

**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 | 郭子阳 |
| 学号 | 1170300520 |
| 班号 | 1703005 |
| 电子邮件 | guoziyang0033@gmail.com |
| 手机号码 | 18800420598 |

**目录**

[1 实验目标概述 1](#_Toc7889959)

[2 实验环境配置 1](#_Toc7889960)

[3 实验过程 2](#_Toc7889961)

[3.1 待开发的三个应用场景 2](#_Toc7889962)

[3.2 基于语法的图数据输入 2](#_Toc7889963)

[3.3 面向复用的设计：CircularOrbit<L,E> 3](#_Toc7889964)

[3.4 面向复用的设计：Track 4](#_Toc7889965)

[3.5 面向复用的设计：CentralObject 4](#_Toc7889966)

[3.6 面向复用的设计：PhysicalObject 5](#_Toc7889967)

[3.7 可复用API设计 5](#_Toc7889968)

[3.8 图的可视化：第三方API的复用 5](#_Toc7889969)

[3.9 设计模式应用 7](#_Toc7889970)

[3.10 应用设计与开发 7](#_Toc7889971)

[3.10.1 StellarSystem 7](#_Toc7889972)

[3.10.2 AtomStructure 8](#_Toc7889973)

[3.10.3 PersonalAppEcosystem 8](#_Toc7889974)

[3.11 应对应用面临的新变化 8](#_Toc7889975)

[3.11.1 StellarSystem 8](#_Toc7889976)

[3.11.2 AtomStructure 8](#_Toc7889977)

[3.11.3 PersonalAppEcosystem 9](#_Toc7889978)

[3.12 Git仓库结构 9](#_Toc7889979)

[4 实验进度记录 10](#_Toc7889980)

[5 实验过程中遇到的困难与解决途径 10](#_Toc7889981)

[6 实验过程中收获的经验、教训、感想 10](#_Toc7889982)

[6.1 实验过程中收获的经验和教训 10](#_Toc7889983)

[6.2 针对以下方面的感受 10](#_Toc7889984)

# 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

实验环境：

系统：macOS Mojave 10.14.3，Oracle JDK 11.0.3

ide：Eclipse IDE for Eclipse Committers 2018-12 (4.10.0)

编辑器：Visual Studio Code 1.31.1

构建工具：Gradle，Travis CI，Drone CI

版本管理：git

代码托管：Github

配置过程：

Gradle构建脚本build.gradle使用上一次实验的脚本即可。

关于在新分支上开发，使用命令git checkout -b 312change即可创建新的分支，使用git checkout master即可再切换回master分支。

为drone.ci配置了配置文件.drone.yml，使项目支持drone.ci自动构建。

增加了项目对日志系统的支持，日志系统采用slf4j框架和log4j2系统，在支持ASNI字符的控制台可输出彩色日志。配置文件为log4j2.xml。

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

https://github.com/ComputerScienceHIT/Lab3-1170300520

# 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）。

* Stellar System
* Atom Structure
* Personal App Ecosystem

这三个应用场景，都使用了圆形轨道，并且轨道物体在某一时刻都依附于某条确定的轨道上。这三个轨道系统的中心物体和轨道物体都较类似，都是immutable的。差异在于，Stellar System中的轨道物体有确定的在轨道上的位置和自身的运行速度，而Atom Structure和Personal App Ecosystem则不需要轨道上的确切位置，只需要知道依附于哪条轨道即可。

## 基于语法的图数据输入

要求从文件中读入配置并构建出相应的系统。由于配置文件中的各个配置的顺序不固定且可能出现空行，故首先将整个配置文件读取为一整个String，再使用正则表达式去分别匹配。

为了各个类分工明确，创建了一些Reader类来专门负责对配置文件的读取和解析，并返回一个封装了配置文件所需信息的Configuration类，以便于对信息的提取。所有的Reader类实现了ConfigurationReader接口，都实现了Configuration readFile()和void parseConfiguration(String line)方法。所有的Configuration类实现Configuration接口。

对于正则表达式匹配配置，主要还是使用零宽断言式来匹配前后缀。遇到的问题有正则表达式的贪婪问题，当使用(?<=App ::= <)([A-Za-z0-9]+,){2}[a-zA-Z0-9.\_-]+,\".+\",\".+\"(?=>)来匹配形如App ::= <Wechat,Tencent,13.2,"The most popular social networking App in China","Social network">的文本时，由于正则表达式默认匹配的贪婪模式，会尽可能地匹配更多的字符，“.”可能会将剩下的几行全都包括，导致多条配置被当作一条，这时，只需要在限定词后面加上?即可取消贪婪匹配。

对于数字的读入，有两种读法：普通数字和科学计数法，就需要使用|连接，匹配数字的表达式为((\\d+[\\.]?\\d\*e\\d+)|(\\d+[\\.]?\\d\*))。

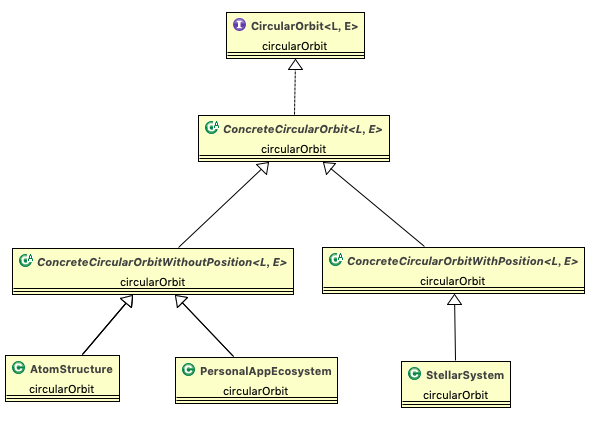
在读取日期时，首先使用SimpleDateFormat类将字符串转化为Date类，再转化为Calendar类保存。

在读取StellarSystem的配置文件时，表示行星速度的量使用的是线速度，建议转化为角速度再保存。

在读取PersonalAppEcosystem的配置文件时，需要按照时间间隔将日志划分，建立多个PersonalAppEcosystem，而不是一个类，多个PersonalAppEcosystem保存在PersonalAppEcosystem类下的静态变量ecosystems中。

## 面向复用的设计：CircularOrbit<L,E>

要求设计轨道系统类，根据设计使用四层架构，主要的系统差异在于轨道上的物体是否有具体位置。第一层是CircularOrbit接口，用于约定类的实例方法；第二层为ConcreteCircularOrbit抽象类，实现了CircularOrbit接口，实现了一些无论有无位置的轨道都一致的方法，如addTrack()等；第三层有两个抽象类：ConcreteCircularOrbitWithPosition和ConcreteCircularOrbitWithoutPosition类，都继承了ConcreteCircularOrbit抽象类，用于具体实现有无位置的轨道不同的方法，如addPhysicalObject()方法，由于参数列表的不同，而无法在底层实现，需要分别实现。第四层则是各个应用的具体实现类，分别继承自第三层的两个抽象类。

UML类图如下：

由于Track是immutable的，无法记录轨道物体与轨道的对应关系，PhysicalObject也是immutable的，无法记录轨道物体之间的联系，所以这些对应关系都应当在轨道系统层记录，故需要在轨道系统层设立成员变量Map<E, Position> positions、Map<E, Map<E, Double>> physicalRelations和Map<E, Double> centralRelations，来分别记录轨道物体对应关系、轨道物体之间的联系和轨道物体与中心物体之间的联系。记录位置的Position是一个抽象类，有两个类具体实现了该类：AnglePosition和CommonPosition，CommonPosition中只有一个Track成员变量，而AnglePosition中还有一个Double类型的成员变量来记录具体的位置。

其中，PersonalAppEcosystem类在读取文件建立时，需要按照日志的时间分割为多个不同的轨道系统，于是在PersonalAppEcosystem中创建静态变量List<PersonalAppEcosystem> ecosystems，以保存一个文件产生出的多个轨道系统。

## 面向复用的设计：Track

Track类由于是immutable的，无需表述与轨道物体之间的对应关系，唯一需要保存的属性只有一个轨道半径radius，将radius设置为final，在构造时传入，并且设置一个getter方法即可，getter方法应使用防御性拷贝。

## 面向复用的设计：CentralObject

CentralObject由于也是immutable的，无需表示与轨道物体之间的联系，所以只需要一个String name属性即可。CentralObject是一个抽象类，让诸如AtomicNucleus、Stellar和User继承该抽象类，并添加自己独有的属性即可。

## 面向复用的设计：PhysicalObject

PhysicalObject也是immutable的，无需表示轨道物体之间的关系，也无需表示与中心物体的关系。但是轨道物体有两种：带有速度和位置的和不带速度和位置的，于是创建抽象类PhysicalObject表示不带有速度和位置的，以及PhysicalObjectWithSpeed抽象类继承PhysicalObject表示带有速度和位置的轨道物体，其中PhysicalObjectWithSpeed有属性speed、clockwise和startAngle来表示角速度、是否顺时针旋转和初始的角度。再让App、Planet和Electron按类型分别继承这两个类。

## 可复用API设计

共需要设计四个可复用的API，可以说这是一个工具类。

计算轨道系统的熵使用信息熵的公式：

其中，是第i条轨道上的物体占总物体个数的比例。

计算两个轨道物体之间的逻辑距离，由于所选择的应用中，两物体之间的关系没有具体数值，都可以看作1，只需要对Relations构成的图进行广度优先搜索即可，从初识物体出发，在第几层搜索到了目标物体，则与目标物体的逻辑距离就是几。

计算轨道物体之间的物理距离，对于在轨道上有具体位置的系统，计算两物体的物理距离只需要计算出两物体的直角坐标后计算即可，对于没有具体位置的系统，物理距离为两物体所在轨道的半径差。

计算多轨道系统之间的差异，对于不需要表达物体差异的轨道系统，使用普通的Difference类，遍历每条轨道，统计物体数量的差异即可。对于需要表达物体差异的轨道系统，创建DifferenceWithObject类继承了Difference类，其中有一个Map来表示各个轨道上物体的差异。对这两个类重写toString()即可。

## 图的可视化：第三方API的复用

图的可视化使用了Java原生的可视化库swing，并且实现了动态可视化，可观察轨道物体的运动情况。

窗体使用javax.swing包下的JFrame类，在JFrame中创建新的类JPanel作为主面板，通过重写public void paint(Graphics g)方法，在JPanel上绘制。将Graphics对象转换成Graphics2D对象，再分别绘制出中心物体、轨道、轨道物体和轨道物体之间的无向关系。

在JPanel类中设置了一个成员变量offset，定义为最大的轨道与窗体宽的商，用于让轨道系统可以完美地处于窗体中，所有的轨道半径在绘制的时候都要乘以offset，以实现等比例绘制。

Graphics2D对象可以通过drawOval(int x, int y, int width, int height)方法绘制出一个椭圆，当width和height相同时，即可绘制出正圆。需要注意的是，参数列表中的x和y，不是圆心的坐标，而是圆的左上角的坐标，即圆外接正方形左上角的坐标。也可以通过drawLine(int x1, int y1, int x2, int y2)绘制出两个点之间的一条线段。绘制实心点时，需要借助Ellipse2D.Double对象，Ellipse2D.Double circle = new Ellipse2D.Double(x, y width, height); g2.fill(circle);。

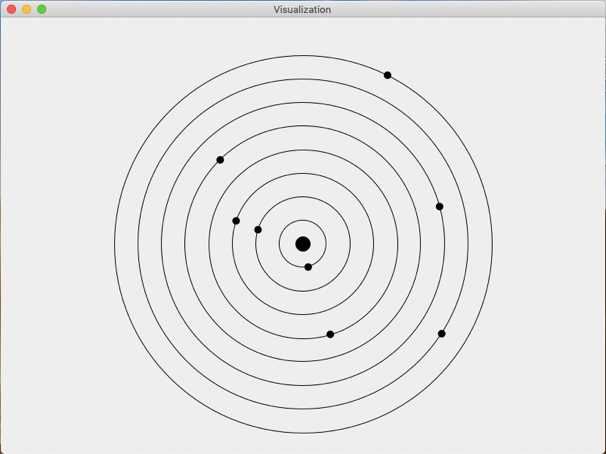
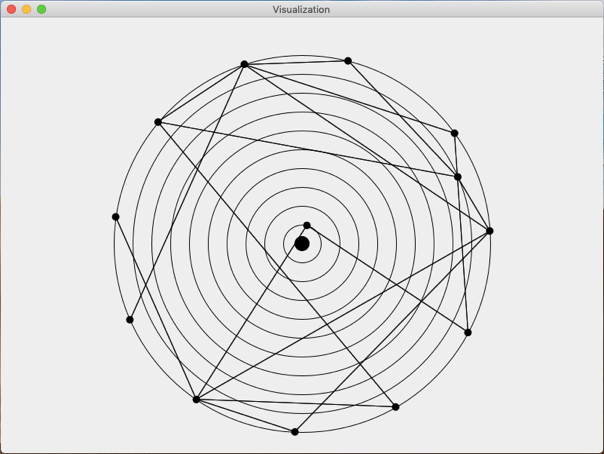
实现运动效果可以使用多线程绘制的方法，令自己的JPanel实现Runnable接口，实现public void run()方法，在run()方法中使用一个while(true)循环，每次循环将panel中的一个成员变量time加一，并且休眠一段时间，再调用repaint()方法，休眠的时间和每秒钟刷新的次数有关，如要实现每秒24帧刷新，休眠的时间就设置为41（毫秒）。要实现动态化，绘制物体和关系的代码就需要根据成员变量time实时地计算出物体的坐标后再绘制。

对于没有速度和位置的系统（如Atom Structure），预先设定了一个全局的速度，可以实现其动态效果。

动态绘制的时候有一个需要注意的点，就是重写哪一个组件的paint()方法。由于JFrame是一个重量级组件，大量调用repaint()方法会导致图像闪烁，甚至出现白屏的问题，而JPanel属于轻量级组件，不会出现闪烁问题，故应该在JPanel层重写paint()方法。

对接应用的时候，需要将DefaultCloseOperation设置为JFrame. DISPOSE\_ON\_CLOSE而不是JFrame.EXIT\_ON\_CLOSE，以避免在窗体关闭后整个JVM也随之关闭的情况。

可视化效果如下：



## 设计模式应用

创建Track和每个PhysicalObject时，使用了工厂方法模式，为每个类创建了一个工厂类，创建对象是使用对应的工厂类创建。

创建ConcreteCircularOrbit对象时，使用了抽象工厂模式，创建了一个工厂类，在类中有多个方法用于创建不同的ConcreteCircularOrbit对象。

使用Iterator模式，为每个系统创建了相应的迭代器，用于按照特定顺序遍历所有的PhysicalObject。

使用了façade模式，所有的API放置在helper类CircularOrbitAPIs类中。

## 应用设计与开发

### StellarSystem

1 . 从文件中创建系统

2 . 可视化当前系统

3 . 添加轨道

4 . 添加物体

5 . 删除物体

6 . 删除轨道

7 . 计算系统的熵

8 . 计算t时刻行星位置

9 . 计算恒星与行星的物理距离

10. 计算行星之间的物理距离

### AtomStructure

1 . 从文件中创建系统

2 . 可视化当前系统

3 . 添加轨道

4 . 添加物体

5 . 删除物体

6 . 删除轨道

7 . 计算系统的熵

8 . 电子跃迁

### PersonalAppEcosystem

1 . 从文件中创建系统

2 . 可视化当前系统

3 . 添加轨道

4 . 添加物体

5 . 删除物体

6 . 删除轨道

7 . 计算系统的熵

8 . 不同时间段轨道的差异

9 . 计算App的逻辑距离

## 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。

添加StellarTrack类，继承自Track类，添加私有属性radius2，则默认的radius为椭圆的长半轴长，而radius2为短半轴长。修改对应的工厂方法，并将StellarSystem中所有的Track改为StellarTrack即可。

### 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。创建一个接口Nucleon（核子），并创建Proton（质子）类和Neutron（中子）类实现该接口。在AtomicNucleus类中添加一个Set，存有Nucleon，并开放相应的添加方法即可。

### PersonalAppEcosystem

为轨道系统增加扇区的特性，对业务领域进行聚类。在轨道系统中增加属性Map<String, Set<App>> fields，并在遍历App时按照领域分别存入相应的Set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。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 2019-3-31 | 20:02-23:36 | 完成轨道系统的基础架构 | 按时完成 |
| 2019-4-15 | 19:30-23:46 | 完成AtomStructure和StellarSystem | 按时完成 |
| 2019-4-23 | 11:50-12:35 | 完成可视化 | 按时完成 |
| 2019-4-30 | 12:00-12:45 | 完成PersonalAppEcosystem | 按时完成 |
| 2019-5-3 | 18:02-19:14 | 完成312change | 按时完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| Track、PhysicalObject和CentralObject都是immutable的 | 重构项目 |
| 轨道系统可视化 | 不使用第三方库，而是使用Java原生的swing库绘制 |
| 设计模式的使用 | 在线学习 |

# 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

学会了面向ADT编程和项目重构的方式，使用了不同的设计模式，并体会到了代码复用的便捷。

## 针对以下方面的感受

面向ADT编程，需要提前思考整个项目的结构，以期做到较高的代码复用程度。在开始项目之前，就需要经过许多深思熟虑，项目过程中也可能出现许多次的重构。使用设计模式更是可以大大增加代码的复用程度。由于没有办法得知实验四和五的要求，只能尽可能地对整个项目进行抽象，以期望在需求变动的时候可以以较小的代价完成变动。本实验难度较大，工作量较大，但是时间较长。