

**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 | 罗瑞欣 |
| 学号 | 1170300821 |
| 班号 | 1703008 |
| 电子邮件 | [3102595709@qq.com](mailto:3102595709@qq.com) |
| 手机号码 | 15048260039 |

**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

Windos10专业版，eclipse，IDEA，jdk-9.01，Junit5.

在这里给出你的GitHub Lab3仓库的URL地址（Lab3-学号）。

git@github.com:ComputerScienceHIT/Lab3-1170300821.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
* SocialNetworkCircle

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。

## 基于语法的图数据输入

## 面向复用的设计：CircularOrbit<L,E>

|  |  |
| --- | --- |
| L | 中心物体 |
| E | 轨道上的物体 |

|  |  |
| --- | --- |
| 方法 | 用途 |
| public void addCentralObject(L object); | 添加中心物体（考虑后续功能扩展，放入set中） |
| public void addTrack(Track t); | 添加轨道到set中 |
| public void removeTrack(Track t); | 从set中去除轨道 |
| public void addObjectToTrack(E object, Track t); | 添加轨道到物体的映射关系 |
| public void addRealationship(E object1, E object2); | 添加物体到物体的映射关系 |
| public void readFiles(String filename); | 从文件中读取系统基本信息 |

ConcreteCircularOrbit<L, E> implements CircularOrbit<L, E> ：

|  |  |
| --- | --- |
| L | 中心物体 |
| E | 轨道上的物体 |
| public Set<Track> trackSet = new HashSet<>(); | 轨道的集合 |
| public Set<E> objectSet = new HashSet<>(); | 轨道上物体的集合 |
| public Set<L> centralObjectSet = new HashSet<>(); | 中心物体的集合 |
| public Map<Track, Set<E>> relationMap = new HashMap<>(); | 轨道和物体的对应关系 |
| public Map<E, E> objectRelation = new HashMap<>(); | 轨道上物体间的关系 |

|  |  |
| --- | --- |
| 方法 | 用途 |
| public void addCentralObject(L object); | 添加中心物体（考虑后续功能扩展，放入set中） |
| public void addTrack(Track t); | 添加轨道到set中 |
| public void removeTrack(Track t); | 从set中去除轨道 |
| public void addObjectToTrack(E object, Track t); | 添加轨道到物体的映射关系 |
| public void addRealationship(E object1, E object2); | 添加物体到物体的映射关系 |
| public void readFiles(String filename); | 从文件中读取系统基本信息 |

## 面向复用的设计：Track

Track抽象类：

|  |  |
| --- | --- |
| protected int trackRadius; | Track的轨道半径 |

|  |  |
| --- | --- |
| 方法 | 作用 |
| Getter | Getter |
| public boolean equal(Track track) | 半径相等时认定两个track相等 |

TrackGame的GameTrack：

|  |  |
| --- | --- |
| protected int trackRadius; | Super |
| private int Group; | 标记轨道所在的分组  （用于后续改进） |

|  |  |
| --- | --- |
| 方法 | 作用 |
| Getter | Getter |

AtomStructure的AtomStructureTrack：

|  |  |
| --- | --- |
| private int trackNumber; | 电子轨道层数 |
| private String atomName; | 电子对应原子名 |

|  |  |
| --- | --- |
| 方法 | 作用 |
| Getter | Getter |

SocialNetworkCircle的SocialNetworkCircleTrack：

|  |  |
| --- | --- |
| protected int trackRadius; | Super |

## 面向复用的设计：L

L复用作为中心物体

|  |  |
| --- | --- |
| Orbit类 | 中心物体类 |
| ConcreteCircularOrbit<L, E> | CentralObject |
| TrackGame | 无 |
| AtomStructure | AtomCentralObject |
| SocialNetworkCircle | SocialNetworkCentralObject |

CentralObject：

|  |  |
| --- | --- |
| protected String name | 中心物体名 |
| Getter | Getter |

AtomCentralObject：

|  |  |
| --- | --- |
| protected String name | 原子名（扩展后可作为质子、中子） |

SocialNetworkCentralObject：

|  |  |
| --- | --- |
| private int age; | 中心用户的年龄 |
| private String gender; | 性别 |
| Getter | Getter |

## 面向复用的设计：PhysicalObject

|  |  |
| --- | --- |
| Orbi类 | 轨道物体类 |
| ConcreteCircularOrbit<L, E> | PhysicalObject |
| TrackGame | Athlete |
| AtomStructure | Electron |
| SocialNetworkCircle | Friend |

PhysicalObject：

|  |  |
| --- | --- |
| protected String objectName = new String(); | 轨道物体名 |
| protected Position position = null; | 轨道物体的位置 |
| Getter | Getter |
| hashCode | hashCode |

Athlete

|  |  |
| --- | --- |
| private int number; | 号码 |
| private String country = new String(); | 国籍 |
| private int age; | 年龄 |
| private double bestgrade; | 本年度最好成绩 |
| Getter | Getter |
| hashCode | hashCode |
| equals | 判断 |

Electron

|  |  |
| --- | --- |
| private String atomName | 对应原子名 |
| Getter | Getter |
| hashCode | hashCode |

Friend

|  |  |
| --- | --- |
| private int age; | 姓名 |
| private String gender; | 年龄 |
| Getter | Getter |

## 可复用API设计

CircularOrbitAPIs<L, E>：

|  |  |
| --- | --- |
| L | 中心物体 |
| E | 轨道上的物体 |
| public double getObjectDistributionEntropy(CircularOrbit c) | 计算多轨道系统中各轨道上物体分布的熵值 |
| public int getLogicalDistance(CircularOrbit c, E e1, E e2) | 计算任意两个物体之间的最短逻辑距离 |
| public double getPhysicalDistance(CircularOrbit c, E e1, E e2) | 计算任意两个物体之间的物理距离 |
| public Difference getDifference(CircularOrbit c1, CircularOrbit c2) | 计算两个多轨道系统之间的差异 |

TrackGameAPI extends CircularOrbitAPIs ：

|  |  |
| --- | --- |
| public int getLogicalDistance(CircularOrbit c, E e1, E e2) | 计算任意两个物体之间的最短逻辑距离 |
| public Difference getDifference(CircularOrbit c1, CircularOrbit c2) | 计算两个多轨道系统之间的差异 |
| public void randomSort(TrackGameOrbit trackC) | 随机安排运动员分组 |
| public void S2B\_Sort(TrackGameOrbit trackC) | 根据成绩分组 |
| public boolean moveInGroup(TrackGameOrbit C1, Athlete a1, Athlete a2) | 在组内调整赛道安排 |
| public boolean moveBetweenGroup(TrackGameOrbit C1, Athlete a1, Athlete a2) | 不同分组间调整赛道安排 |
| private Set<String> writeDifference(Map<GameTrack, Set<Athlete>> mapC, GameTrack track) | 写difference |

|  |  |
| --- | --- |
| public double getObjectDistributionEntropy(CircularOrbit c) | 计算多轨道系统中各轨道上物体分布的熵值 |
| public void removeElectronic(AtomStructureOrbit C,int i) | 去除某个轨道上的电子 |
| public Difference getDifference(CircularOrbit c1, CircularOrbit c2) | 计算两个多轨道系统之间的差异 |
| public void electronictransition(AtomStructureOrbit C1, int t1, int t2) | 一个电子跃迁t1→t2 |

AtomStructureAPI extends CircularOrbitAPIs

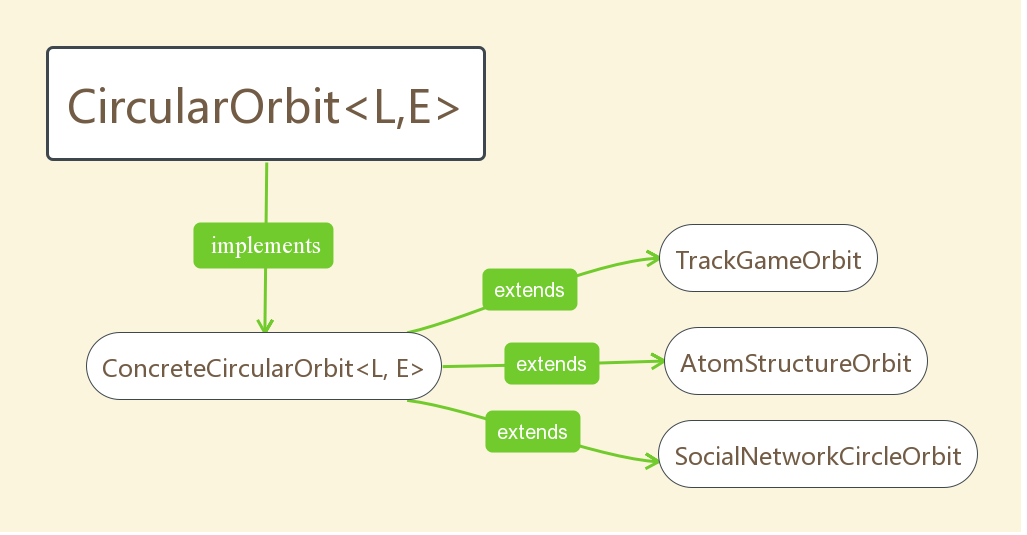
|  |  |
| --- | --- |
| L |  |
| E |  |

|  |  |
| --- | --- |
| L |  |
| E |  |

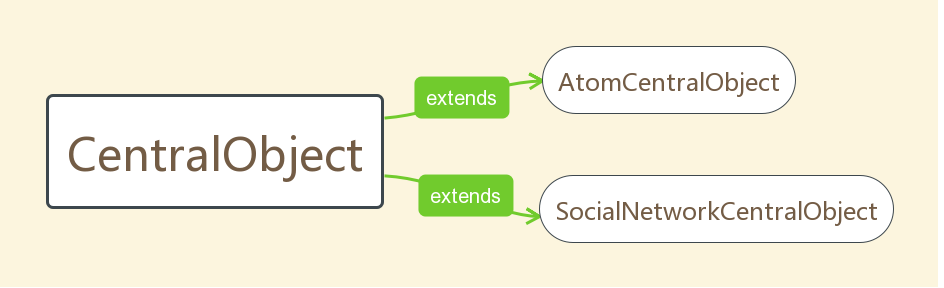
## 图的可视化：第三方API的复用

## 设计模式应用

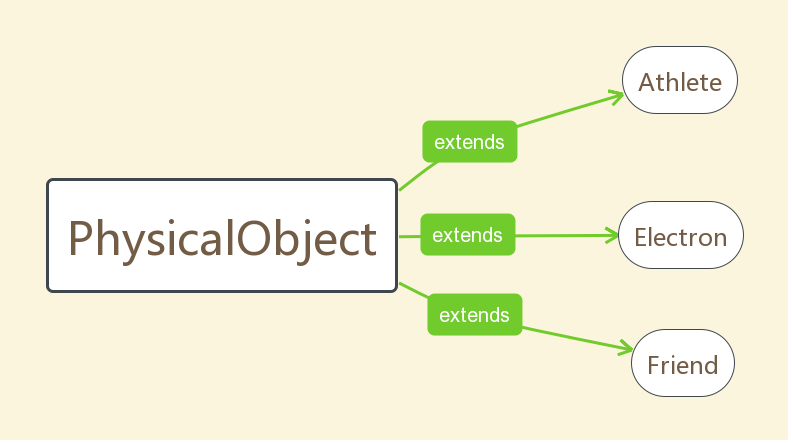
CircularOrbit<L,E>



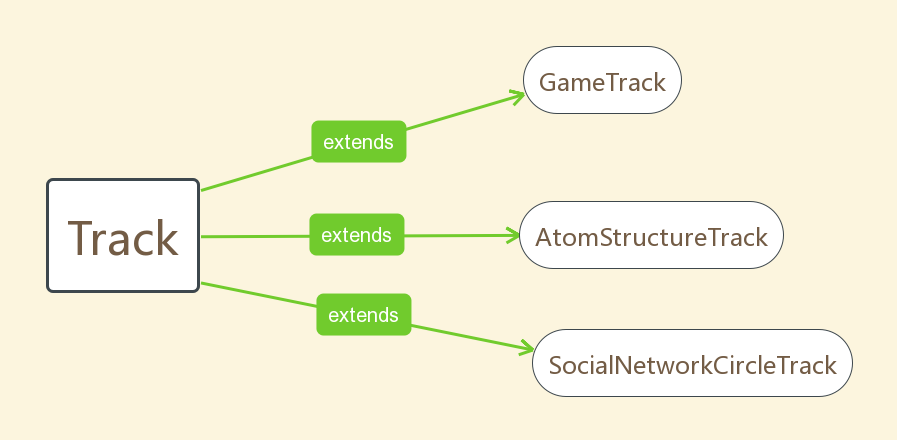
CentralObject



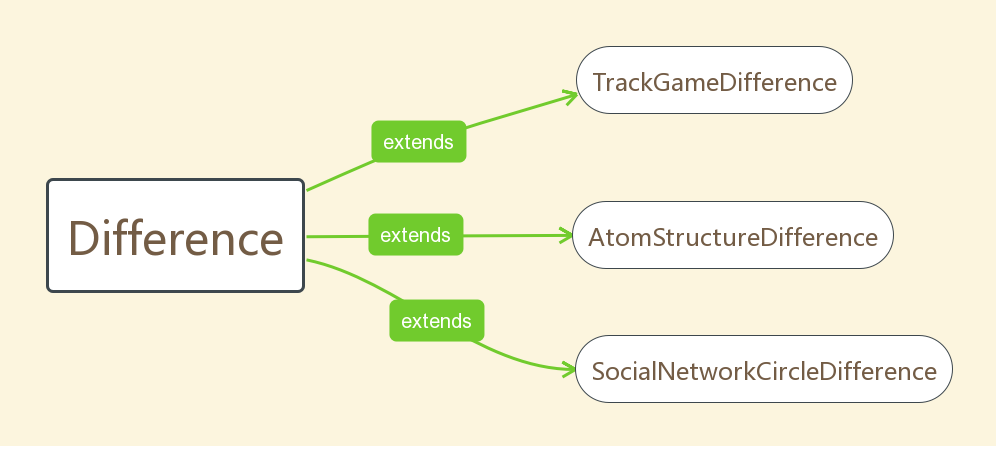
PhysicalObject



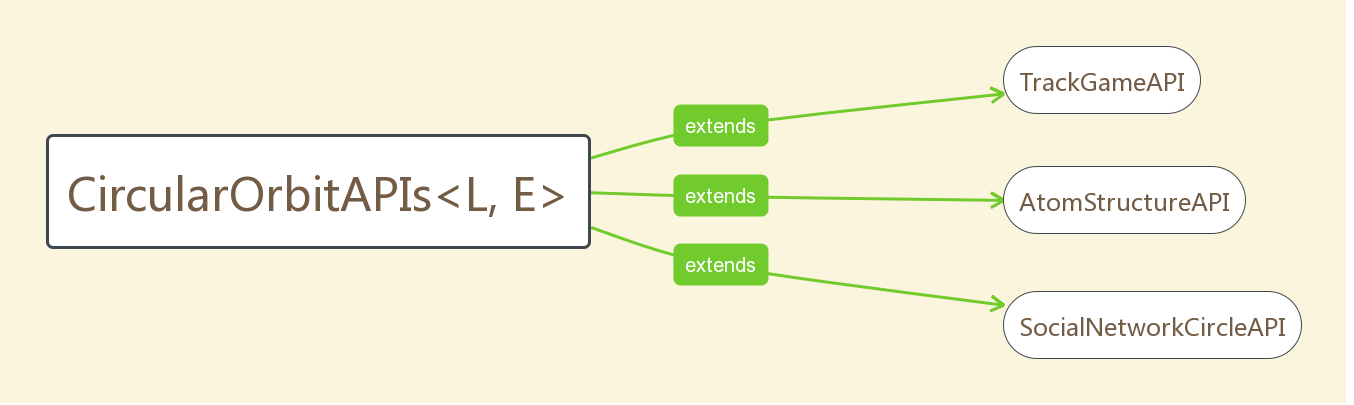
Track



Difference



APIs



请分小节介绍每种设计模式在你的ADT和应用设计中的具体应用。

## 应用设计与开发

### TrackGame

1. 初始化和清空
2. 计算距离
3. 计算差异
4. 调整分组
5. 增删运动员

### AtomStructure

1. 初始化和清空
2. 计算距离
3. 计算差异
4. 调整分组
5. 增删电子

### SocialNetworkCircle

1. 初始化和清空
2. 计算距离
3. 计算差异
4. 调整分组
5. 增删朋友

## 应对应用面临的新变化

### TrackGame

存储分组情况的数据结构改为Set<Map<GameTrack, Set<Athlete>>> ，每个分组的map中GameTrack指向一个Set<Athlete>，即可实现接力赛四人一个跑道的情况。

### AtomStructure

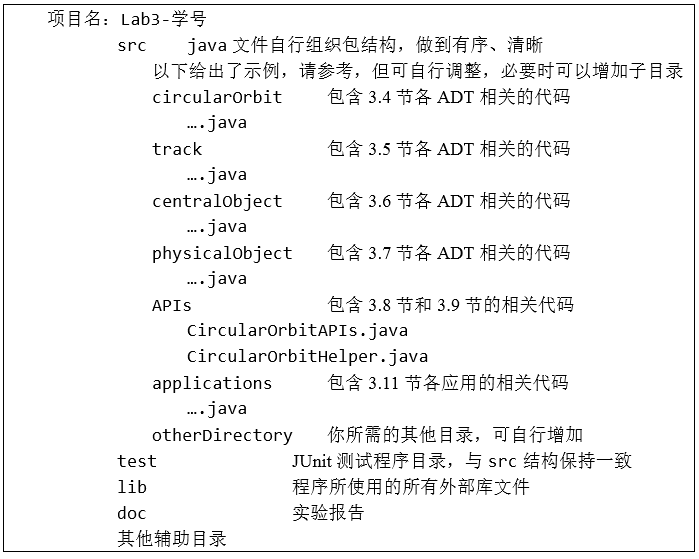
将轨道中心物体存储在Set<AtomCentralObject>中，可以对AtomCentralObject命名为“Proton”“neutron”来区分质子中子。

### SocialNetworkCircle

设置Map<SocialNetworkCircleTrack, Set<Friend>>存储社交层次到friend的映射，Map<Friend,Set<Friend>> 存储每个人（包括中央用户）的直接朋友关系。

## 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 |
| 4.1 |  | 学习正则表达式 | 海星 |
| 4.4 |  | 构建基本框架 | 完成 |
| 4.7 |  | Centralobject | 完成 |
| 4.9 |  | Track | 完成 |
| 4.11 |  | Object | 完成 |
| 4.15-4.20 |  | Orbit | 部分完成 |
| 4.30 |  | Orbit | 完成 |
| 5.1-5.3 |  | API | 完成 |
| 5.3-5.5 |  | APP | 完成 |
| 5.5-5.6 |  | GUI | 部分完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 正则表达式的match和正则语法 | Google |
| GUI的Swing设计 | Google和舍友研讨 |
|  |  |

# 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？

答：根据共有的类型设计ADT，简化问题复杂度；简化编程，更有目的性，分解难度。

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？

答：检查一致性，确保类内部数据的相对正确；吼啊

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？

答：很难正确的分解任务，随着项目的进展会出现更多问题

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？

答：使项目进程更规范，不易出错

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？

答：对正则语法和match有了刻骨铭心的认识

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？

答：需求会不断变化，很难不更改之前的ADT；不断试错

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？

答：没看，但是预留了很大程度的扩展性

1. 关于本实验的工作量、难度、deadline。

答：大、难、早

1. 到目前为止你对《软件构造》课程的评价。

答：吼啊