

**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 | 李震宇 |
| 学号 | 1170300110 |
| 班号 | 1703001 |
| 电子邮件 | 1137859144@qq.com |
| 手机号码 | 18800419432 |

**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：

 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）

# 实验环境配置

Java Eclipse WindowBuilder

GitHub Lab3仓库的URL地址:

https://github.com/ComputerScienceHIT/Lab3-1170300110.git

# 实验过程

## 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
* SocialNetwork

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。

都是轨道系统，由中心物体、轨道、轨道上物体组成。

不同：

TrackGame中心物体Sport并不需要编写。轨道上的物体是Athlete。

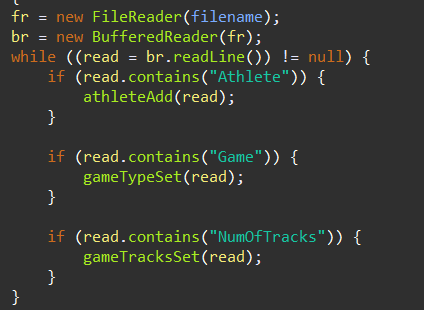
AtomStructure中心物体是CentralAtom，轨道上物体是Electron。

SocialNetwork中心物体是Person，轨道上物体是People。

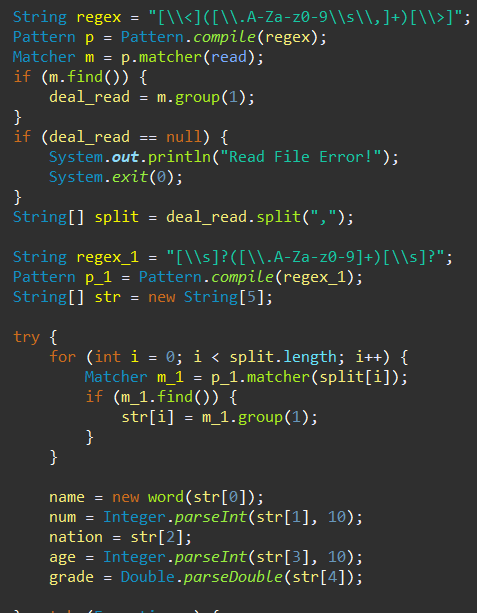
## 基于语法的图数据输入

按行读入文件，首先判断每行读入的信息是什么，然后分别调用处理函数，对字符串进行正则匹配，提取出用来构造轨道系统的信息。

判断每行读入信息：



正则匹配提取信息：



## 面向复用的设计：CircularOrbit<L,E>

接口类型。

主要描述一下接口中定义的方法。

### public E getCertainObject(int num);

找到编号为num的物体，将其返回，若找不到，则返回null。

### public Track getCertainTrack(int num);

找到编号为num的轨道，将其返回，若找不到，则返回null。

### public void addTrack(Track t);

从轨道系统中增加t这条轨道。

### public void deleteTrack(Track t);

从轨道系统中删除t这条轨道。

### public void setCentralObject(L object);

设置轨道系统的中心物体为object。

### public void addObjectToTrack(Track t, E object);

将物体object添加到轨道t上。

### public void readFile(File filename) throws IOException;

读取文件 // 在每个app中具体实现。

### public void transit(E object, Track t);

移动物体object至轨道t。

### public int getTrackNum();

得到轨道数量。

### public int getObjectNum();

得到物体数量。

### *@Override* public MyIterator<L, E> iterator();

迭代器设置。

## 面向复用的设计：Track

Track中定义了半径、编号两个属性(但是后续app中没用到半径，还是写上了。

Track是一个imutable类型的ADT，主要在设计时考虑到防止暴露的问题。

其中所有的属性设置为private。不提供set方法给用户。

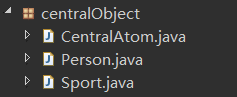
重写了Track中的equals方法认为编号相等的Track即相同的Track。

重写了HashCode方法，将hash值设置为编号大小。

## 面向复用的设计：L

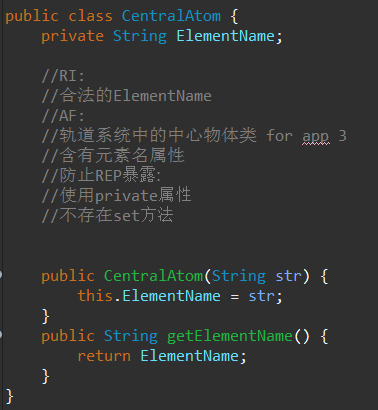
中心物体L。

并没有发现三个APP有什么相似之处，所以依次设计。



分别是原子结构、社交网络、TrackGame的L中心物体。

### CentralAtom

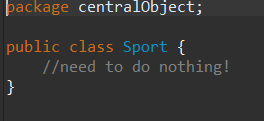


### Person

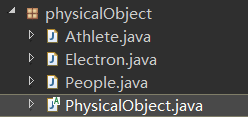


### Sport

不需要设计。



## 面向复用的设计：PhysicalObject



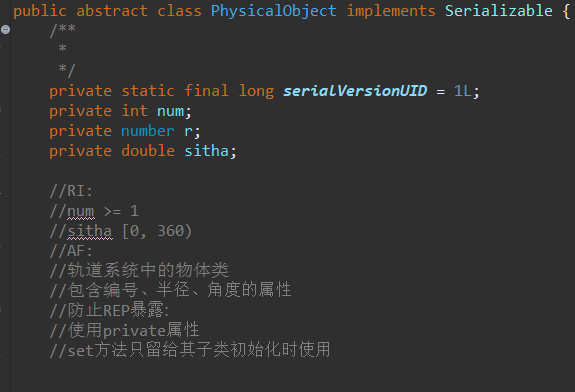
发现了一些相似之处，所以写在了一个抽象类PhysicalObject中。

比如每个轨道物体都有编号属性。

都有(r, θ)属性。

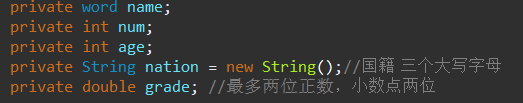
都应该有equals方法。

### 抽象类：PhyscialObject



### Athlete

运动员类，含有：



名字、编号、年龄、国籍、成绩五个属性。

因为inmmutable都设置为私有属性，并不提供set方法。

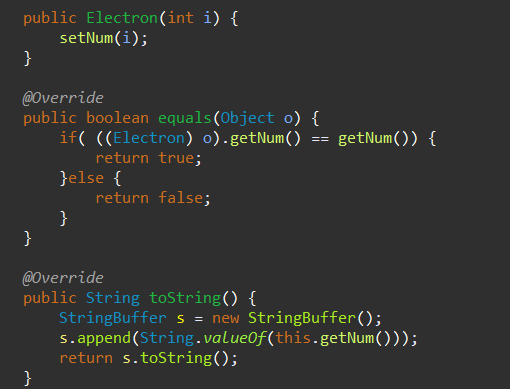
此外，equals和toString方法被重写：



### Electron

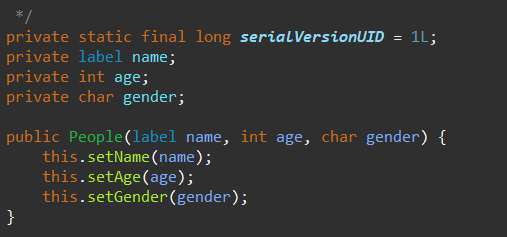
仅仅含有编号属性就可以。

重写equals和toString方法。



### People

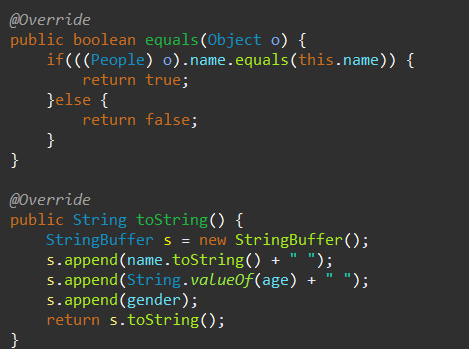
属性：



名字、年龄、性别。

为了保证inmmutable，属性设置为private，set方法设置为private并只提供给其构造器方法使用。

重写equals和toString方法：

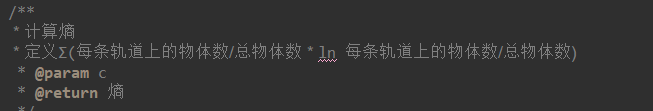


## 可复用API设计

一共是五个API，绘图API在3.8中描述。3.7.5是3.7.4要使用的内部ADT

### double getObjectDistributionEntropy(CircularOrbit c)

比较简单，主要是信息熵的定义，维基百科了一下。



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

得到逻辑距离。

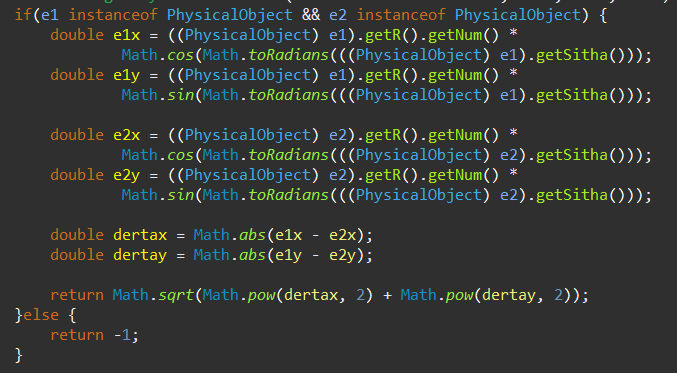
主要是为APP5服务的，也就是那个社交网络。

用广度优先搜索算法就OK

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

得到物理距离。

用数学方法计算就OK



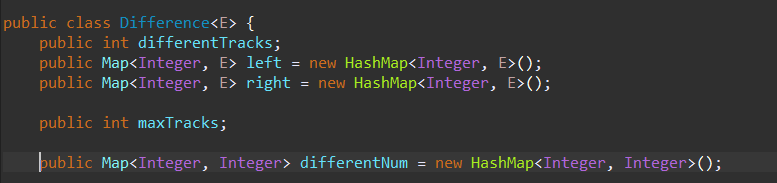
### Difference getDifference (CircularOrbit c1, CircularOrbit c2)

得到两个轨道系统的Difference

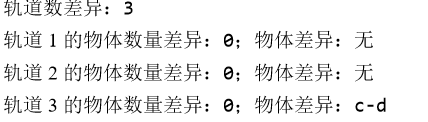
主要就是对比两个轨道系统，然后用判断的方法得到轨道数目差异、每个轨道上物体的差异，对于1/5APP来说，轨道上的物体需要区分，对于原子系统APP来说轨道上的电子不需要区分。

判断轨道系统属性之后，先计算轨道数目差异，之后再每个轨道依次的进行差异的对比。

### Difference



对应要求



differentTracks属性就是那个轨道数差异

differenNum映射是哪个轨道x的物体数量差异y

left是物体差异左边那个。

right是物体差异右边那个。

这是一个mutable类型的ADT，getDifference方法中被构造。

构造之后，主要就是可视化的问题了，就是要用要求的打印方法打印出来，所以就是重

写一个toString方法。



## 图的可视化：第三方API的复用

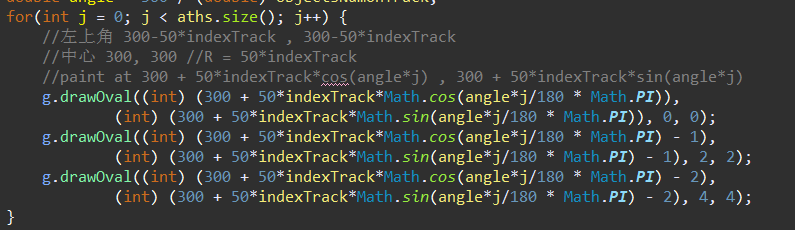
使用Paint()方法画图。

主要就是计算坐标。

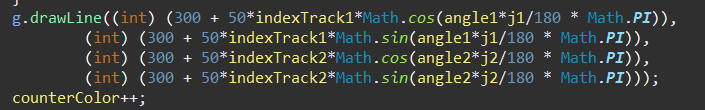
对于APP1/3来说只需要画轨道上的点就可以。

对于APP5来说还需要画物体之间的关系，即点之间的连线。

计算坐标：

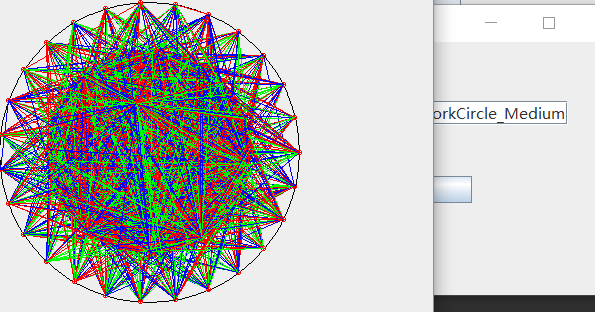


连接直线：

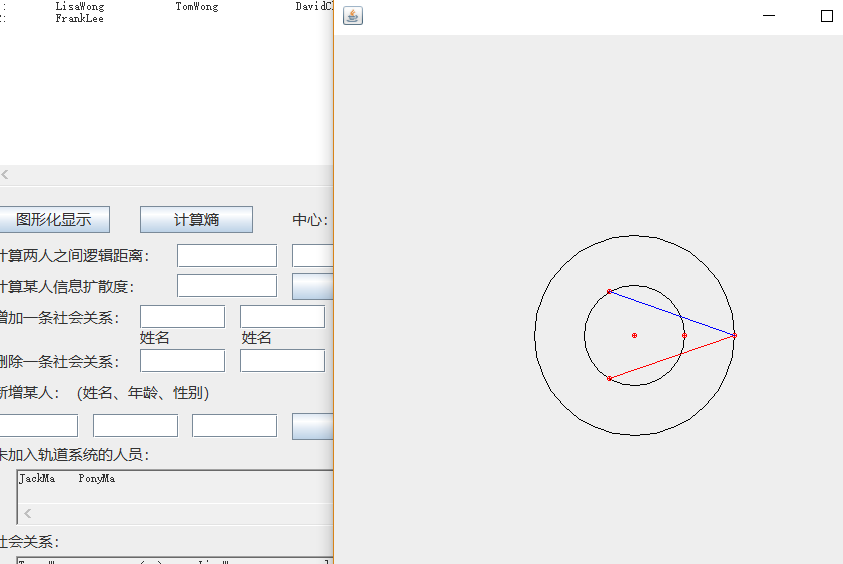


绘画效果：

数据样本太多了：



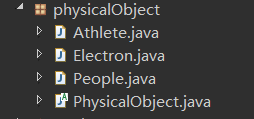
正常的情况：



## 设计模式应用

### PhysicalObject factory 设计模式

设计一个抽象类，然后子类继承它，实现不同的轨道物体类的编写。



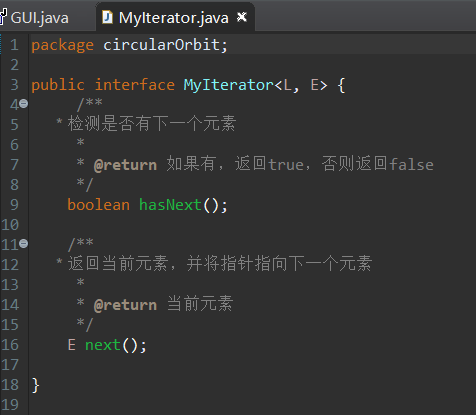
在声明时可以使用PhysicalObject o = new Athlete的形式。

### ConcreteCircularOrbit factory\Iterator 设计模式

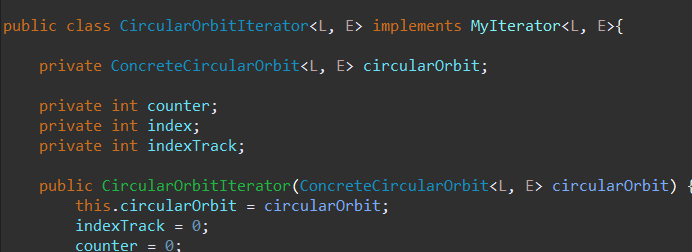
首先factory设计模式，和上边的差不多，子类如TrackGame继承ConcreteCircularOrbit

然后是Iterator设计模式，为其设计迭代器。

首先是接口设计。

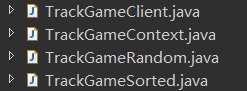


然后实现这个接口：



### TrackGame strategy设计模式

将random和sorted放置运动员的方法分别放在不同的类中。



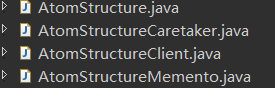
由Context对传入的游戏进行不同的处理：



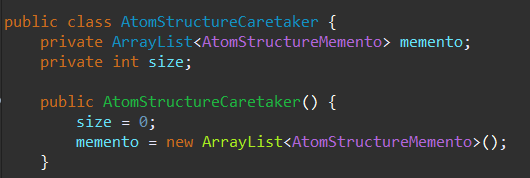
这样就可以让client选择使用不同的放置运动员的方法初始化轨道系统了。

### AtomStructure memento设计模式

备忘录设计模式。



存档时client调用save方法创建新的备忘录Memento，读档时client调用对应函数，通过caretaker中存储的Memento列表加索引恢复存档。



caretaker提供set、get方法设置、读取memento

## 应用设计与开发

### TrackGame

#### 编排比赛方案

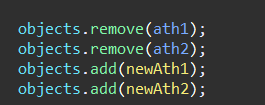
随机或按成绩好坏顺序对运动员进行比赛分组、赛道安排。

随机的话就是对整个运动员objects随机排列组合，然后随机的放到跑道上。

排序的话用Collection.sort方法，按运动员的成绩进行排序，之后分跑道数量奇偶两种情况，放置运动员，如果是奇数，如5，那么放置策略为：3,2,4,1,5，如果是偶数则放置策略为，如4，为2,3,1,4。

#### 交换两个运动员

change方法，就是将两个运动员信息提取出来，创建两个新的对象，然后将原来的对象删掉，然后将新的对象加进去，将轨道信息调整一下即可。



### AtomStructure

#### 电子跃迁

给定指定轨道，将电子跃迁到对应的轨道上。这个也很简单，就该一下电子的轨道信息

和轨道的映射信息即可。

#### 存档读档



和上边的解释很相近，就是用备忘录模式实现这个信息的备份和还原。

### SocialNetworkCircle

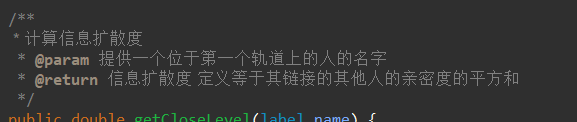
#### 判定用户在哪个轨道上

在GUI图像一栏中，每个轨道上有那些个用户已经可视化的标识出来了，所以GUI内

就没写这个功能。

起始也比较简单，遍历一下轨道到物体的映射就OK了。

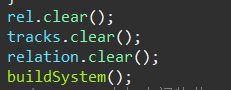
#### 计算信息扩散度



有了定义，遍历一下轨道到物体映射和关系集合就可以很容易的计算出来信息扩散度了。

#### 增加、删除一条社会关系

增加、删除一条社会关系之后，重新构建轨道系统就可以了。

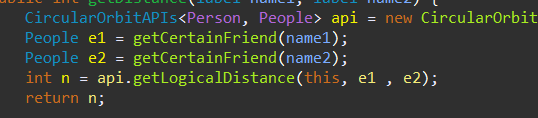


#### 计算用户逻辑距离

将中心物体作为People加入到二维邻接矩阵中，然后使用广度优先搜索，得到用户逻

辑距离。

这里调用API中的方法就可以得到结果了。



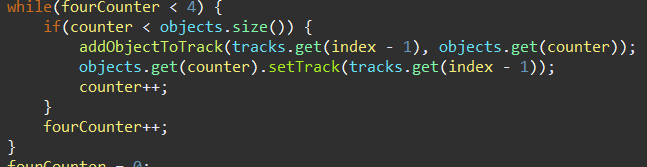
## 应对应用面临的新变化

以下各小节，只需保留和完成你所选定的应用即可。

### TrackGame

要求四个运动员为一组，进行接力比赛。

每次放置运动员都放置四名，最后不满尽量填充，就可以满足实验要求了。



### AtomStructure

要求中心原子有多个，那么就将中心物体属性的类型变成一个List，存储多个原子信息就可以了。

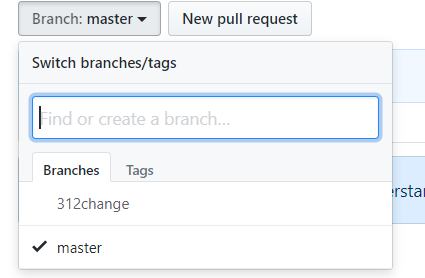


### SocialNetworkCircle

要求单项性，再构建系统的时候，原来是按照无向图构建的，对称的二维数组位置也设置为1，在这里就只将一边设置为1，对称位置不进行操作即可。



## 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 |
| 3.28 | 晚上 | 写CircularOrbit接口 | 完成 |
| 3.29 | 晚上 | 写接口实例化 | 未完成 |
| 3.30 | 晚上 | 写接口实例化 | 完成 |
| 4.1 | 晚上 | 写Track | 完成 |
| 4.2 | 晚上 | 写PhysicalObject | 完成 |
| 4.3 | 晚上 | 写CentralObejct | 完成 |
| 4.5 | 晚上 | 写TrackGame | 未完成 |
| 4.6 | 晚上 | 写TrackGame | 未完成 |
| 4.7 | 晚上 | 写TrackGame | 完成 |
| 4.10 | 晚上 | 写Atom | 未完成 |
| 4.11 | 晚上 | 写Atom | 未完成 |
| 4.12 | 晚上 | 写Atom | 完成 |
| 4.13 | 晚上 | 写SocialNetwork | 未完成 |
| 4.15 | 晚上 | 写SocialNetwork | 未完成 |
| 4.16 | 晚上 | 写SocialNetwork | 完成 |
| 4.17 | 晚上 | 写API | 未完成 |
| 4.18 | 晚上 | 写API | 完成 |
| 4.19 | 晚上 | GUI | 未完成 |
| 4.20 | 晚上 | GUI | 完成 |
| 4.21 | 晚上 | paint | 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 |
| 设计模式不清晰，不太会。 | 看博客，按博客中的类图，一点点写完。 |
| 一些抽象类不太会用。 | 百度学会了。 |
| GUI不会写、paint不会用。 | 看百度学会了。 |

# 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。
10. 使用ADT更抽象，复用性更好，方便以后的编程。
11. 防止意外的错误发生，愿意。
12. 能。比较复杂，要求比较多，比较乱。
13. 在不同情况下运用不同的设计模式，能更方便编写代码。
14. 更方便，但程序写的正则表达式帮你交复杂。
15. 凭空设计，要考虑到好多东西，要抽象。难就难在抽象，主要是多思考相同点，如何抽象ADT。
16. 没有参考，觉得要凉。
17. 多，难，早
18. 海星，挺实用的。