1`;//轨道半径

方法

<code>public double getTrackRadius()</code>	得到轨道半径
<code>public void setTrackRadius(double trackRadius)</code>	设置轨道半径
<code>@Override public String toString()</code>	重写 toString 方法

### 3.6.2 StellarSystemObject 类（子类）

该类继承自 PhysicalObject 类，由于行星系统轨道物体特性很多，所以属性和方法很多。

属性

<code>private String planetName = new String();</code>	名字
<code>private String planetState = new String();</code>	状态
<code>private String planetColor = new String();</code>	颜色
<code>private double planetRadius = -1;</code>	行星半径
<code>private double revolutionSpeed = -1;</code>	公转速度
<code>private String revolutionDiretion = new String();</code>	公转方向
<code>private double angle = -1;</code>	初始角度

方法

<code>public StellarSystemObject(String planetName, String planetState, String planetColor, double planetRadius, double revolutionSpeed, String revolutionDiretion, double angle)</code>
--



<code>public String getPlanetName()</code>
<code>public void setPlanetName(String planetName)</code>
<code>public String getPlanetState()</code>
<code>public void setPlanetState(String planetState)</code>
<code>public String getPlanetColor()</code>
<code>public void setPlanetColor(String planetColor)</code>
<code>public double getPlanetRadius()</code>
<code>public void setPlanetRadius(double planetRadius)</code>
<code>public double getRevolutionSpeed()</code>
<code>public void setRevolutionSpeed(double revolutionSpeed)</code>
<code>public String getRevolutionDiretion()</code>
<code>public void setRevolutionDiretion(String revolutionDiretion)</code>
<code>public double getAngle()</code>
<code>public void setAngle(double angle)</code>
<code>@Override public String toString()</code>

### 3.6.3 Friend 类（子类）

该类继承自 `PhysicalObject` 类，且是社交系统的中心物体和轨道物体的共用类属性

<code>private String friendName = new String();</code>	姓名
<code>private int age = -1;</code>	年龄
<code>private String sex = new String();</code>	性别
<code>private Map&lt;Friend, Double&gt; socialTieMap = new HashMap&lt;&gt;();</code>	亲密度关系
<code>private LinkedList&lt;Friend&gt; allFriends = new LinkedList&lt;&gt;();</code>	所有朋友

方法

<code>public String getFriendName()</code>	得到姓名
<code>public void setFriendName(String friendName)</code>	设置姓名
<code>public int getAge()</code>	得到年龄
<code>public void setAge(int age)</code>	设置年龄
<code>public String getSex()</code>	得到性别
<code>public void setSex(String sex)</code>	设置性别
<code>public boolean addSocialTie(Friend friend, double intimacy)</code>	添加关系
<code>public boolean deleteSocialTie(Friend friend)</code>	删除关系
<code>public double getSocialTie(Friend friend)</code>	得到亲密度
<code>public LinkedList&lt;Friend&gt; getAllFriends()</code>	得到朋友
<code>@Override public String toString()</code>	toString

### 3.7 可复用 API 设计

API 设计中的像许多功能已在 Application 类中重复实现。  
故设计了两个类，如下。

```
public double getObjectDistributionEntropy(CircularOrbit<L, E> c)
public Difference<L, E> getDifference(CircularOrbit<L, E> c1, CircularOrbit<L, E> c2)
```

->这两个方法分别用于计算熵值和获取两个系统的差异  
设计思想是委托到其他的类中实现具体的功能，具体表现如下

```
CircularOrbitAPIs.java
1 package APIs;
2
3 import circularOrbit.CircularOrbit;
4
5
6 public class CircularOrbitAPIs<L, E> {
7
8     public double getObjectDistributionEntropy(CircularOrbit<L, E> c) {
9         return c.getSystemEntropy();
10    }
11
12    public Difference<L, E> getDifference(CircularOrbit<L, E> c1, CircularOrbit<L, E> c2) {
13        return new Difference<>(c1, c2);
14    }
15
16 }
17
```

### 3.8 设计模式应用

#### 3.8.1 迭代器设计模式

- 设计接口： `public interface CircularOrbit<L, E> extends Iterable<E>`
- 完善接口： `public Iterator<E> iterator()`  

```
public Iterator<E> iterator() {
    return new PhysicalObjectIterator();
}
```
- 在 ConcreteCircularOrbit 类中加入了一个私有类 `private class PhysicalObjectIterator implements Iterator<E>`
- 完善该类的方法，如截图

```

private class PhysicalObjectItertor implements Iterator<E> {

    private int countObject = 0; // 当前物体计数
    private int numObject = 0; // 物体总数目

    public PhysicalObjectItertor() { // 得到系统物体总数目
        for (int i = 0; i < getTracksNumber(); i++) {
            numObject += getTrackObjectsNumber(i + 1);
        }
    }

    public boolean hasNext() { // 判断是否还有下一个物体
        return countObject < numObject;
    }

    public E next() {
        if (hasNext()) {
            return getEByNum(++countObject); // 通过当前物体的序号得到该物体
        }
        throw new NoSuchElementException();
    }
}

```

有了该迭代器设计，便可以对系统进行遍历，且遍历顺序是有内层轨道向外层轨道，轨道内随即遍历（具体实现是按 list 内的储存顺序遍历）

### 3.8.2 备忘录设计模式

为了实现该设计模式，新设计了一个 Package 名为 `memento`

包内有四个类，分别是 `State` 类，`Memento` 类，`Caretaker` 类，`Originator` 类

▼ `memento`

- > `Caretaker.java`
- > `Memento.java`
- > `Originator.java`
- > `State.java`

- `State` 类

需要保存的状态，为了面对将来可能出现的变化，所以设计此类而不是直接传入 `double` 参数。

属性

```
private double trackRadius = -1;
```

方法

```
public State(double trackRadius)
```

```
public double getTrackRadius()
```

```
@Override public String toString()
```

- `Memento` 类

属性

```
private State state; // state to save
```

方法

```
public Memento(State state)
```

```
public State getState()
```

- CareTaker 类

该类储存所有的历史纪录，并可继续添加属性

```
private List<Memento> mementos = new ArrayList<>(); // save all the trackNum in it
```

方法

```
public void addMemento(Memento m)
```

```
public Memento getMemento()
```

- Originator 类

该类管理记录，可以保存和回滚属性

```
private State state;
```

方法

```
public void setState(State state)
```

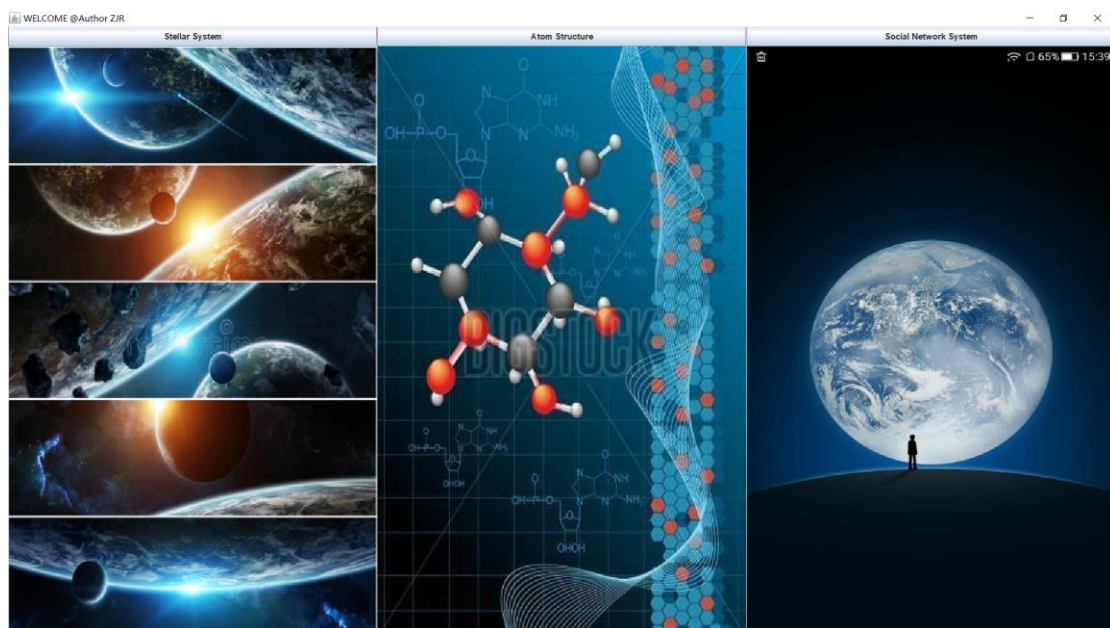
```
public Memento save()
```

```
public void restore(Memento m)
```

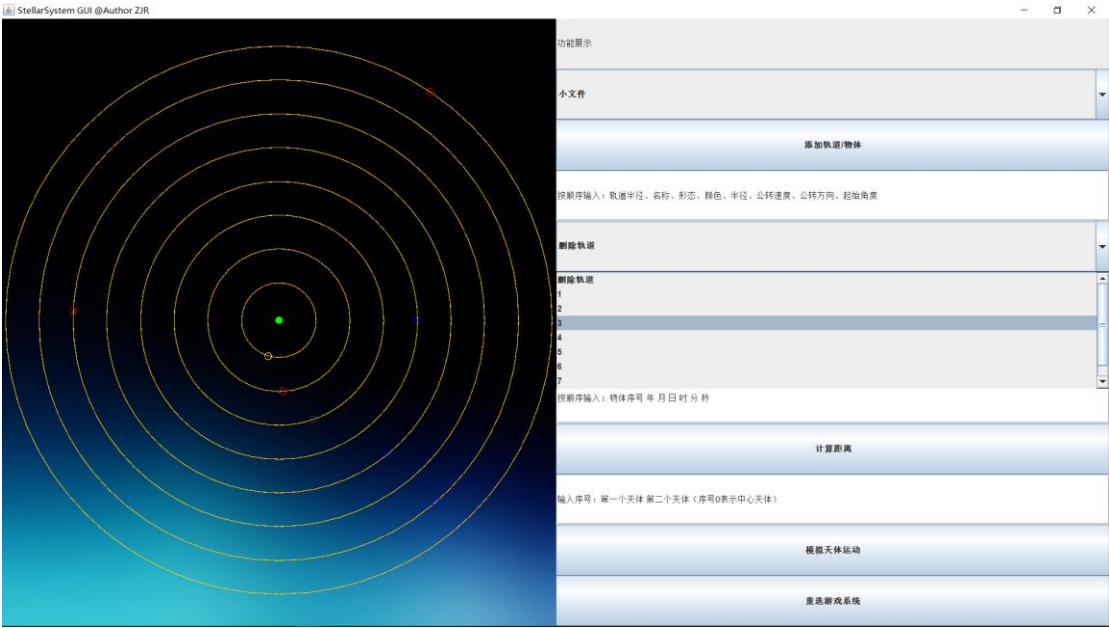
### 3.9 应用设计开发与 GUI

GUI 演示界面如下

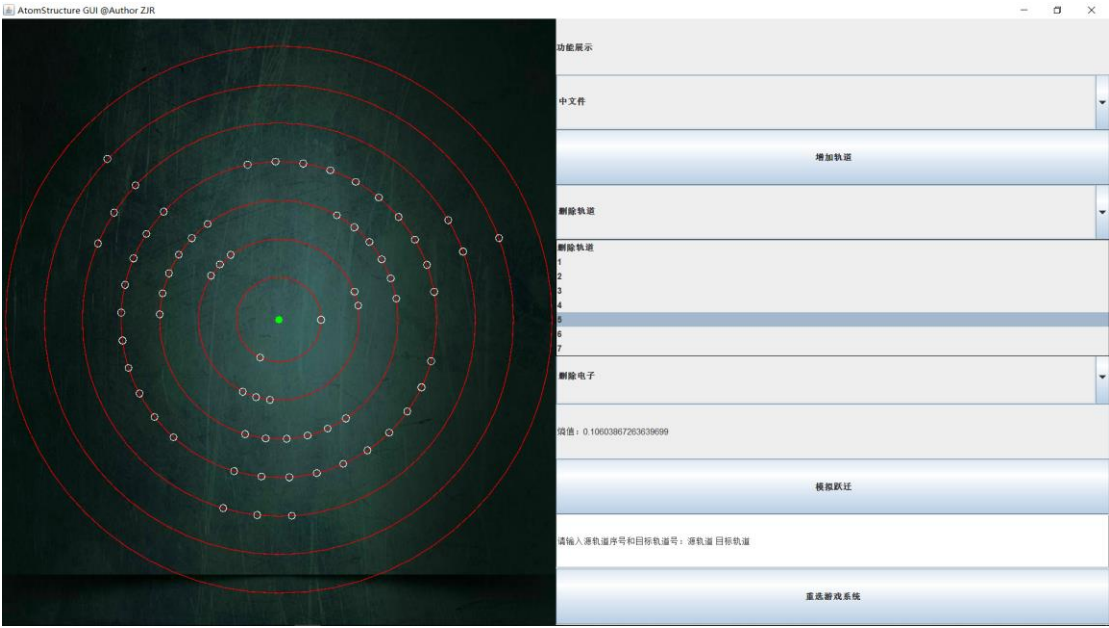
1，系统选择界面



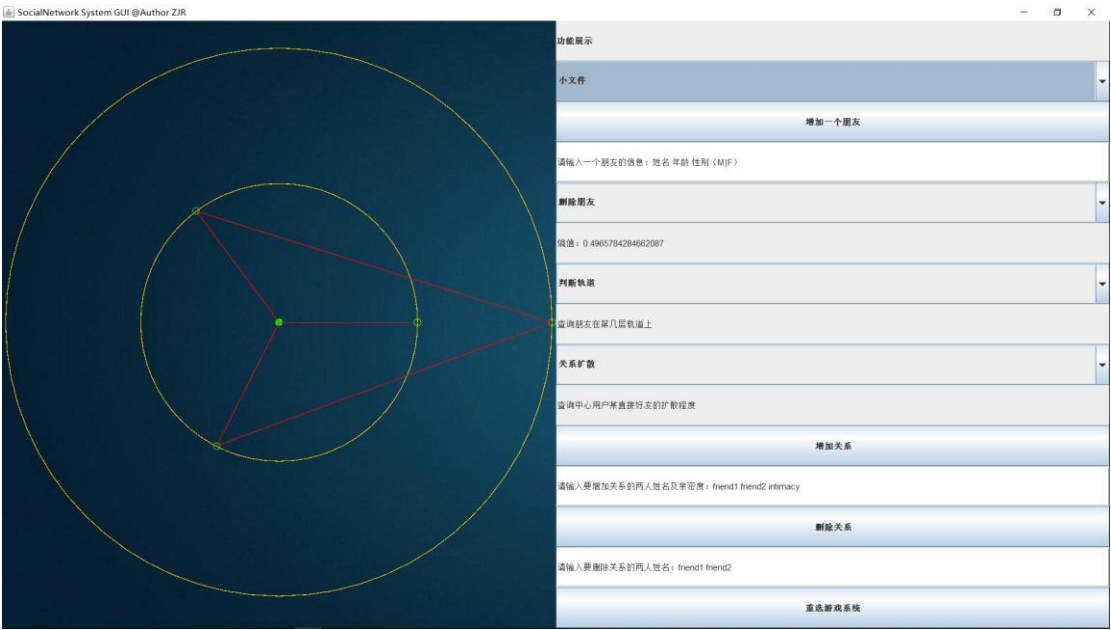
2.1，行星系统 GUI 实现（有 button 和 comboBox 下拉选项框，同时实现了行星运动）



2.2，原子系统 GUI 实现（有 button 和 combobox 下拉选项框）



2.3，社交系统 GUI 实现（有 button 和 combobox 下拉选项框）



3.9.1 StellarSystem

StellarSystem 类设计

该类继承自 ConcreteCircularOrbit 类，是行星系统类

方法

<code>public void readFileAndCreateSystem(File file)</code>	读行星文件并建立系统
<code>public void setReadTime(Calendar readTime)</code>	设置读取文件的时间为系统初始时刻
<code>public Calendar getReadTime()</code>	得到系统初始时刻
<code>public double calculatePosition(StellarSystemObject planet, Calendar newTime)</code>	计算系统新的时刻的角度位置
<code>public double getPhysicalDistance(StellarSystemObject planet1, StellarSystemObject planet2)</code>	计算两个行星的距离
<code>public double getPhysicalDistanceStar(StellarSystemObject planet)</code>	计算某行星与恒星之间的距离

3.9.2 AtomStructure

AtomStructure 类设计

该类继承自 ConcreteCircularOrbit 类，是原子系统类

方法

<code>public boolean transit(PhysicalObject physicalObject, Track&lt;PhysicalObject&gt; track)</code>	轨道上某电子跃迁
<code>public boolean addPhysicalObject(PhysicalObject physicalObject)</code>	添加一个电子

<b>public boolean</b> deletePhysicalObject(PhysicalObject physicalObject)	删除一个电子
<b>public void</b> readFileAndCreateSystem(File file)	读原子文件建立系统

### 3.9.3 SocialNetworkCircle

SocialNetworkCircle 类设计

该类继承自 ConcreteCircularOrbit 类，是社交系统类

属性

```
private List<Friend> allFriends = new ArrayList<>(); // 储存所有的朋友
```

方法

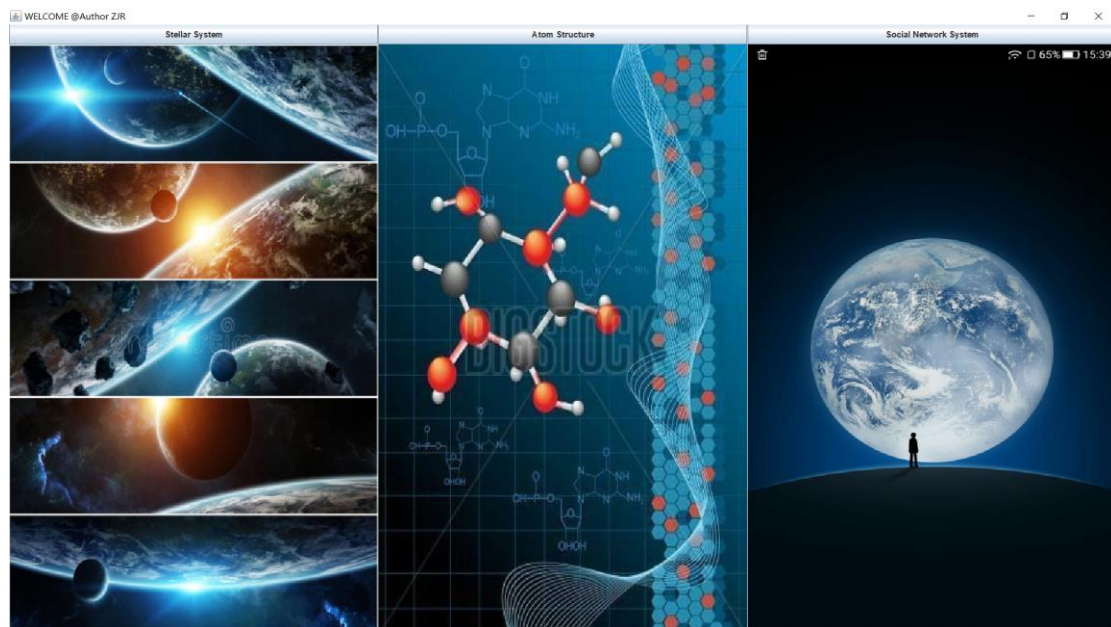
<b>public void</b> readFileAndCreateSystem(File file)	读取社交系统文件建立系统
<b>public boolean</b> addFriend(Friend friend)	添加朋友
<b>public boolean</b> addCentralUser	添加中心用户
<b>public int</b> getDistance(Friend friend1, Friend friend2)	得到两人的社交距离
<b>public</b> Friend getFriendByName(String name)	通过名字获取 friend
<b>public boolean</b> addFriendOnTrack(Friend friend)	将一个朋友添加在轨道上
<b>public int</b> getFriendTrackNum(Friend friend)	获取 friend 的轨道半径
<b>public int</b> getLogicalDistance(Friend friend1, Friend friend2)	获取两个朋友的逻辑距离
<b>public int</b> Informationdiffusivity(Friend friend)	获取中心用户的一级朋友的扩散度
<b>public void</b> addRelationAndRefactor(Friend friend1, Friend friend2, <b>double</b> intimacy)	添加关系重构系统
<b>public boolean</b> deleteRelationAndRefactor(Friend friend1, Friend friend2)	删除关系并重构系统



## 3.10 application 类与我的 GUI

### 3.10.1 App 类-用户类

- **main 函数类**, 用户通过该类可以直接进入到 GUI 界面-系统选择界面, 如下。



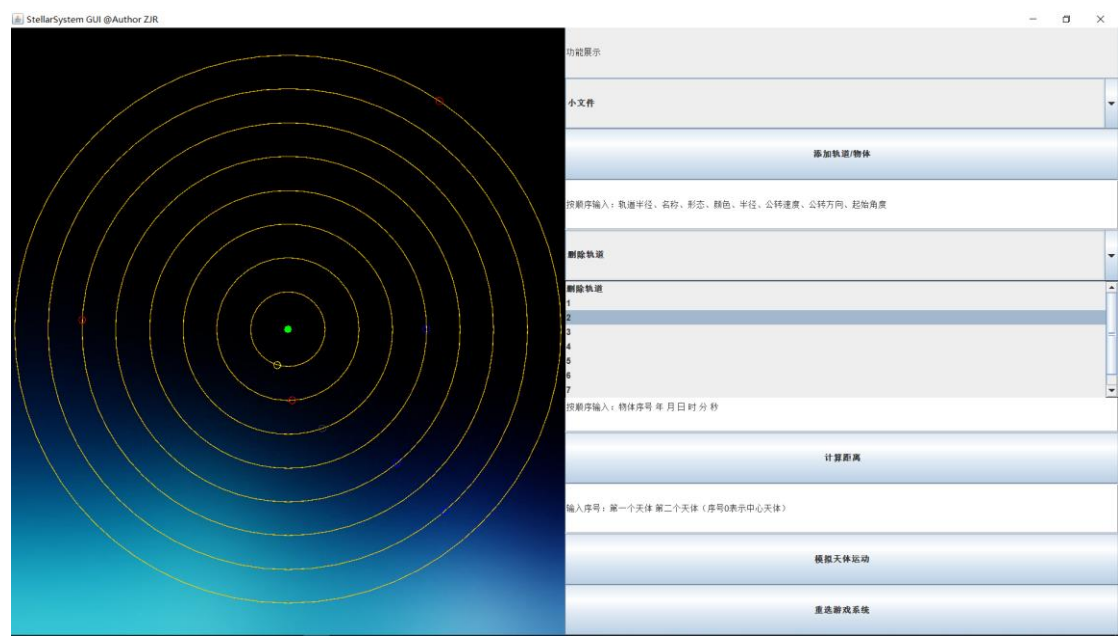
- **设计思路**: 一个 JFrame+三个 JPanel+三个 JButton+三个图片+3 个 ActionListener。
- **事件监听**: 分别点击三个 button, 执行: 销毁当前 JFrame, 并调用三个具体的系统 GUI 类中的 GUI 方法, 即可切换 GUI 界面, 如截图

```
button1.addActionListener(new ActionListener() {  
  
    public void actionPerformed(ActionEvent e) {  
        frame.dispose();  
        StellarSystemGUI stellarSystemGUI = new StellarSystemGUI();  
        stellarSystemGUI.GUI();  
    }  
});
```

### 3.10.2 StellarSystemGUI 类

- **行星系统应用 GUI**, 有许多私有属性和方法, 仅有一个 public 方法为 GUI, 用于 GUI 显示。





- 组件: 1 个 JFrame+2 个 JPanel+1 个 JLabel+2 个 JComboBox+5 个 JButton+4 个 JTextField

2 个 JComboBox（下拉选项框）	JcomboBox1	选择读取的文件
	JcomboBox2	删除轨道
5 个 JButton	Jbutton1	添加轨道/物体
	Jbutton2	计算某时刻行星角度
	Jbutton3	计算两物体的距离
	Jbutton4	模拟行星运动
	Jbutton5	重选系统
4 个 JTextField	JtextField1	输入新物体参数
	JtextField2	熵值展示
	JtextField3	输入行星时刻参数
	JtextField4	输入两个物体序号

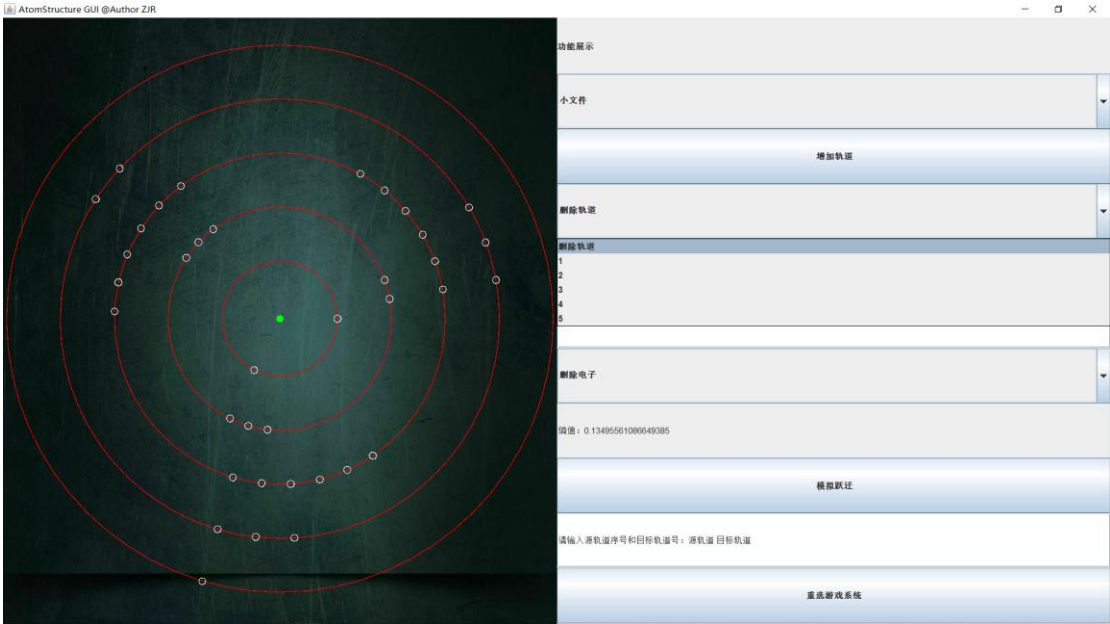
- 为每一个 button 和 combobox 设置了事件监听，如代码所示

```
button2.setText("添加轨道/物体");
button2.setFocusable(false);
JTextField textField = new JTextField("按顺序输入：轨道半径、名称、形态、颜色、半径、公转速度、公转方向、起始角度");
panel2.add(textField);
button2.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        try {
            String[] strings = textField.getText().split(" ");
            if (stellarSystem.getTrackByRadius(Double.parseDouble(strings[0])) == null) {
                Track<StellarSystemObject> track = new Track<>(Double.parseDouble(strings[0]));
                StellarSystemObject planet = new StellarSystemObject(strings[1], strings[2], strings[3],
                    Double.parseDouble(strings[4]), Double.parseDouble(strings[5]), strings[6],
                    Double.parseDouble(strings[7]));
                planet.setTrackRadius(Double.parseDouble(strings[0]));
                stellarSystem.addTrack(track);
                stellarSystem.addTrackObject(track, planet);
                initFunc3();
                comboBox3.repaint();
                textField4.setText("熵值: " + stellarSystem.getSystemEntropy());
            }
        } catch (Exception e2) {
            textField.setText("输入格式非法!");
        }
    }
});
```

3.10.3 AtomStructureGUI 类

- 原子系统 GUI 类，有许多私有属性和方法，只有一个 public 方法 GUI，用于主函数调用。



- 组件：1 个 JFrame+2 个 JPanel+4 个 JButton+3 个 JComboBox+3 个 JTextField+1 个 JLabel

4 个 JButton	Jbutton1	增加轨道
	Jbutton2	增加电子
	Jbutton3	模拟跃迁
	Jbutton4	重选系统，回到主函数
3 个 JComboBox	Jcombobox1	选择读取文件
	Jcombobox2	删除轨道
	Jcombobox3	删除电子
3 个 JTextField	JtextField1	增加电子的轨道序号
	JtextField2	熵值展示
	JtextField3	模拟跃迁的源和目标轨道序号

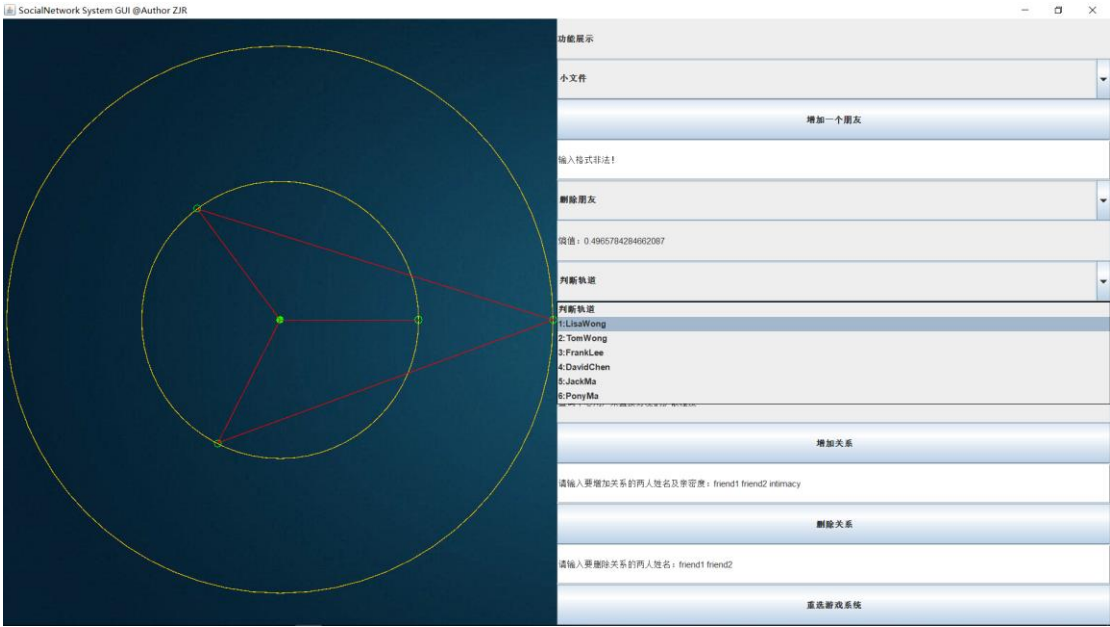
- 为每一个 JButton 和 JComboBox 添加了事件监听，如代码

```
comboBox1.addItem("小文件");
comboBox1.addItem("中文件");
comboBox1.addItemListener(new ItemListener() {

    public void itemStateChanged(ItemEvent e) {
        try {
            if (e.getStateChange() == ItemEvent.SELECTED) {
                if (e.getItem().equals("小文件")) {
                    changeAtomStructure(1);
                } else if (e.getItem().equals("中文件")) {
                    changeAtomStructure(2);
                }
            }
            initFunc3();
            initFunc5();
            textField4.setText("请在此处输入要增加电子的轨道序号");
            textField6.setText("熵值: " + atomStructure.getSystemEntropy());
            textField7.setText("请输入源轨道序号和目标轨道号: 源轨道 目标轨道");
        }
        catch (Exception e2) {
            System.out.println("未找到文件! ");
        }
    }
});
```

3.10.4 SocialNetworkSystemGUI 类

- 社交系统 GUI 类，有许多私有属性和方法，只有一个 public 方法 GUI，用于主函数调用。



- 属性：1 个 JFrame+2 个 JPanel+1 个 JLabel+4 个 JComboBox+4 个 JButton+6 个 JTextField

4 个 JComboBox	JcomboBox1	选择读取文件
	JcomboBox2	删除朋友
	JcomboBox3	判断朋友所处轨道

	JcomboBox4	判断一级好友关系扩散
4 个 Jbutton	Jbutton1	增加一个朋友
	Jbutton2	增加关系
	Jbutton3	删除关系
	Jbutton4	重选系统, 调 main 函数
6 个 JTextField	JtextField1	增加的朋友参数
	JtextField2	熵值展示框
	JtextField3	朋友所在的轨道结果框
	JtextField4	关系扩散程度结果框
	JtextField5	增加关系参数框
	JtextField6	删除关系参数框

- 为每一个 Jbutton 和 JcomboBox 设置了事件监听，如代码

```
private void initFunc1() {
    comboBox1.addItem("小文件");
    comboBox1.addItem("中文件");
    comboBox1.addItem("大文件");
    comboBox1.addItemListener(new ItemListener() {

        public void itemStateChanged(ItemEvent e) {
            try {
                if (e.getStateChange() == ItemEvent.SELECTED) {
                    if (e.getItem().equals("小文件")) {
                        changeAtomStructure(1);
                    } else if (e.getItem().equals("中文件")) {
                        changeAtomStructure(2);
                    } else if (e.getItem().equals("大文件")) {
                        changeAtomStructure(3);
                    }
                }
                initFunc3();
                textField2.setText("请输入一个朋友的信息: 姓名 年龄 性别 (M|F) ");
                comboBox3.repaint();
                textField4.setText("熵值: " + socialNetworkCircle.getSystemEntropy());
                initFunc5();
                comboBox5.repaint();
                textField5.setText("查询朋友在第几层轨道上");
                initFunc6();
                comboBox6.repaint();
                textField6.setText("查询中心用户某直接好友的扩散程度");
                textField7.setText("请输入要增加关系的两人姓名及亲密度: friend1 friend2 intimacy");
                textField8.setText("请输入要删除关系的两人姓名: friend1 friend2");
            }
            catch (Exception e2) {
                System.out.println("未找到文件!");
                e2.printStackTrace();
            }
        }
    });
}
```

### 3.11 Difference 类

设计思路及目的：用此类保存两个系统的差异属性

```
int trackNumbersDifference = 0; // 表示轨道数目差异
```

```
List<Integer> ObjectDifferences = new ArrayList<>(); // 储存所有轨道上的物体数
```

## 差异

## 方法

**public** Difference(CircularOrbit<L, E> c1, CircularOrbit<L, E> c2)

->基本思路是：遍历所有的轨道，遍历轨道上的所有物体

```
this.trackNumbersDifference = c1.getTracksNumber() - c2.getTracksNumber();
System.out.println("轨道数差异: " + trackNumbersDifference);
int smaller = trackNumbersDifference > 0 ? c2.getTracksNumber() : c1.getTracksNumber();
int bigger = smaller + trackNumbersDifference;
for (int i = 0; i < smaller; i++) {
    ObjectDifferences.add(c1.getTrackObjectsNumber(i + 1) - c2.getTrackObjectsNumber(i + 1));
    System.out.println("轨道" + (i + 1) + "的物体数目差异: " + ObjectDifferences.get(i) + "物体差异"
        + c1.getTrack(i + 1).toString() + c2.getTrack(i + 1).toString());
}
for (int i = smaller; i < bigger; i++) {
    ObjectDifferences.add(
        trackNumbersDifference > 0 ? c1.getTrackObjectsNumber(i + 1) : -c2.getTrackObjectsNumber(i + 1));
    System.out.println("轨道" + (i + 1) + "的物体数目差异: " + ObjectDifferences.get(i) + "物体差异"
        + c1.getTrack(i + 1).toString() + c2.getTrack(i + 1).toString());
}
```

## 3.12 新变化

## 3.12.1 StellarSystem

具体量化可由 github 来看到，如截图  
第一部分工作变化如下（基础类变化）：

Showing 12 changed files with 225 additions and 115 deletions.		Unified	Split
src/APIs/CircularOrbitAPIs.java	+2 -6	■■■■■	
src/applications/AtomStructureGUI.java	+4 -4	■■■■■	
src/applications/SocialNetworkSystemGUI.java	+2 -2	■■■■■	
src/applications/StellarSystemGUI.java	+39 -28	■■■■■	
src/circularOrbit/AtomStructure.java	+52 -8	■■■■■	
src/circularOrbit/CircularOrbit.java	+0 -2	■■■■■	
src/circularOrbit/ConcreteCircularOrbit.java	+11 -51	■■■■■	
src/circularOrbit/SocialNetworkCircle.java	+51 -7	■■■■■	
src/circularOrbit/StellarSystem.java	+57 -0	■■■■■	
src/difference/Difference.java	+2 -2	■■■■■	
src/physicalObject/Friend.java	+1 -1	■■■■■	
src/track/StellarTrackChange.java	+4 -4	■■■■■	

第二部份工作变化如下（GUI 功能变化）：

Showing 3 changed files with 27 additions and 14 deletions.

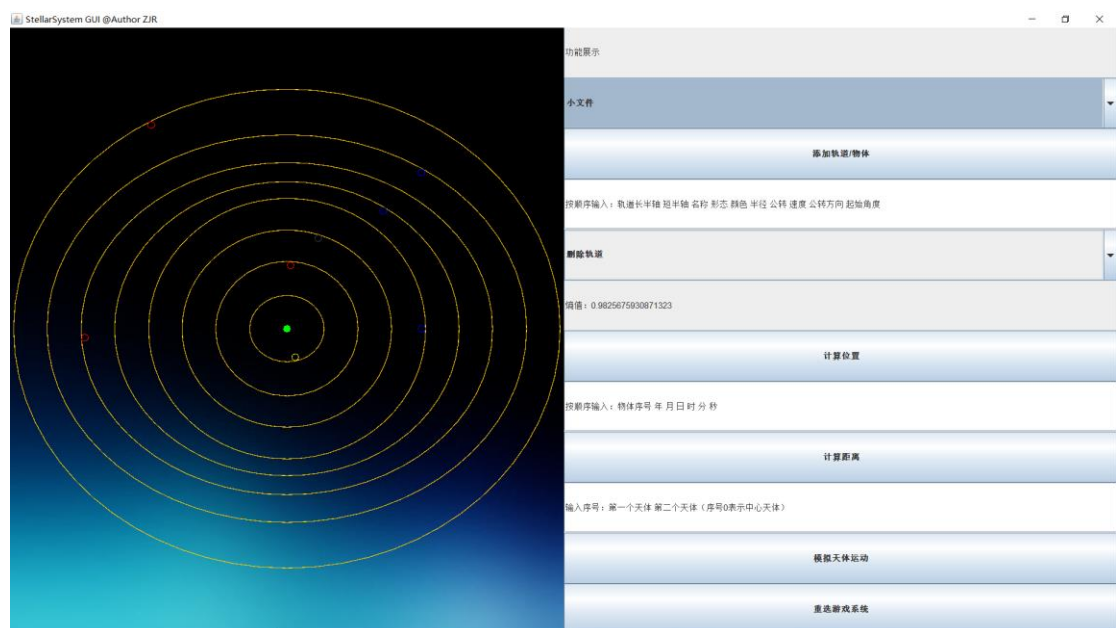
- 此变化改动较大，表现在新加了一个泛型，新建了一个椭圆长半轴和短半轴
- 修改读取的文件，增加轨道参数（只修改了一个小文件，中文件和大文件并未修改），如截图

```

- Planet ::= <Earth,Solid,Blue,6378.137,1.49e8,29.783,20.783,CW,0>
- Planet ::= <Mercury,Solid,Dark,1378.137,1.49e7,69,40,CW,20.085>
- Planet ::= <Saturn,Liquid,Red,2378,1.49e6,2.33e5,1.99e5,CCW,39.21>
- Planet ::= <Jupiter,Gas,Blue,1637.007,2e8,30,20,CW,70>
- Planet ::= <Mars,Solid,Red,637.137,9.99e10,1000.93,800,CCW,110>
- Planet ::= <Neptune,Liquid,Yellow,6627.137,1.49e5,9293.05,7000.45,CCW,360>
- Planet ::= <Uranus,Gas,Blue,637.137,1.49e11,1e5,8e4,CW,359>
- Planet ::= <Venus,Solid,Red,6378.137,1.49e20,203.24,190.25,CCW,181.23>
+ Planet ::= <Earth,Solid,Blue,6378.137,1.49e8,1.4e8,29.783,CW,0>
+ Planet ::= <Mercury,Solid,Dark,1378.137,1.49e7,1.4e7,69,CW,20.085>
+ Planet ::= <Saturn,Liquid,Red,2378,1.49e6,1.4e6,2.33e5,CCW,39.21>
+ Planet ::= <Jupiter,Gas,Blue,1637.007,2e8,1.7e8,30,CW,70>
+ Planet ::= <Mars,Solid,Red,637.137,9.99e10,8e10,1000.93,CCW,110>
+ Planet ::= <Neptune,Liquid,Yellow,6627.137,1.49e5,1.3e5,9293.05,CCW,360>
+ Planet ::= <Uranus,Gas,Blue,637.137,1.49e11,1.2e11,1e5,CW,359>
+ Planet ::= <Venus,Solid,Red,6378.137,1.49e20,1.3e20,203.24,CCW,181.23>

```

- 修改了 GUI 画图：将圆形轨道变为椭圆形轨道。如截图展示。



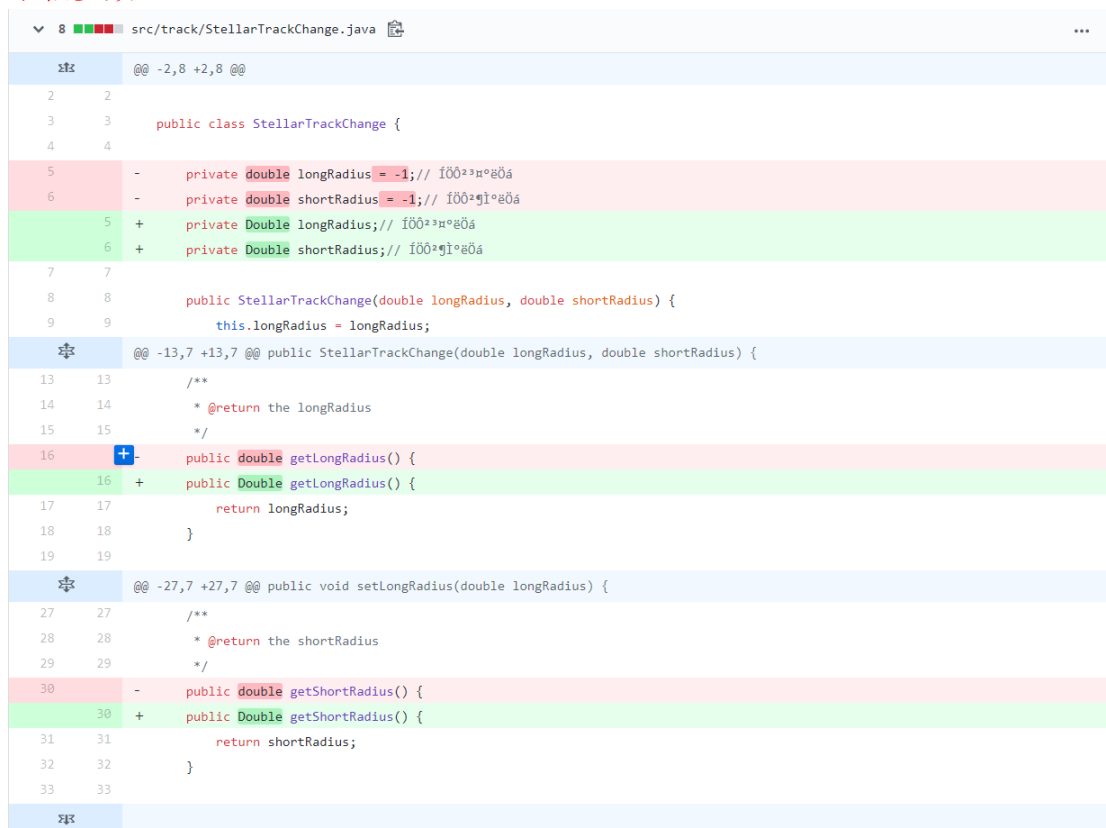
- 修改了运动函数：将原本的圆形运动改为椭圆形运动  
根据数学原理，计算得到任意时刻的位置（为了简便同时考虑到理论上的近似可行性，近似假设角速度不变），具体实现代码如截图

```

if (angleRenew > 90 && angleRenew < 270) {
    double x1 = d * (d * k)
        / Math.pow((d * k) * (d * k) + Math.pow(d * Math.tan(angleRenew), 2), 0.5);
    double y1 = x1 * Math.tan(angleRenew);
    x1 = 0 - x1;
    y1 = 0 - y1;
    g2d.draw(new Ellipse2D.Double(x + x1 - 5, y - y1 - 5, 10, 10));
} else if (angleRenew == 90 || angleRenew == 270) {
} else {
    double x1 = d * (d * k)
        / Math.pow((d * k) * (d * k) + Math.pow(d * Math.tan(angleRenew), 2), 0.5);
    double y1 = x1 * Math.tan(angleRenew);
    g2d.draw(new Ellipse2D.Double(x + x1 - 5, y - y1 - 5, 10, 10));
}

```

- 新增了一个类，名为 StellarTrackChange，用于保存椭圆轨道的长半轴和短半轴参数。



```


src/track/StellarTrackChange.java
1  @@ -2,8 +2,8 @@
2  2
3  3      public class StellarTrackChange {
4  4
5  -     private double longRadius = -1; // f00*3n0e0a
6  -     private double shortRadius = -1; // f00*3i0e0a
7  +     private Double longRadius; // f00*3n0e0a
8  +     private Double shortRadius; // f00*3i0e0a
9
10
11     public StellarTrackChange(double longRadius, double shortRadius) {
12         this.longRadius = longRadius;
13
14     @@ -13,7 +13,7 @@ public StellarTrackChange(double longRadius, double shortRadius) {
15
16     /**
17      * @return the longRadius
18      */
19     public double getLongRadius() {
20     +     public Double getLongRadius() {
21         return longRadius;
22     }
23
24     @@ -27,7 +27,7 @@ public void setLongRadius(double longRadius) {
25
26     /**
27      * @return the shortRadius
28      */
29     public double getShortRadius() {
30     +     public Double getShortRadius() {
31         return shortRadius;
32     }
33

```

- 修改了绝大多数基本类，增加一个泛型。  
该泛型用于代表轨道半径类。

### 3.12.2 AtomStructure

具体量化可由 github 来看到，如截图

 Showing 5 changed files with 80 additions and 6 deletions.

- 需要修改生成系统的外部读取文件，添加参数：质子数和中子数



```

3 src/Spring2019_HITCS_SC_Lab3-master/AtomicStructure.txt
... @@ -1,3 +1,4 @@
1 1 ElementName ::= Rb
2 2 NumberOfTracks ::= 5
3 - NumberOfElectron ::= 1/2;2/8;3/18;4/8;5/1
3 + NumberOfElectron ::= 1/2;2/8;3/18;4/8;5/1
4 + AtomicNucleus ::= 37/36

```

```

3 src/Spring2019_HITCS_SC_Lab3-master/AtomicStructure_Medium.txt
... @@ -1,3 +1,4 @@
1 1 ElementName ::= Er
2 2 NumberOfTracks ::= 6
3 - NumberOfElectron ::= 1/2;2/8;3/18;4/30;5/8;6/2
3 + NumberOfElectron ::= 1/2;2/8;3/18;4/30;5/8;6/2
4 + AtomicNucleus ::= 68/70

```

- 增加了一个中心物体类：AtomicNucleus

```

46 src/centralObject/AtomicNucleus.java
... @@ -0,0 +1,46 @@
1 + package centralObject;
2 +
3 + import java.util.ArrayList;
4 + import java.util.List;
5 +
6 + public class AtomicNucleus extends CentralObject {
7 +
8 +     private List<CentralObject> protons = new ArrayList<>();
9 +     private List<CentralObject> neutrons = new ArrayList<>();

```

- 修改 AtomStructure 类文件：17 处增加，3 处删除

```

20 Structure.java
17 additions & 3 deletions

```

将中心物体由 CentralObject 改为 AtomicNucleus，同时增加了读取质子数和中子数的正则表达式，如截图



```

113 + String regex3 = "AtomicNucleus[ ]*::=[ ]*([1-9][0-9]*)/([1-9][0-9]*)";
114 + Pattern pattern3 = Pattern.compile(regex3);
115 + Matcher matcher3 = pattern3.matcher(bfReader.readLine());
116 + if (matcher3.find()) {
117 +     int num1 = Integer.parseInt(matcher3.group(1));
118 +     for (int j = 0; j < num1; j++) {
119 +         this.getCentralPoint().addProton();
120 +     }
121 +     int num2 = Integer.parseInt(matcher3.group(2));
122 +     for (int j = 0; j < num2; j++) {
123 +         this.getCentralPoint().addNeutron();
124 +     }
125 + }
126 +

```

● 修改 Atom StructureGUI 类：13 处增加，1 处删除

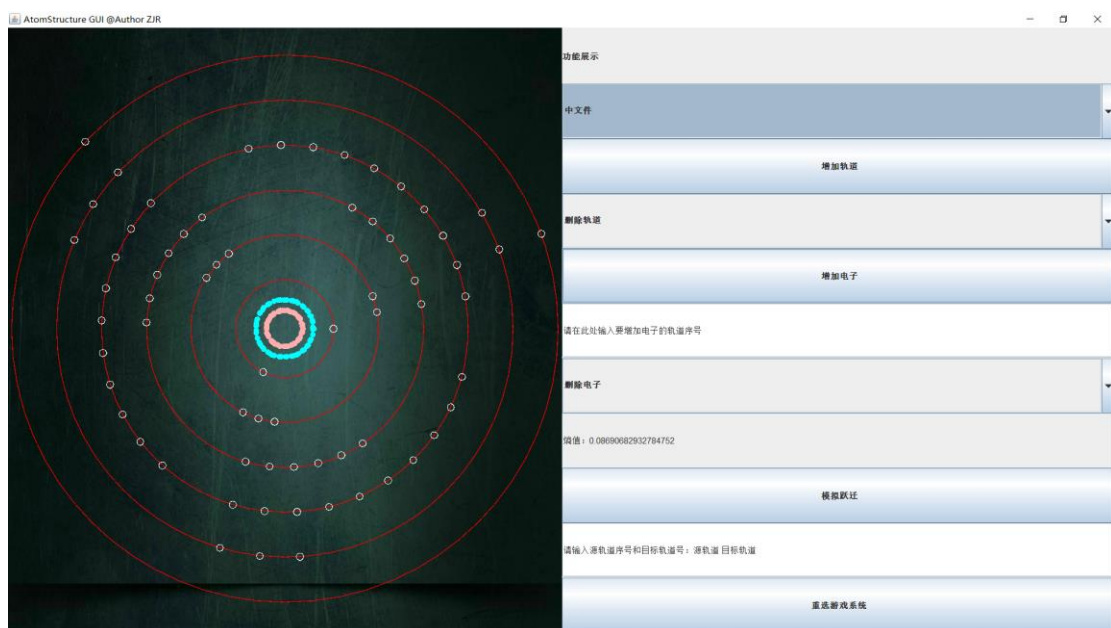
14 13 additions & 1 deletion StructureGUI.java

```

@@ -92,7 +92,19 @@ private void initPanel1() {
92 92 g.setColor(Color.green);
93 93 double r = (double) (x - 10) / count;
94 94 if (atomStructure.getCentralPoint() != null) {
95 - g.fillOval(x - 5, y - 5, 10, 10);
95 + int num1 = atomStructure.getCentralPoint().getProtonsNum();
96 + int num2 = atomStructure.getCentralPoint().getNeutronsNum();
97 +
98 + for (int i = 0; i < num2; i++) {
99 + g.setColor(Color.pink);
100 + g.fillOval((int) (x + Math.cos(i * 360 / num2) * 25) - 4,
101 + (int) (y + Math.sin(i * 360 / num2) * 25) - 4, 8, 8);
102 + }
103 + for (int j = 0; j < num1; j++) {
104 + g.setColor(Color.CYAN);
105 + g.fillOval((int) (x + Math.cos(j * 360 / (num1+1)) * 40) - 4,
106 + (int) (y + Math.sin(j * 360 / (num1+1)) * 40) - 4, 8, 8);
107 + }

```

将原本的画一个中心物体的语句删除，并增加语句在轨道中央画多个质子和中子，如演示截图。

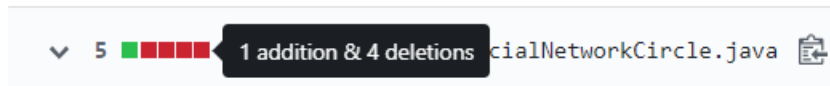


### 3.12.3 SocialNetworkCircle

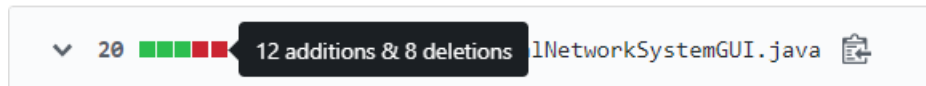
具体量化可由 github 来看到，如截图

Showing 2 changed files with 13 additions and 12 deletions.

SocialNetworkCircle.java 类的变化为 1 处添加 4 处删除



SocialNetworkCircleGUI 类的变化为 12 处增加，8 处删除



- 变化较小，花费较少时间即完成
- 改动的地方有

`public void readFileAndCreateSystem(File file)` 在读取文件构造系统的时候删除了边的双向联系；

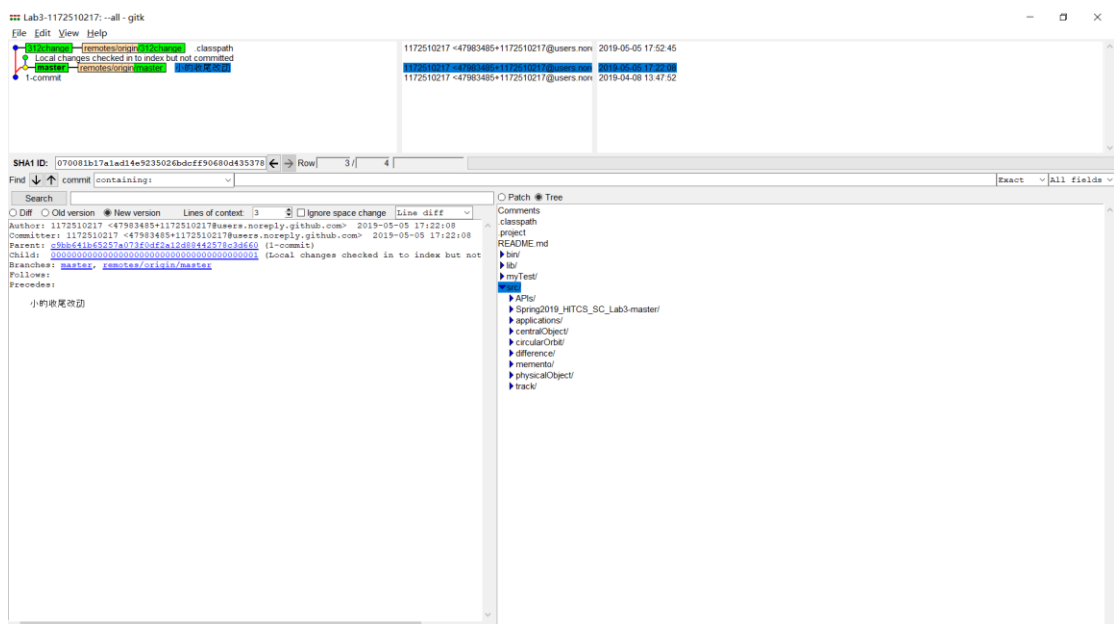
`public void addRelationAndRefactor(Friend friend1, Friend friend2, double intimacy)` 修改为只是添加单向的关系；

`public boolean deleteRelationAndRefactor(Friend friend1, Friend friend2)` 修改为只是删除单向关系；

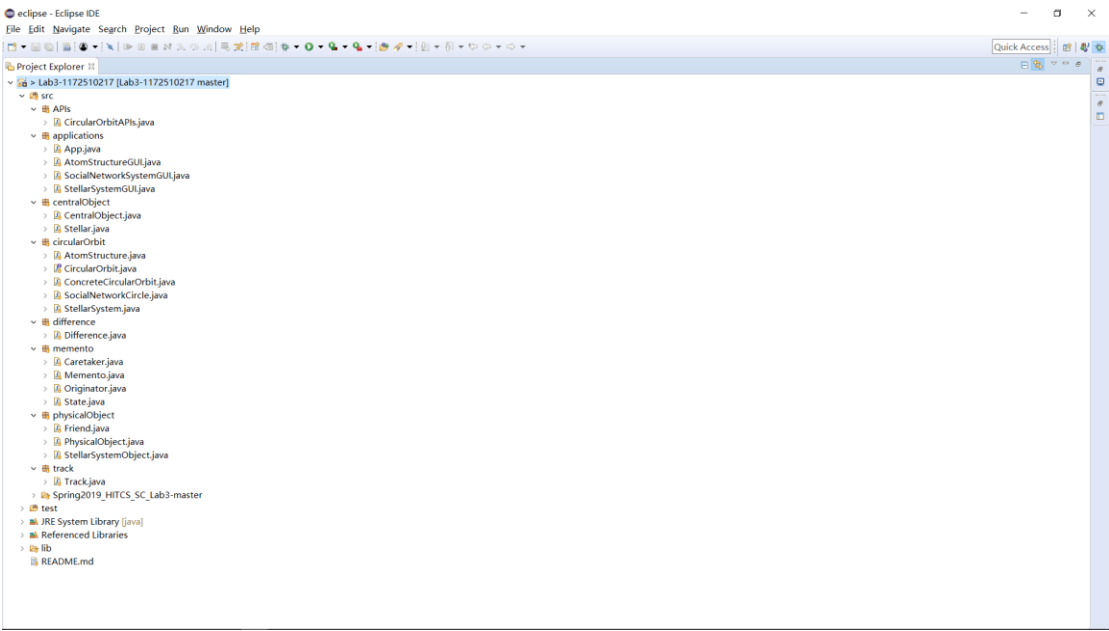
SocialNetworkSystemGUI 添加关系连线的地方进行修改-改为如果前者轨道层数大于后者，则不进行画线；

同时修改部分 GUI 显示信息文本修改：friend1 成功与 friend2 建立了关系改为 friend1 成功认识了 friend2 等。

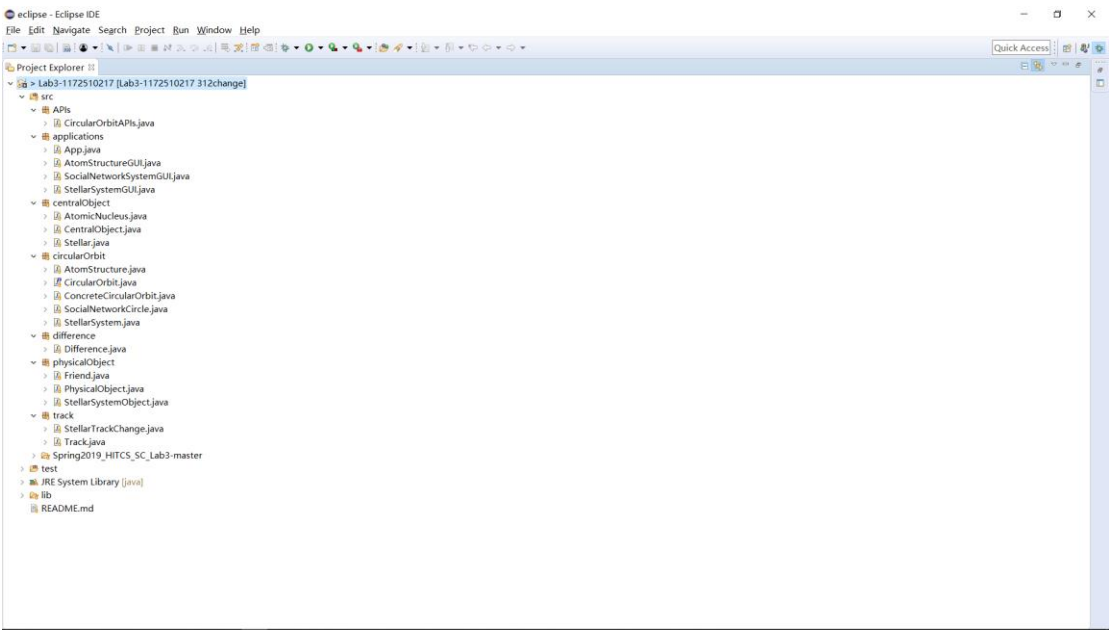
### 3.13 Git 仓库结构



目前为止我的仓库主分支目录为



目前为止我的仓库的 312change 分支目录为



## 4 实验进度记录

日期	时间段	计划任务	实际完成情况
19-04-10	下午 1~2 节	通读实验要求，并理解	正常完成
19-04-19	上午 3~4 节	学习 GUI 基本知识	初步了解 JButton 和 ActionListener
19-04-20	晚上	设计基础类	完成部分

19-04-21	晚上	设计基础类	完成绝大部分
19-04-22	晚上	设计基础类	全部完成
19-04-28	晚上	GUI 插件学习	一无所成
19-04-29	晚上	GUI 插件学习	仍无头绪决定放弃使用插件
19-05-01	全天	实现行星系统 GUI	完成
19-05-02	全天	实现社交和原子系统 GUI	完成
19-05-03	全天	设计主调用类+312change	312change 未完成
19-05-04	全天	312change	312change 完成
19-05-05	全天	实验报告书写	完成

## 5 实验过程中遇到的困难与解决途径

遇到的难点	解决途径
正则表达式比较困难	自学+练习
GUI 设计比较困难，打算使用插件 JformDesigner	插件咋都不会用，白白浪费了好多天的时间，决定手动书写 GUI 代码
设计模式使用	仔细复习上课所学的设计模式思想，不断尝试

## 6 实验过程中收获的经验、教训、感想

### 6.1 实验过程中收获的经验教训

- 真的不能拖延，一个月的代码量真是多得可怕
- 还是要不断提高自己的代码熟练度
- 不断进行自我学习与自我完善
- 要学会阅读 API 文档

### 6.2 针对以下方面的感受

- (1) 重新思考 Lab2 中的问题：面向 ADT 的编程和直接面向应用场景编程，你体会到二者有何差异？本实验设计的 ADT 在五个不同的应用场景下使用，你是否体会到复用的好处？
- (2) 重新思考 Lab2 中的问题：为 ADT 撰写复杂的 specification, invariants, RI, AF，时刻注意 ADT 是否有 rep exposure，这些工作的意义是什么？你是否愿意在以后的编程中坚持这么做？
- (3) 之前你将别人提供的 API 用于自己的程序开发中，本次实验你尝试着开发给别人使用的 API，是否能够体会到其中的难处和乐趣？
- (4) 在编程中使用设计模式，增加了很多类，但在复用和可维护性方面带来了

收益。你如何看待设计模式？

- (5) 你之前在使用其他软件时，应该体会过输入各种命令向系统发出指令。本次实验你开发了一个解析器，使用语法和正则表达式去解析输入文件并据此构造对象。你对语法驱动编程有何感受？
- (6) Lab1 和 Lab2 的大部分工作都不是从 0 开始，而是基于他人给出的设计方案和初始代码。本次实验是你完全从 0 开始进行 ADT 的设计并用 OOP 实现，经过三周之后，你感觉“设计 ADT”的难度主要体现在哪些地方？你是如何克服的？
- (7) 你在完成本实验时，是否有参考 Lab4 和 Lab5 的实验手册？若有，你如何在本次实验中同时去考虑后续两个实验的要求的？
- (8) 关于本实验的工作量、难度、deadline。
- (9) 到目前为止你对《软件构造》课程的评价。
  - 1, 体会到了复用的好处，极大的降低了代码重叠率
  - 2, 设计模式提高了代码可维护性以及可改变性，比较好，决定以后多多使用
  - 3, 感觉正则表达式十分强大，需要多加掌握
  - 4, 本实验工作量较大，但 deadline 合适
  - 5, 设计 ADT 困难在一开始的构思：多画图，多找共性与异性。