

**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 | 李东炫 |
| 学号 | L173710101 |
| 班号 | 1737101 |
| 电子邮件 | [3034125066@qq.com](mailto:3034125066@qq.com) |
| 手机号码 | 15561860820 |

**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 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： λ 子类型、泛型、多态、重写、重载 λ 继承、代理、组合 λ 常见的 OO 设计模式 λ 语法驱动的编程、正则表达式 λ 基于状态的编程 λ 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）。

# 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。

实验环境设置请参见 Lab-0 实验指南。

本次实验在 GitHub Classroom 中的 URL 地址为：

https://classroom.github.com/a/aMg3ti15

请访问该 URL，按照提示建立自己的 Lab3 仓库并关联至自己的学号。

本地开发时，本次实验只需建立一个项目，统一向 GitHub 仓库提交。实验包

含的多个任务分别在不同的包内开发，具体目录组织方式参见各任务最后一部分

的说明。请务必遵循目录结构，以便于教师/TA 进行测试。

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

<https://github.com/ComputerScienceHIT/Lab3-L173710101>

# 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）。

* StellarSystem
* AtomStructure
* PersonalAppEcoSystem

共性：

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。

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：

在 StellarSystem 应用中，请使用 decorator 设计模式，为某些行星增加一

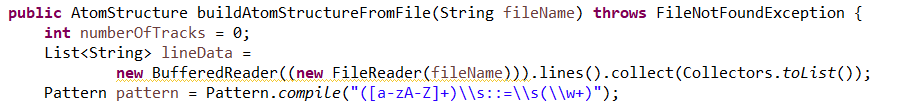
颗或多颗“卫星”——原需求中，轨道上的行星是单个的；在该新需求中，

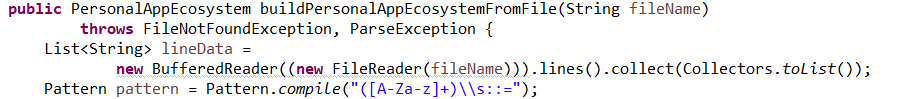
轨道上的行星可能携带一颗或多颗卫星。

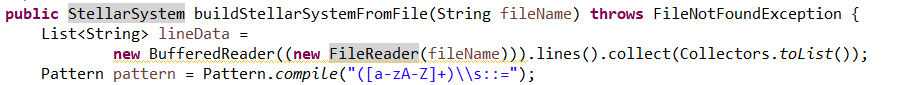
在 **AtomStructure** 应用中，请使用 state 和 memento 设计模式管理电子跃迁 的状态，并可进行状态的恢复。意即：可保存电子每次跃迁前后的轨道信息。

## 基于语法的图数据输入

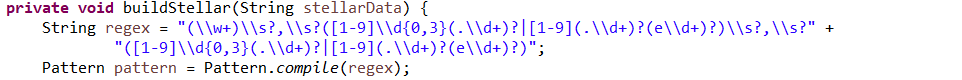
以下分别是三个应用的在输入处理中设计的正则表达式：

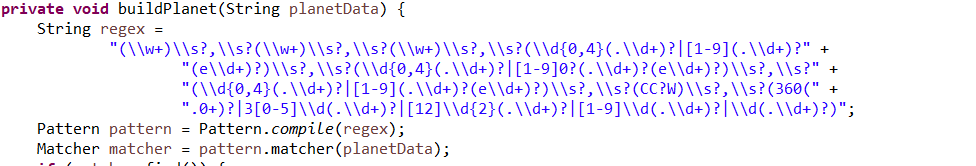






对于在文件中读入的每一行，尝试用不同的pattern进行匹配并进行对应的处理（包括split）。





这里遇到了非常大问题。最开始使用了非常朴素的办法，直接用split来分割，遇到了特别大的问题，对于不同的输入文件经常会犯错误。之后学习了正则表达式，得到了调整解决掉了问题。

## 面向复用的设计：CircularOrbit<L,E>

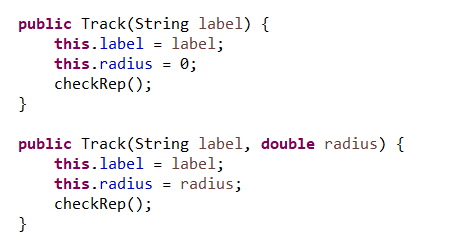
域的设计：

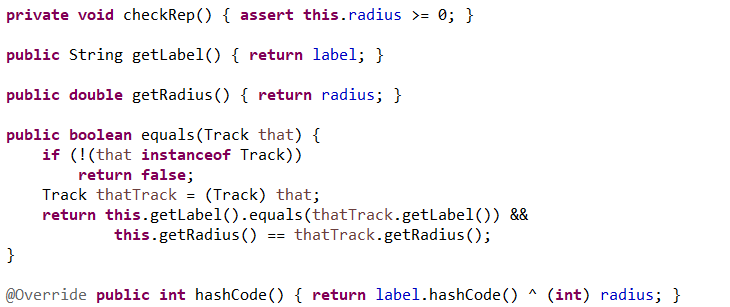
|  |  |
| --- | --- |
| Field | 作用 |
| L centralObject; | 中心物体 |
| Set<E> physicalObjects | 轨道上的物体 |
| List<Track> tracks | 轨道 |
| TracksPhysicalObjectRelation<Track, E> tracksPhysicalObjectRelation | 轨道上物体的关系 |
| CentralPhysicalRelation<L, E> centralPhysicalRelation | 中心物体与轨道上物体的关系 |
| PhysicalObjectRelation physicalObjectRelation | 物体的关系 |

|  |  |
| --- | --- |
| Method | Implementation |
| List<Track> getTracks() | 返回轨道 |
| L getCentralObject() | 得到中心物体 |
| Set<E> getPhysicalObjects() | 得到轨道物体的集合 |
| public void addTrack(Track track) | 添加轨道 |
| public boolean removeTrack(Track track) | 移除轨道 |
| public void setCentralObject(L centralObject) | 设置中心物体 |
| public boolean addObject(Track target, E physicalObject) | 添加物体 |
| public boolean removeObject(E physicalObject) | 移除物体 |
| public boolean addRelationOfCentralPhysical(E physicalObject) | 添加关系 |
| public boolean addRelationOfPhysicalObject(E physicalObject\_1, E physicalObject\_2) | 添加物体间关系 |
| public boolean hasRelation(E centralObject, Track track) | 判断是否有关系 |
| public boolean hasRelation(E physicalObject) | 判断是否有关系 |
| public boolean hasRelation(E physicalObject\_1, E physicalObject\_2) | 判断是否有关系 |
| public boolean transit(E physicalObject, Track target) | 跃迁 |
| public double getRadius(E physicalObject) | 获取半径 |
| public Set<E> getPhysicalObjectsOnTrack | 获取轨道上的物体 |
| public Track getTrackOfObject(E physicalObject) | 获取轨道 |
| public Set<E> getRelatedPhysicalObjects(E physicalObject) | 获取有关的物体 |
| public Set<E> getRelatedPhysicalObjects() | 获取有关的物体 |
| int indexOfTrack(Track track) | 获取下标 |

## 面向复用的设计：Track

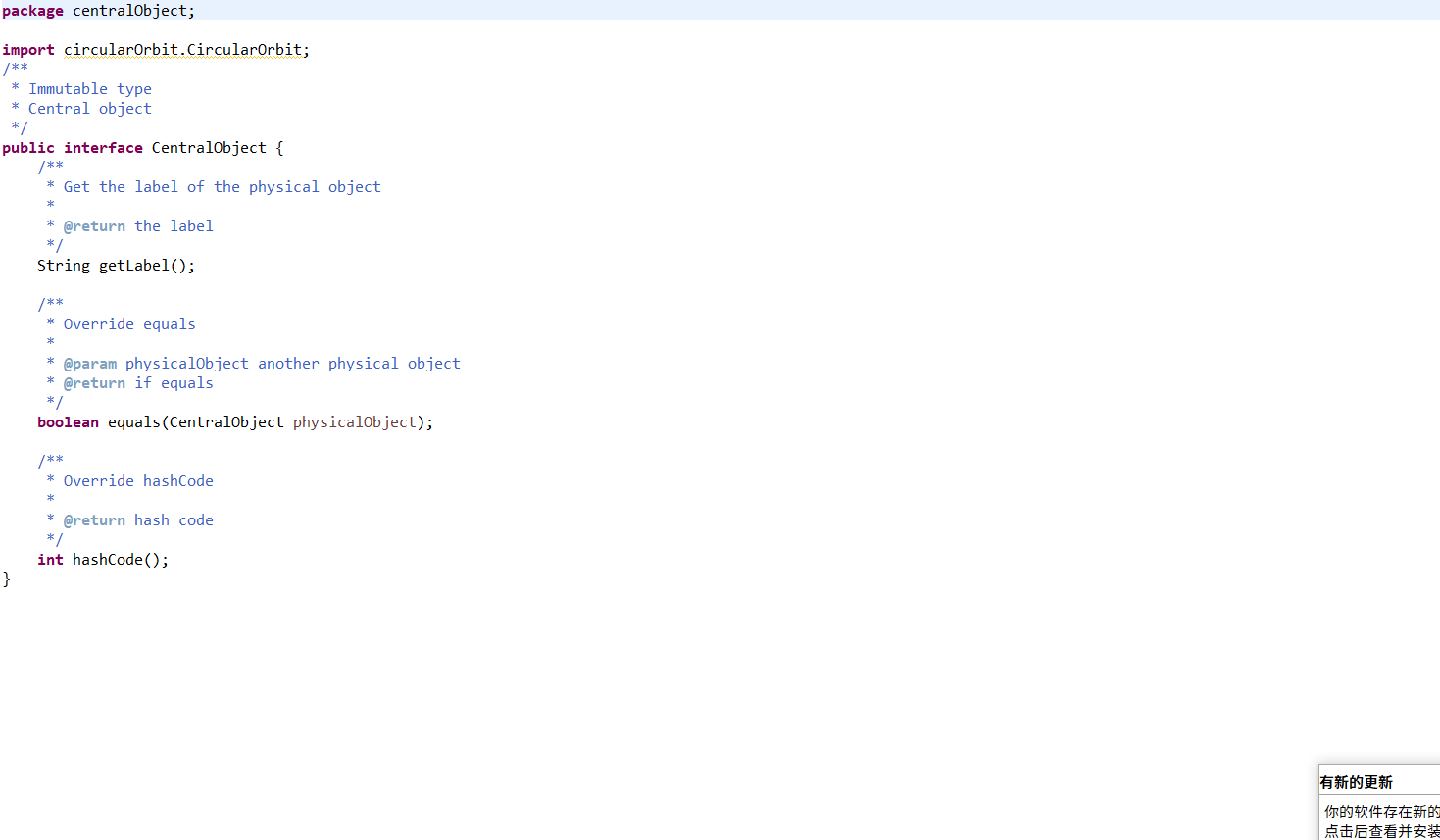
Immutable类。只有label 和radius域。

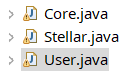




## 面向复用的设计：L

构造接口类CentralObject。

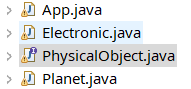




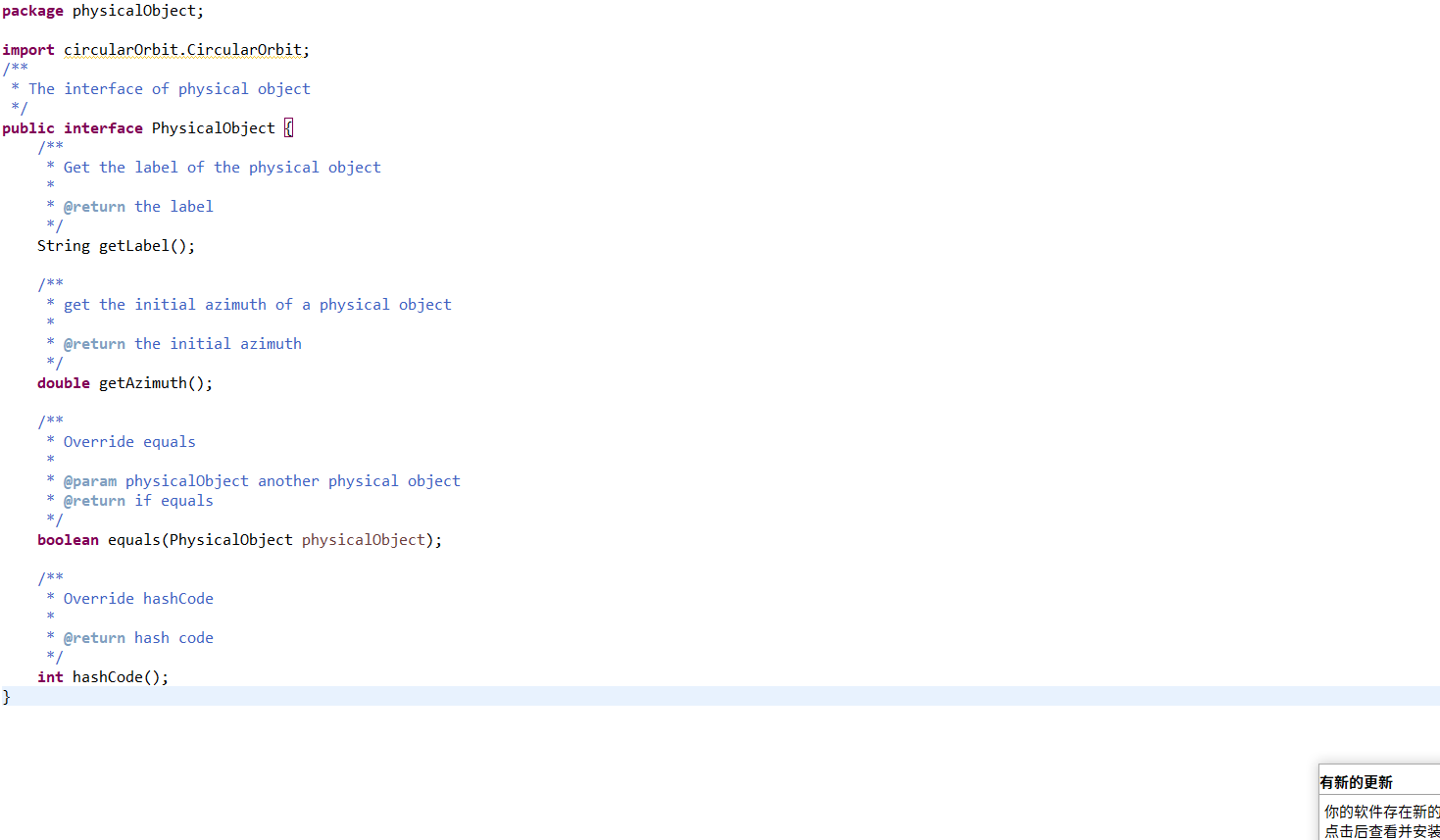
Core,Stellar,User实现CentralObject类

## 面向复用的设计：PhysicalObject

定义接口类PhysicalObject。



这三个类是PhysicalObject的实现。



## 可复用API设计

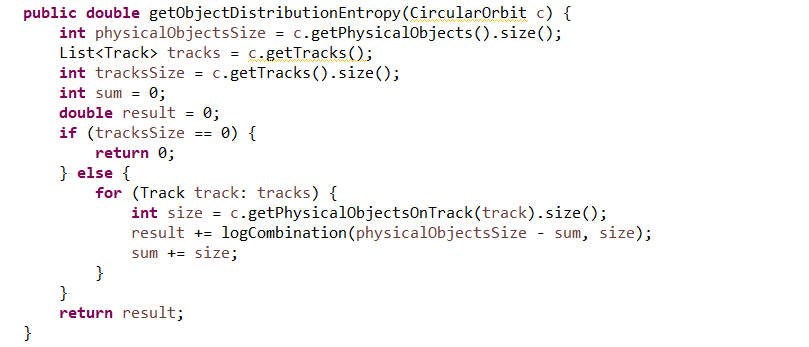
计算多轨道系统中各轨道上物体分布的熵值。有关熵的概念可参见 https://en.

wikipedia.org/wiki/Entropy\_(information\_theory)。通俗的说，如果所有物体都

分布在同一条轨道上，其熵值最低；如果所有物体均匀的分布在每一条轨道

上，整个系统的熵值最高。

double getObjectDistributionEntropy(CircularOrbit c)



 计算任意两个物体之间的最短逻辑距离。这里的逻辑距离是指：e1 和 e2 之

间通过最少多少条边（relation）即可连接在一起。两个物体之间若无关系，

则距离无穷大。（你在 Lab1 的 P3 里已经实现了类似功能）

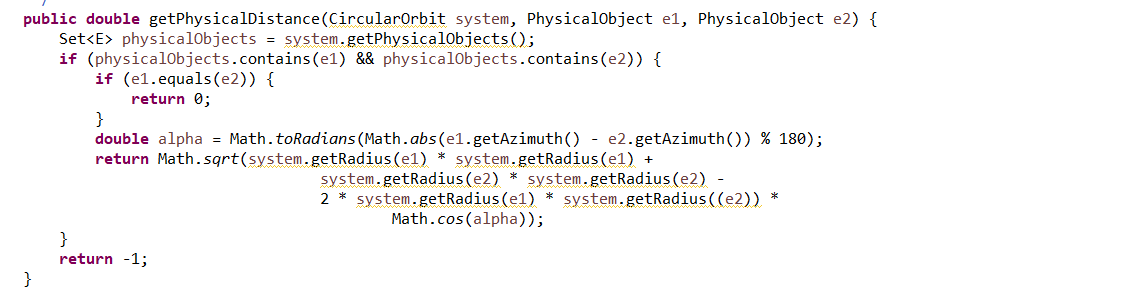
int getLogicalDistance (CircularOrbit c, E e1, E e2)



 计算任意两个物体之间的物理距离。若物体有具体位置，则可在直角坐标系

里计算出它们之间的物理距离。

double getPhysicalDistance (CircularOrbit c, E e1, E e2)



 计算两个多轨道系统之间的差异。请自行设计 Difference 这个 ADT，需要

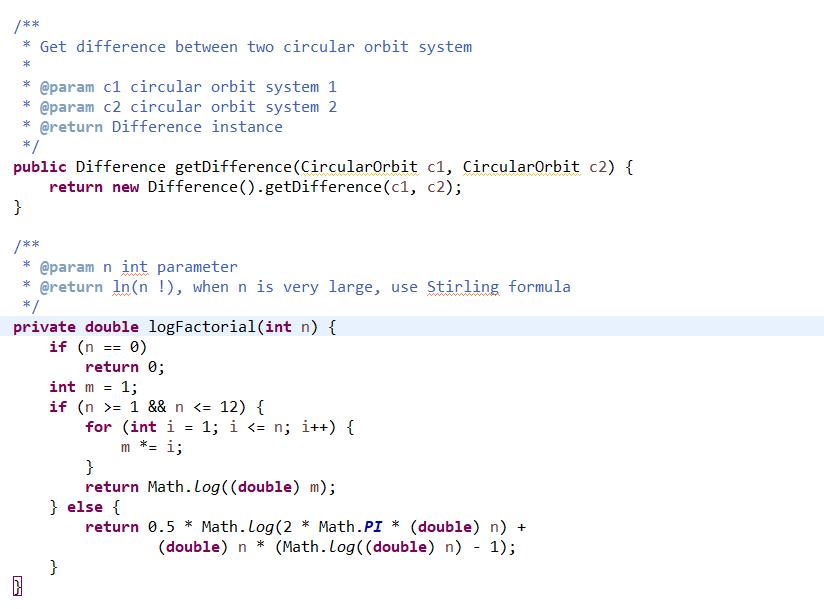
表达出：轨道数的差异、具有相同次序的轨道上物体数量的差异和物体的差

异（如果物体不需要区分，则无需给出物体的差异，只需给出数量差异）。c1

和 c2 必须为同类型的轨道才可以比较（例如不能比较一个太阳系和一个 100

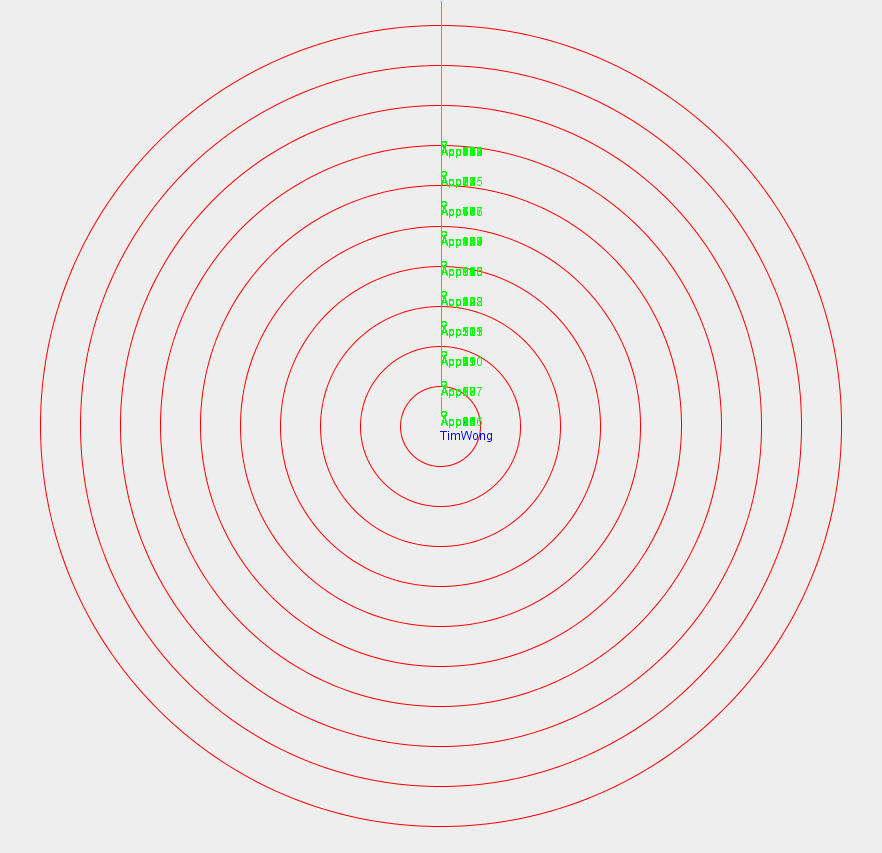
米比赛）。

Difference getDifference (CircularOrbit c1, CircularOrbit c2)



## 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。只完成了简单的轨道系统可视化。完成度比较低。



GUI实现思路：

其实这么简单的实现不需要想太多控件相关的内容，只需要使用一个Jpanel然后花圈就行了…所以实现的不好。

## 设计模式应用

Track,PhysicalObject等对象使用静态工厂方法实现。

对于ConcreteCircularOrbit的构造使用builder设计模式。

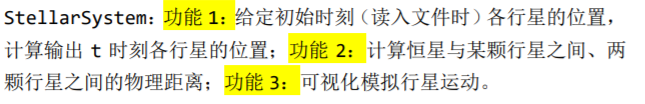
## 应用设计与开发

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。

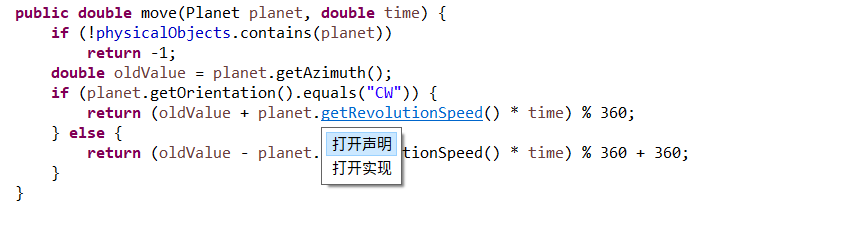
### StellarSystem

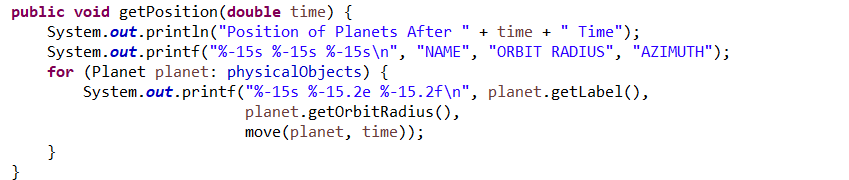
这里实现了实验手册中



前两条实现了。

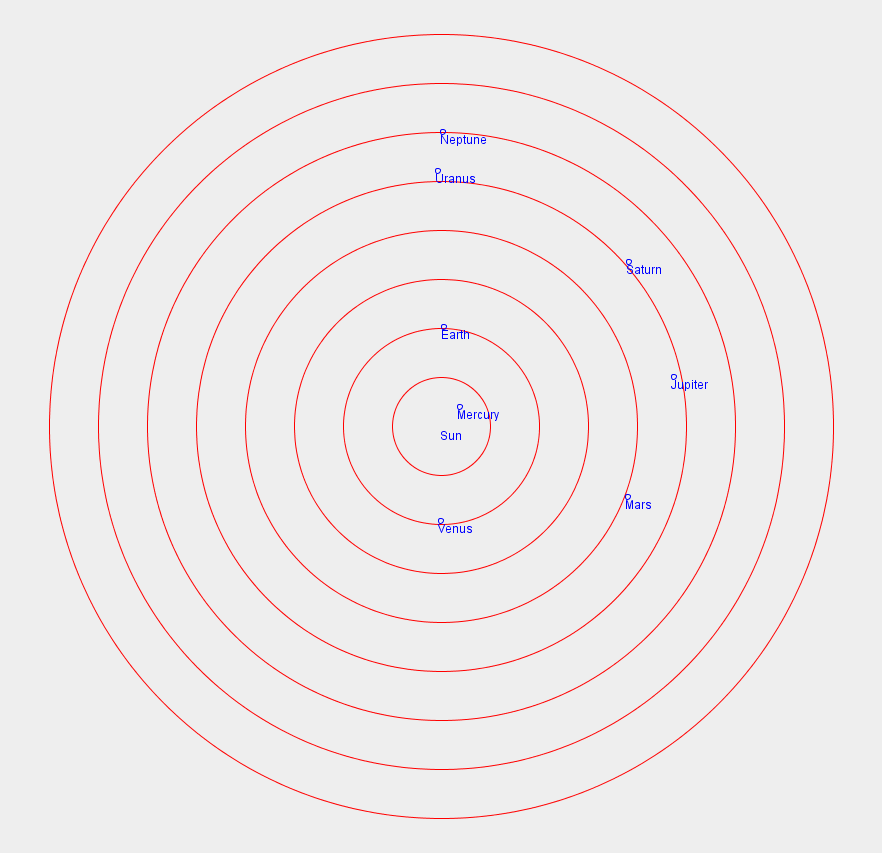
使用了builder模式







第三条由于不会让可视化的图动起来而失败。

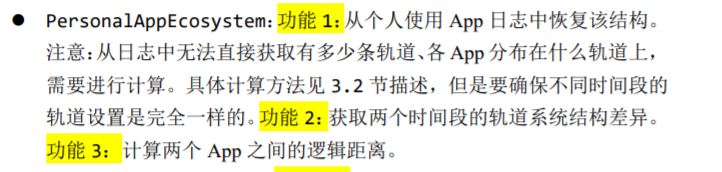


### AtomStructure

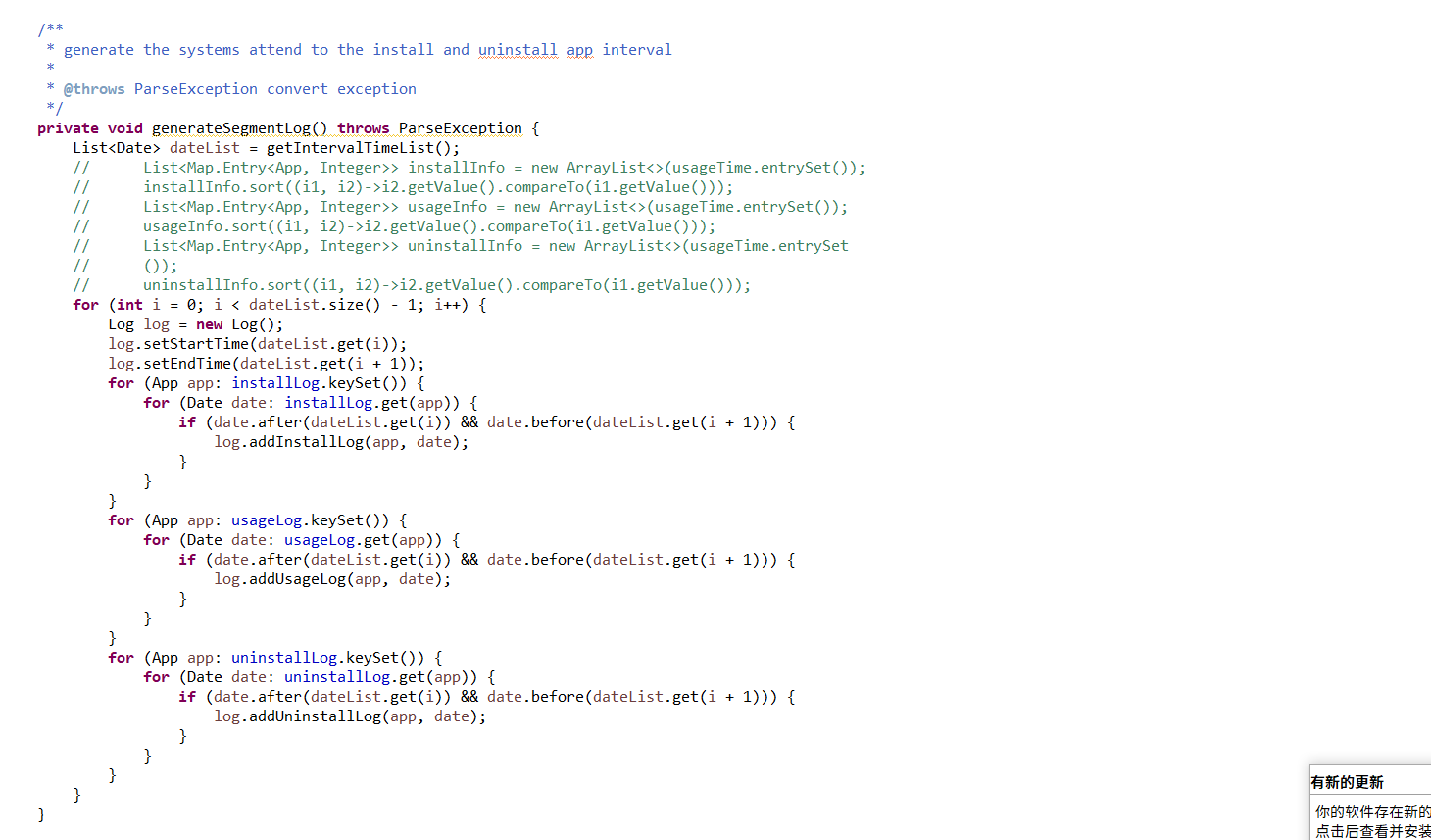


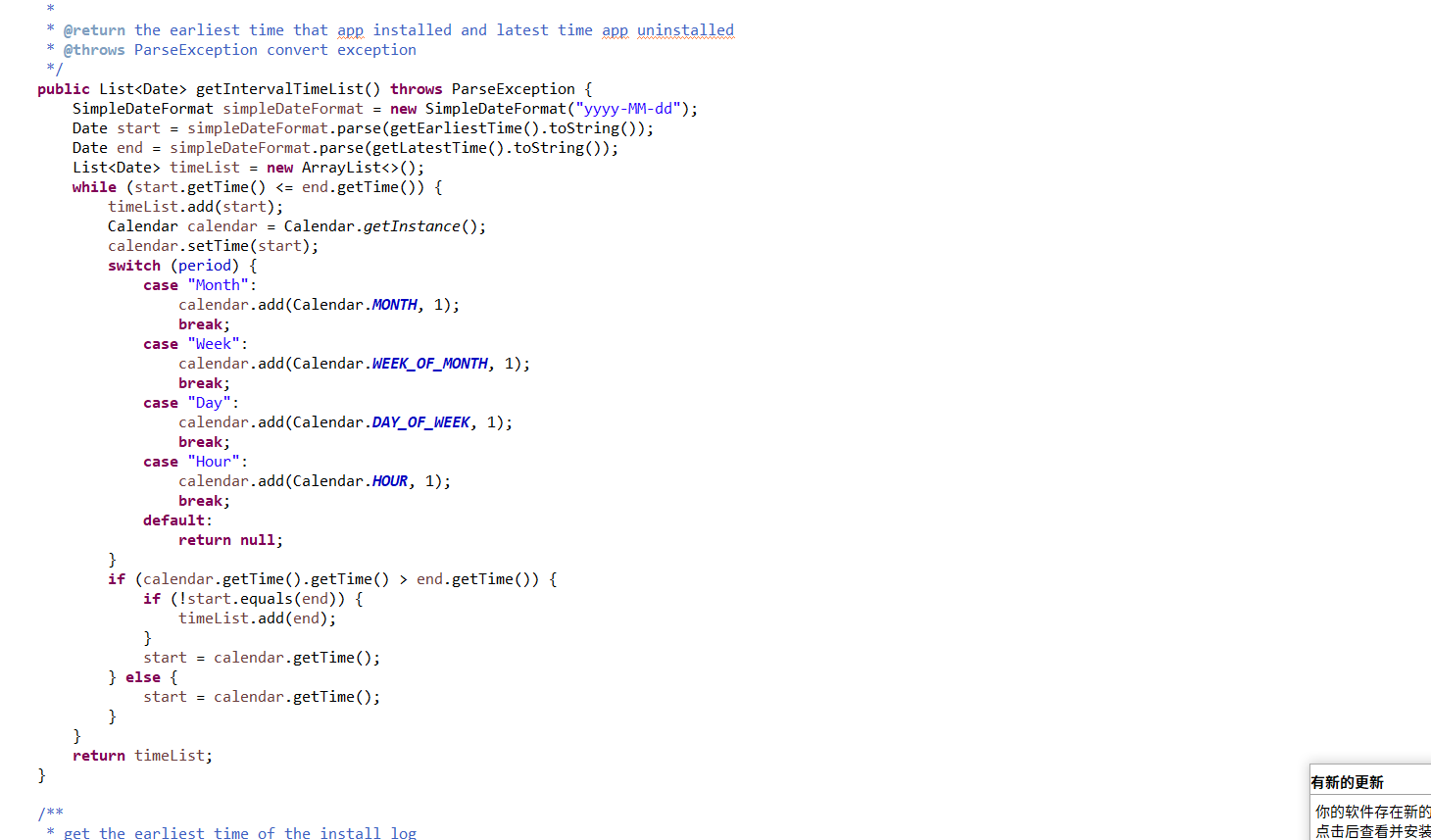
已经在Concrete中实现。

### PersonalAppEcosystem



使用了Builder模式来实现，





## 应对应用面临的新变化

### StellarSystem

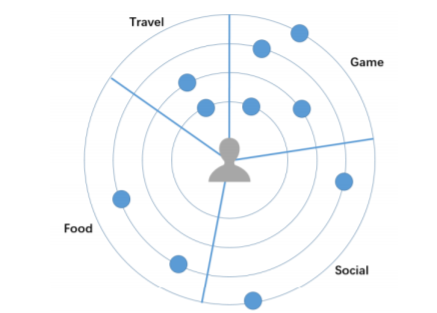
行星轨道变成符合实际的椭圆形轨道（意即轨道的属性从“半径”变化成描述椭圆的多个参数）

### 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

### PersonalAppEcosystem

为轨道系统增加“扇区”特性，根据其中包含的所 有 App 的“业务领域”属性进行聚类，隶属于同一 业务领域的所有 App 放在同一个扇区里。例如在下 图中分为了四个领域：Game、Social、Food、Travel。



## Git仓库结构

# 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 |
| 2019/4/20-  2019/4/2 | 13：30-22：00 | 写ADT | 未完 |
| 2019/4/29 | 13：40-20：00 | 写ADT | 写完 |
| 2019/5/1-2019/5/5 | 14：00-4：00 | 写 | 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 |
| 可视化不太会 | 现在还是不太会。只是用了基础的功能。 |
| ADT设计很困难。 | 确实很困难。用了好久才想明白。 |

# 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？

难度更大了。复用确实减少了很多工作量。

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？

事实上提高了设计的效率。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？
2. 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？

大部分设计模式，是让你在在面向对象的基础上尽量消除继承的手段。一个设计合理的系统，因为解耦充分，各个模块独立性强，所以单元测试应该是比较容易写的多态的本质是运行期动态决定程序的分支走向，也就是“更好的if”，而设计模式，至少是《设计模式》那本书中提到的那些模式，基本上是基于多态的。所以如果你合理的利用设计模式，你设计出的代码应该有较少的if，如果你的代码越使用设计模式if越多，或者更直观地说，缩进越多，你一定犯了错误。

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？

不是一件容易的工作。对于用户指令的难以预期以及正则表达都是一个挑战。

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？

设计之初就要考虑到使用各种设计模式，从0开始构建。实际上我在项目中还是参照了之前的实验的一些经验才得以完成。

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，我写的时候还没有这两个手册可以看。

1. 关于本实验的工作量、难度、deadline。

工作量很大，难度也很大。感谢劳动节。

1. 到目前为止你对《软件构造》课程的评价。

实验难，我认为在课程中还是更多的实际写代码 从思想与实践角度去讲。直接看PPT出来的代码似乎不能更有助于思考。