

**2023年春季学期  
计算机学院《软件构造》课程**

**Lab 3实验报告**

|  |  |
| --- | --- |
| 姓名 | 房昕玉 |
| 学号 | 2021113290 |
| 班号 | 2137102 |
| 电子邮件 | 2354598747@qq.com |
| 手机号码 | 18629831988 |

**目录**

[1 实验目标概述 1](#_Toc1822)

[2 实验环境配置 1](#_Toc28452)

[3 实验过程 1](#_Toc5938)

[3.1 待开发的三个应用场景 1](#_Toc23708)

[3.2 基于语法的图数据输入 2](#_Toc31186)

[3.3 面向复用的设计：CircularOrbit<L,E> 3](#_Toc10050)

[3.4 面向复用的设计：Track 5](#_Toc26474)

[3.5 面向复用的设计：L 6](#_Toc1654)

[3.6 面向复用的设计：PhysicalObject 6](#_Toc8185)

[3.7 可复用API设计 7](#_Toc28655)

[3.8 图的可视化：第三方API的复用 8](#_Toc26942)

[3.9 设计模式应用 9](#_Toc17840)

[3.10 应用设计与开发 12](#_Toc2924)

[3.10.1 AtomStructure 12](#_Toc19569)

[3.10.2 TrackGame 16](#_Toc10340)

[3.10.3 SocialNetworkCircle 16](#_Toc30324)

[3.11 应对应用面临的新变化 17](#_Toc27071)

[3.11.1 TrackGame 17](#_Toc11742)

[3.11.2 AtomStructure 18](#_Toc9266)

[3.11.3 SocialNetworkCircle 19](#_Toc25360)

[3.12 Git仓库结构 19](#_Toc17173)

[4 实验进度记录 20](#_Toc1686)

[5 实验过程中遇到的困难与解决途径 21](#_Toc9488)

[6 实验过程中收获的经验、教训、感想 21](#_Toc5020)

[6.1 实验过程中收获的经验和教训 21](#_Toc6969)

[6.2 针对以下方面的感受 21](#_Toc14097)

# 实验目标概述

本次实验覆盖课程第 2、3 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：

l 子类型、泛型、多态、重写、重载

l 继承、委派、CRP

l 语法驱动的编程、正则表达式

l 设计模式

本次实验给定了多个具体应用，学生不是直接针对每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

配置maven框架支持，junit测试，Java swing画图GUI

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

[git@github.com:ComputerScienceHIT/HIT-Lab3-2021113290.git](mailto:git@github.com:ComputerScienceHIT/HIT-Lab3-2021113290.git)

<https://github.com/ComputerScienceHIT/HIT-Lab3-2021113290.git>

# 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。

TrackGame:轨道为圆形，每个轨道只有一名运动员，无中心物体，运动员之间不存在关系，运动员不能切换轨道；

AtomStructure：轨道为圆形，每个轨道有多个电子，有中心物体（原子核），电子之间不存在关系，电子能切换轨道；

SocialNetworkCircle：轨道为圆形，每个轨道有多个朋友，有中心物体（人），人与人之间存在关系；

## 基于语法的图数据输入

先读文本文件的数据并将文本文件的数据转化成字符串，然后使用正则表达式，将文本文件匹配成相应的数据结构。

**通用的数据读取：**

我们的输入文件：常规思路就是::=前面是key,后面是value，

可以用pair，key是Athlete，value是<Bolt,1,JAM,38,9.94>，可以看作是数组

Pair<String, List<String>>对应的就是Athlete ::= <Bolt,1,JAM,38,9.94>

返回结果举例子：

{ a-{p,o,i} , b-{u,y,t} , c-{r,e,w} }

0 1 2

//先匹配::=前面的

String pattern1="([a-z,A-Z,0-9]+)( +)?:"; //a-z,A-Z,0-9一次或多次出现； （空格）一次或多次出现或不出现---》直接用 （空格）\*

String pattern2="=( +)?(.\*)";//匹配的是整个

**针对不同的系统将读取的数据划分：**

对于Pair<String,List<String>> pair中的每一项key与文本文件中的::=前面的名字匹配；然后value对应文本文件中::=后面的内容，匹配添加到轨道系统中。

这里以AtomStructure为例：

1. if (pair.getKey().equals(InAtomType.ElementName.toString())){
2. this.addCentralObject(new AScenter(pair.getValue().get(0)));
3. }else if (pair.getKey().equals(InAtomType.NumberOfTracks.toString())){
4. int numOfTracks= Integer.parseInt(pair.getValue().get(0));
5. for (int i = 1; i <=numOfTracks; i++) {
6. this.addTrack(new TrackOfAtom(i));
7. }
8. } else if (pair.getKey().equals(InAtomType.NumberOfElectron.toString())) {
9. /\*\*对每一条轨道，都要对该轨道上的电子编号；到下一条轨道时，不重新编号轨道号 /该轨道上的电子数量\*/
10. int Electronicbianhao=0;
11. Random random=new Random();
12. for(String eachString:pair.getValue()){
13. String fenshu[]=eachString.split("/");
14. for(int i=1;i<=Integer.valueOf(fenshu[1]);i++){
15. Integer orbitbianhao=Integer.valueOf(fenshu[0])-1;//拿到轨道编号
16. Electron addedElectronic=new Electron(++Electronicbianhao);
17. this.addPhysicalObject(addedElectronic,tracks.get(orbitbianhao));
18. setPosition(addedElectronic);    positionOfPhysicalObjects.put(addedElectronic,newPosition(((TrackOfAtom)tracks.get(orbitbianhao)).getRadius(), random.nextDouble()\*2\*Math.PI, 2));
19. }
20. }
21. }

## 面向复用的设计：CircularOrbit<L,E>

接口中含有如下方法：

1. public static CircularOrbit empty(String SystemName,String file)  //静态工厂方法
2. public boolean addTrack(Track track);  //添加轨道
3. public boolean removeTrack(Track track);  //删除轨道
4. public boolean addCentralObject(L Object);  //添加中心物体
5. public boolean addPhysicalObject(E Object,Track track);  //添加轨道物体
6. public boolean removePhysicalObject(E Object,Track track);  //删除轨道物体
7. public boolean addRelationCtoP(L Object1, E Object2, Relation relation);  //添加中心物体和轨道物体关系
8. public boolean addRelationPtoP(E Object1, E Object2, Relation relation);  //添加轨道物体和轨道物体关系
9. public List<Pair<String, List<String>>> readFile(String fileName);  //读文件
10. public boolean transit(E Object, Track track);  //轨道物体移到track轨道上
11. public Iterator<E> iterator();  //迭代器
12. public Map<Track,List<E>> getTrackAndObjects();  //轨道和上面的物体的映射关系
13. public int getLogicalDistance(String a,String b);  //两物体之间的逻辑距离
14. public ApplicationType getType();  //app类型
15. public L getCentralObject();
16. public List<Track> getTracks();
17. public List<E> getPhysicalObjects(Track track);
18. public Position getPhysicalObjectPosition(E object);
19. public List<Pair<Position, Position>> getStraight();
20. public void setPosition(E object);  //设置位置

用ConcreteCircularOrbit.java实现接口中的方法，针对不同的子类，设计他们各自独有的方法：

**TrackGame**：分组时要对分好的组添加运动员信息。这里使用策略模式。具体在3.0阐述。



**AtomStructure**：重写transit函数——电子跃迁

**SocialNetWorkCircle：**



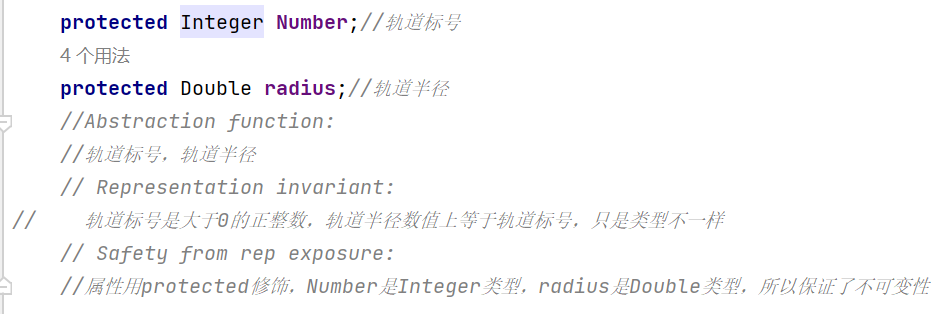
makeGraph通过文本中的数据转化成图结构，使用lab2中用过的ConcreteVerticesGraph()完成；

重写增删改查轨道，轨道物体：对图中数据操作；

重写增加关系函数，补充删除关系函数。

## 面向复用的设计：Track

Track抽象类里有两个属性：编号和半径；



三个子类TrackOfAtom，TrackOfGame，TrackOfSocial，要重写hashcode和equals。

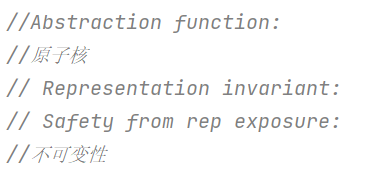
## 面向复用的设计：L

设计CentralObject接口：只有一个方法public String getName();

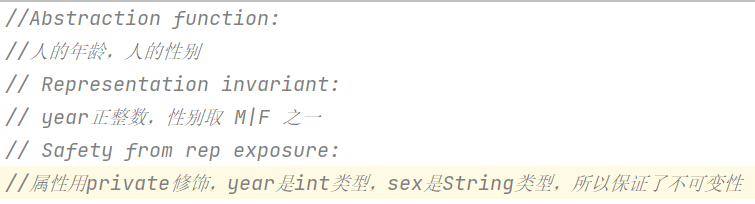
设计抽象类ConcreteCentralObject实现CentralObject接口，增加成员变量name和hashcode和equals；

三个子类：

AScenter；



SNcenter：增加两个成员变量年龄和性别，hashcode和equals；



TGcenter：无中心物体，name为“null”

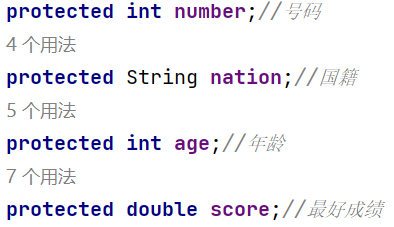
## 面向复用的设计：PhysicalObject

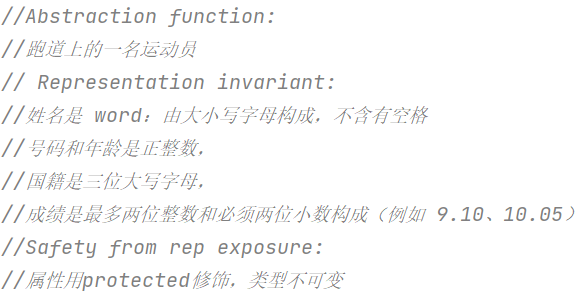
设计PhysicalObject接口：只有一个方法public String getName();

设计抽象类ConcreteCentralObject实现CentralObject接口，增加成员变量name；

三个子类：都要重写hashcode and equals

Athlete：





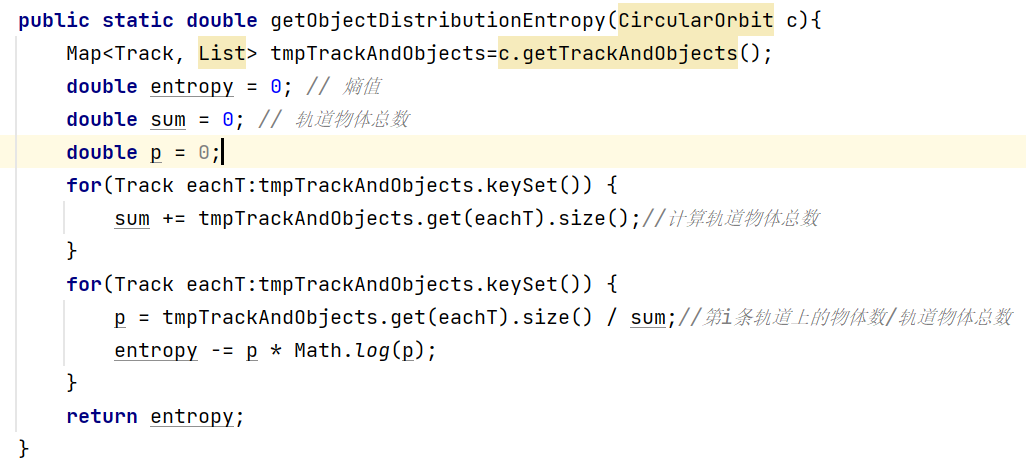
Electron：name即编号（number）

Friend：和中心物体属性，方法一样

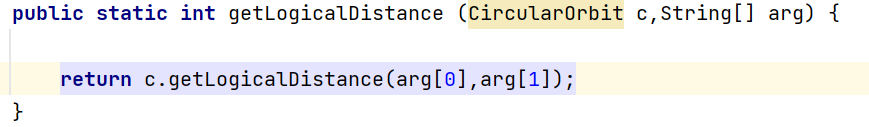
## 可复用API设计

编写四个函数：计算熵，计算逻辑距离，计算物理距离，删除轨道

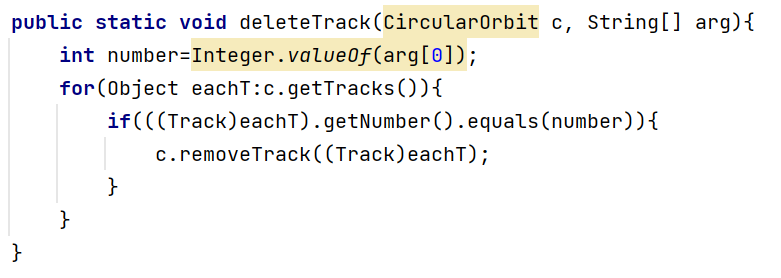
1. public static double getObjectDistributionEntropy(CircularOrbit c)



1. public static double getPhysicalDistance (CircularOrbit c, String[] arg)
2. public static int getLogicalDistance (CircularOrbit c,String[] arg)



1. public static void deleteTrack(CircularOrbit c, String[] arg)



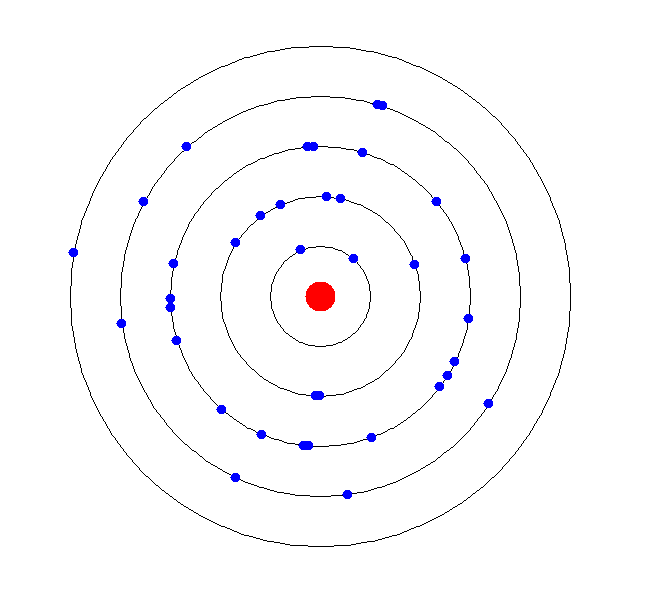
## 图的可视化：第三方API的复用

使用java swing，这里注意SocialNetworkCircle的GUI实现，调用getStraight()，此图较AtomStructure的GUI实现，添加了中心物体对于轨道物体的映射连线，和轨道物体之间映射的连线。

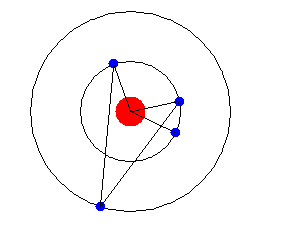
可视化内容继承自JPanel。在Visual类里面重写paintComponent方法

静态方法public static void visualizes(CircularOrbit c)获取轨道系统GUI所需要的参数，包括轨道数目，中心物体和轨道物体的映射。

可视化AtomicStructure.txt：



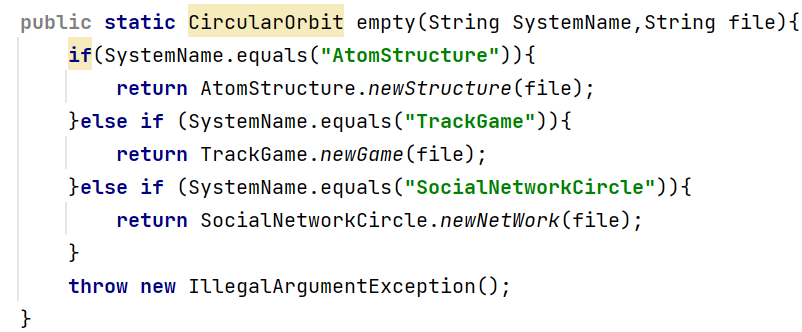
可视化SocialNetworkCircle.txt：



## 设计模式应用

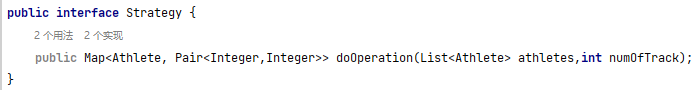
1. 初始化建立轨道系统时使用静态工厂方法

在CircularOrbit接口中写如下方法：对应各个不同子类的根据文本文件初始化后的轨道系统。



1. TrackGame里面为运动员分配跑道时使用策略模式

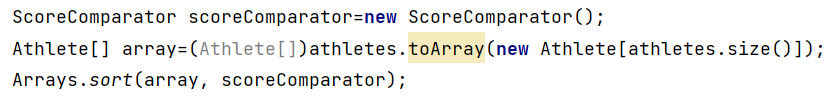
一个接口Strategy，两个类OperationRandom和OperationSort实现该接口。



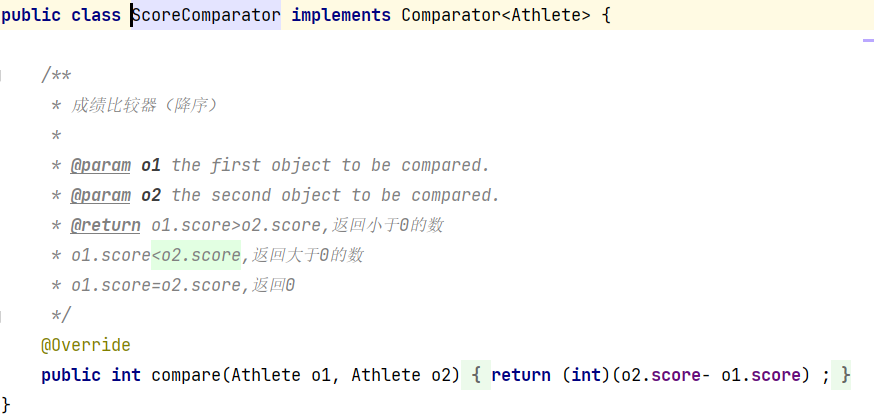
随机分配：



按成绩分配：分配时使用分数比较器，其余和按随机分配代码一样

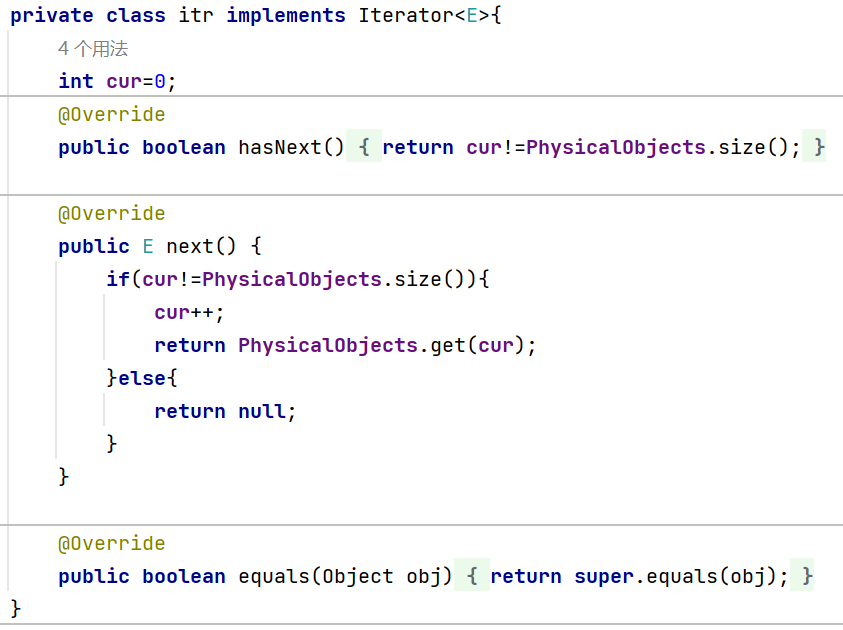


分数比较器：



1. 使用 Iterator 设计模式，设计迭代器，客户端可在遍历 CircularOrbit 对

象中的各 PhysicalObject 对象时使用，遍历次序为：从内部轨道逐步向外，同一轨道上的物体按照其角度从小到大的次序（若不考虑绝对位置，则随机遍历）



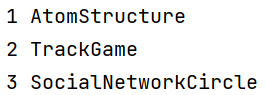
## 应用设计与开发

利用上述设计和实现的ADT，实现手册里要求的各项功能。

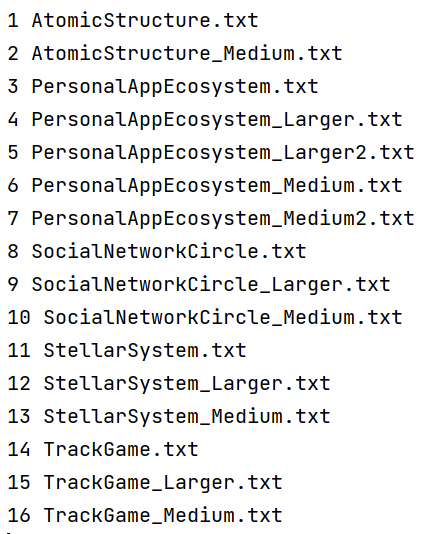
以下各小节，只需保留和完成你所选定的应用即可。

**三个app通用的功能：**

**首先选择app：**



选择后展示文件目录下面所有的数据文本文件

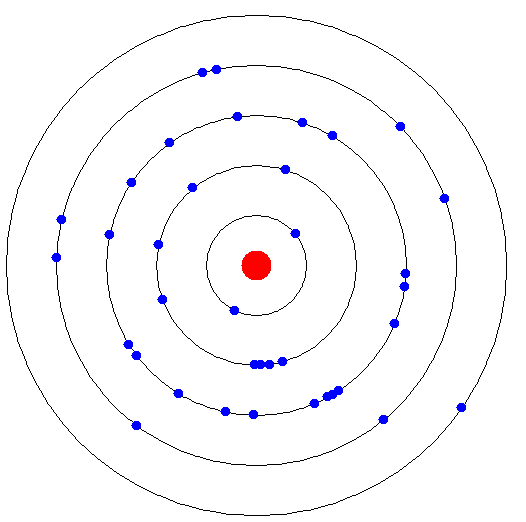


选择文件后，自动弹出画图窗口，画出根据文本文件中的数据生成的轨道系统

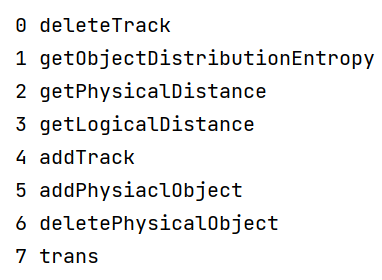
展示功能菜单，前四个功能是通用的，适用于三个轨道系统，

### AtomStructure

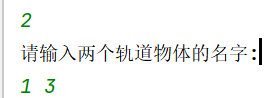
选择文件1：可视化文件数据的轨道系统

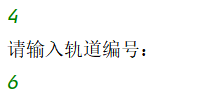


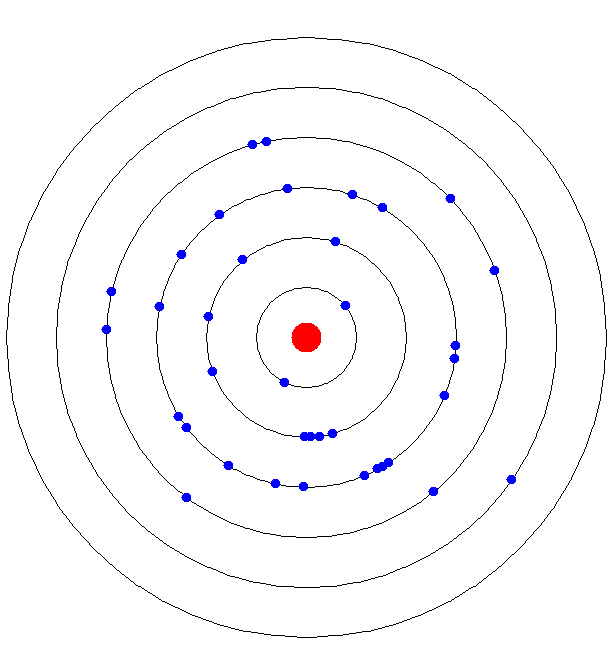
展示功能：（前四个是通用的功能）

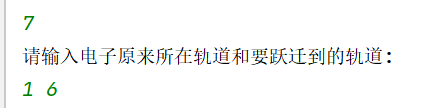


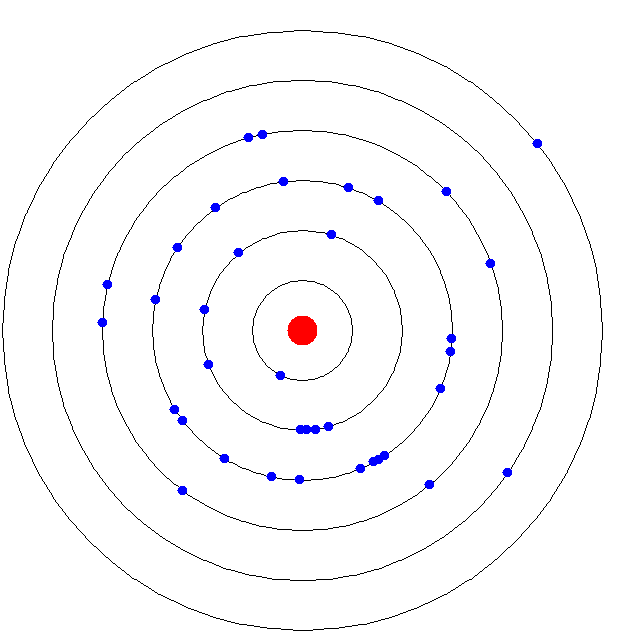
1：计算熵：

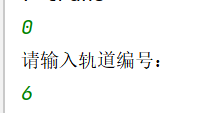


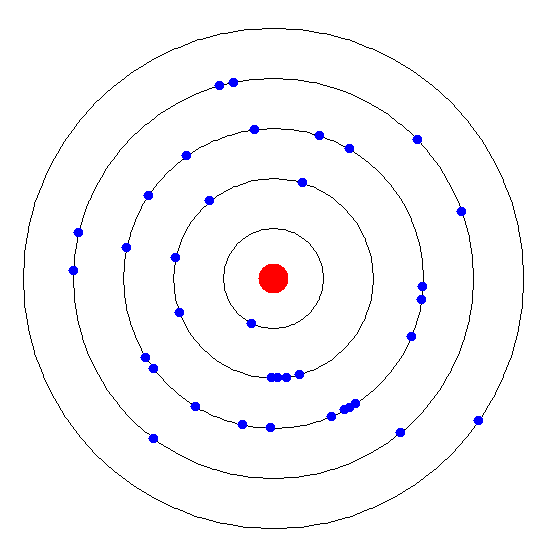










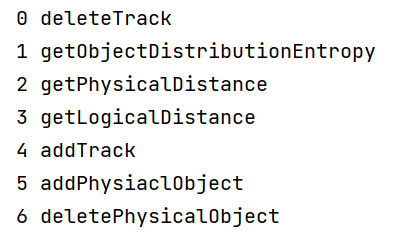


### TrackGame

选择文件后，还需要选择要进行的策略：

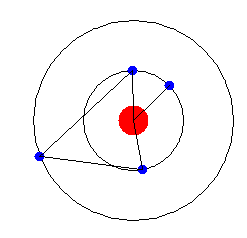


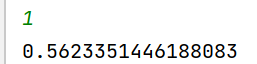
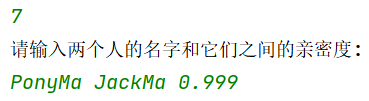
该app所含的功能：

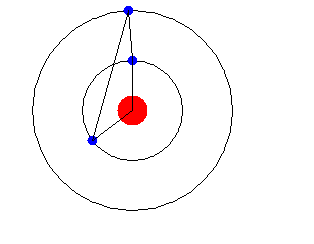


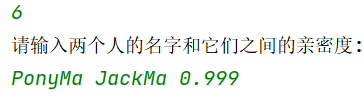
### SocialNetworkCircle

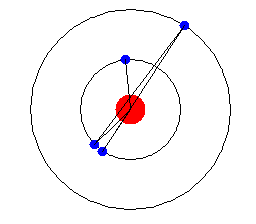
1.选择文件8：



1. 
2. 



4.



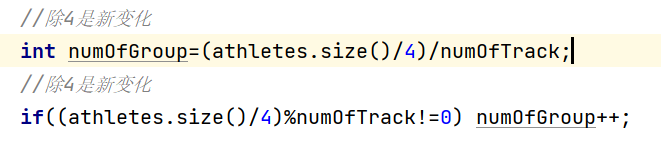
## 应对应用面临的新变化

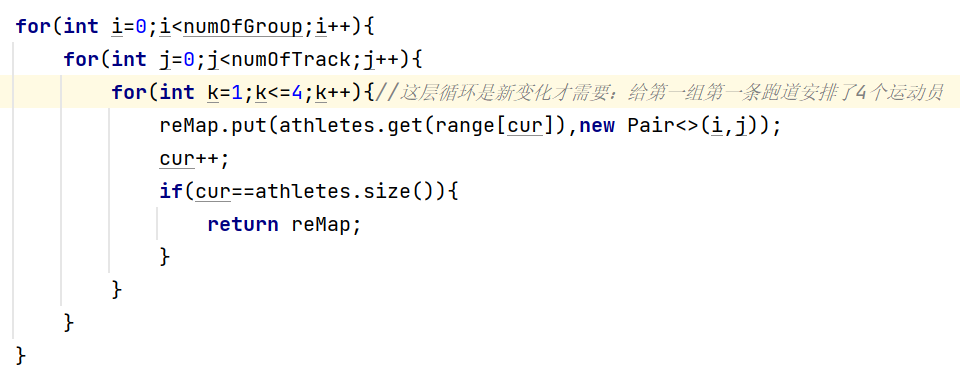
以下各小节，只需保留和完成你所选定的应用即可。

### TrackGame

新变化：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。

在策略模式中的两个模式中为跑到设置运动员时修改：

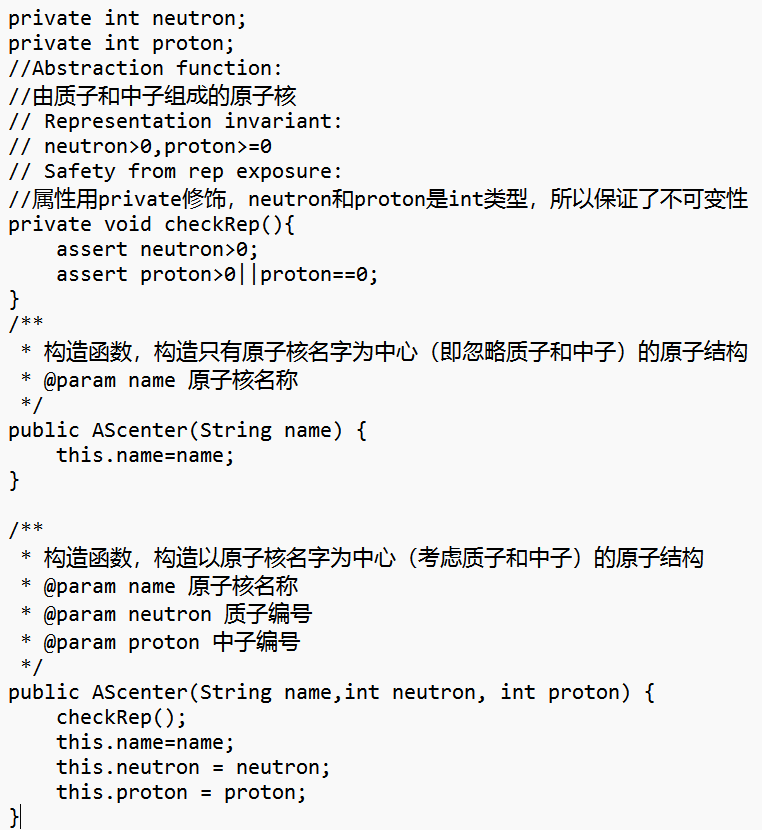




### AtomStructure

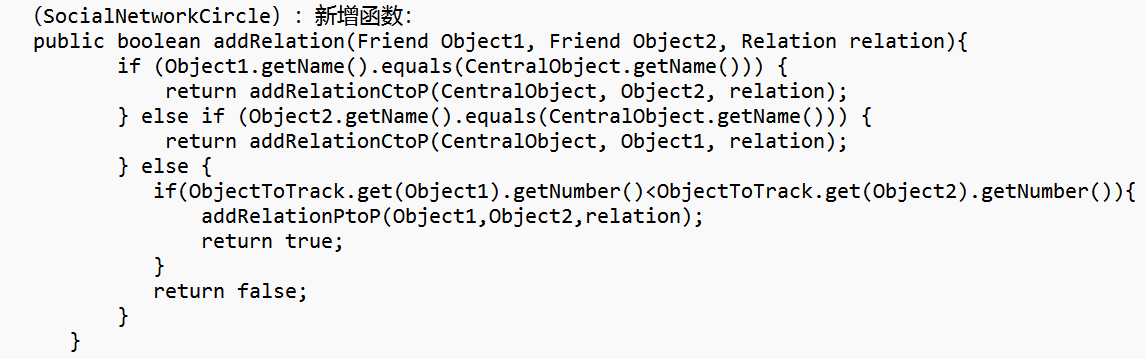
原子核需要表达为多个质子和多个中子，即处于中心点的物体可以是多个物体构成的集合

修改原子核中心物体的属性及其构造函数：



### SocialNetworkCircle

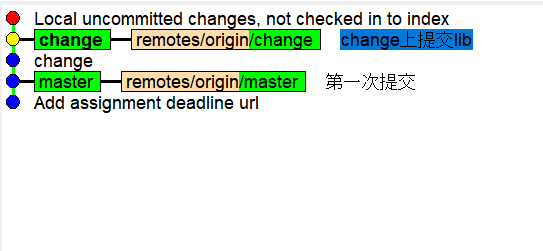
在子类中新增方法：里面添加亲密度时具体实现要判断两个人所在的轨道编号大小关系



## 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 |
| 2023/4/20 | 17：00-21：30 | 研究实验手册 | 未完成，有一些不理解 |
| 2023/4/21 | 15：30-18：00 | 理解分析实验要求 | 完成 |
| 2023/4/21 | 19：00-23：00 | 完成CircularOrbit接口和ConcreteCircularOrbit抽象类 | 按计划完成 |
| 2023/4/22 | 15：30-17：30 | 完成Track接口，轨道物体接口，中心物体接口设计 | 按计划完成 |
| 2023/4/23 | 9：00-12：00 | 完成轨道，轨道物体，中心物体的三个子类设计 | 按计划完成 |
| 2023/4/23 | 14：30-18：00 | 设计通用读文件的类 | 按计划完成 |
| 2023/4/24 | 9：30-12：00 | 完成trackgame类具体设计 | 未完成，延迟了一个小时 |
| 2023/4/24 | 14：00-17:30 | 完成AtomStructure类具体设计 | 按计划完成 |
| 2023/4/24 | 18:30-00:00 | 完成SocialNetWork类具体设计 | 按计划完成 |
| 2023/4/25 | 9：00-14:30 | 完成测试类并通过测试 | 延迟1小时完成 |
| 2023/4/26 | 9：40-12：00 | 完成APIs | 按计划完成 |
| 2023/4/26 | 15：00-18：30 | 可视化轨道系统 | 未完成 |
| 2023/4/27 | 9：00-14：00 | 可视化轨道系统 | 按计划完成 |
| 2023/4/28 | 18：00-22：00 | 设计applications的框架 | 延迟2小时完成 |
| 2023/5/1 | 8:00-18:00 | 完成applications | 按计划完成 |
| 2023/5/2 | 8：00-10：00 | 完成change | 按计划完成 |
| 2023/5/2 | 14：30-18：00 | 写报告 | 按计划完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 任务量太大，不知从哪里开始 | 仔细研读手册 |
| 不太懂设计模式 | 到图书馆看相关的书，精读《图解设计模式》中常用的几个设计模式，有了一定理解 |
| 可视化不知从何下手 | 询问同学，并上网搜所相关教学料，了解Java swing的用法 |
| 测试时遇到测试失败 | 上网搜索发现：总结出了两个测试时容易出现的错误：  1.变量的类型得是包装数据类型，才能用hashcode  2.有的时候测试死活过不去可能是因为导错包了，sun....都不行，测试类所在的包要与具体类一致 |
| 问题出现：给轨道编号命名时，new的时候开辟了新的空间，不是名字相同就是同意条轨道，故推测是equals和hashcode的问题 | 重写后测试结果正确 |
| 给原子结构的跃迁做测试时，测试失败。 | 原因找到了，是因为要先把轨道物体添加到相应的轨道上才能跃迁，不然对于新建的轨道物体来说没有原轨道beforeTrack |
| 测试makeGraph失败 | 回去查看makeGraph源代码，发现是Init的时候 PhysicalObjects.clear()； this.makeGraph();这两句写到了循环里面 |

# 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

了解java swing，对设计模式的应用有了具体实现，掌握正则表达式匹配文本的方法,学习了子类型、泛型、多态、重写、重载，继承、委派、CRP 的应用

## 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. 面向ADT编程要求对编写的程序整体有充分的认识，要考虑接口、抽象类、实例类的整体设计，要考虑到哪些部分能够复用、哪些部分需要具体实现；而面向应用场景编程则是就事论事，编程思路简单，但是工作量大，复用性低。
3. 体会到了
4. 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？

一定要为ADT撰写spec、AF、RI、rep exposure，不要嫌麻烦！在简单的程序中可能用处不大觉得多余，但是大型程序开发时，这些内容可以时刻提醒自己本程序要做什么、输入输出是什么、我的rep对于client来说代表什么抽象的形象、在写程序时要时刻注意满足规约、防止表示泄露的发生！

愿意。

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？

开发API给别人用是一个十分困难的过程，不仅仅要求其正确性，还需要其他人能够理解自己的想法，学会用法，但是如果能够帮助到他人的开发是一个十分好的方法。

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？

在自己开发的过程中体会到了设计模式的好处，认为它存在的意义十分重大，使用起来对于维护和改变会有很大的帮助

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？

语法驱动器能够更加准确的识别文件中的内容，相比用if-else这种方法识别，效率大大提高了

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？

ADT的设计难度主要在于设想与实现的不同步。首先抽象十分困难，初次抽象出来的内容和最终的实可能会有很大出入，需要我们在开发过程中不断去完善自己的设想。

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？

没有

1. 关于本实验的工作量、难度、deadline。

大，难，不是deadline的问题，会的可能很快做完，不会的在一个小点就会卡很久，更别说工作量这么大的整个实验了。主要是目前所有的知识与要应用的不匹配，所以要花大量时间来自学，然后消化理解，才能完成实验。

1. 到目前为止你对《软件构造》课程的评价。

实验是特色，完成实验后感觉自己变强了不少，不错的课程。就是上课讲的太浅，想要完成实验不够用，可以加课时。