

**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 | 汤添凝 |
| 学号 | 1170300728 |
| 班号 | 1703007 |
| 电子邮件 | ttn912@126.com |
| 手机号码 | 18016240627 |

**目录**

[1 实验目标概述 1](#_Toc3655)

[2 实验环境配置 1](#_Toc7247)

[3 实验过程 2](#_Toc28538)

[3.1 待开发的三个应用场景 2](#_Toc15649)

[3.2 基于语法的图数据输入 3](#_Toc487)

[3.3 面向复用的设计：CircularOrbit<L,E> 4](#_Toc24789)

[3.4 面向复用的设计：Track 5](#_Toc22779)

[3.5 面向复用的设计：Centralobjects 5](#_Toc7077)

[3.6 面向复用的设计：PhysicalObject 6](#_Toc21638)

[3.7 可复用API设计 6](#_Toc24756)

[3.8 图的可视化：第三方API的复用 7](#_Toc11112)

[3.8.1 TrackGame 7](#_Toc645)

[3.8.2 AtomStructure 8](#_Toc2223)

[3.8.3 SocialNetworkCircle 8](#_Toc32)

[3.9 应用设计与开发 9](#_Toc14000)

[3.9.1 TrackGame 9](#_Toc19526)

[3.9.2 AtomStructure 10](#_Toc28642)

[3.9.3 SocialNetworkCircle 10](#_Toc26472)

[3.10 应对应用面临的新变化 12](#_Toc27515)

[3.10.1 TrackGame 12](#_Toc10814)

[3.10.2 AtomStructure 12](#_Toc11307)

[3.10.3 SocialNetworkCircle 13](#_Toc20019)

[3.11 Git仓库结构 13](#_Toc2401)

[4 实验进度记录 13](#_Toc27129)

[5 实验过程中遇到的困难与解决途径 14](#_Toc28665)

[6 实验过程中收获的经验、教训、感想 14](#_Toc3081)

[6.1 实验过程中收获的经验和教训 14](#_Toc32089)

[6.2 针对以下方面的感受 14](#_Toc10716)

# 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

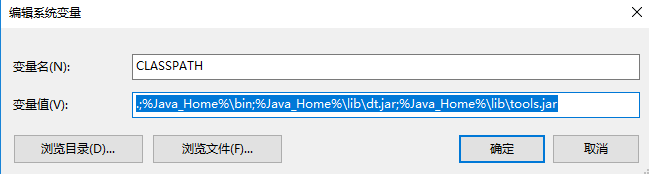
<http://web.mit.edu/6.031/www/fa18/getting-started/>中的安装Eclipse和Gitd的指南非常详细，按照文字和图片即可完成，但是JDK的官网里没有8、9、10版本的，所以我先下载了JDK11，后来在其他网站找到了JDK8版本。

java环境的配置：

安装好JDK后，计算机→属性→高级系统设置→高级→环境变量→系统变量→新建JAVA\_Home变量（填写JDK的安装路径如下图） 

接下来找到并编辑Path变量，在变量值最后输入“%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;”

最后新建CLASSPATH变量，变量值填写“ .;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar ”，

系统变量就配置好了。

URL：[https://github.com/ComputerScienceHIT/Lab3-1170300728](https://github.com/ComputerScienceHIT/Lab2-1170300728)

# 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）。

* 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）。

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，以及可视化。

差异：

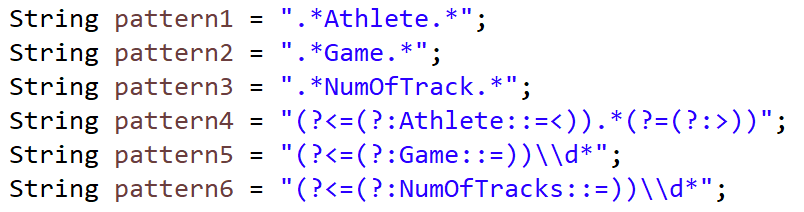
TrackGame需要构造多个轨道系统，而其他两个只需要从文件中读入一个，需要编排。

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中轨道上的电子都是值相同的对象，并且需要实现电子跃迁。SocialNetworkCircle中需要实现物体关系及对应操作，需要计算信息扩散度。

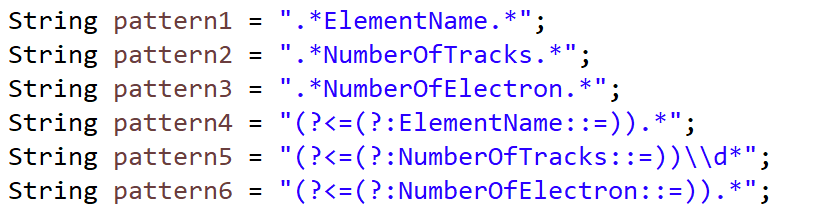
## 基于语法的图数据输入

以下分别是三个应用的在输入处理中设计的正则表达式：

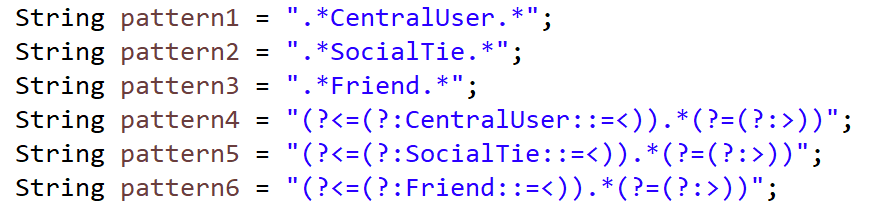
TrackGame：



AtomStructure：



SocialNetworkCircle：



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Social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（嵌套）

## 面向复用的设计：CircularOrbit<L,E>

这一部分需要从所选定的应用中进行 ADT 抽象，设计 CircularOrbit 应提供的接口方法。主要考虑所谓轨道系统的通用方法。例如：

public void addTrack(float radius); 添加一条半径为radius的轨道

public void removeTrack(float radius); 删除半径为radius的轨道

public void addL(L l); 添加一个中心元素l

public void addToTrack(E e,float r);

向半径为radius的轨道上添加一个轨道元素e

public void addLToE(L l,E e);

添加中心元素l到轨道元素e的有向关系

public void addEToE(E e1,E e2);

添加轨道元素e1到轨道元素e2的有向关系

public void addEAndE(E e1,E e2);

添加轨道元素e1到轨道元素e2的无向关系

public Map<E,List<E>> sourceMap();

返回轨道系统所有轨道元素以及从它延伸而出的有向关系

public Map<Track, Map<E, List<E>>> getTrackMap();

返回基于轨道的轨道系统数据

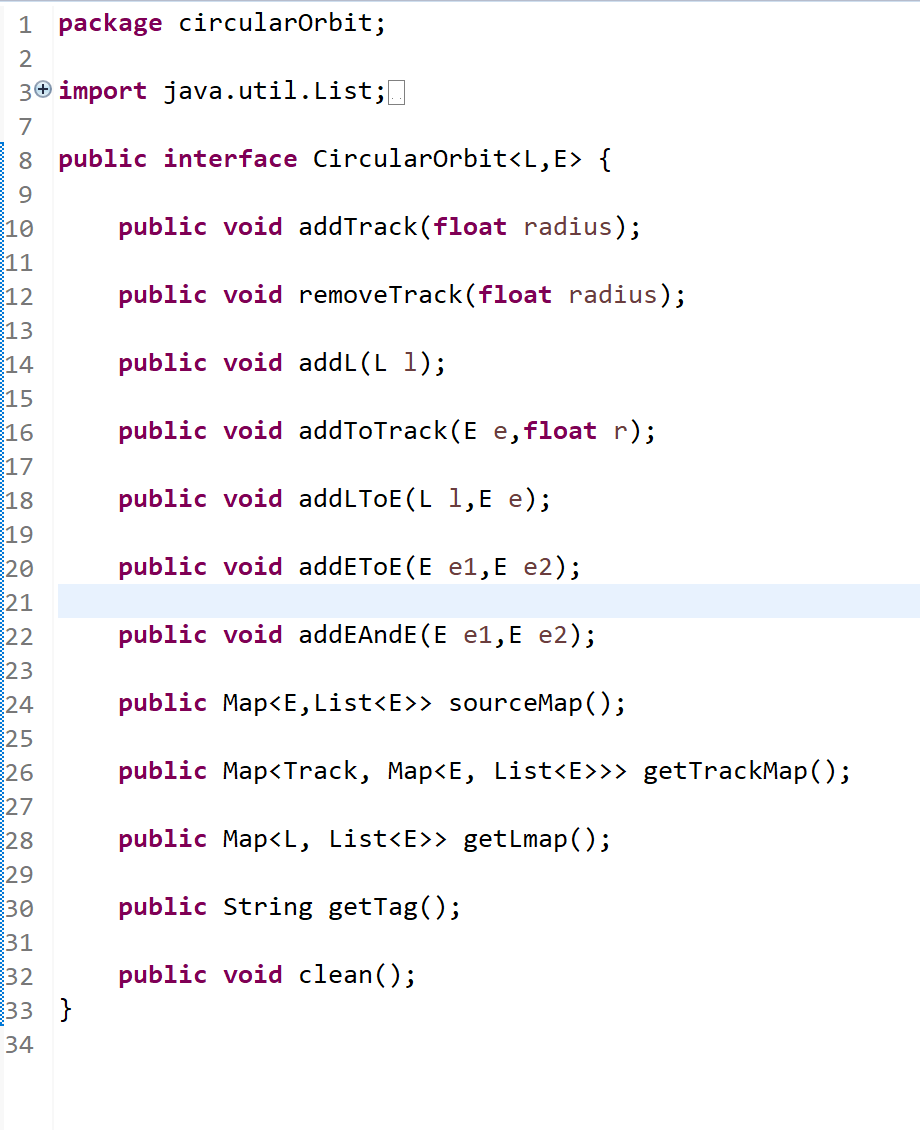
public Map<L, List<E>> getLmap();

返回轨道系统所有中心元素以及从它延伸而出的有向关系

public String getTag(); 返回轨道系统的标签

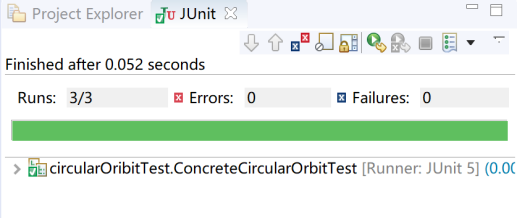
public void clean(); 清空轨道系统

部分代码截图如下：



由于我认为这一部分比较关键，写了较为具体的test，部分test以及test结果截图如下：





## 面向复用的设计：Track

Immutable类。只有一个radius域。

方法：

getradius，返回轨道半径。

## 面向复用的设计：Centralobjects

Immutable类。

构造centralobjects类，其中有两个子类（TrackGame没有中心物体）：

1、atomic

方法：getter

2、CentralUser

方法：getter

在应用中调用centralobjects作为中心物体。

## 面向复用的设计：PhysicalObject

Immutable类。

构造physicalobjects类，其中有三个子类：

1、Athlete

方法：getter

2、electron

方法：getter

3、FriendUser

方法：getter

在应用中调用physicalobjects作为轨道物体。

## 可复用API设计

计算轨道系统熵值。

经过学习可知熵值的计算方法，核心问题是如何计算ln值，而java的math库自带log（）函数满足了这个功能。于是，我们逐轨道计算-xlnx并求和即可。

获取最短逻辑距离。

在前几次实验中多次使用了最短路径算法，这里主要问题在于如何获得邻接矩阵，我选择了通过轨道物体集合对应二维数组横纵坐标来实现，部分代码如下：

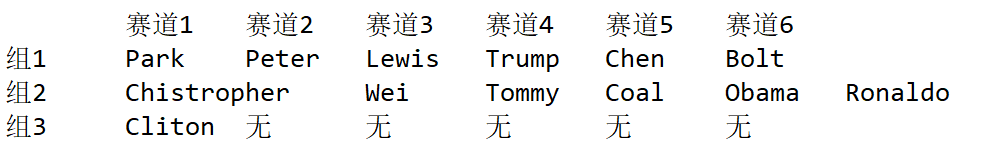


获取物理距离：因为在三个应用中不考虑物理位置，所以不予实现。

## 图的可视化：第三方API的复用

### TrackGame

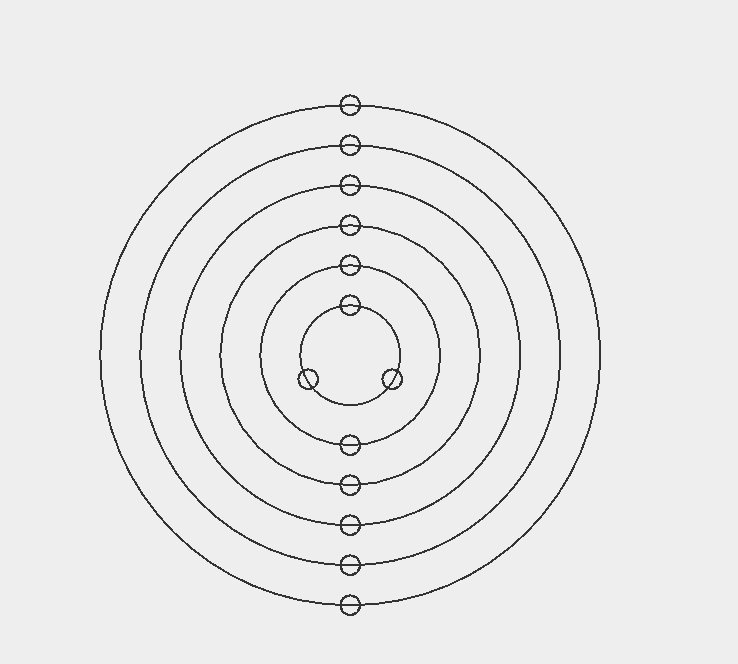
无序的赛道安排表：



有序的赛道安排表：

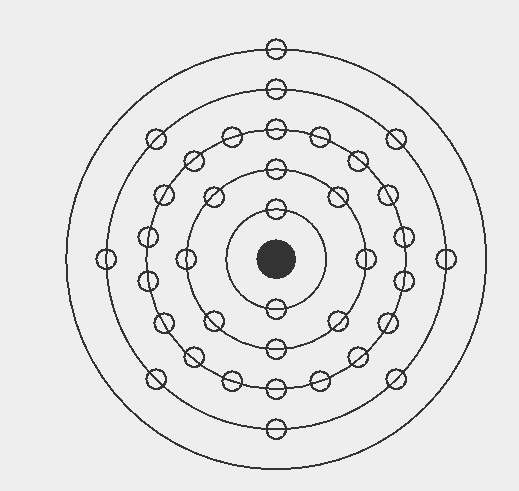


赛道制成的轨道模型示意图



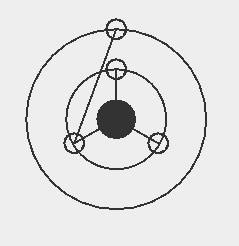
### AtomStructure

元素的轨道模型示意图：



### SocialNetworkCircle

人际关系网的轨道示意图：



## 应用设计与开发

### TrackGame

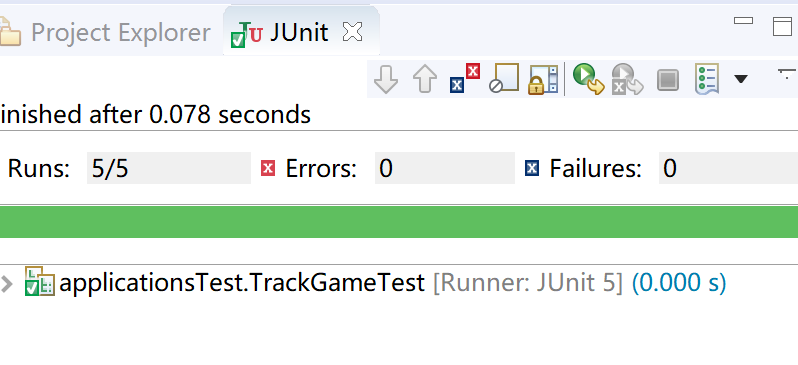
功能1：分别将划分轨道系统和轨道的任务委托给autogame1与autogame2完成编排任务。

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。

部分Test以Test结果截图如下：





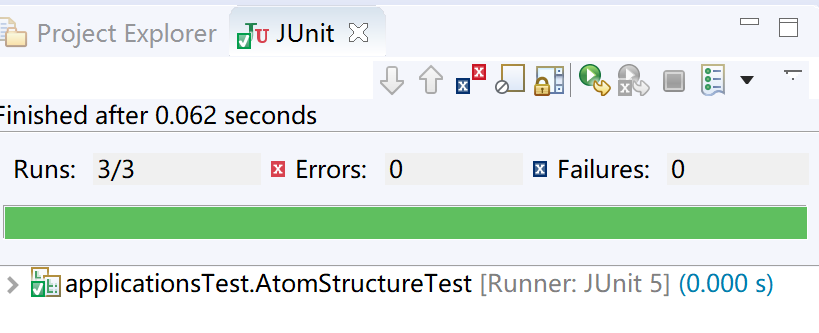
### AtomStructure

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

判断合法性：无。

部分Test以Test结果截图如下：





### 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。

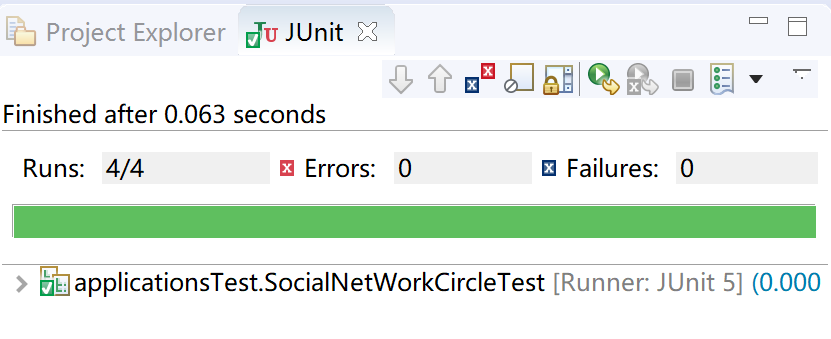
功能3：增加/删除一条社交关系。增加删除关系可以直接调用ConcreteCircularOrbit中实现了的函数，不过考虑到关系改变会引起图的结构的改变（有的节点需要删除，有的节点需要改变所处的轨道），需要调用adjustFriendLocation来调整整个physicalObjectMap，这里采用的（暴力）方法是重复功能1中的BFS过程，重新计算所处轨道，然后删除不能连通的节点。

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

对于应用三，删除轨道上的物体同样会引起图结构的改变，只需要调用ConcreteCircularOrbit中方法后重新调整图结构即可。

部分Test以Test结果截图如下：





## 应对应用面临的新变化

### TrackGame

在最初询问赛制时添加接力跑选项，分别将划分轨道系统和轨道的任务委托给autogame3与autogame4完成编排任务，分别对autogame1与autogame2的组进行变动，使4人一组即可。

部分截图如下：

无序接力赛赛程表：



有序接力赛赛程表：



（由于例子人数不足，这里全部分在一组中进行编组，可以通过增加运动员来实现多组比赛）

### 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，在原先的类中添加名字，分别记录质子或中子，用来区分不同的分子。原本设计中便是以几何形式保存中心元素，这里正常操作即可。

### 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即可。

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。

## 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 |
| 2019-04-29 | 13:00-16:30 | 编写CircularOrbit函数并进行测试 | 按计划完成 |
| 2019-05-01 | 14:00-16:00 | 编写Track\ Centralobjects\  PhysicalObject函数并进行测试 | 按计划完成 |
| 2019-05-02 | 13:45-15:30 | 编写基于语法的图数据输入 | 遇到困难，延时完成 |
| 2019-05-03 | 13:00-17:00 | 编写CircularOrbitAPIs\  CircularOrbitHelper函数 | 按计划完成 |
| 2019-05-04 | 9:00-19:00 | 编写application包游戏函数 | 按计划完成 |
| 2019-05-05 | 10:45-20:30 | 编写test文件以及3.12内容 | 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 |
| 图形可视化问题 | 上网搜索资料并自学Graphics，成功画出轨道示意图 |
| 过多内容如何整合串联 | 类似Lab2的棋类游戏，由小及大书写，可以较为清晰地写完。 |
| 读文件依然是个困难 | 正则表达式掌握不熟练 |

# 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

Java语言是面向对象的典型编程语言。这次实验遇到了许多java语言可视化、大规模编程上的困难，但是经过查阅书籍、上网搜索或者寻求助教以及同学的帮助之后都成功解决了。

这次实验复习并巩固了以前学习的编程知识，加强了面向ADT编程的计算思维。还学会了各种eclipse的操作方法，有助于java编程。

## 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？

前者是从底层开始设计编程内容，而后者是从应用场景直接设计。由于实验内容简单，这次实验没有彻底体现前者的优势，但当编程规模巨大的时候，前者的思路会更为信息。

复用可以大幅减少工作量。

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？

为了使编程更清晰，后续改动更明确。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？

能，很有趣。

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？

设计模式可以使得大规模编程在局部的思路清晰，很优秀。

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？

这种方法更常规和贴近机器，值得学习和使用。

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？

最难的在于如何串联各个类，沟通好类的关系更容易实现代码设计。

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。

工作量巨大，难度高，deadline较近，使变成难度加大。

1. 到目前为止你对《软件构造》课程的评价。

还不错。