

**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 | 杜钰欣 |
| 学号 | 1170300320 |
| 班号 | 1703003 |
| 电子邮件 | 1485422651@qq.com |
| 手机号码 | 18745718808 |

**目录**

[1 实验目标概述 1](#_Toc3922069)

[2 实验环境配置 1](#_Toc3922070)

[3 实验过程 1](#_Toc3922071)

[3.1 待开发的三个应用场景 1](#_Toc3922072)

[3.2 基于语法的图数据输入 1](#_Toc3922073)

[3.3 面向复用的设计：CircularOrbit<L,E> 1](#_Toc3922074)

[3.4 面向复用的设计：Track 2](#_Toc3922075)

[3.5 面向复用的设计：L 2](#_Toc3922076)

[3.6 面向复用的设计：PhysicalObject 2](#_Toc3922077)

[3.7 可复用API设计 2](#_Toc3922078)

[3.8 图的可视化：第三方API的复用 2](#_Toc3922079)

[3.9 设计模式应用 2](#_Toc3922080)

[3.10 应用设计与开发 2](#_Toc3922081)

[3.10.1 TrackGame 2](#_Toc3922082)

[3.10.2 StellarSystem 2](#_Toc3922083)

[3.10.3 AtomStructure 2](#_Toc3922084)

[3.10.4 PersonalAppEcosystem 2](#_Toc3922085)

[3.10.5 SocialNetworkCircle 2](#_Toc3922086)

[3.11 应对应用面临的新变化 2](#_Toc3922087)

[3.11.1 TrackGame 3](#_Toc3922088)

[3.11.2 StellarSystem 3](#_Toc3922089)

[3.11.3 AtomStructure 3](#_Toc3922090)

[3.11.4 PersonalAppEcosystem 3](#_Toc3922091)

[3.11.5 SocialNetworkCircle 3](#_Toc3922092)

[3.12 Git仓库结构 3](#_Toc3922093)

[4 实验进度记录 3](#_Toc3922094)

[5 实验过程中遇到的困难与解决途径 3](#_Toc3922095)

[6 实验过程中收获的经验、教训、感想 4](#_Toc3922096)

[6.1 实验过程中收获的经验和教训 4](#_Toc3922097)

[6.2 针对以下方面的感受 4](#_Toc3922098)

# 实验目标概述

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，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）。

# 实验环境配置

简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

# 实验过程

请仔细对照实验手册，针对每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## 待开发的三个应用场景

首先请列出你要完成的具体应用场景（至少3个，1和2中选一，3必选，4和5中选一，鼓励完成更多的应用场景）。

* TrackGame
* AtomStructure
* SocialNetworkCircle

分析你所选定的多个应用场景的异同，理解需求：它们在哪些方面有共性、哪些方面有差异。

共性：

1. 三个应用要求有某些相同的功能，例如增加轨道，删除轨道，向某条轨道添加物体，将某个物体删除。
2. 三个应用的图像显示是类似的，都是同心圆上多个物体分布在不同的轨道上

差异：

1. 每个应用都有不同的独特应用，例如田径比赛生成分组情况；原子结构中的电子可以发生跃迁；社交网络中各轨道的人之间也有社交关系，并有关系参照值，信息扩散度。
2. 每个应用的中心物体类型不同
3. 每个应用轨道物体类型不同

## 基于语法的图数据输入

此项用到的是正则表达式

TrackGame

游戏类型：String reg0 = "Game ::= ([0-9]+)";

跑道数目：String reg1 = "NumOfTracks ::= ([0-9]+)";

运动员信息：

Stringreg2="Athlete ::=<([^,]+),([^,]+),([^,]+),([^,]+),([^,]+)>";

AtomStructure

元素名称String reg0 = "ElementName ::=\\s+([A-Z])([A-Za-z0-9])?";

轨道数目String reg1 = "NumberOfTracks ::=\\s+([0-9])";

轨道号及对应电子数量String reg2 = "NumberOfElectron ::=\\s+(((\\d)/(\\d)+;)+((\\d)/(\\d)+))";

SocialNetworkCircle

中心人物String reg0 = "CentralUser ::= <([\\w]+),([0-9]+),([M|F])>";

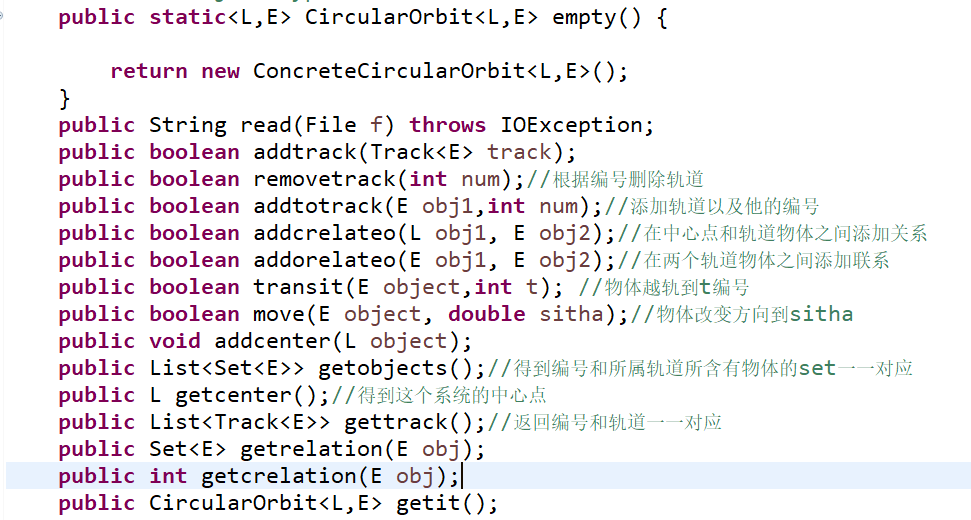
轨道人物String reg1 = "Friend ::= <([\\w]+),([0-9]+),([M|F])>";

社交关系String reg2 = "SocialTie ::= <([^,]+),([^,]+),([^,]+)>";

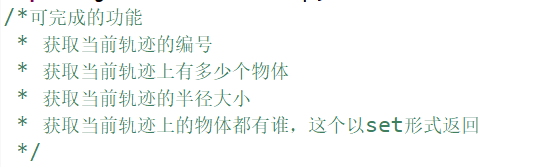
## 面向复用的设计CircularOrbit<L,E>

一个可以被实现成为具体类的接口

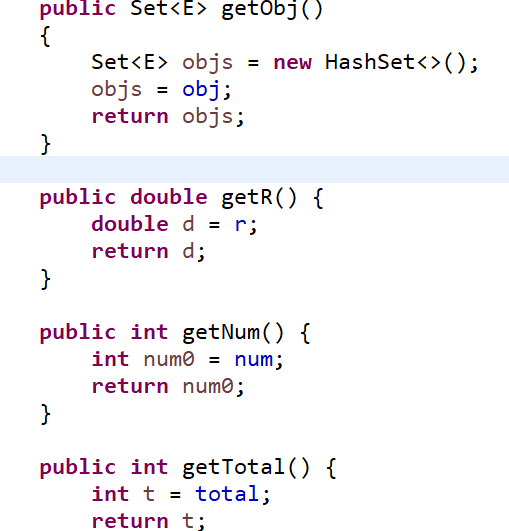
实现的功能有：



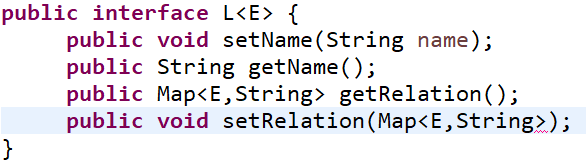
## 面向复用的设计：Track



三个应用共用一个轨道系统

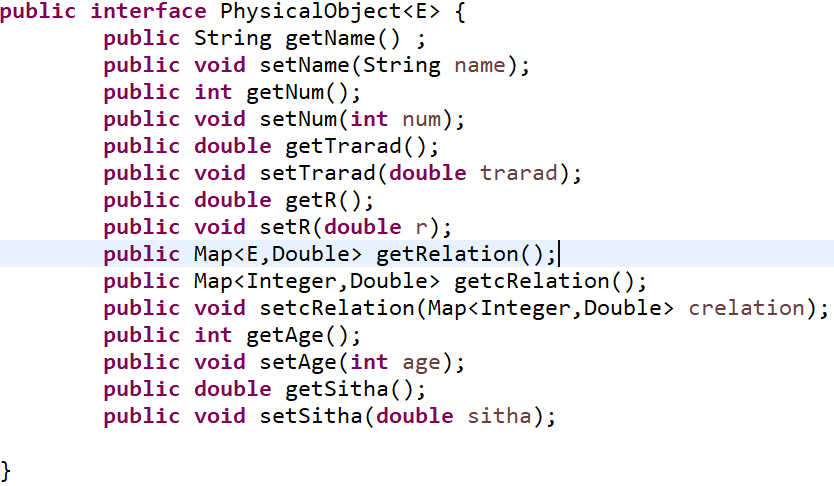


## 面向复用的设计：L



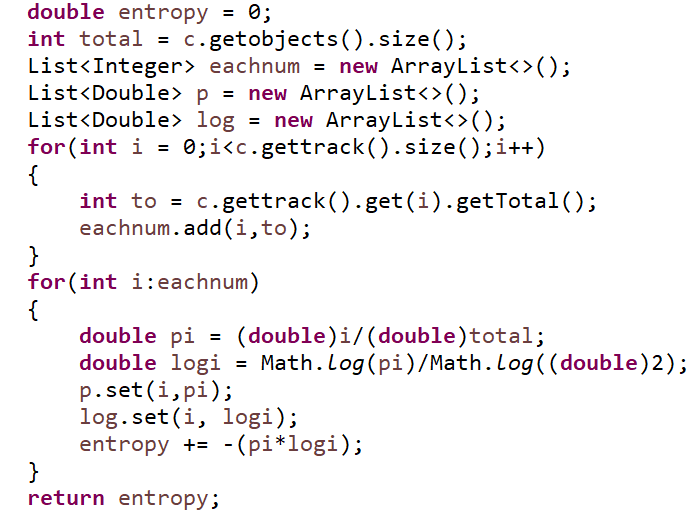
在具体的类中实现这个接口的功能，再添加上每个应用独特的功能。在具体实现类中变量定义为private，获取的时候定义一个新变量作为中介，从而防止其暴露

**面向复用的设计**：PhysicalObject



## 可复用API设计

计算多轨道系统中个轨道上物体分布的熵值

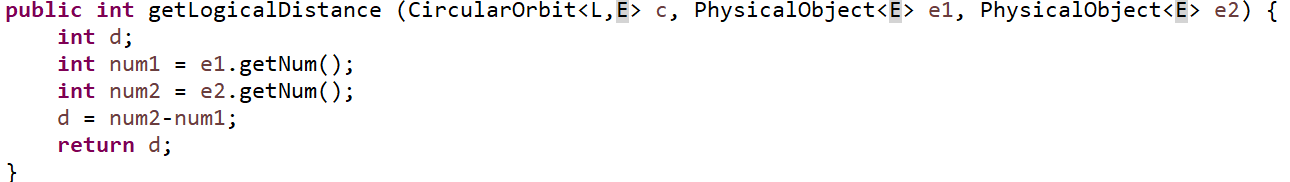


运行结果实例：原子结构



计算任意两个物体之间的最短逻辑距离

就是计算两个物体所在轨道编号的差



计算任意两个物体之间的物理距离

通过每个物体Position的值横坐标平方和纵坐标平方加和开方即可

计算两个多轨道系统之间的差异

多轨道差异分两种情况，一种是轨道物体之间不互相区分一种是轨道上的物体是互不相同的

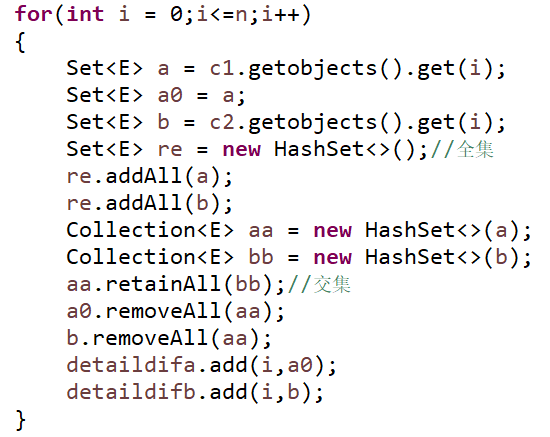
在这里我新建了一个单独的差异类，将差异存储在此类中

轨道上物体不互相区分指的就是原子和田径比赛这种轨道

只需要分别比较每条轨道上有多少个物体然后分别存储起来就可以了

轨道上物体互不相同指的是类似于社交网络的轨道

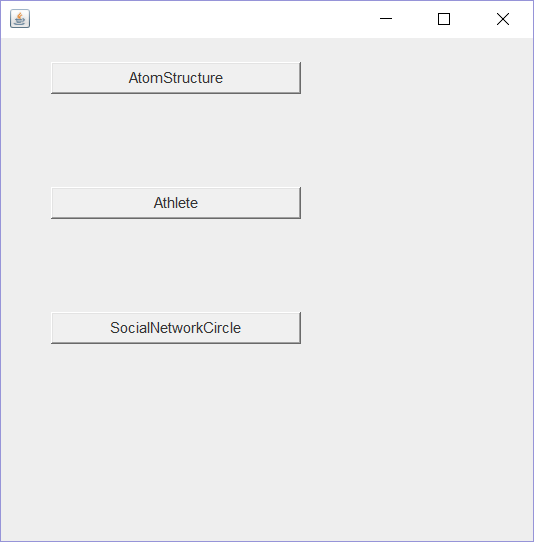
在这种轨道上进行比较的时候要用到集合的交并运算，需要用到collection库中的几个方法



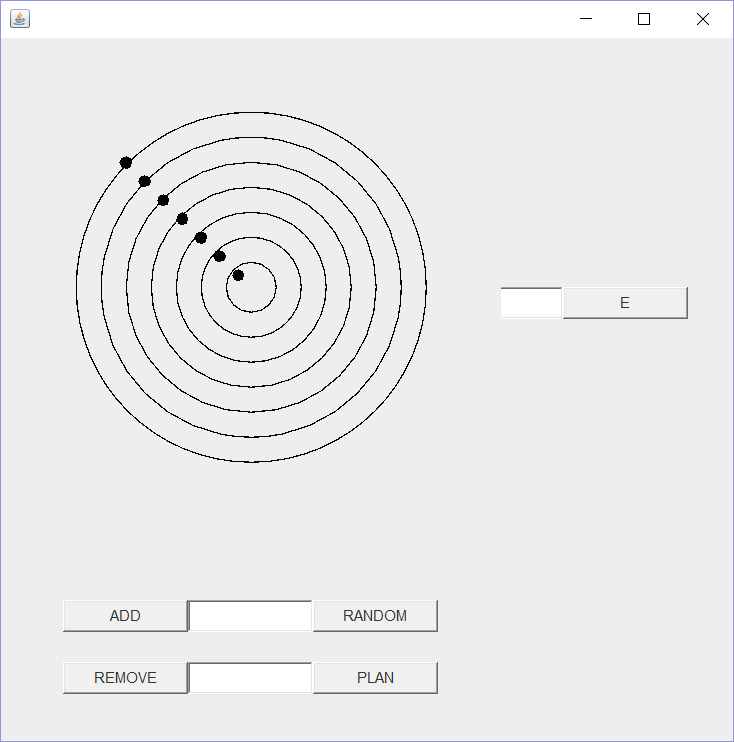
## 图的可视化：第三方API的复用

在此类中我用到的第三方API是Panel,不需要额外导入新的包以供操作，只要extends Panel即可

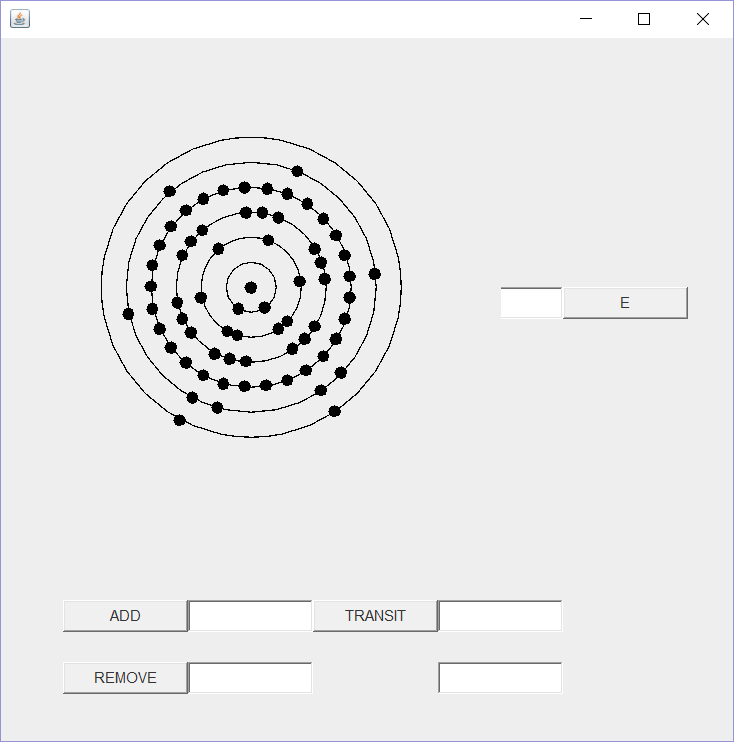
初始界面



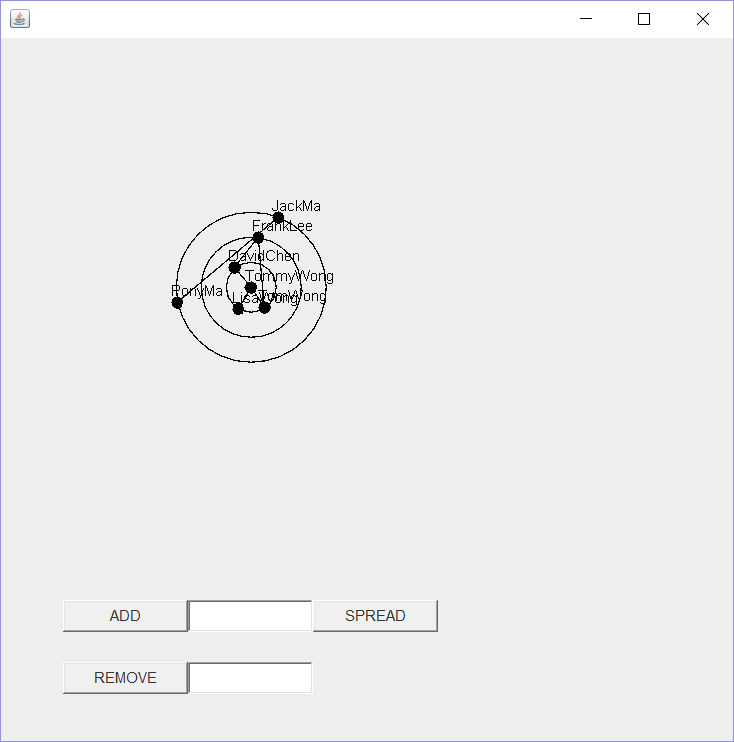
田径比赛可视化



原子可视化

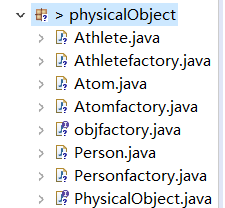


社交网络可视化



## 设计模式应用

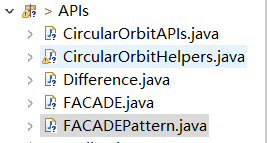
工厂模式：在轨道物体中应用，设计抽象产品，抽象工厂，具体工厂，具体物体的类，抽象工厂提供产品的生产即返回抽象产品类，不同的轨道物体实现抽象产品之后添加自己特有的功能，不同的具体工厂实现抽象工厂，返回相应的具体工厂生产的具体产品。



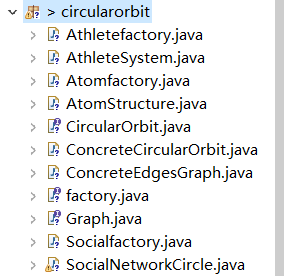
备忘录模式：这项模式应用在原子结构应用上，运用这种模式可以记录下用户在改变电子位置发生跃迁前和后原子都处于哪条轨道上。备忘录模式包括发起人，备忘录和管理者，发起人就是改变电子位置，使其目前为止改变的类，备忘录实时的记录下每个时刻电子都在那个位置上，而管理者则是可以设置备忘录和获取当前备忘录内容。



门面模式：将API要实现的所有功能放在一起作为外观角色，他们的共同类叫做FAÇADE，当用户想要使用某种功能的时候只需要在一个外部的FACADEPattern类里面整个的调用FAÇADE即可，在FAÇADE类中的多个功能又分别由具体实现的类，这样就避免了用户在使用的时候需要自己一个一个的调用功能，如果使用某项功能只需要全部计算一遍然后找到自己需要的数据即可



抽象工厂模式：与工厂模式不一样的就是一个抽象工厂可以生产多个产品，在此实验中一个每个应用是一个抽象工厂，每个抽象工厂都能产生轨道物体和中心物体两个类，其中 每个具体应用实现抽象工厂类，每个具体的中心物体和轨道物体根据上面的工厂模式可以得到

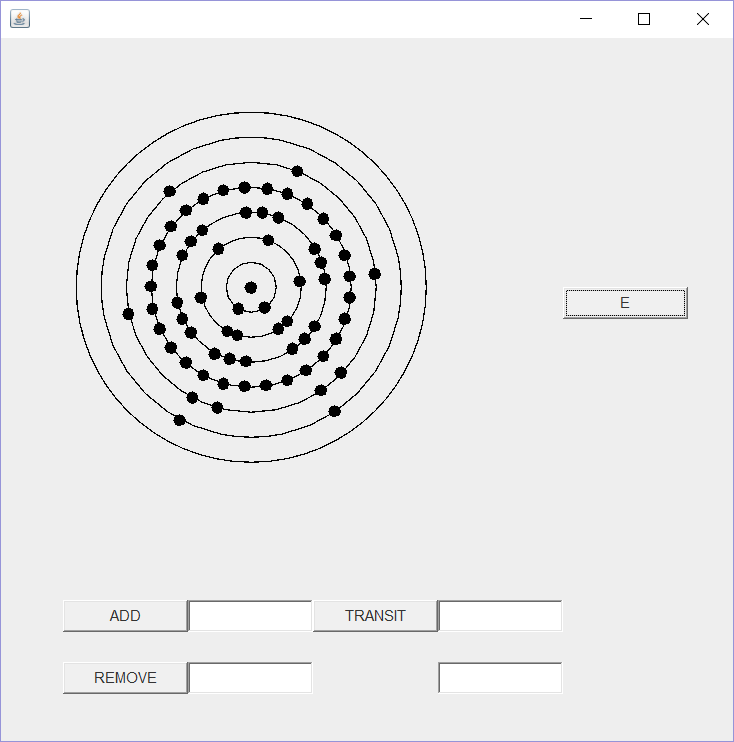


## 应用设计与开发

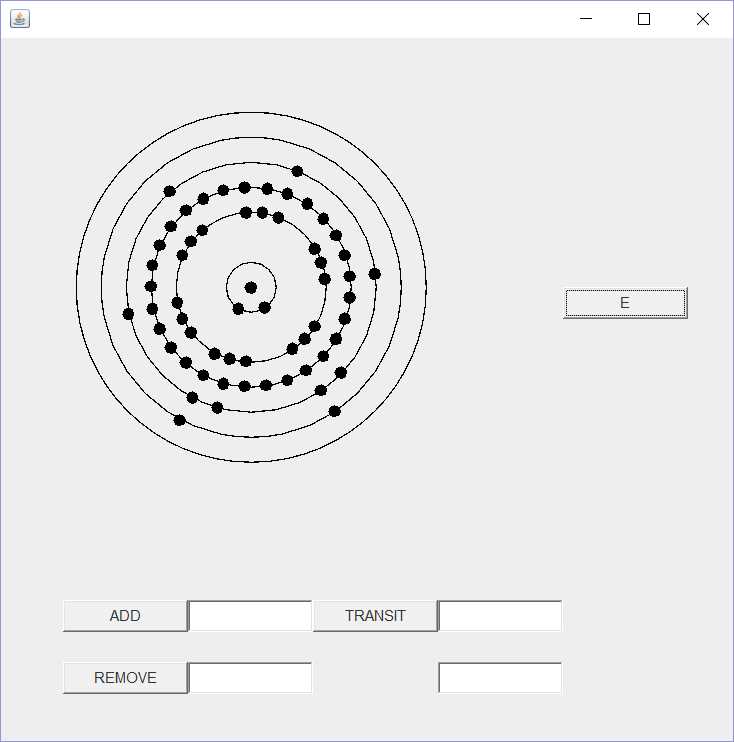
共同功能

（向轨道添加物体和从轨道删除物体时跃迁的两个步骤，这里不用可视化表示）

添加轨道



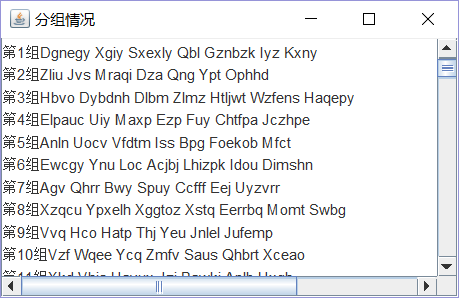
删除轨道



### TrackGame

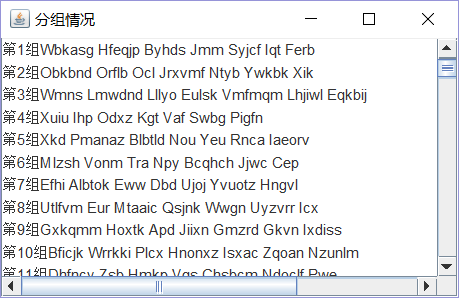
功能一：自动编排比赛（方案一：完全随机；方案二：按运动员成绩排序）

方案一：从左至右跑道从内圈到外圈



方案二：从左至右跑道从内圈到外圈

由于编排后的比赛要求成绩好的人所处赛道越靠中间，所以要先取轨道数量的中位数，依次向两边安排赛道，直到最小为1最大为赛道总数量



功能二：手动调整比赛

交换Ujehe和Uschht两人的组

在AthleteSystem中加入函数exchange



结果如下

交换前



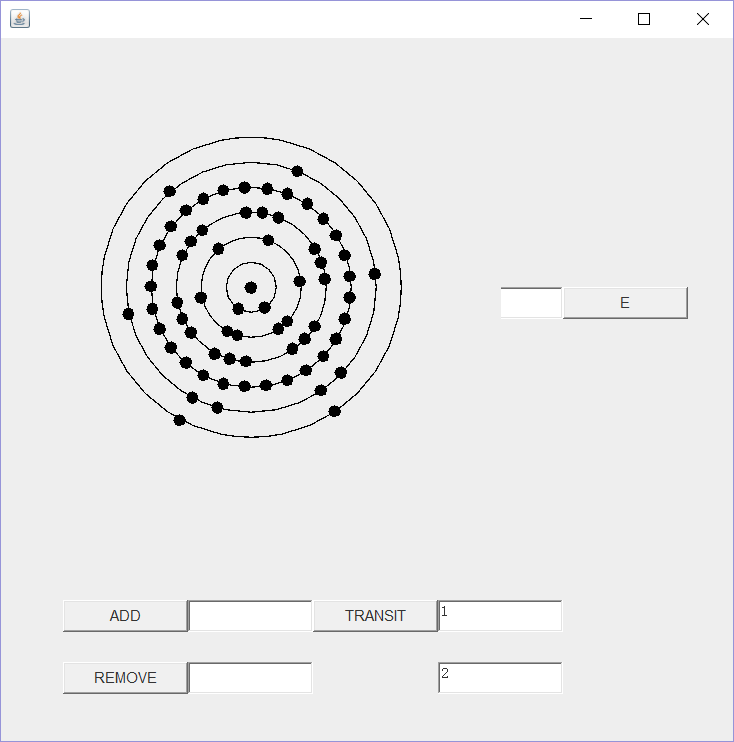
交换后



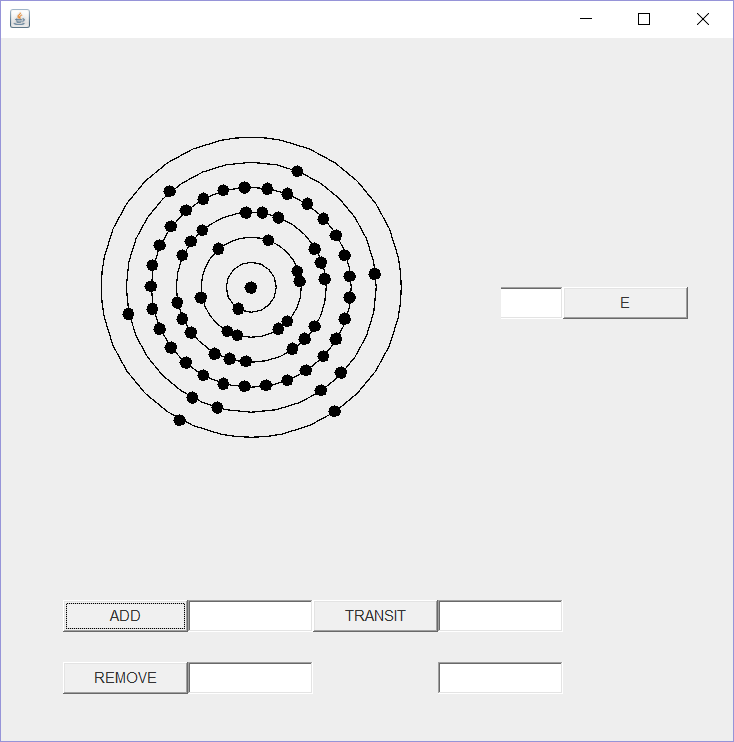
### AtomStructure

给定源轨道，目标轨道，模拟电子跃迁

跃迁前



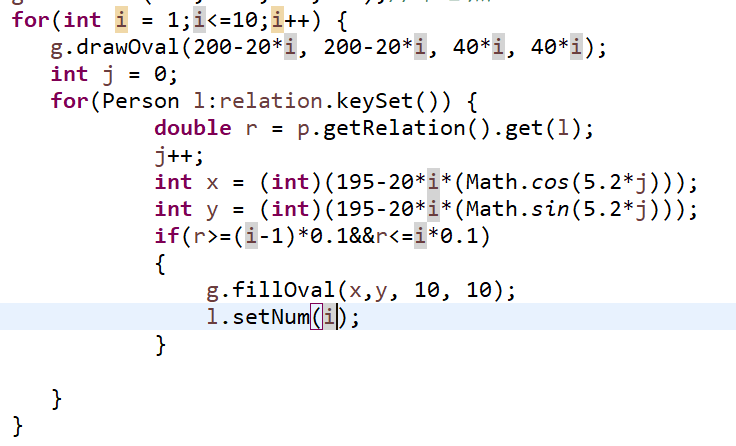
跃迁后



### SocialNetworkCircle

功能一：判定每个用户是在哪个轨道上

在画图的时候，我根据轨道人物与中心人物亲密程度分配轨道，每画一个点都确定这个点的轨道编号



功能二：计算某个处于第一条轨道上的某个好友的信息扩散度

所用策略：根据第一条轨道上人物向外延申的好友的亲密度加和即可

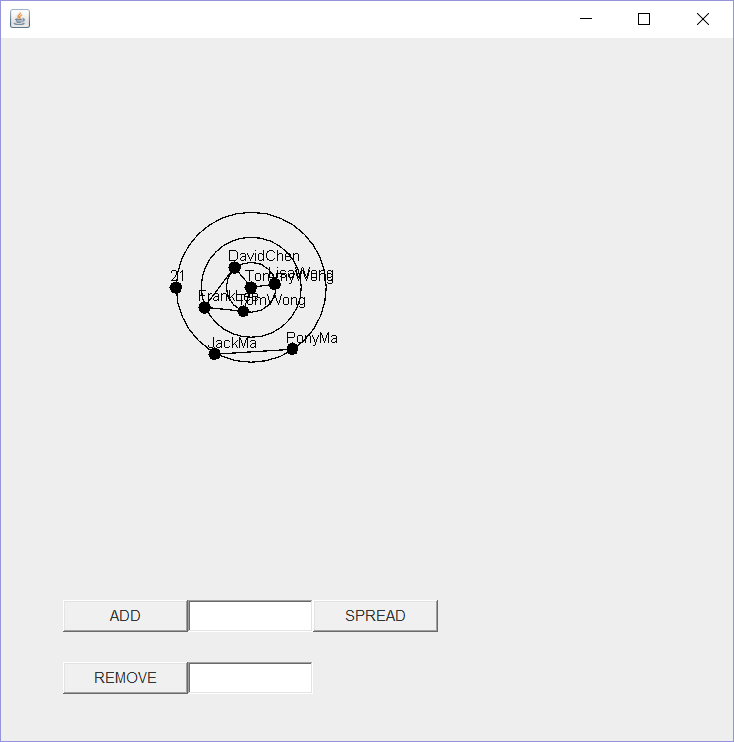
因为关系亲密度越低加和结果也越小，正好满足信息扩散度低



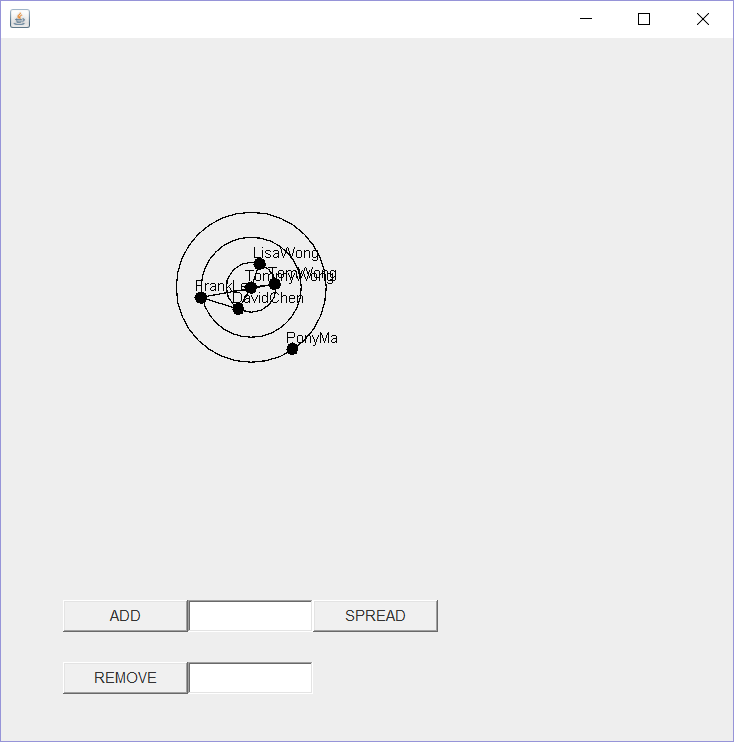
功能三：增加或删除一条社交关系

在此功能中我使用了lab2中的图结构，把每个人当作一个点，每种关系是一条有向边，所以增加或删除关系利用lab2就是重新建构一个graph这个在lab2中已经实现

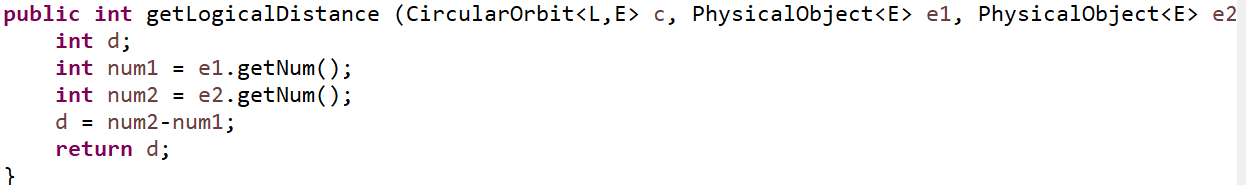
增加一个人物名字为“21”



删除关系中的一个人物名字为“JackMa”可以发现原来和PonyMa相连的线已经消失



功能四：计算两个轨道上用户之间的逻辑距离



## 应对应用面临的新变化

以下各小节，只需保留和完成你所选定的应用即可。

### TrackGame

需求变化：可支持接力比赛（每个跑道一支队伍，每支队伍 4 人），从而每个跑道上可以出现 4 个物体，但无需 考虑 4 个物体的绝对位置

更改：将每次循环录入组别的人数调整一下即可



### AtomStructure

需求：原子核需要表达为多个质子和多个中子

更改：添加质子和中子的集合，分别设置Getter和Setter方法

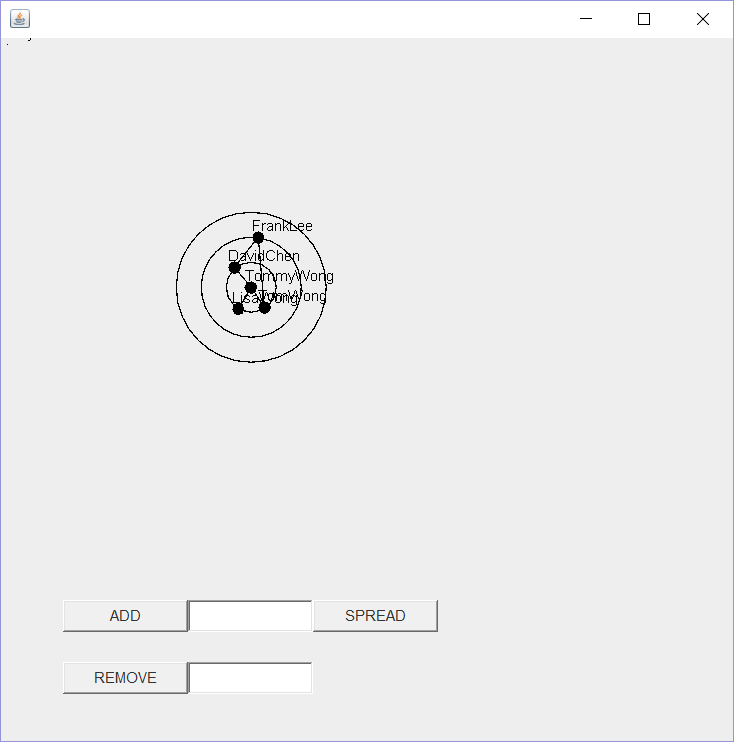




### SocialNetworkCircle

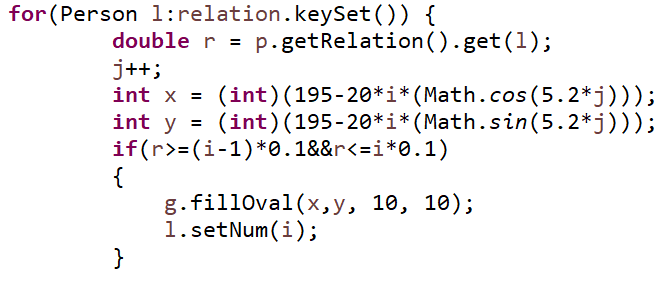
需求：只考虑中心点向外的人际关系，忽略非中心人物的社交关系

更改：单独取出中心人物，只考虑由他发起的关系

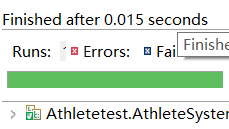




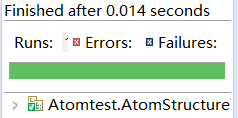




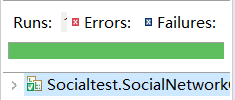
**测试**



田径比赛的测试主要检查了添加轨道，删除轨道，以及将某个运动员添加到某条轨道上，交换两个运动员的跑道，以及按成绩排跑道这些功能



原子结构的测试主要检查了添加轨道删除轨道以及电子跃迁这些功能



社交网络的测试主要检查了添加轨道删除轨道以及建立网络上两人物之间的关系，这种关系分包含中心人物和不包含中心任务两种情况，最后还对信息扩散度进行了检测。

## Git仓库结构

请在完成全部实验要求之后，利用Git log指令或Git图形化客户端或GitHub上项目仓库的Insight页面，给出你的仓库到目前为止的Object Graph，尤其是区分清楚312change分支和master分支所指向的位置。

# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 4.15 | 19:00-23：00 | 编写circularorbit | 完成 |
| 4.16 | 19：00-23：00 | 设计轨道类 | 完成 |
| 4.17 | 19：00-23：00 | 设计中心物体和轨道物体类 | 完成 |
| 4.18 | 18：00-23：00 | 设计api | 未完成 |
| 4.19 | 18：00-23:00 | 设计API | 完成 |
| 4.20 | 13：00-23：00 | 修改中心物体和轨道物体类 | 未完成 |
| 4.21 | 13：00-23：00 | 修改中心物体和轨道物体类 | 完成 |
| 4.28 | 18：00-23：00 | 更改行星为田径比赛并完成改动 | 完成 |
| 4.29 | 19：00-23：00 | 完成原子结构的可视化 | 完成 |
| 4.30 | 22：00-3：00 | 完成田径比赛的可视化以及编排人员 | 完成 |
| 5.1 | 10：00-16：00 | 完成社交网络的可视化部分 | 完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 设计轨道的过程中由于实验要求Immutable所以一开始设计的有错误 | 在实验要求改动之后，设计变得简单方便了很多 |
| 选择的应用都是初始位置没有设置角度的，随机分布在轨道上的函数没有设置好 | 在同一条轨道上记录一个点就给一个值加一利用这个值得到一个sin或cos这样可以固定在同一条圆形轨道上 |
| 因为map可以定义成hashmap的原因，在将map传进或者传出的过程中总是遇到指向空指针的异常 | 上网查阅相关异常和解决方法之后，每一次都明确map为hashmap这样就不存在从map指向hashmap或者是从hashmap指向map的情况出现，空指针异常得以解决 |
| 结构认识不清，代码整体重构 | 与同学交流之后，重新确定实验代码的结构，重写 |
| 社交网络的轨道上人员安排问题 | 某些人与中心点没有关系却与轨道上物体有关，这些人按照亲密程度乘以有关系的轨道人员，再确定轨道位置，这样就完美的利用了所有的轨道 和信息。 |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训

在写代码之前首先要对本次实验代码的结构已经确定好，不然写着写着就会发现整体的结构有错误，这就会使你之前花费的时间都白白浪费掉起不到一点作用。先将每次实验用到的知识了解清楚再开始动手，本次实验模式部分因为之前并不懂怎么建立所以也没有加入模式，使得后来加入模式麻烦了许多，远远不如刚开始定好结构，按模式来写方便。

## 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？

真正体会到了复用的好处，因为可以通过实现接口和继承父类所以省去了很多三个应用都需要完成的功能的书写，再加上泛型的应用使得整个程序的结构更加清晰，在总体的功能需要更改的时候变得更加方便。

1. 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？

每一次书写的时候都是在提醒自己定义变量时候的注意事项，使用的时候注意不要进行违背规定的操作。

1. 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？

自己使用别人开发的API的时候要严格按照别人指定的过程进行使用，但使用别人的API毕竟比自己从零开始要方便许多。当自己开发API的时候要注意保护好某些不想被他人更改的参数，所以在设计的时候要格外注意，自己开发的API可以随时根据实际情况更改其中的操作甚至是变量类型，修改更加随意。

1. 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？

设计模式虽然看起来使得类变得更加复杂，但如果仔细分析，其实在设计模式是将各个类分割成几部分，使得在调用的时候更加安全和方便，模式的应用还使得编写代码的程序员在对代码做出改进和修复的时候更加简单，不用一次性删减或者增加大量的新代码

1. 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？

语法驱动编程这项操作在限制了某些输入规范之后，用户可以一次性在文本中输入所有想录入的信息，并且在之后可以反复使用，如果没有运用这项操作的话，需要每次运行都在控制台输入大量的信息以供使用，非常的麻烦而且在第二次想使用同一组数据的时候还不方便。

1. 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？

设计ADT我认为关键在于首先要知道现在所设计的ADT是一个具体类还是接口还是抽象类，他在整个代码结构中的用处是什么，起到什么样的作用，然后进行设计的时候注意编写注释，时刻提醒自己运用的时候要注意什么。

1. 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？

完成本实验的时候还没仔细看过实验4和5的实验手册，但在和同学交流的过程中了解到一些，希望设计模式和接口复用之类的设计可以对之后的4，5有帮助。

1. 关于本实验的工作量、难度、deadline。

本实验工作量很大，在开始实验的时候甚至在课堂上老师还没有讲关于设计模式部分的内容。刚开始写这个实验的时候对于整体结构把握的也不到位，进行了重构很多次，最后通过和同学交流才将整个实验结构理解清晰。本实验的难度不仅体现在没有任何给定的框架需要自己来设计，还体现在这是一个综合性很强的实验，它不仅需要完成基本计算，手写代码解析给好的信息，还需要将基本计算的结构带入图中，画出一个包含信息的图像，甚至在生成图像之后还修改了之前的信息需要重新进行调整。但看这次实验的截止日期还算合理但由于在实验截止前和截止后都分别有考查课的考试，使得这次的实验做起来时间很紧张，在后期有些慌乱。

1. 到目前为止你对《软件构造》课程的评价。

实验三结束后本学期的一般课程已经结束，软件构造这门课程授课内容很多，不及时复习和实践来巩固的话是得不到一个好的效果的，但在其他课程也很繁忙的情况下，只能通过认真完成实验来得到熟悉内容的目的，还是希望老师在课堂上多加入一些实验的内容，这样在开始每个实验的时候不会很迷茫不知道怎么下手。