哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

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	
学号	
班号	
电子邮件	
手机号码	

目录

1 实验目标概述……1
2 实验环境配置……1
3 实验过程……1
3.1 待开发的三个应用场景……1
3.2 基于语法的图数据输入……1
3.3 面向复用的设计：CircularOrbit<L,E>……1
3.4 面向复用的设计：Track……2
3.5 面向复用的设计：L……2
3.6 面向复用的设计：PhysicalObject……3
3.7 可复用 API 设计……3
3.8 图的可视化：第三方 API 的复用……3
3.9 设计模式应用……3
3.10 应用设计与开发……3
3.10.1 TrackGame……4
3.10.2 StellarSystem……5
3.10.3 AtomStructure……5
3.10.4 PersonalAppEcosystem……6
3.10.5 SocialNetworkCircle……6
3.11 应对应用面临的新变化……7
3.11.1 TrackGame……7
3.11.2 StellarSystem……7
3.11.3 AtomStructure……7
3.11.4 PersonalAppEcosystem……7
3.11.5 SocialNetworkCircle……7
3.12 Git 仓库结构……8
4 实验进度记录……8
5 实验过程中遇到的困难与解决途径……9
6 实验过程中收获的经验、教训、感想……9
6.1 实验过程中收获的经验和教训……9

6.2 针对以下方面的感受 9

1 实验目标概述

编写具有可服用行和可维护性的软件。

为了实践上述目标，本次试验将开发一个应用程序，程序将包含三个模块：

1. TrackGame：模拟跑步比赛，后续改进为接力赛；
2. AtomStructure：模拟原子结构，原子核为一个物体，后续改进为多个中子和质子；
3. SocialNetworkCircle：模拟人际关系网络。

2 实验环境配置

https://github.com/ComputerScienceHIT/Lab3-1173710229.git

3 实验过程

3.1 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
- SocialNetworkCircle

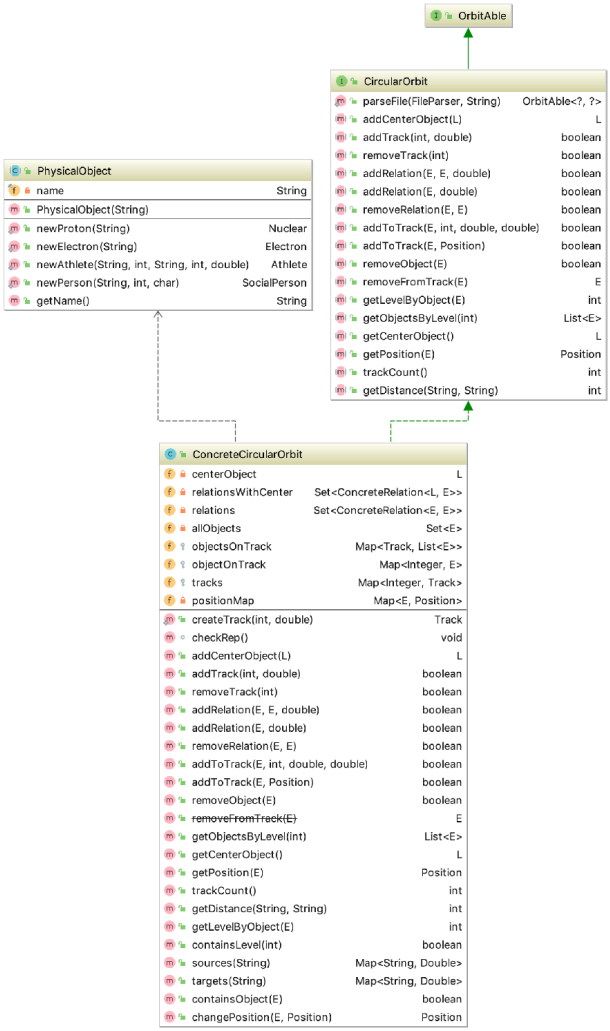
AtomStructure 是最基本的轨道模型：有中心物体和多个轨道、多个轨道物体。而 TrackGame 与之不同的是没有中心物体，SocialNetworkCircle 在原子模型的基础上增加了关系。

3.2 基于语法的图数据输入

3.3 面向复用的设计：`CircularOrbit<L,E>`

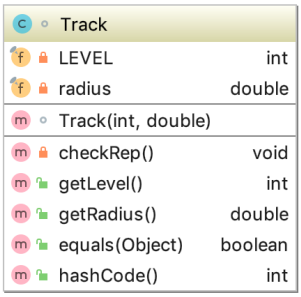
这个类设计为可变类，是轨道模型的基本实现方法，其他具体应用全都与此类有关联。L 代表中心物体，E 代表轨道物体。

类图如下：



3.4 面向复用的设计：Track

这个类是轨道类，设计为不可变类。类图如下：

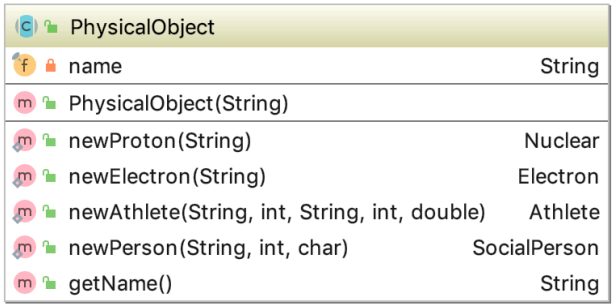


3.5 面向复用的设计：L

L 被设计为 CenterObject，被多个子类继承表示具体的中心物体，另外设置 EmptyObject 为跑道模型服务。

3.6 面向复用的设计：PhysicalObject

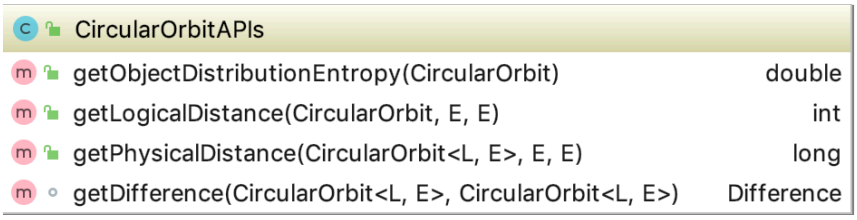
PhysicalObject 设计为一个抽象类，所有轨道物体都要继承它。此外提供多个工厂方法。



现有 Electron, Nuclear, Athlete, Person 类是其子类。

3.7 可复用 API 设计

设计 CircularOrbitAPIs，类图如下：



3.8 图的可视化：第三方 API 的复用

可视化使用 AWT 工具包，

3.9 设计模式应用

所有 Track 和 PhysicalObject 对象都是用工厂设计模式创建的；

每个文件解析类的对象使用了抽象工厂；

使用 strategy 设计模式实现多种比赛编排策略；

3.10 应用设计与开发

运行 Application 中的 Application.java，刚进入程序时要求用户输入三个数字来决定启动的具体程序。

3.10.1 TrackGame

首先令用户选择文件进行读取来初始化运动员和比赛信息，如果选取的文件不合法，用户会收到提示（其他应用也拥有相同功能，不再赘述）

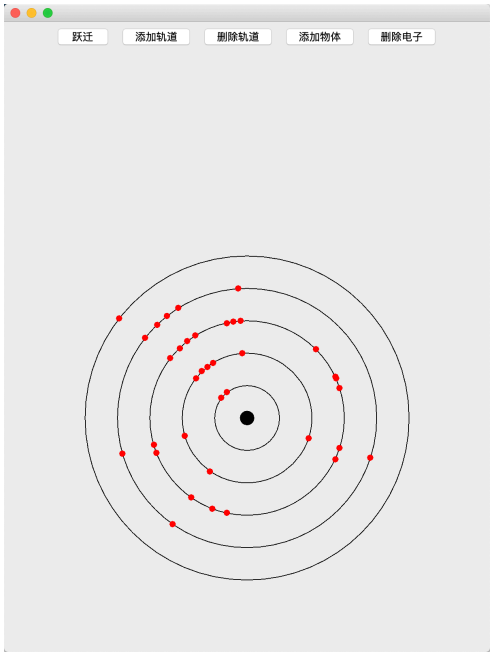
读取正确的文件之后，提供如下基础操作：

“随机安排”指随机分配所有运动员的组、跑道等信息；“强者后出场”指将成绩好的运动员更晚出场，且同组之内，成绩好的运动员更占据中间赛道。
“将安排写入文件”按钮将比赛安排写入文件，如果没有安排过比赛，将不允许执行此操作。
“交换两个运动员的组”也需要在安排过比赛后才允许执行，点击后会要求输入两个运动员的姓名，然后他们两个的比赛安排会被交换，此后如果在写入文件，会创建“Customized Arrangement.xlsx”文件。
剩下的四个操作如字面义，如果执行了这四个操作，则在写入文件时，文件名也是 Customized Arrangement.xlsx。

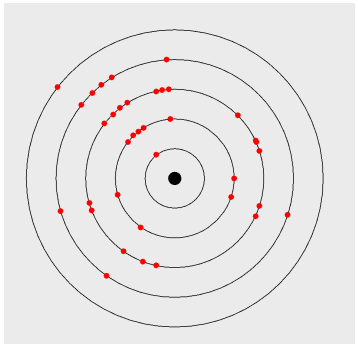
3.10.2 StellarSystem

3.10.3 AtomStructure

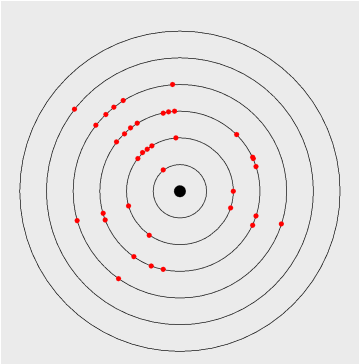
此部分提供如下功能：



“跃迁”：要求用户输入三个数据：从哪个轨道跃迁，跃迁到哪个轨道，跃迁几个电子，确定后绘制新的原子结构。例如将上图的轨道 1 的 1 个电子跃迁到轨道 2，操作完后轨道如图：



“添加轨道”：用户输入添加第几条轨道后绘制新的原子模型，例如添加 6 号轨道：

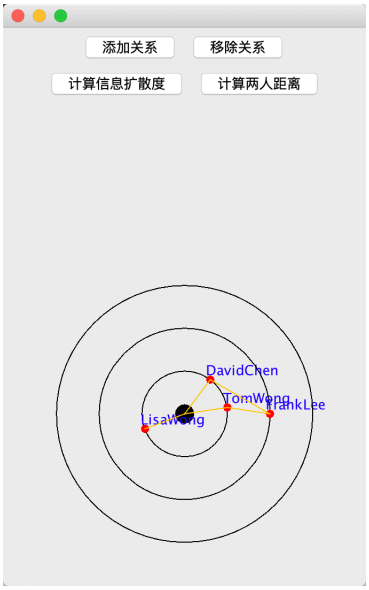


“删除轨道”：用户输入轨道序号，删除该轨道和其上所有电子。
“添加电子”和“删除电子”需要用户提供轨道号，然后程序将添加或删除电子。

3.10.4 PersonalAppEcosystem

3.10.5 SocialNetworkCircle

此应用提供如下功能：



因为这个应用的特殊性，所以不能添加单个人到轨道模型中，必须以关系的形式添加。所以不提供直接添加物体的功能
“添加关系”：用户输入两个人名，特殊的如果两个人名都没出现在图中，则提示添加失败，因为这个关系对于此模型来说是一个不合法的关系，无法在图中显示。
“移除关系”：输入两个人名，移除他们之间的关系。
“计算信息扩散度”：输入人名，计算有多少人和他有直接或者间接的联系。
“计算两个人的距离”：输入两个人名，计算他们之间的逻辑距离。

3.11 应对应用面临的新变化

3.11.1 TrackGame

新建一个类为 Team，TrackGame 的“运动员”是 Team 而不是 Athlete。Team 类设计为一个列表，只能包含少于等于 4 个运动员

3.11.2 StellarSystem

3.11.3 AtomStructure

创建新的类 Kernel，其组成为一个列表，可以储存中子和质子，AtomStructure 继承的 ConcreteCircularOrbit<Nuclear, Electron> 更改为 ConcreteCircularOrbit<Kernel, Electron>，另外封装获得质子和中子的方法。

```
public class AtomStructure extends ConcreteCircularOrbit<Kernel, Electron> {

    public static AtomStructure empty() { return new AtomStructure(); }

    public boolean removeElectronFromTrack(int level) {
        if (tracks.get(level) == null) return false;        // track does not exist

        List<Electron> electrons = objectsOnTrack.get(tracks.get(level));

        removeFromTrack(electrons.get(0));
```

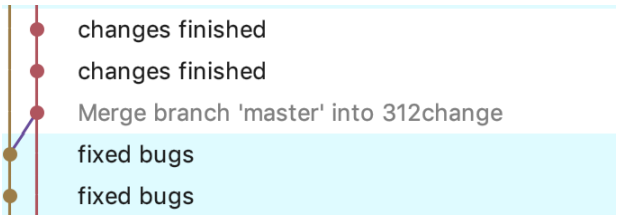
```
List<PhysicalObject> kernel = new ArrayList<>();

public void addNeutron(Neutron neutron) { kernel.add(neutron); }

public void addProton(Proton proton) { kernel.add(proton); }

public List<Neutron> getAllNeutrons() {
    List<Neutron> neutrons = new ArrayList<>();
    for (PhysicalObject physicalObject : kernel) {
        if (physicalObject.getClass() == Neutron.class) {
            neutrons.add((Neutron) physicalObject);
        }
    }
    return Collections.unmodifiableList(neutrons);
}

public List<Proton> getAllProtons() {
    List<Proton> protons = new ArrayList<>();
    for (PhysicalObject physicalObject : kernel) {
        if (physicalObject.getClass() == Proton.class) {
            protons.add((Proton) physicalObject);
        }
    }
    return Collections.unmodifiableList(protons);
}
```

3.11.4 PersonalAppEcosystem

3.11.5 SocialNetworkCircle

本来为了实现双向关系，在添加关系方法中会添加两次，以此来实现双向关系，现在去掉一个即可实现单向关系。

3.12 Git 仓库结构



4 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
4.1	13:45-15:30	初步设计并建立实验的框架	完成
4.8	14:00-17:20		
未记录			
4.20	14:40-15:50	设计 Social Network Circle 的 ADT	完成
4.21	8:45-17:05	实现 Social Network Circle	未完成
	19:10-19:40		
4.22	14:00-15:20	实现 Social Network Circle + 修复 Track Game 的 Bug	完成
4.23	10:00-11:30	设计主程序框架+写 Track Game 测试	Track Game 未完成
4.24	17:40-20:00	写完剩余测试	未完成
4.25	16:00-16:30	修复已发现的问题	完成
4.29	14:30-15:30	写完剩余测试	完成
	18:30-20:30		
5.1	10:00-12:50	修复 Social Network Circle 的错误，改进计算速度	完成
5.1	15:30-16:20	编写主程序	未完成
5.2	11:50-13:00		
	15:00-22:00		
5.3	9:50-17:40		
5.4	一天	编写主程序，修复已知问题	未完成
5.5	8:00-14:00	完成编写主程序	未完成
5.5	15:30-16:30	完成主程序	基本完成
5.6	8:00-10:00	完成主程序，实验报告	主程序完成
	13:45-15:30	修复错误，实验报告	未完成

	15:45-17:30	修复错误，实验报告	未完成
	18:30-22:00	修复错误，实验报告	完成

5 实验过程中遇到的困难与解决途径

遇到的难点	解决途径
人际关系网络的添加和移除人和其他的有点不同，有的关系是不能显示在图中的	改为输入关系添加
移除轨道、移除物体的操作较为麻烦，需要考虑移除关系	重写了多次相关方法，重新整理逻辑，改善 ADT 后解决

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验和教训

因为吸取了上个实验的教训，所以这次开始大面积写代码前，我仔细地对 ADT 进行了思考，每个类应该拥有哪些字段，应该提供哪些方法，所以这次实验的前期比较顺利。但是随着功能的不断增多，逐渐发现最初设想的那些方法根本不够用，所以又要再次修改或者添加方法，有的方法实现起来效率极低，又要再次修改成员变量，因此在已知任务的情况下应该考虑多一步，当然，要预留够修改的空间，这样可以避免修改时改动大量代码；

第二，要灵活应用各种设计模式，例如工厂设计模式使用起来有对外屏蔽细节的有点，能够保护类本身，并且提高客户端易用性。

6.2 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？

面向 ADT 编程难度相对较低，因为只需要完成 ADT 要求的功能即可，但是面向场景编程更为抽象，ADT 也需要自己设计，设计不好时未来还有大量改动，所以我认为 ADT 服务于场景，场景又反过来验证 ADT 的合理性，所以不断互相补充，成为一个更加复杂的过程。

(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？

这些东西能够时刻提醒自己应该遵守的规则，不要盲目编程，同时还能保证类的安全性，以后我也坚持。

(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？

能，为了封装出良好的 API，需要考虑比较多的问题，最基础要求能有正确的结果，其次还要有一定的容错能力，保证自己的内部不被破坏。

(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？

设计模式某些时候会使目前的代码量增加，但是通常能够减少未来的代码量，而且更容易维护，活用各种设计模式能够提高开发效率。

(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？

语法是一种规范，语法驱动编程能够锻炼开发者的素养——遵守规矩的同时利用规矩方便开发的进行。

(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？

最难在于设计 ADT，最开始总是难以预料后来要发生的变化，因此最初设计的 ADT 在用着用着就会发现存在“不够用”的情况，所以又要反过来增加一些方法，这时候就要仔细考量新增方法是否具有实用价值，我添加这个方法是否能很好的被复用。

(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？

参考了 Lab4 的一小部分要求，例如用户选择不合要求的文件进行解析的时候，当前的程序已经能够提示用户文件非法，要求更换文件。

(8) 关于本实验的工作量、难度、deadline。

本次实验工作量太大。

难度适中，如果光谈 deadline 完全能说时间足够，但是考虑到实际情况，即课程不是只有软件构造一门，再结合这个代码量，只能说工作量真的很大，如果说这门课程的学分是 5，其他课程是 2，那么应该投入 2.5 倍的时间，但实际情况并非如此，投入的时间远超 5 倍，所以我觉得要结合实际情况综合考虑实验工作量和 deadline 的要求。

(9) 到目前为止你对《软件构造》课程的评价。

课程很好，实验有点累人。