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	李大鑫
学号	1170300825
班号	1703008
电子邮件	
手机号码	

目录

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

环境：IDEA，JFormerDesigner ，swing

Lab3 地址：https://github.com/ComputerScienceHIT/Lab3-1170300825

3 实验过程

3.1 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）需要轨道系统中基本存在的对象，包括轨道、中心物体、轨道物体。其中物体都不考虑绝对位置。轨道都为圆形。

2）都需要完成的功能有：添加/删除轨道，在某一轨道上添加/删除物体，获得轨道系统的熵值，获得逻辑距离，比较两个同类型轨道系统的差异，检查轨道系统是否合法，可视化。

差异：

1）TrackGame 需要构造多个轨道系统，而其他两个只需要从文件中读入一个。

2）TrackGame 需要实现编排策略需要实现对轨道物体交换组/轨道系统。AtomStructure 轨道物体都是值相同的对象，需要实现物体跃迁。SocialNetworkCircle 中需要实现物体关系及对应操作，需要计算信息扩散度。

3.2 基于语法的图数据输入

以下分别是三个应用的在输入处理中设计的正则表达式：

TrackGame：

```
String athletePattern = "Athlete\\s*::=\\s*<([a-zA-Z]+),(\\d+),([A-Z]{3}),(\\d+),(\\d{1,2}\\.\\d{2}+)>";
String gamePattern = "Game\\s*::=\\s*([100|200|400,\\]+)";
String tracksPattern = "NumOfTracks\\s*::=\\s*([4-9,10]+)";
```

AtomStructure：

```
String elementNamePattern = "ElementName\\s*::=\\s*([A-Z]{1}[a-z]{0,1})";
String tracksPattern = "NumberOfTracks\\s*::=\\s*(\\d+)";
String electronPattern = "NumberOfElectron\\s*::=\\s*([\\d+\\/\\d+,;]+)";
```

SocialNetworkCircle：

```
String centralUserPattern = "CentralUser\\s*::=\\s*<([A-Za-z0-9]+),\\s*(\\d+),\\s*([MF])>";
String friendPattern = "Friend\\s*::=\\s*<([A-Za-z0-9]+),\\s*(\\d+),\\s*([MF])>";
  使用?:表示不捕获该括号内的元素（非捕获组），如果出现嵌套的括号，匹配规则是由左向右，由外向内
String socialTiePattern = "SocialTie\\s*::=\\s*<([A-Za-z0-9]+),\\s*([A-Za-z0-9]+),\\s*([01](?:\\.[1]\\d{1,3}){0,1})>";
```

对于在文件中读入的每一行，尝试用不同的 pattern 进行匹配并进行对应的处理（包括 split），如果不能匹配则抛出运行时异常 MyExp（自定义异常）。

其中 SocialTie 的正则表达式中用到了?:代表该括号是一个非捕获组，对于有多个括号的捕获情况，捕获顺序是从左到右，从外到内。

3.3 面向复用的设计：`CircularOrbit<L,E>`

类的域：

域名	作用
L centralObject	中心物体
Map<Track,List<E>> physicalObjectMap	轨道->轨道上物体的映射
List<Relation<L,E>>relOfCobj2TraObj	中心物体与其朋友的关系
Map<E,List<Relation<E,E>>> relOf2TraObjs	轨道物体与周围朋友的关系

方法以及实现：

方法名	方法实现
Public Boolean addTrack(Track newTrack)	向 physicalObjectMap 中添加新轨道
Public Boolean removeTrack(Track rmTrack)	从 physicalObjectMap 中删除轨道，轨道上物体随之被删除，调用 removePhysicalObject。
Public void addCentralObject(L co)	添加中心物体
Public void AddPhysicalObj2Track(E po,Track tk)	向轨道 tk 上添加新物体 po，修改 relOf2TraObjs
Public void addRelationOfCentralObj2TrackObj(L co,E po,double weight)	向 relOfCob2TraObj 中添加关系
Public void removeRelationOfCentralObjs2TraObj(L eo,E po)	在 relOfCob2TraObj 中删除该关系。
Public boolean addRelationOf2TrackObs(E po1,E po2,double weight)	在 relOf2TraObjs 添加该关系
Public Boolean removeRelationOf2TrackObs	在 relOf2TrackObs 中删除该关系
Public void transit(E oldObj,E newObj,Track t)	将原来的轨道物体 oldObj（Immutable)修改基本信息后生成的新物体 newObj 放到轨道 t 上。
Public void move(E oldObject,E newObject)	将原有的轨道物体 oldObject 修改基本信息为新的轨道物体 newObject。
Public double getObjectDistributionEntropy()	获取轨道系统的熵值，每一条轨道上的物体数目/轨道系统所有物体数目作为 p,使用 entropy=-sigma(pi*log2(pi))公式计算轨道系统熵值。
Public int getLogicalDistance(E	使用 BFS 算法计算 e1 和 e2 之间

e1,E e2)	的逻辑距离。
Public Difference getDifference(CircularOrbit<L,E> that)	获得自身轨道系统与同类型轨道系统 that 之间的系统差异。获得两者经过排序之后的轨道（这里的轨道比较按照由内向外因此比较而不是按照半径相等对应比较），将进行比较的两轨道传入 Difference 对象中，交给 Difference 对象处理。最后返回 Difference 对象。
Public Boolean checkOrbitAvailable()	检查系统是否合法，直接返回 true 交给具体应用系统进行重写。
Public Iterator<E> iterator()	返回 iterator 迭代器。

注意到实验要求中的文件读入是不符合本实验中三个应用的共性的，因为 TrackGame 需要读入多个轨道系统，于是另设计构造类分别是 TrackGame,AtomStructure,SocialNetworkCircle，负责读入文件，调用 builder 构造轨道系统（因此构造类中包含具体轨道系统的引用），负责作为 GUI 与 Orbit 轨道系统中的桥梁（获得 Orbit 基本信息，响应 GUI 事件，刷新 GUI 显示信息）。

3.4 面向复用的设计：Track

Immutable 类。只有一个 radius 域。

方法：

getter，静态工厂方法 getInstance，compareTo（Comparable 接口），equals，hashCode。

其中，覆盖 equals 方法，Track 使用 Radius 判断两个 Track 是否相等。

3.5 面向复用的设计：L

构造 CommonObject 作为 CentralObject 和 PhysicalObject 的子类。其中包括 obName 与 pos 分别代表物体名称以及物体位置，Position 是 Immutable 类型。

方法：getter，drawGraphics,hashCode,toString

声明：Public void drawGraphics(Graphics g,GraphicsPainter painter)：接受 Graphics 类与画笔信息类在 g 上根据画笔信息画出物体，所谓画笔信息类是一个 mutable 类，承载物体中心在 g 上的位置、物体绘制颜色、字体信息。在物体类中声明绘制方法是想将具体绘制方法委托到具体的类，而不是通过获取物体信息然后统一绘制。

Immutable 对象类。CentralObject 继承自 CommonObject。
方法：getter，equals

3.6 面向复用的设计：PhysicalObject

Immutable 类，继承自 CommonObject。
方法：equalsObject，compareTo(Comparable 接口)
声明：1）public boolean equalsObject(Object obj)，比较当前 object 与 obj 是否值相等，这里不覆盖 equals，equals 依旧是通过内存地址判断相等。

3.7 可复用 API 设计

1. 计算轨道系统熵值。在 ConcreteCircularOrbit 中已经具体实现。
2. 获取最短逻辑距离。在 ConcreteCircularOrbit 中已经具体实现。
3. 获取物理距离：因为在三个应用中不考虑物理位置，所以不予实现。
4. 计算两个多轨道系统之间的差异：

在 ConcreteCircularOrbit 中我们将两个对应的轨道物体集合添加到 Difference 对象中。下面声明 Difference 的类设计。

每一对比较的轨道，各自形成集合只保留各自轨道上独有的轨道物体（去除交集，这里的比较实用 equalsObject 进行值比较），通过两个集合构造 trackDifference 对象，所以一个轨道系统的 Difference 由多个 trackDifference 构成。在 trackDifference 中提供 toString 方法将差异转化为字串，需要注意的是，如果两个集合都为空则说明两轨道上物体完全相同，这时候不输出“物体差异”。

3.8 图的可视化：第三方 API 的复用

本实验中使用 Swing 实现可视化功能。同时，在简单的轨道系统可视化基础上，添加了一部分简单控件用于优化交互体验。GUI 如图 1-4（图 4 显示了实现了在添加新物体时的 GUI 面板设计，验收的时候出了点问题）

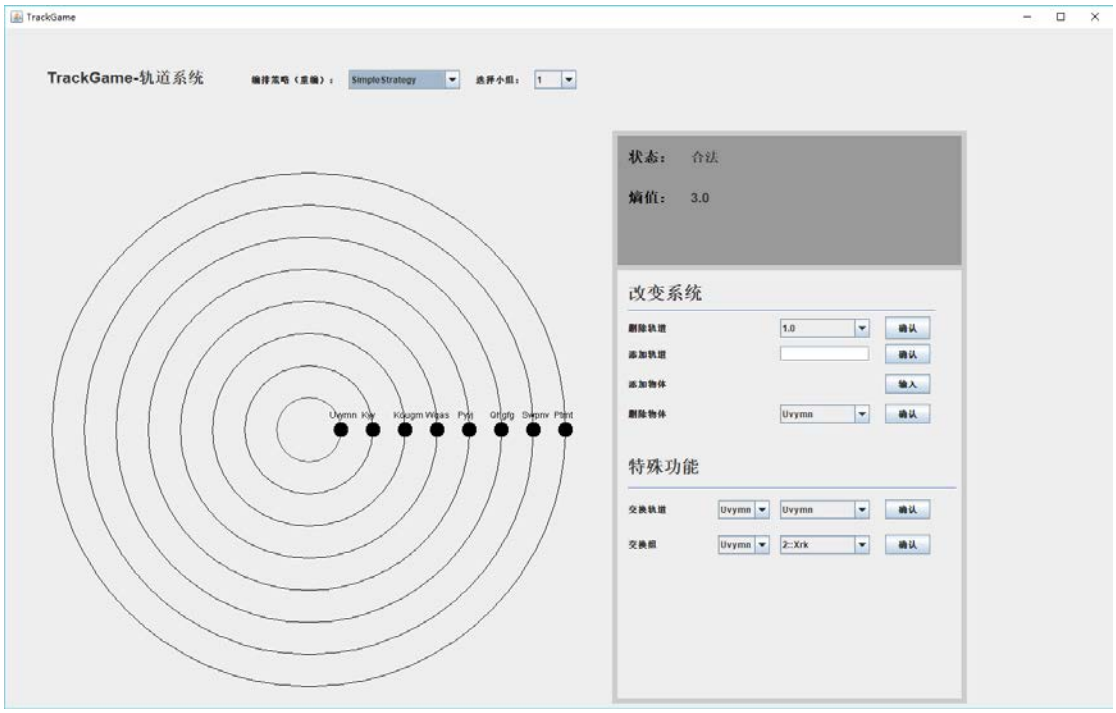

图 1. TrackGame 轨道系统 GUI 实现

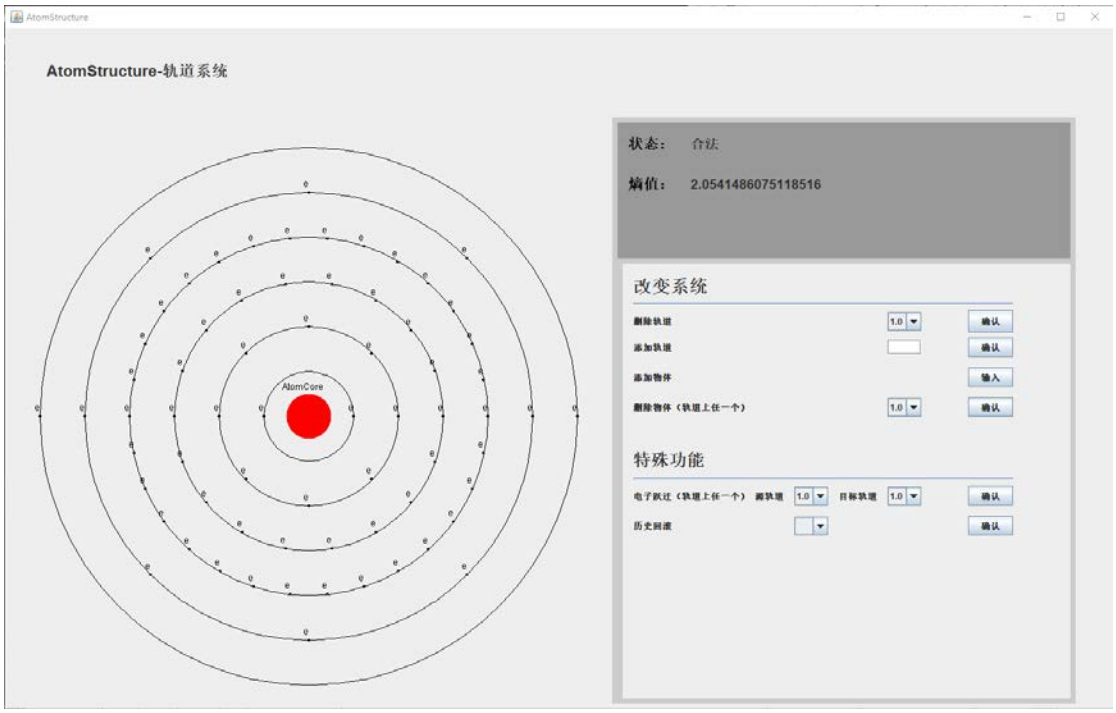

图 2. AtomStructure 轨道系统 GUI 实现

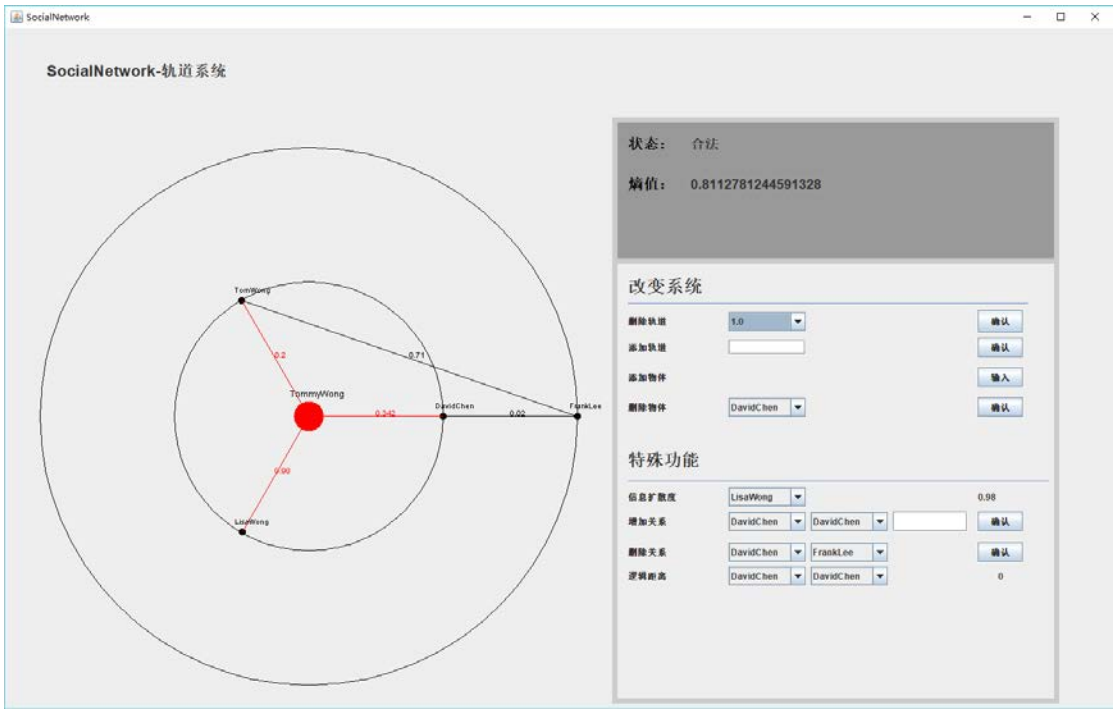

图 3.SocialNetwork 轨道系统 GUI 实现

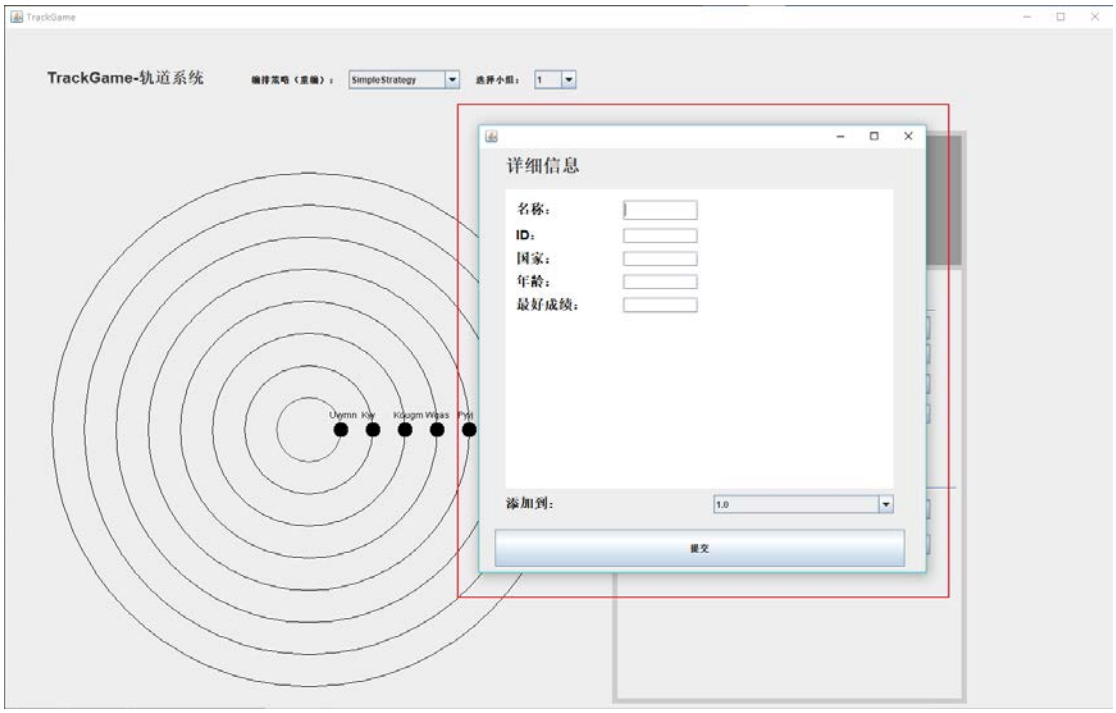

图 4. TrackGame 轨道系统——添加新轨道物体功能面板

GUI 实现思路：

1）使用 IDEA 插件 JFormerDesigner 进行 UI 布局，所谓 UI 布局是为了设计各种控件（Button，ComboBox，Label 等）在面板上的位置，将一个命名为 drawPanel 的 JPanel 放在左侧用来承载轨道系统。

2）对于每一个应用类，实现 visualizeContentPanel 方法，在这个方法中，

构造并返回一个 JPanel，在该 JPanel 中重写 paint 方法，在 paint 中根据当前 Orbit 的具体信息绘制当前轨道系统。visualizeContentPanel 是暴露给 CircularOrbitHelper 的 visualize 使用的，在其中调用 visualizeContentPanel 方法获得 JPanel 只需要将这个将 JPanel 添加到一个 JFrame 中就可以显示该轨道系统。

3）对于每一个应用类，实现 visualize(JPanel panel)方法，该方法将 visualizeContentPanel 中获得到的 Jpanel 添加到 panel 中。这个方法是暴露给像 TrackGame 这样的构造类的，在构造类中获得面板上的 drawPanel 引用，将 drawPanel 传入到应用类的 visualize 方法中即可显示轨道系统。之所以要这样设计是因为首先面板上不只有轨道系统还有很多信息交互的控件，在其中 drawPanel 只是用作布局占位用，可以看做容器，在 Swing 中绘图得通过重写 paint 实例化一个 JPanel 类来实现。

4）布局类（extends Jpanel）。使用 JFormerDesigner 可以通过可视化的方法生成一个布局类（布局可参考图 1-4），在本次实验中，三个应用所实现的都是继承自 JPanel 的布局类，分别为 TrackGamePanel，AtomStructurePanel，SocialNetworkCirclePanel，为布局类添加构造函数，传入构造类的引用。为了实现面板交互功能，添加事件监听器，包括对于不同控件的点击、选择等事件监听，在事件监听的回调函数中，调用构造类中的方法响应事件。

5）构造类。布局类在构造类中被创建，构造类传入引用。在事件响应中，构造类操作调用具体的应用类获得显示信息，然后将信息传入布局类中，布局类接受到信息后刷新控件内容。

GUI 实现缺点：

布局类（目测）不能继承，所以需要分别实现三个布局类，所以需要分别实现三个构造类，冗余代码比较多。

3.9 设计模式应用

3.9.1 高级类

1）Track,PhysicalObject 等对象使用静态工厂方法实现，在类中设计 getInstance 方法获得对象。

2）对于 ConcreteCircularOrbit 的构造使用 builder 设计模式，设计抽象类 ConcreteCircularOrbitBuilder，包括 buildTrack，buildeObjects，buildRelation，通常传入的参数添加轨道系统组成。

每个具体应用类的 Builder 继承自 ConcreteCircularOrbitBuilder，覆盖 createConcreteCircularOrbit 方法本别创建对应的轨道

3）Iterator 设计模式。设计 MyIterator 类，构造方法中传入轨道系统的

PhysicalObjectMap，首先按照轨道排序，之后按照轨道物体位置排序（因为不考虑轨道物体绝对位置，所以不予实现），在其中添加一个迭代下标，hashNext 判断下标是否越界，next 返回移动下标返回值。

4）façade 设计模式，实现 API 类。

3.9.2 TrackGame

实现 strategy 设计模式。

设计接口类 AssignmentStrategy，其中只有一个 assign(List<Track> tracks,List<Runner> runnerList)声明。

分别实现具体类，简单编排 SimpleStrategy，随即编排 RandomStrategy，按 bestscore 排序编排 SortedScoreStrategy。具体类中实现 assign 函数。

3.9.3 AtomStructure

实现 Memento 设计模式管理电子跃迁的状态。

设计 ElectronTransitMemento 类，保存电子轨道跃迁信息，分别保存跃迁电子 electron，源轨道 fromTrack，目标轨道 toTrack。

设计 ElectronTransitCareTaker 类，负责保存所有的动作列表，回退历史信息（返回删除点之后的所有操作）。回退操作主要通过从 CareTaker 类中获得删除点之后的所有历史信息，然后传入轨道系统中反方向执行操作完成。

3.9.4 SocialNetworkCircle

不需要实现设计模式

3.10 应用设计与开发

3.10.1 TrackGame

功能 1：分别将划分轨道系统和轨道的任务委托给 RandomStrategy 与 SortedScoreStrategy 完成编排任务。

功能 2：调整比赛方案，输入两个运动员的名称（这里假定了每个运动的名称都各不相同），调整轨道：调用在某一轨道上删除/添加一个物体的函数即可完成操作；调整轨道系统：首先需要找到各自的轨道系统、轨道，然后在其中删除目标物体，然后将两个物体分别添加到对方的轨道系统的轨道上。

判断合法性：按照要求实现。

3.10.2 AtomStructure

功能：模拟电子跃迁。调用 ConcreteCircularOrbit 中实现的 transit 函数即可。

判断合法性：无。

3.10.3 SocialNetworkCircle

功能 1：从社交关系日志中恢复结构，首先将所有的输入信息读入，这时候我们得到了中心节点，与中心节点相邻的节点，节点之间的关联信息，在这个图上进行 BFS，BFS 的起始节点集合是所有与中心点相邻的节点，于是我们就可以获得哪些节点与中心节点不相邻（删去），各自的节点应该处于哪一条轨道上。之后调用 Builder 构造轨道系统即可。

功能 2：计算第一条轨道上的亲密度因素：这里我们定义第一层轨道上的物体为 V，其他物体为 U，记 v->u 路径上紧密度的乘积为 val(v,u)，则 v 的扩散度为其所有能到达的物体 u’的 val(v,u’)之和。使用 BFS，这里的 BFS 一次的源点为一个第一层轨道上的物体，对每个第一层轨道上的物体都进行一次 BFS。

功能 3：增加/删除一条社交关系。增加删除关系可以直接调用 ConcreteCircularOrbit 中实现了的函数，不过考虑到关系改变会引起图的结构的改变（有的节点需要删除，有的节点需要改变所处的轨道），需要调用 adjustFriendLocation 来调整整个 physicalObjectMap，这里采用的（暴力）方法是重复功能 1 中的 BFS 过程，重新计算所处轨道，然后删除不能连通的节点。

功能 4：计算逻辑距离。调用 ConcreteCircularOrbit 中的 getLogicalDistance 即可。

判断合法性：按照要求实现。

3.11 应对应用面临的新变化

3.11.1 TrackGame

4 人一个轨道：添加编排策略 RelayRaceStrategy，每个轨道分配 4 个人。修改 visualizeContentPanel 方法一个轨道上显示全部的人。

修改后如图 5.

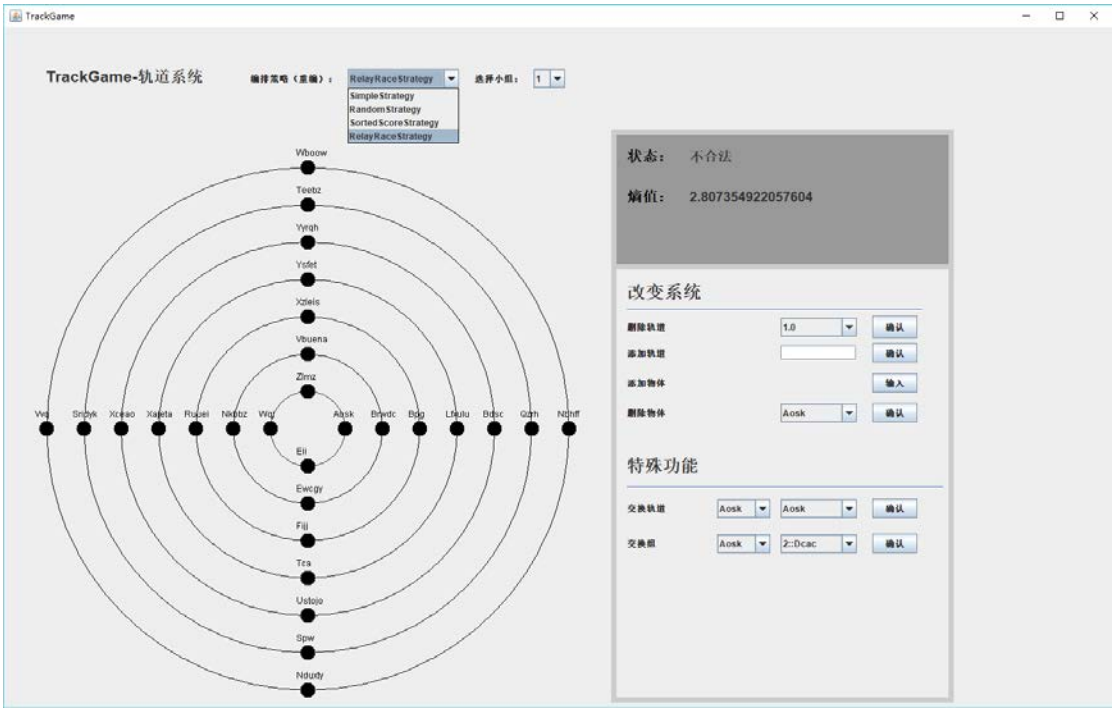

图 5 调用 RelayRaceStrategy 之后编排

3.11.2 AtomStructure

原子核表达为多个质子和多个中子的组合：设计中子质子类，更改 AtomCore，添加中子质子列表。修改 AtomCore，覆盖 drawGraphics 方法，改一下 AtomCore 的显示字符串为多少个中子质子的组合。

具体显示如图 6。

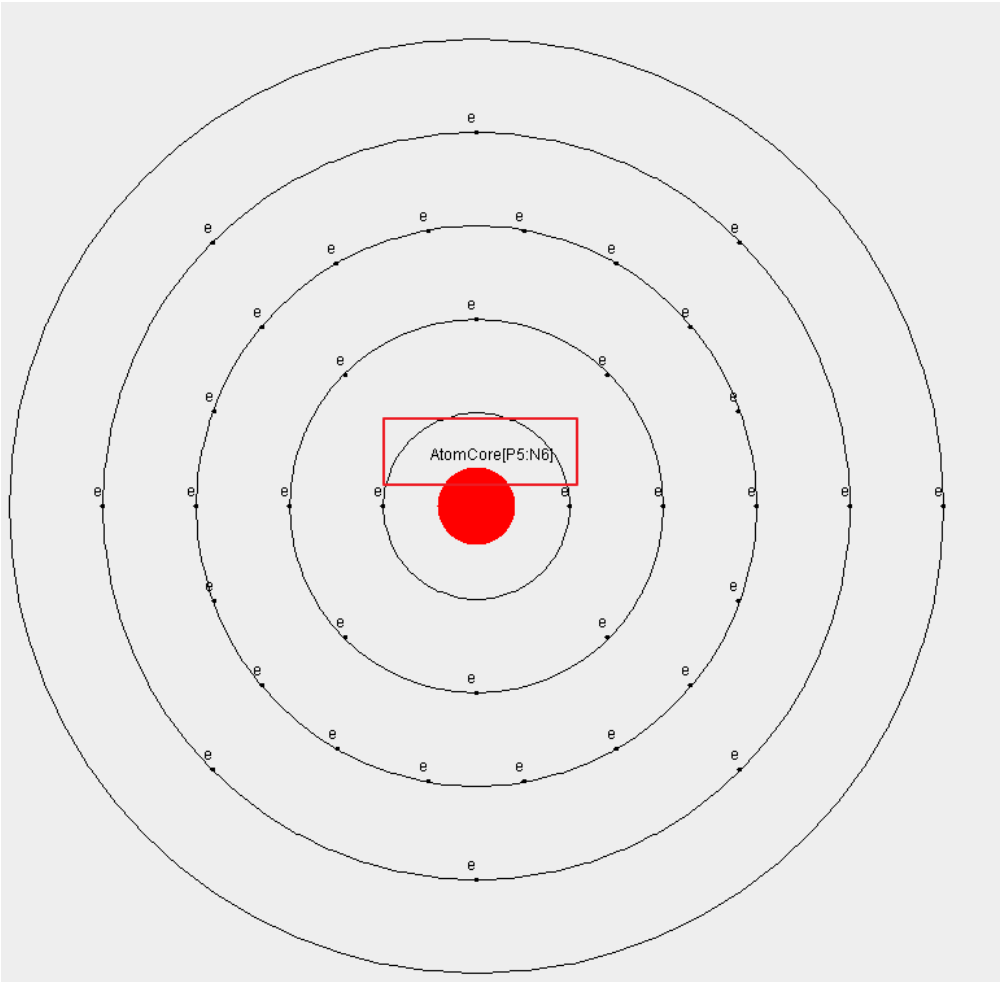

图 6. 修改之后显示原子核组成

3.11.3 SocialNetworkCircle

为关系增加方向性：在原来的实现中，无向边使用两条有向边表示，这里只需要改一下，添加/删除关系的时候只需要操作一条边即可。

忽略边：在读入边的时候，如果存在轨道物体向中心点的边则不操作，然后将其他所有的边加入，这里当然也包括了外层轨道到内层轨道的边，但是这里我们保留它，因为从实际应用角度出发，虽然一条边是从外层轨道到内层轨道的，但是在添加/删除关系之后这条边也可能变成内到外的边。

如何忽略？这里我们求出所有点到中心点的距离，根据距离即可判断内外（也可以求出所在的轨道-更简），每次更改关系结构之后需要重新求一遍距离，所以将求距离操作设计在 adjustFriendsLocation 中。

忽略影响：修改 visualizeContentPanel，不显示外到内的边。修改求扩散度的操作，忽略所有外到内的边。但是如果两点之间是外到内的边，两者依然是相连的关系，这点在面板上的删除边功能上有所体现。

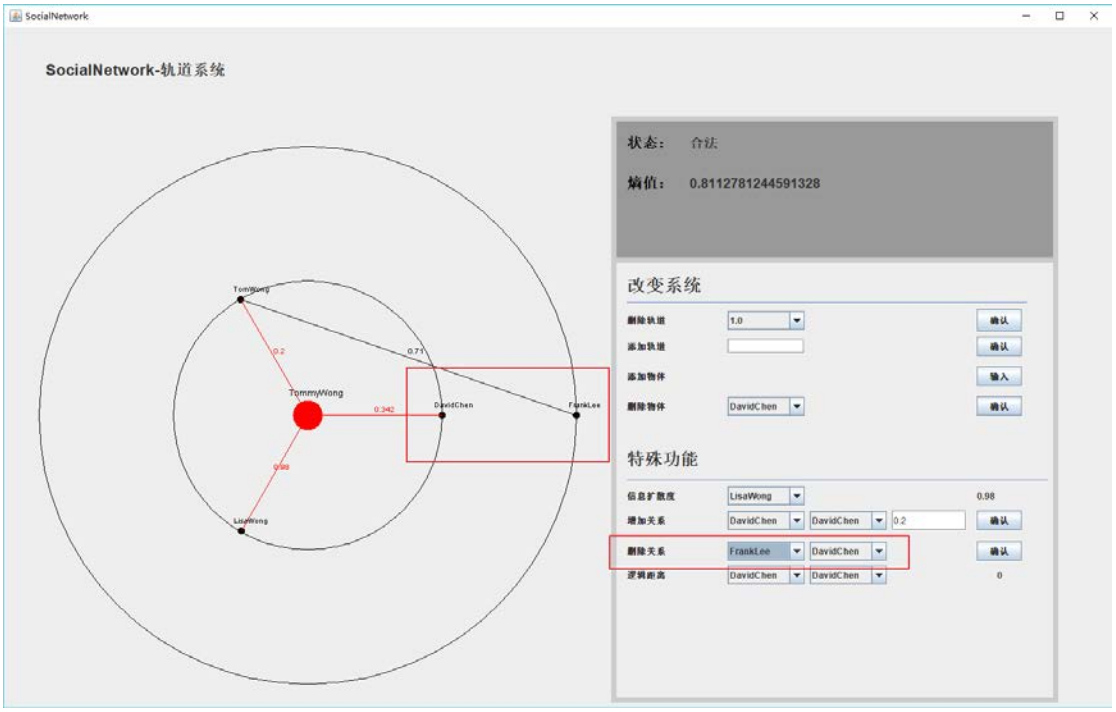

图 6. 修改之后的 SocialNetworkOrbit（解释：在删除关系中有 FrankLee->DavidChen 的一条边，但是因为这条边是由外到内的所以在轨道系统图上被忽略了）

3.12 Git 仓库结构

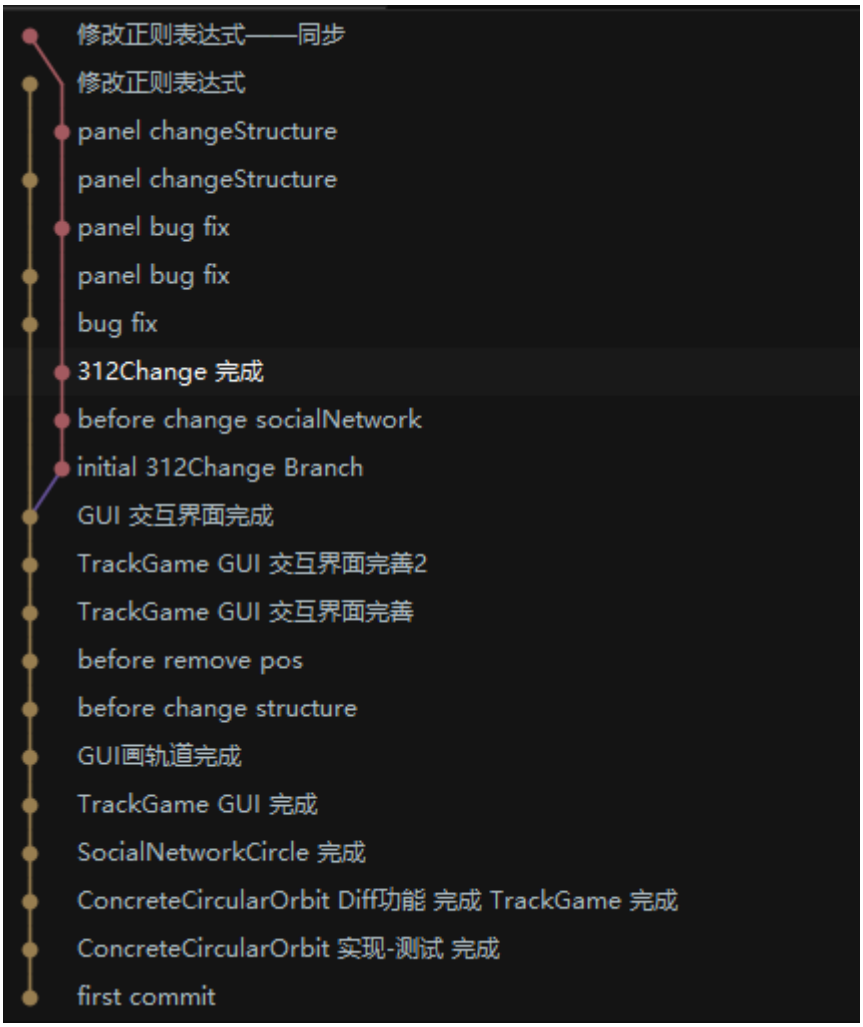

图 7．实验 git 仓库结构

4 实验进度记录

日期	时间段	计划任务	实际完成情况
4/11	10:46-17:05	ConcreteCircularOrbit 大体结构	完成
4/12	20:00-0:02	完成 TrackGame	完成
4/14	18:00-23:35	完成 SocialNetworkCircle	完成
4/20	15:33-22:36	GUI 画轨道完成	完成
4/21	9:57-19:31	TrackGameGUI 完成	完成
4/22	9:23-20:37	完成交互界面，完成 312change 验收	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
SocialNetworkCircle 添加删除关系操作	暴力解决，重新求解图的结构
可视化	一点一点慢慢写，布局类不能继承解决方案也是多谢了点冗余代码。

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和教训

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？

面向 ADT 编程可复用。

复用太好了！真的少打很多代码呢！复用基于应用抽出高级类，不过也不能啥都抽，文档中给出的设计有一些显然不太合理。

(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？

设计 ADT 的复杂是为了他人使用时候的省心。

如果只是给我自己写的 ADT 我显然不愿意这么干，太麻烦了。但是如果想要设计 ADT 给别人用，给公司用，那我是很乐意的！

(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？

好像没什么难处，感觉实验中 API 只是起到了一个暴露接口的作用。但是把

自己做的东西给别人用，别人还用不出 bug 来的感觉还是很棒的！

(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？

设计模式的使用增加了我们程序的可维护性，这一点可以从 TrackGame 中 Strategy 的使用可以看出，我们只需要添加一个对应的马拉松编排算法就能够实现新添加的功能。像 builder 模式将轨道系统的构造分成有序的步骤，client 调用起来更加简单。Iterator 设计模式提供的有序遍历可以用在很多个其他的方法中。设计模式好哇！

(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？

不太理解什么是语法驱动编程。。。

(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？

难在设计吧，我刚开始是按照文档中说的那样写的，但是写了一半之后发现文档中的要求与我的应用是有矛盾的，于是又得重写高级类，这也说明了设计的重要性吧。克服也不是一次就能克服的，我觉得首先得看完题目，不要一味相信实验要求文档吧。。。

(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？

？？？没有。。。药丸 TAT

(8) 关于本实验的工作量、难度、deadline。

写完五个是不可能写完五个的，写三个日子还能过得去。。。

(9) 到目前为止你对《软件构造》课程的评价。

挺好的，能够大幅度提高自己的工程能力吧，比如设计模式就是程序员精髓。