

**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 | 胡宏彬 |
| 学号 | 2021111204 |
| 班号 | 2137101 |
| 电子邮件 | 1953802638@qq.com |
| 手机号码 | 17816078651 |

**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

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

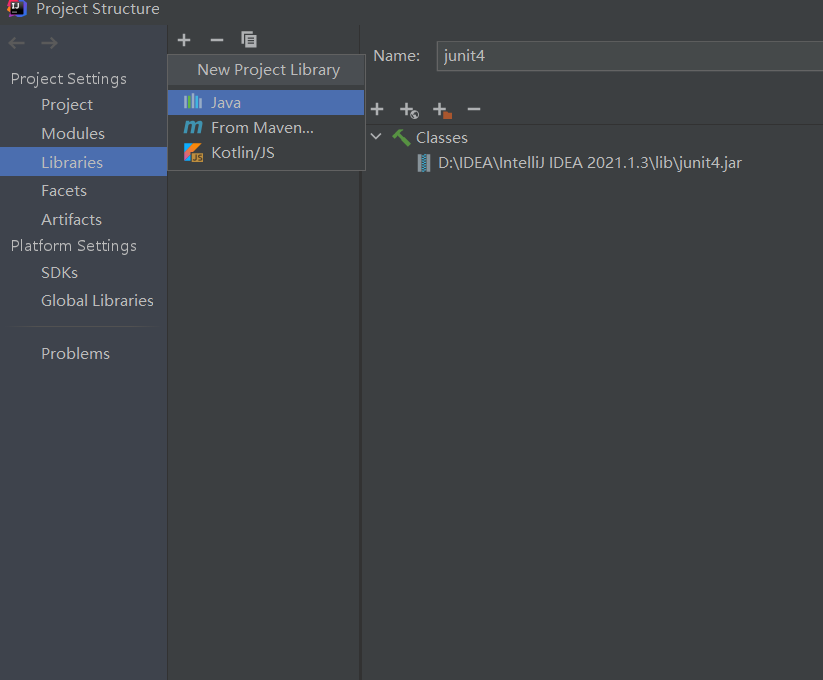
这些应用之间的相似性和差异性，使 ADT 有更大程度的复用（可复用性）和更

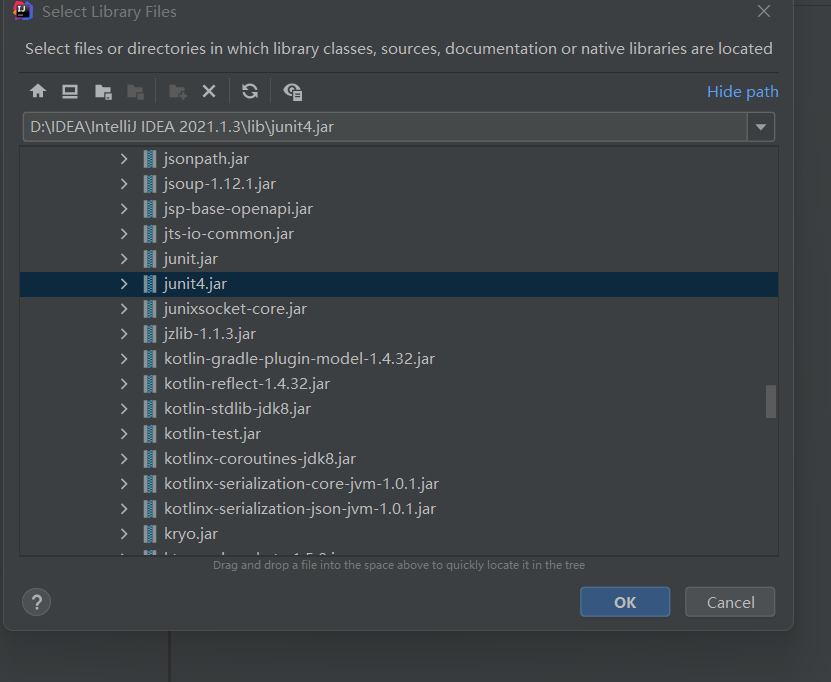
容易面向各种变化（可维护性）。

# 实验环境配置

Junit 配置：

从File 选择Project structure，在Libraries点击+号，选择java，找到IDEA的lib目录下的Junit4.jar，添加并应用即可。





其余环境如IDEA、Java8在之前的实验中已经配置完毕。

GitHub Lab3仓库的URL地址：

https://github.com/ComputerScienceHIT/HIT-Lab3-2021111204.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）。

* 径赛场地赛程编排（TrackGame）
* 原子结构模型(AtomStructure)
* 社交网络的好友分布（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. 整体的系统与我们设计的轨道系统类似，包括轨道、中心物体、轨道物体等。其中物体都不考虑绝对位置，且轨道都为圆形。
2. 都有如下功能：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：

在不同的应用场景中，各条轨道上的物体有的相同，有的不同。例如：

TrackGame:轨道物体之间有区别，为不同的运动员。

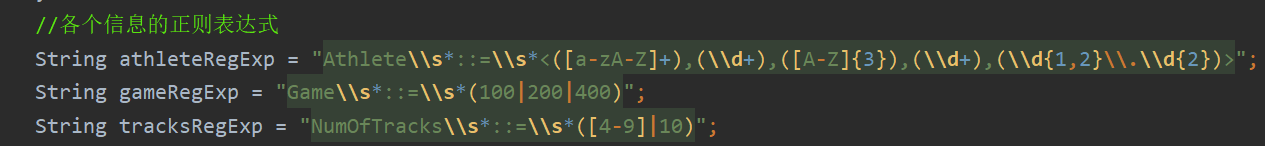
AtomStructure:轨道物体都是同一个对象，均为电子。

SocialNetworkCircle：不同轨道上的好友各不相同，等级各不相同。

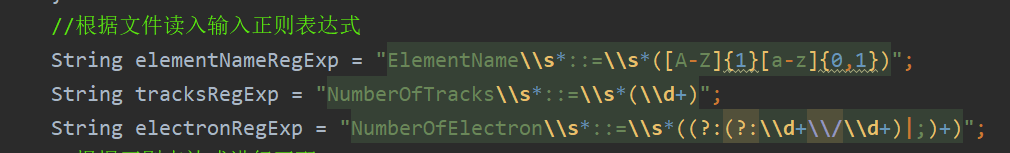
## 基于语法的图数据输入

正则表达式：

TrackGame：



AtomStructure：



SocialNetworkCircle：

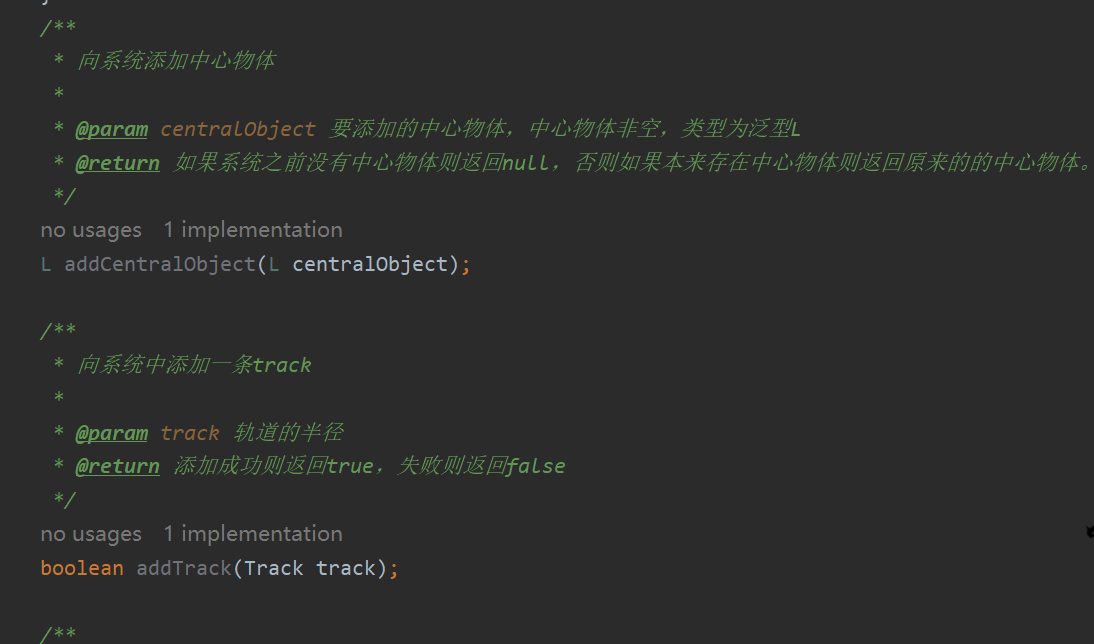
## 面向复用的设计：CircularOrbit<L,E>

按照说明，这是一个mutable的ADT，轨道系统在不同的要求下可以发生变化。

1. 接口CircularOrbit<L,E>

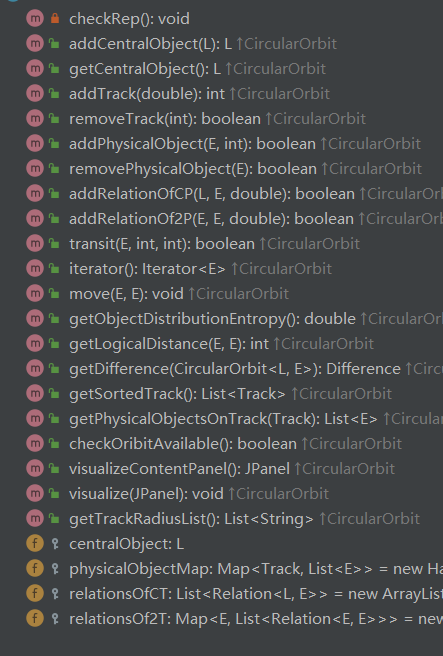
按照3.11要求，由于在完成该ADT时需要用Iterator 设计模式，设计迭代器，迭代轨道上的物体。故而该接口应该继承自Iterable<E>接口。

在接口中定义所需的各种方法并写上spec，部分如下：

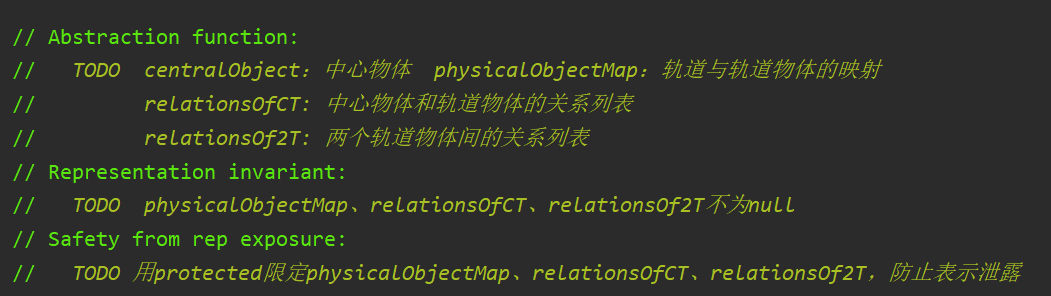


2）具体实现类：ConcreteCircularOrbit

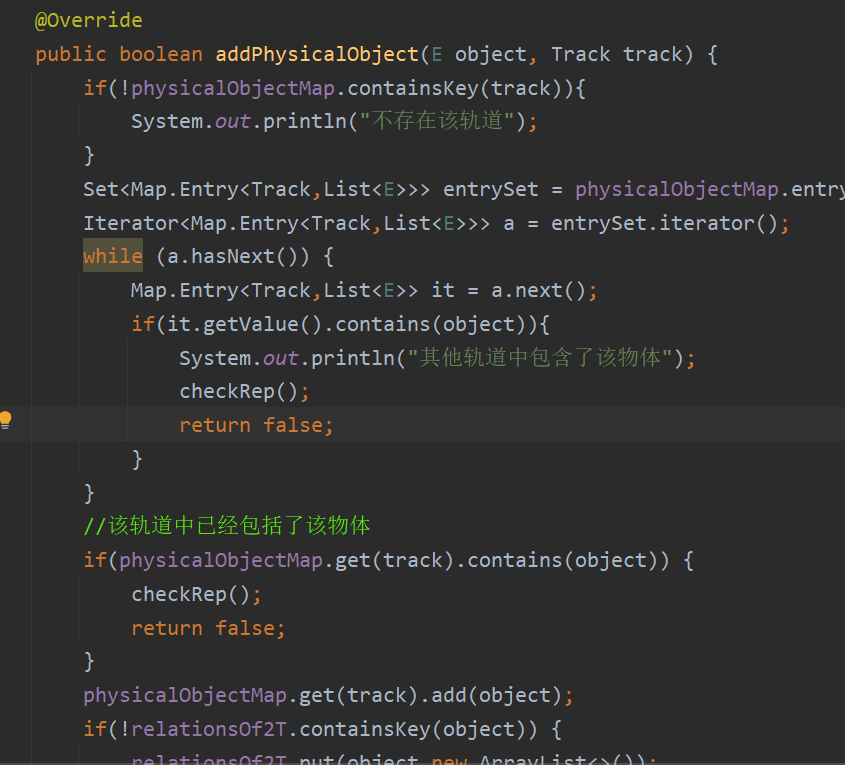
1. ConcreteCircularOrbit的field



1. ConcreteCircularOrbit的AF、RI、safety from rep exposure

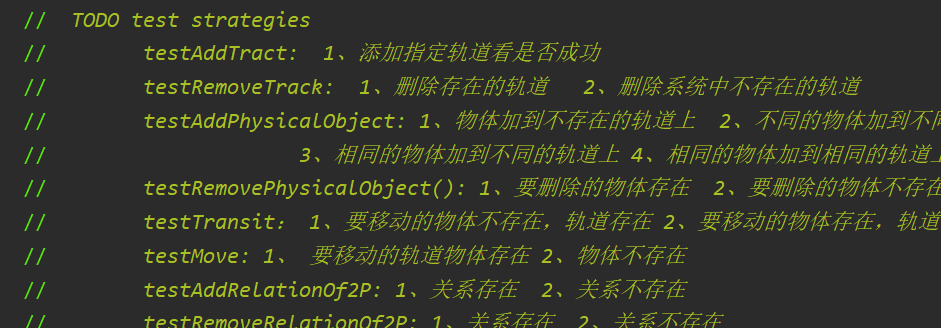


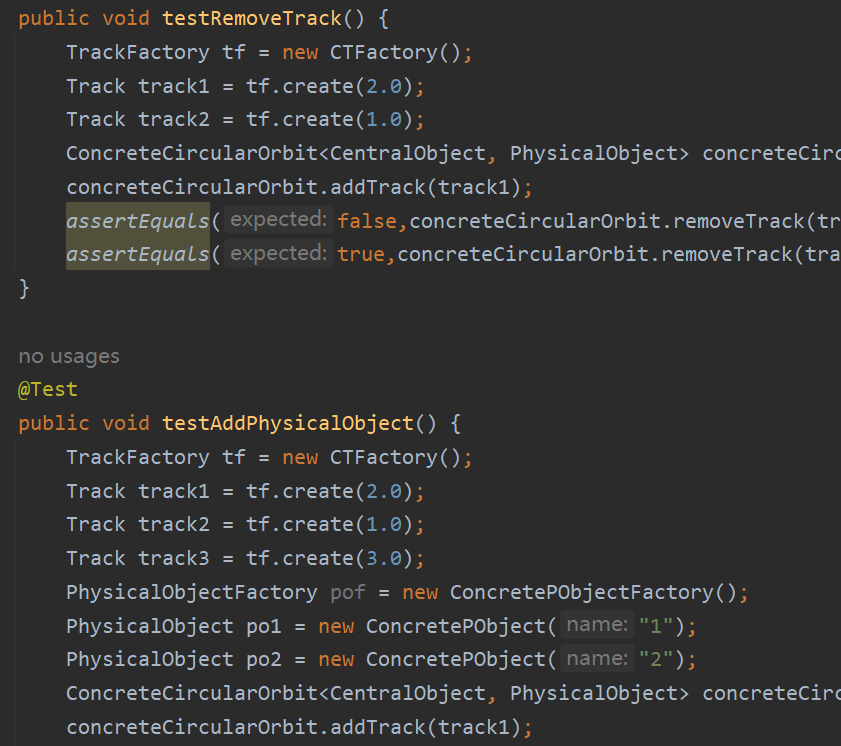
1. ConcreteCircularOrbit的部分代码



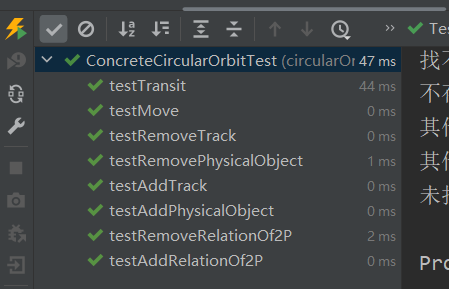
1. 测试

测试策略及部分代码：





测试结果：

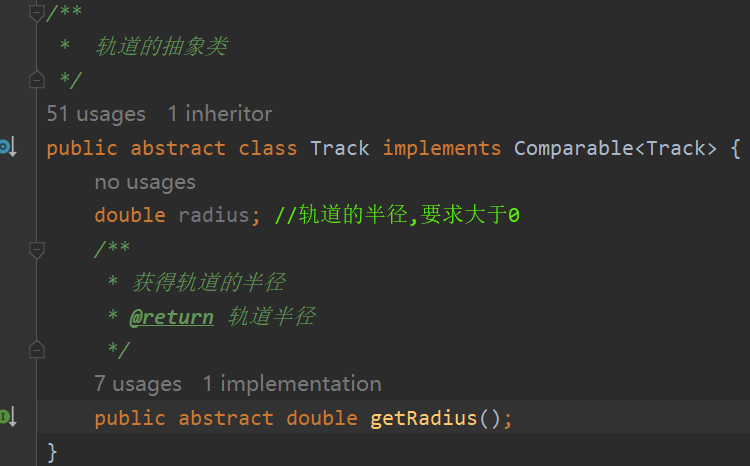


## 面向复用的设计：Track

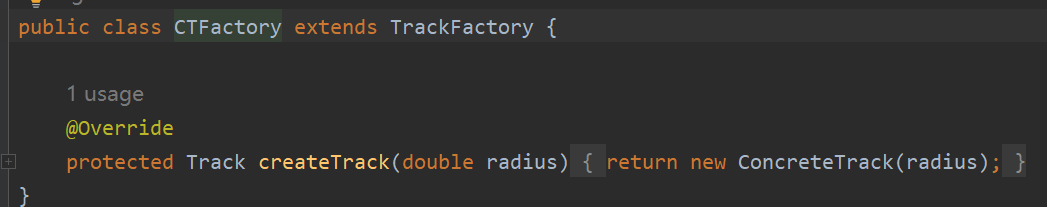
1. 实现TrackFactory类

思路：首先构造TrackFactory的抽象类，然后再用一个具体类CTFactory对其进行继承，从而完成对Track创建的封装，满足工厂设计模式

TrackFactory的部分代码：



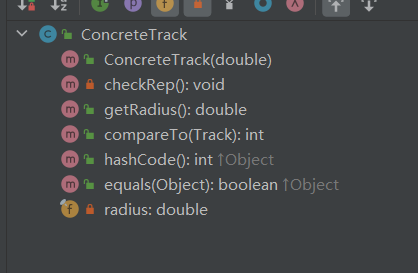
CTFactory的代码：



2）实现Track类

思路：首先构造Track的抽象类，为了方便后续比较，该类需要继承自Comparable接口。然后再用一个具体类ConcreteTrack对其进行继承。

1、ConcreteTrack的field



2、ConcreteTrack的AF、RI、safety from rep exposure



3、ConcreteTrack的部分代码

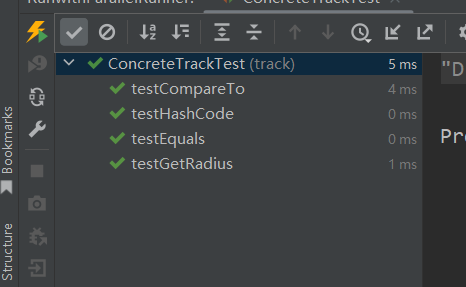


3）测试代码：

测试策略及部分代码：



测试结果：

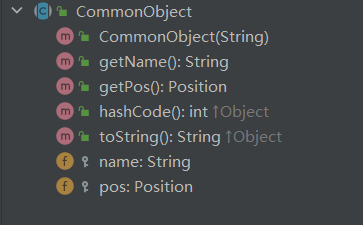


## 面向复用的设计：L

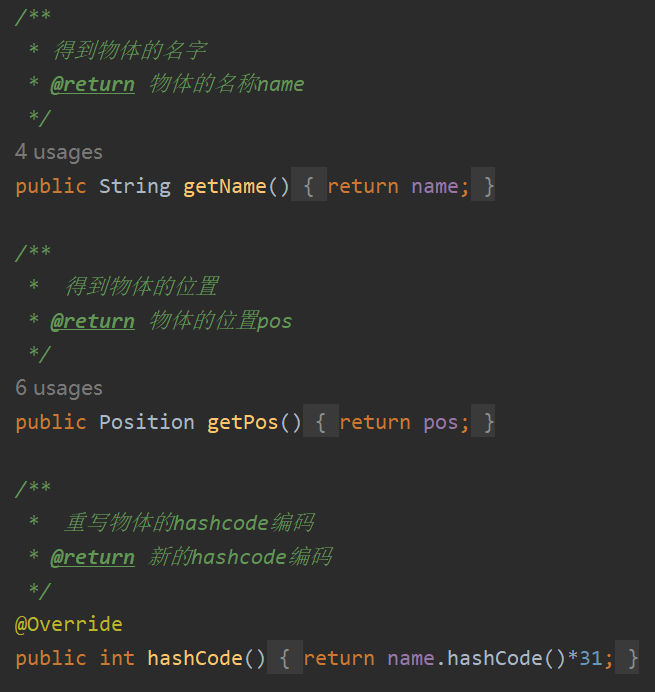
思路：首先设计一个CommomObject类，为PhysicalObject类和CentralObject类的父类。然后根据工厂设计方法对ConcreteCObject进行封装。其中Position是Immutable类型，为抽象出的位置的类。而ConcreteCObject为Immutable对象类。

1. 实现CommomObject抽象类

抽象类的field：

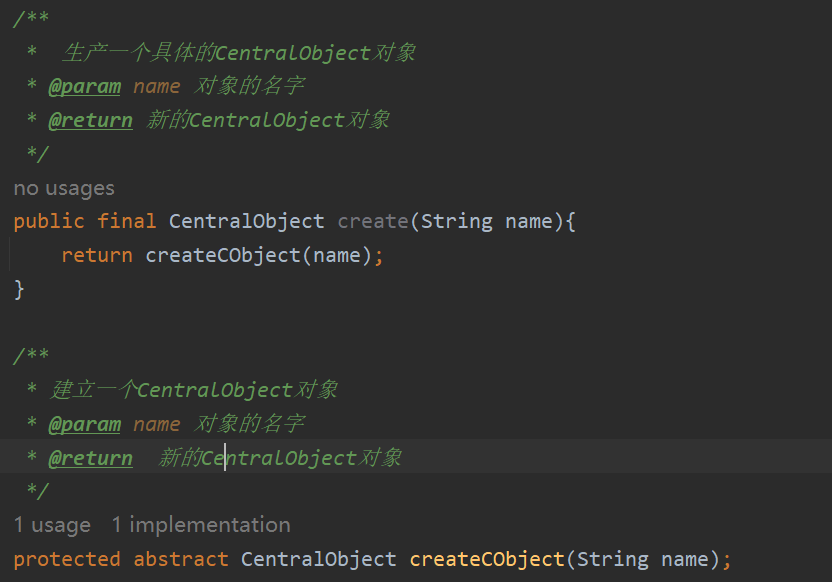


抽象类的spec与部分代码：

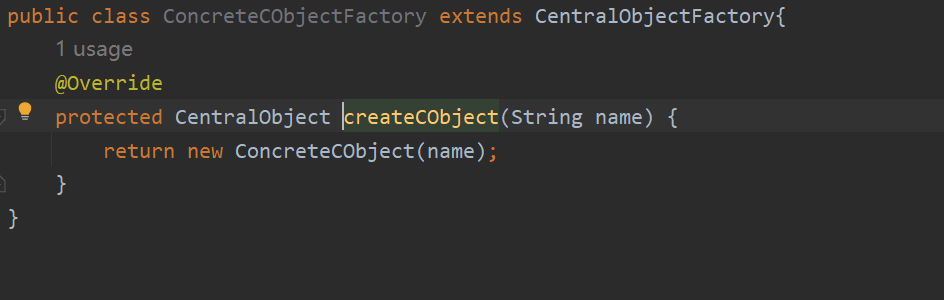


2）实现ConcreteCObjectFactory类

1、CentralObjectFactory的部分代码：

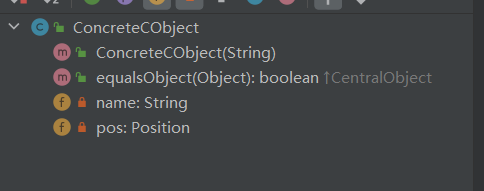


2、ConcreteCObjectFactory的部分代码：



3）实现ConcreteCObject类

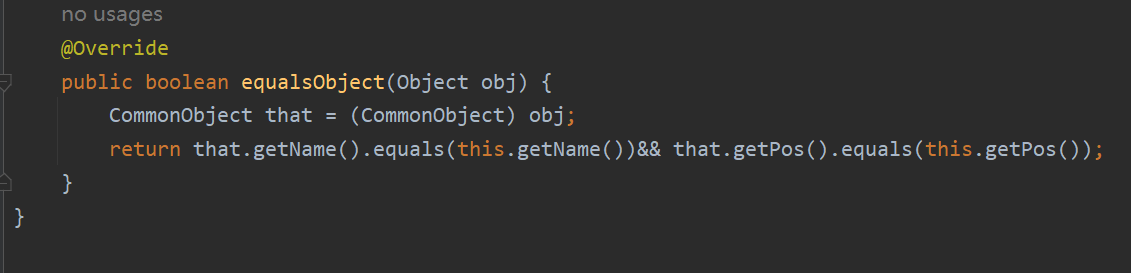
1、ConcreteCObject的field



2、ConcreteCObject的AF、RI、safety from rep exposure

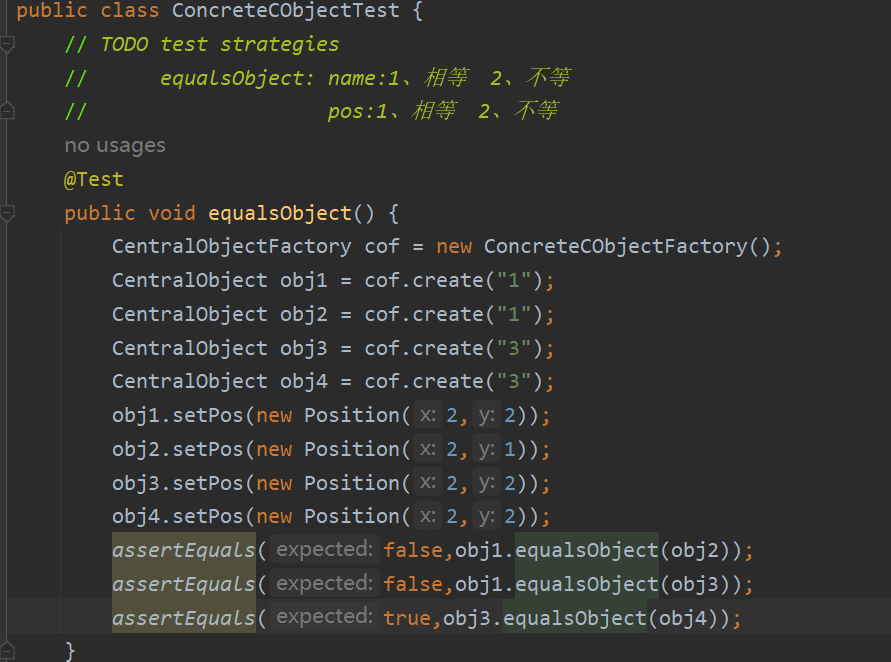


3、ConcreteCObject的部分代码

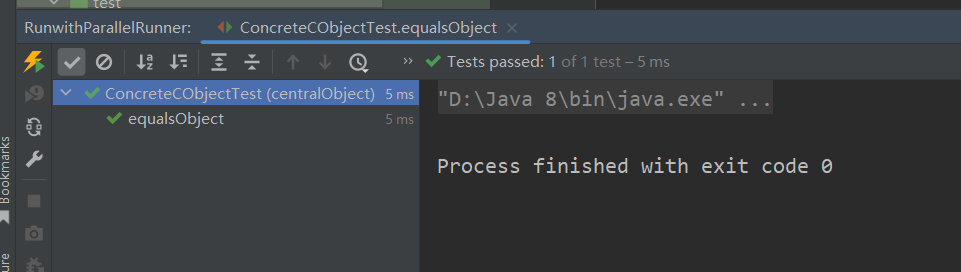


4）测试：

测试策略及部分代码：



测试结果：

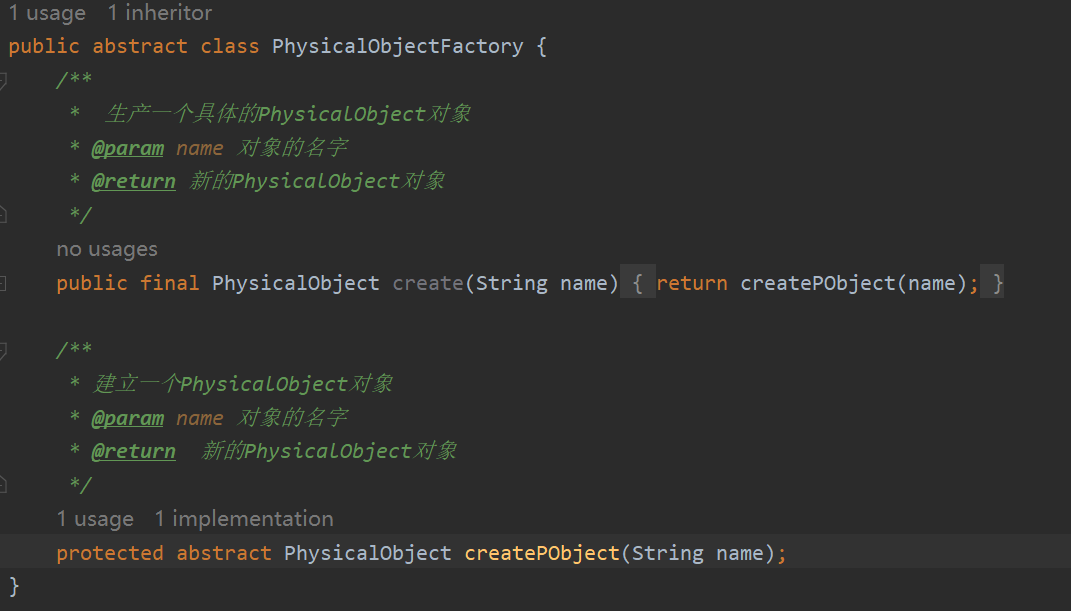


## 面向复用的设计：PhysicalObject

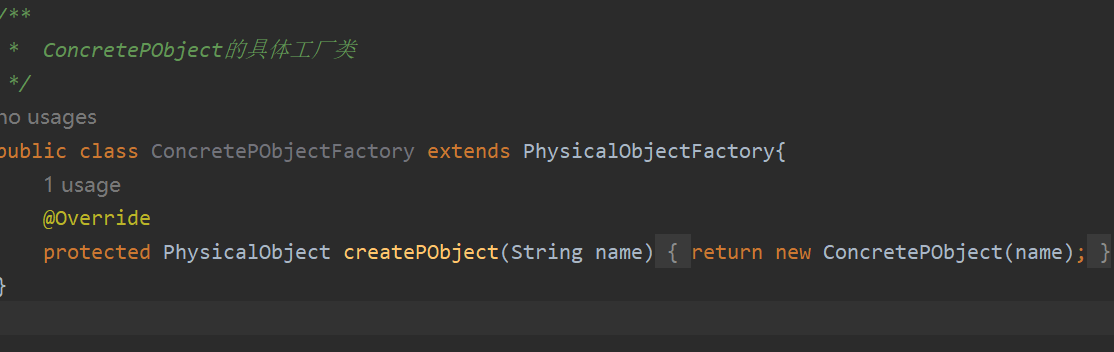
思路：根据工厂设计方法对ConcretePObject进行封装。PhysicalObject是继承自CommonObject的轨道物体的抽象类，而ConcretePObject继承自PhysicalObject类。

类似中心物体，ConcretePObject是一个Immutable对象类。

1. 实现PhysicalObjectFactory
2. 抽象类PhysicalObjectFactory的部分代码

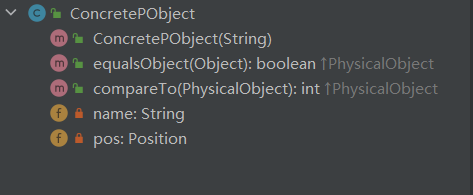


1. 具体类ConcretePObjectFactory的部分代码

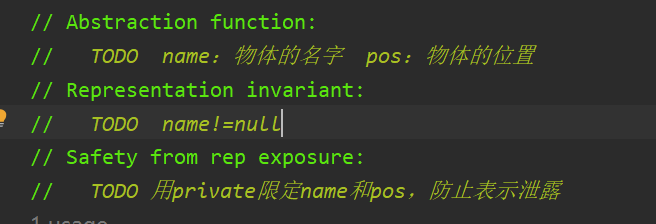


1. 实现ConcretePObject

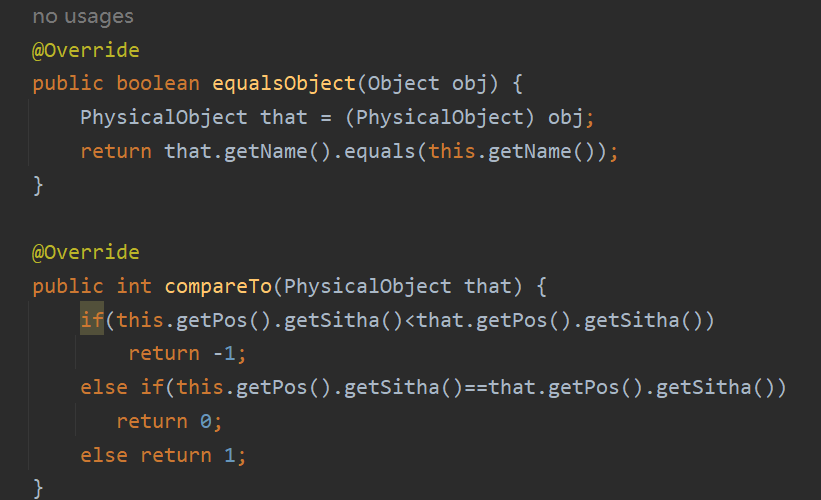
1、ConcretePObject的field



2、ConcretePObject的AF、RI、safety from rep exposure



1. ConcretePObject的部分代码

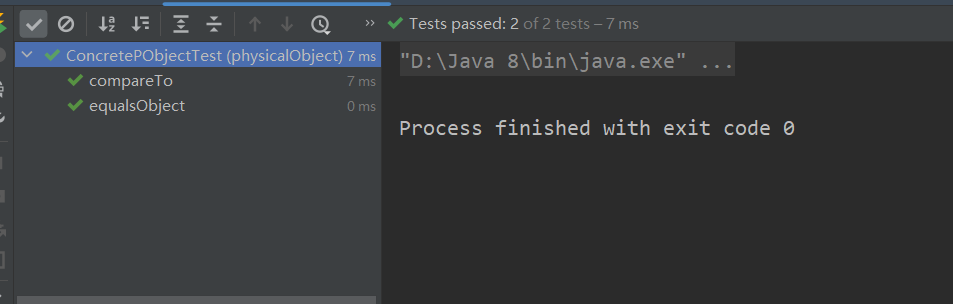


3）测试：

测试策略及部分代码：



测试结果：

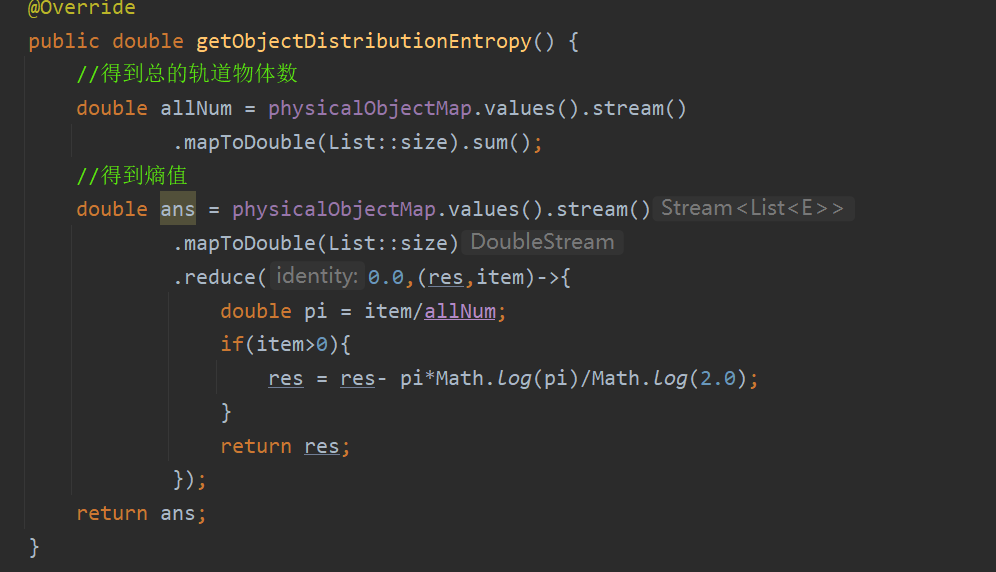


## 可复用API设计

1、计算多轨道系统中各轨道上物体分布的熵值

设计思路:首先熵值的计算公式：W= -Σx\*log(x)。其中x的值为各轨道上物体的个数/总的轨道物体个数的值。根据计算公式编写程序，使用stream流的操作。

部分代码：

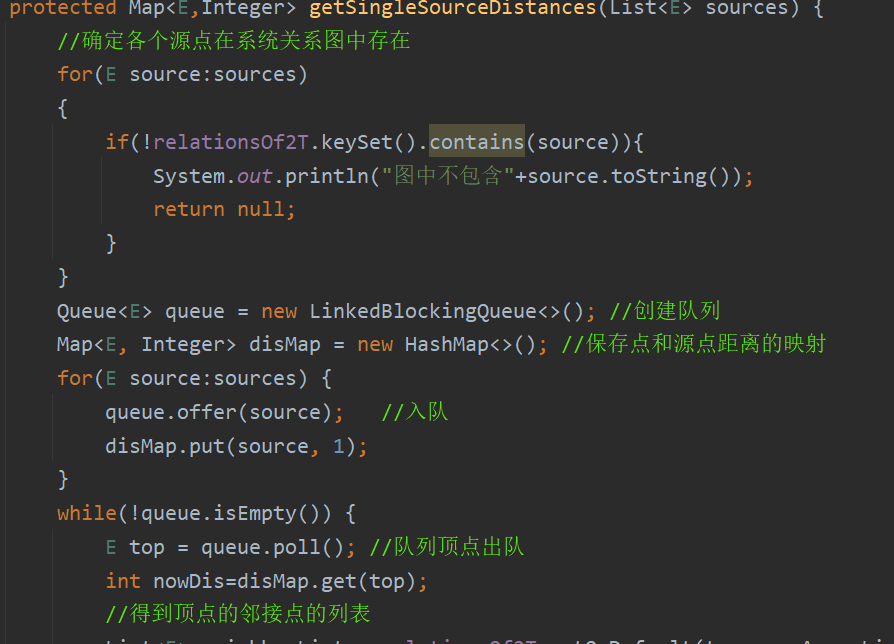


2、计算任意两个物体之间的最短逻辑距离。

设计思路:首先，设计函数getSingleSourceDistances以得到点和最短源点距离的映射。该函数用广度优先搜索的方法容易得到。

调用getSingleSourceDistances函数，将其中一个点作为源点输入，容易得到两个物体之间的最短逻辑距离。

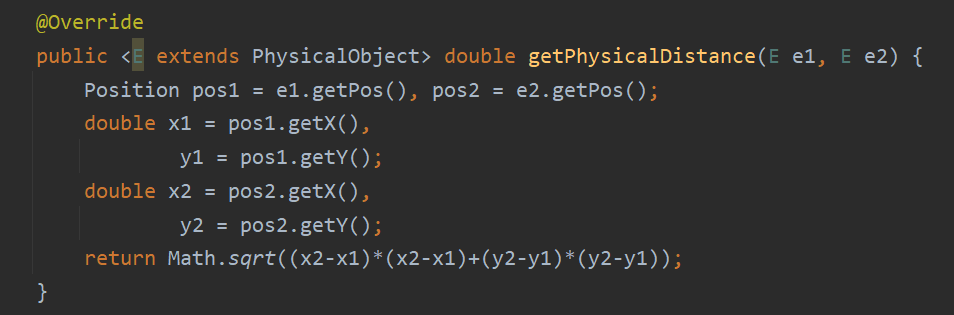
部分代码：



3、计算任意两个物体之间的物理距离。

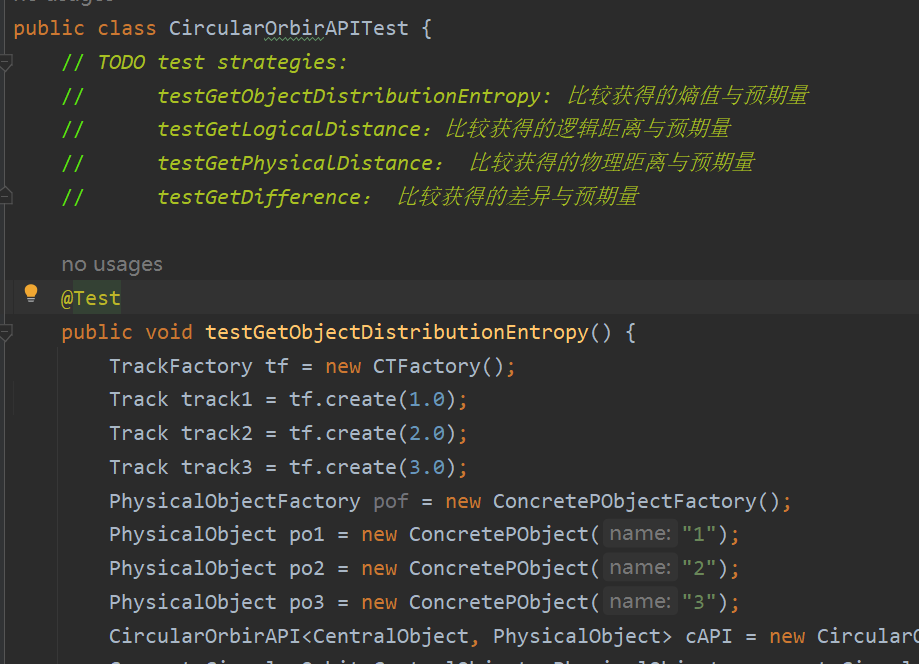
设计思路:得到两个点位置的x，y坐标，然后用数学方法计算即可。

部分代码：

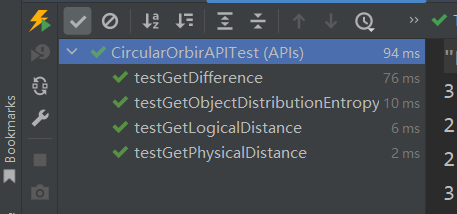


测试：

测试策略及部分代码：



测试结果：



## 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。

设计思路：

使用IDEA插件JFormerDesigner进行UI布局，将一个命名为drawPanel的JPanel放在左侧用来承载轨道系统。

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中根据当前轨道系统的具体信息绘制当前轨道系统。

在复用API中构建CircularOrbitHelper，并在其中实现visualize方法，在其中调用visualizeContentPanel方法获得JPanel，将其添加到一个JFrame中就可显示轨道。

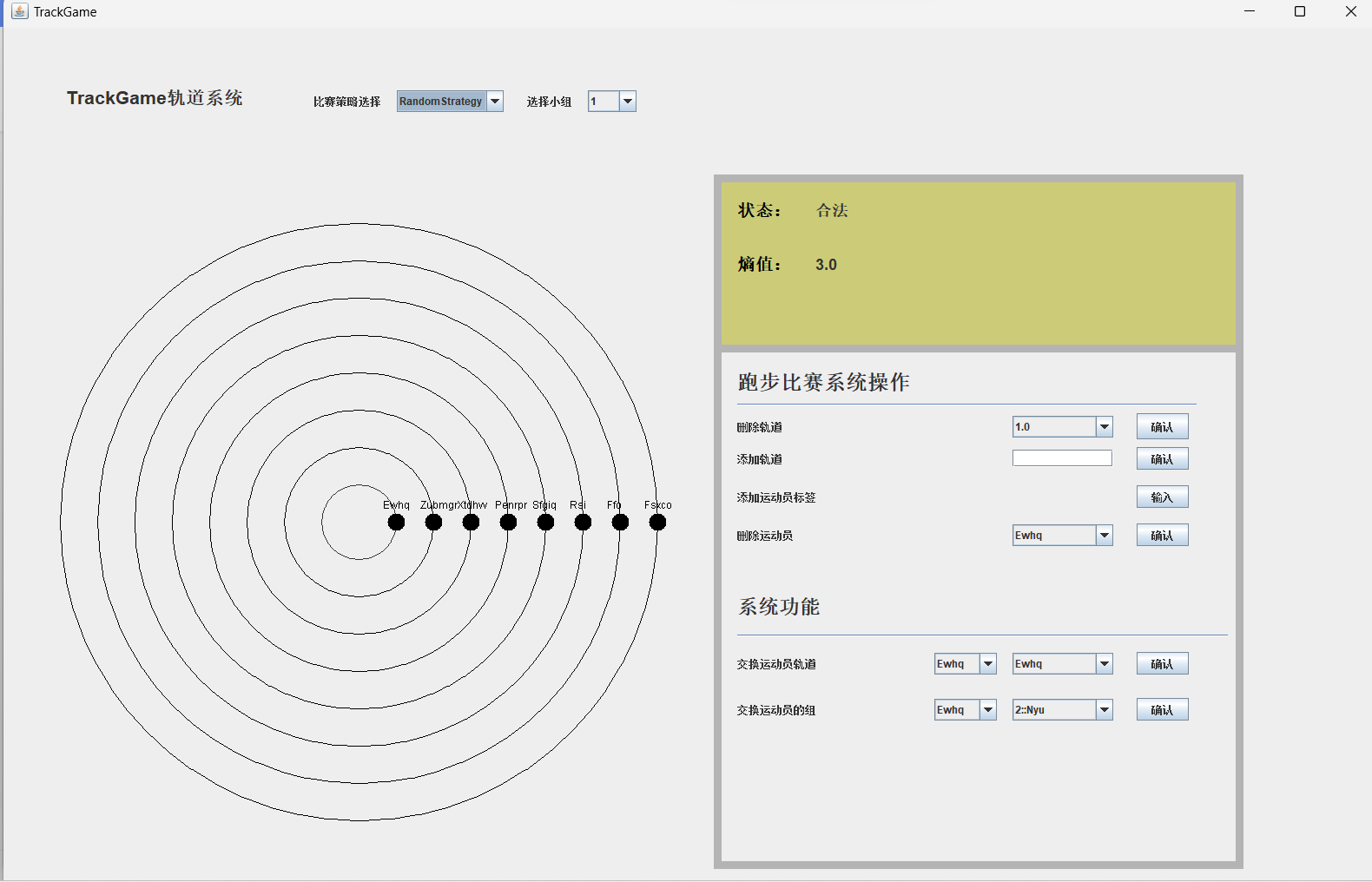
对于每一个应用类，实现visualize(JPanel panel)方法，该方法将visualizeContentPanel中获得到的Jpanel添加到panel中。在各个具体应用的构造类中获得面板上的drawPanel引用，将drawPanel传入到应用类的visualize方法中即可显示轨道系统。

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。

部分代码：



具体实现情况图片：



## 设计模式应用

1. TrackGame

实现了strategy策略设计模式。

设计了一个抽象父类Strategy，并让具体的分类策略RandomStrategy和SortStrategy继承于它。策略原理为根据运动员列表的顺序进行分组，随机分组时打乱运动员列表的顺序，有序分组时按照运动员的最好成绩将运动员列表变为升序，然后按照结果分组、分轨道。

1. AtomStructure

设计一个memory类来保存电子跃迁的轨道记录（包括源轨道、目标轨道等）

然后设计一个careTaker类来调用memory，从而完成历史记录回退等操作。该类同时保存电子跃迁的所有历史记录，根据下标随时调用。

3、SocialNetworkCircle

无特殊的设计模式应用。

## 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。

### TrackGame

设计思路：

1. 首先，由于要使用策略设计模式，以抽象类Strategy为父类的进行比赛分组策略的编写设计，分组时将这部分任务委派给Strategy及其子类即可。
2. 设计Runner类保存运动员的比如ID，姓名，年龄等信息。
3. 设计TrackCircularOrbit类，该类继承自CircularOrbit类，为该跑步比赛系统的具体实现系统，并设计TrackCircularOrbitBuilder类，用于创建该系统，防止TrackCircularOrbit暴露给客户端，防止表示泄露。
4. 设计TrackGame类，在该类里创建TrackCircularOrbit的对象，读取配置文件中的各种信息对轨道、运动员、比赛的各个成员变量完成初始化，并通过方法实现改变运动员的分组、改变运动员轨道等要求的功能。
5. 完成可视化，设计一个TrackGamePanel面板和RunnerFrame框架，通过包导入Miglayout布局，分别保存TrackGame的组件和布局，有关选手信息的各组件及布局。然后将TrackGamePanel面板和RunnerFrame框架委派给TrackGame函数实现可视化。

### AtomStructure

设计思路：

1. 设计AtomCircularOrbit类，继承自CircularOrbit类，并用一个AtomCircularOrbitBuilder进行创建，防止表示泄露。
2. 设计electron类和nucleus类，分别表示原子核和电子，并设计各自的工厂类用于创建对象。
3. 设计一个memory类来保存电子跃迁的轨道记录（包括源轨道、目标轨道等），轨道跃迁用CircularOrbit类的transit功能即可完成。
4. 然后设计一个careTaker类来调用memory，从而完成历史记录回退等操作。该类同时保存电子跃迁的所有历史记录，根据下标随时调用。
5. 可视化设计模式同TrackGame，思路基本相同。

### SocialNetworkCircle

设计思路：

1. 总体框架与前两个应用大致相同，设计Friends类作为轨道物体，设计centralUser作为中心物体，以及对应的轨道系统SocialNetworkCircularOrbit，并创建对应的工厂类。
2. 具体功能通过继承自轨道系统ADT，均能通过其中的方法实现。需要注意，读取文件配置时，与中心用户无关联的朋友不用显示在轨道上，轨道上的朋友通过亲密度（跟中心用户的最短逻辑距离）确定。
3. 可视化设计模式同TrackGame，思路基本相同。

注意：完成所有应用的设计后，再设计一个项目选择面板，通过按钮事件选择要执行的应用。

该界面运行运行结果：



## 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

### AtomStructure

### SocialNetworkCircle

## 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 |
| 04-11 | 19:30-21:50 | 浏览项目，分析结构 | 按计划完成 |
| 04-13 | 20:30-22:00 | 完成ADT部分Track的编写和报告完成 | 按计划完成 |
| 04-14 | 20:00-22:00 | 完成ADT部分L和PhysicalObject部分的代码编写和报告，完成抽象出的共性Position类的编写 | 按计划完成 |
| 04-15 | 18:00-22:00 | 完成ADT部分CircularOrbit接口和其实现类的部分代码 | 按计划完成 |
| 04-17 | 18:45-22:00 | 完成ADT部分CircularOrbit实现类的剩余代码，并完成可复用API的设计和代码编写。 | 按计划完成 |
| 04-18 | 19:20-22:00 | 完成各ADT的测试和可复用API的测试 | 按计划完成 |
| 04-19 | 20:00-22:00 | 完成TrackGame的部分代码 | 按计划完成 |
| 04-21 | 18:00-22:00 | 完成TrackGame的剩余代码并实现可视化 | 按计划完成 |
| 04-25 | 18:00-23:40 | 完成AtomStructure的代码并实现可视化 | 按计划完成 |
| 04-29 | 9:00-22:00 | 完成SocialNetwork的代码并实现可视化，并完成功能选择窗口 | 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 |
|  |  |
|  |  |
|  |  |

# 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

## 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。