**《共享单车管理系统的开发》**



**第八组**

**成员：曾胜坚 唐光宇 谷银珂**

# 1.问题分析

## 1.1背景

随着城市交通的发展和人们对出行方式的需求，共享单车成为了一种受欢迎的出行方式。共享单车企业需要一个有效的管理系统来管理车辆、用户和订单等信息，以提高运营效率和用户体验。

## 1.2需求分析

（1）管理员能够通过系统管理车辆信息，包括添加、修改、查询、删除车辆信息。

（2）管理员能够通过系统管理用户信息，包括添加、修改、查询、删除用户信息。

（3）管理员能够通过系统管理订单信息，包括查询订单、取消订单、生成订单等功能。

（4）管理员能够对车辆进行调度和维修，系统能够记录车辆的调度和维修历史。

（5）管理员能够生成报表和统计数据，以便分析运营情况和做出决策。

# 2.可行性研究

在进行项目的实际开发前，要进行可行性研究，目的是以最小的代价在短时间内确定软件项目是否值得开发，是否可以实现。对共享单车管理系统可以从经济可行性、技术技术可行性和社会可行性三个方面来论证。

## 2.1经济可行性

经济可行性的目的是估算开发成本，确定项目值得投资。共享单车管理系统的开发成本包括软件开发单车的工资、服务器和硬件设备的购买成本等。同时，系统的运营和维护成本也需要考虑。通过对市场需求和预期收入的分析，可以评估系统的经济可行性。

（1）采用传统的手工输入方法，不仅工作繁琐而且需要人力输入多，记录容出错，那样产生了成本高，效果差的弊病，对于一个企业的资料库建立是十分困难和麻烦的。而采用这种系统所发费的成本，精力都远远低于手工，并且此系统上手容易，操作简便。对个人而言，节省了大量时间与精力：对公司而言减少了单车雇员资金及纸质化管理的相应成本；对开发者而言系统维护和随用户及技术的更新易操作。

（2）软件开发周期不应拉的过长，也不能过短，周期过长容易导致软件与预想差距过大，进度拖拉，周期过短导致功能不全面。作为一个实验项目，开发周期应在八到九周比较合理，这样能够有足够的实际来完成规划，构思，实现。

（3）项目中所使用的开发工具软件（Java SE, IDEA Community, MySQL Community）等均为开源软件,降低了项目的经济压力，让开发单车能够专门投入于开发过程中。

综上所述，共享单车信息管理系统在经济上是可行的。

## 2.2技术可行性

技术可行性要对项目的功能和限制条件进行分析，目的是确定项目是否能实现。一般要包括开发风险和技术风险。

在开发风险方面，由于使用的大部分软件都是开源的，且软，硬件要求都不高，所以开发的费用较低，本系统使用的技术都比较成熟，故开发风险较低。

公司单车信息管理系统不受限于系统，可跨平台运行，系统使用的技术大都比较成熟，出错的概率较低，性能出色。在开发技术方面选择Java SE作为实现语言，包括JDBC（java数据库编程），java web在内的所用技术都比较成熟，可用用于本项目，同时使用结构化方法进行开发，在技术层面是完全可行的。

## 2.3社会可行性

现如今，每天有成千上万的公司建立，这些公司都需要一套行之有效的单车信息管理系统来帮助其完成公司的日常管理，那么设计一套科学，有效的信息管理系统就有了现实的意义。该系统投入运行后可以有效减少公司日常运营的压力和开销，将单车的信息管理电子化，统一化，对于关联的公司其他事务会显著提高效率，并且系统使用简单，直观，不会给公司带来其他压力。

综上所述，开发单车信息管理系统具备社会可行性。

# 3. 面向对象的分析

根据问题分析的业务描述，得到系统用例如图3-1所示。每个用例规约如下面各表所示。



**图3-1 系统用例图**

## 3.1 建立用例模型

**表3-1 “管理基本信息”用例规约**

|  |  |
| --- | --- |
| 用例名称： | 管理基本信息 |
| 用例ID： | P1 |
| 参与者： | 管理员 |
| 用例说明： | 对共享单车和单车所属维修人员的基本信息进行管理 |
| 前置条件： | 无 |
| 主事件流： | 1. 添加共享单车信息 2. 输入单车的基本信息 3. 选择添加按钮  * 若该单车信息存在，执行A1 * 若该单车信息不存在，执行（3）  1. 单车信息添加至数据库 2. 修改单车信息 3. 从下拉列表中选中要修改的单车 4. 输入修改的信息，更新至数据库 5. 删除单车信息 6. 从列表中选中要删除的单车信息 7. 点击删除按钮删除信息，更新至数据库中 8. 查询单车信息 9. 根据编号找到目标单车信息 10. 得到查询结果 |
| 备选事件流： | 无 |
| 异常事件流： | A1：提示“单车信息存在” |
| 后置条件： | 更新后的单车信息显示在列表中 |

**表3-2 “维修人员管理”规约**

|  |  |
| --- | --- |
| 用例名称： | 维修人员管理 |
| 用例ID： | P2 |
| 参与者： | 管理员 |
| 用例说明： | 管理维修人员信息 |
| 前置条件： | 无 |
| 主事件流： | 1. 修改维修人员信息   （1）选中要修改的维修人员，点击修改信息  （2）提交修改选项，更新信息到数据库  2. 新建维修人员   * 1. 输入维修人员的基本信息   2. 选择添加按钮 * 若该维修人员名或维修人员工号存在，执行A1 * 若该维修人员信息不存在，执行（3）   1. 维修人员信息添加至数据库  1. 删除维修人员    1. 在下拉列表选中要删除的维修人员    2. 点击删除，更新信息至数据库中 |
| 备选事件流： | 无 |
| 异常事件流： | A1：提示“该维修人员已经存在” |
| 后置条件： | 维修人员信息被正确更新 |

**表3-3 “管理单车维修”用例规约**

|  |  |
| --- | --- |
| 用例名称： | 管理单车维修 |
| 用例ID： | P3 |
| 参与者： | 管理员 |
| 用例说明： | 管理员管理单车的维修 |
| 前置条件： | 无 |
| 主事件流： | 1. 单车维修 2. 选择目标单车 3. 选择维修的维修人员，提交 4. 查询维修历史   （1）在列表中查找想要查询的维修历史 |
| 备选事件流： | 无 |
| 异常事件流： | 无 |
| 后置条件： | 维修后的单车信息被正常显示，更新 |

## 3.2 建立类模型

从上文的用例建模中可以得出本系统的实体类如下单车实体类。



**图3-1 单车管理系统类图**

## 3.3创建顺序图

有了用例图和用例规约，可以对系统有一个初步的了解，为了更加全面地掌握系统的处理流程，下一步可以选用顺序图描述系统的处理顺序。顺序图是将交互关系表示为一个二维图。纵向是时间轴，时间沿竖线向下延伸。横向轴代表了在协作中各独立对象的类元角色。类元角色用生命线表示。当对象存在时，角色用一条虚线表示，当对象的过程处于激活状态时，生命线是一个双道线。本文对单车信息管理和反馈处理建立顺序图，其它的用例所创建的顺序图效果是类似的。



**图3-4单车信息管理顺序图**

# 4数据库设计

数据库设计就是根据业务系统的具体需求，结合我们所选用的DBMS（数据库管理系统），为这个业务系统构造出最优的数据存储模型。并建立数据库中的表结构以及表与表之间的关联关系的过程。使之能有效的对应用系统中的数据进行存储，并可以高效的对已存储的数据进行访问。本系统采用MySQL 8.03作为数据库实现，数据库名：单车信息管理系统；标志：hrdatabase。

## 4.1类模型向关系模型的转换

对上一节进行分分析，得到单车，维修人员，用户，计费，维修记录五个实体类。将各个实体类分别映射成数据库中的关系模式。

根据系统需求，可以设计以下数据库表：

**实体：**

用户，

单车，

计费，

维修人员表，

维修记录表。

**属性：**

用户：U\_ID,用户姓名,用户年龄,用户电话，用户余额，使用类型，是否可用。

单车：B\_ID,状态,维修人员姓名，

计费：COST\_ID,U\_ID,B\_ID,使用时长，使用距离。

维修人员表：R\_ID,维修人员姓名,维修人员年龄,维修人员电话,是否空闲。

维修记录表：RR\_ID, B\_ID,R\_ID,报修时间,维修详情,维修结果。

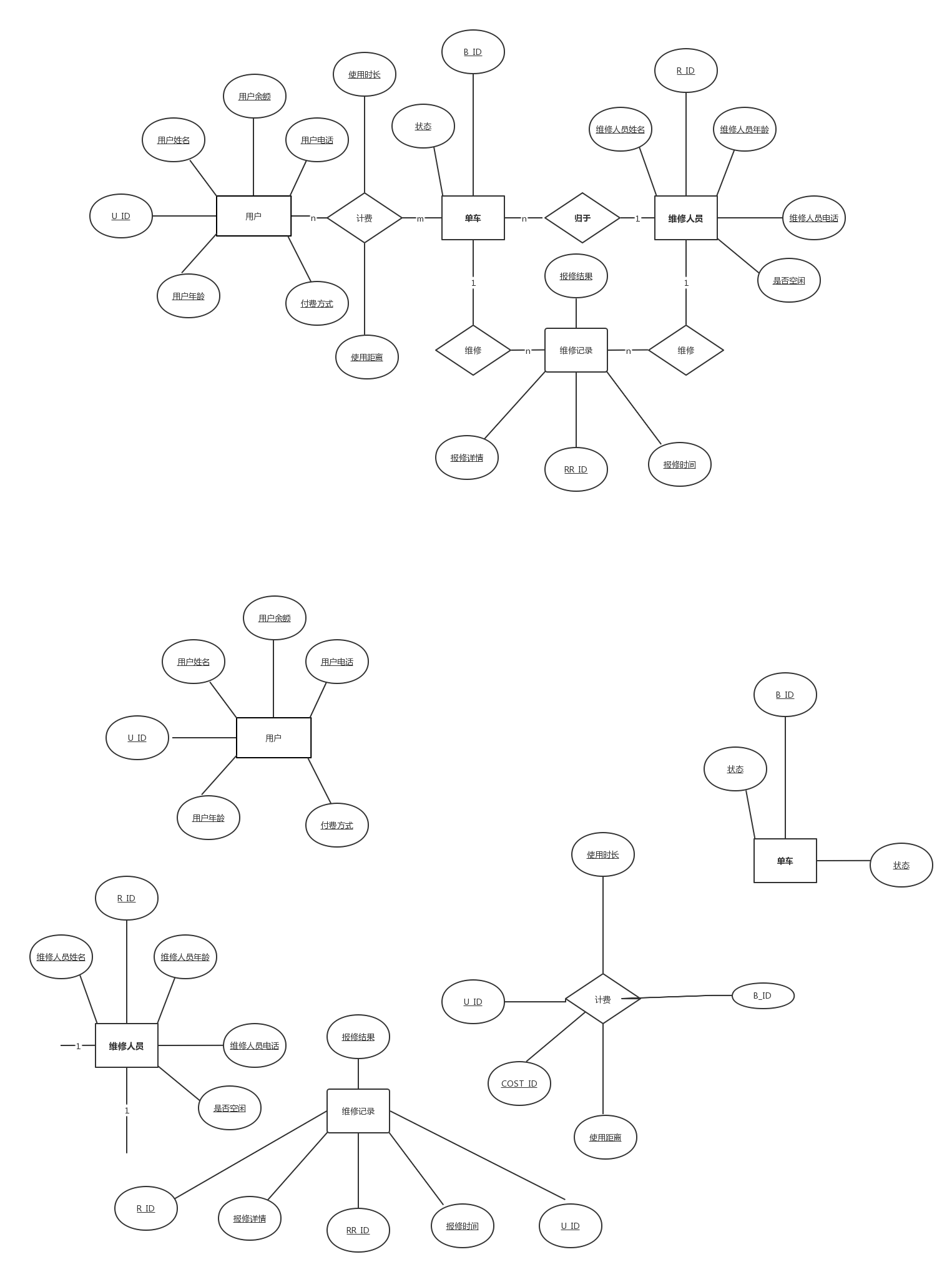
用户（U\_ID,用户姓名,用户年龄,用户电话，用户余额，使用类型），

单车（B\_ID,状态），

计费（使用时长，使用距离），

维修人员表（R\_ID,维修人员姓名,维修人员年龄,维修人员电话,是否空闲），

维修记录表（RR\_ID,报修时间,维修详情,维修结果）。



**图4-1数据库中建立的表**

# 5. 面向对象的设计

## 5.1 设计软件类

### 1. 实体类

实体类主要是作为数据管理和业务逻辑处理层面上存在的类别； 它们主要在分析阶段区分 实体类的主要职责是存储和管理系统内部的信息，它也可以有行为，甚至很复杂的行为，但这些行为必须与它所代表的实体对象密切相关。在OOA阶段得到的类图中的类都是实体（关联）类，包括车辆实体、用户类、订单管理类、统计分析类。每个实体类在实现的时候要对应一个数据对象。

### 2. 边界类

边界类是一种用于对系统外部环境与其内部运作之间的交互进行建模的类。用于描述外部参与者与系统之间的交互，对系统中依赖于环境的那些部分进行建模。在本系统中每个边界类应对应一个实例，每个边界类设计一个专门的界面来实现。

### 3. 控制类

控制类用于对一个或几个用例所特有的控制行为进行建模。控制对象（控制类的实例）通常控制其他对象，因此它们的行为具有协调性质。控制类将用例的特有行为进行封装。

本系统的控制类建立在边界类上，图形化界面组件既是边界类又成为了控制类，也就是说边界类另一方面起到了控制类的作用，换一种说法，边界类和控制类被集成在了一个类中，这种集合的类在本系统中有多个，下面从控制类的角度列出。

（1） 添加车辆信息控制类。

（2） 修改车辆信息控制类。

（3） 删除车辆信息控制类。

（4） 查询车辆信息控制类。

（5） 用户管理控制类

（6） 订单管理控制类

（7） 统计分析控制类

## 5.2 软件的体系架构

本软件体系架构简单，主要遵循两个设计原则：单一职责原则，开放-封闭原则。在上文提到的四个类的基础上增加了一个数据库类，每个类的职责单一，每个类有且仅有一个使其发生变化的原因，防止因职责的变化而削弱或抑制这个类完成其他职责的能力，同时对修改类是封闭的，对扩展类是开放的。

下面列出了各个业务的模块

1. AssessChangeHistory

用于处理考核管理及历史记录

1. Database

用于连接数据库及实现相关的数据库操作

1. RepairBean

部门类，实现部门的相关操作类

1. RepairHistory

部门调动管理类，主要是记录调动历史

1. PersonBean

人员信息管理类

1. BikeBean

单车信息管理类

1. StatisticBean

统计分析类，用于实现对车辆使用情况的统计和分析

# 6. 面向对象的编程

面向对象的设计已经将整个应用程序使用的类、方法和体系描述得比较清楚了，在详细设计阶段已经考虑了程序最终的实现方式。与生命周期开发方法的实现阶段不同，OO方法在系统实现的过程中，不需要再对应用程序的编码方案进行设计，主要考虑将设计方案对应到Java代码的过程以及页面的实现。

本系统主要使用Javax.swing和JDBC实现，javax.swing用于实现前端图形化界面，JDBC用于完成数据库操作。

## 6.1建立公用文件

本小节列出了本系统中主要用到的公用类，简要解释了其作用，具体的成员方法可以在上一节面向对象的设计中的软件体系架构一节查看，这里不再列出。

1. AssessChangeHistory

用于处理考核管理及历史记录

1. Database

用于连接数据库及实现相关的数据库操作

1. RepairBean

部门类，实现部门的相关操作类

1. RepairHistory

部门调动管理类，主要是记录调动历史

1. PersonBean

人员信息管理类

1. BikeBean

单车信息管理类

1. StatisticBean

统计分析类，用于实现对车辆使用情况的统计和分析

## 6.2创建主界面

### 1. BikeMain

BikeMain类完成了主界面的创建工作，配合接下来的各个模块完成了所有界面的创建。

# 7. 软件测试与维护

## 7.1 软件测试

### 1. 测试范围

本次对该软件的测试范围包括基本信息模块，车辆信息模块，用户管理模块，订单管理模块。

### 2. 测试的种类

考虑到本系统各个模块之间耦合性不高，相互之间较为独立，例如添加车辆信息和删除车辆信息之间的关系仅在数据库方面有交叉，故本文在集成测试都时候选用深度优先的结合方法。

本文采用的测试种类有单元测试，集成测试（深度优先），白盒测试，黑盒测试。

### 3. 测试的数据

数据1：

数据2：

数据3：

### 4. 软件测试过程

（1）集成测试，黑盒测试（第一组数据）

首先使用数据一完成加入车辆信息，调整用户信息，生成订单，删除车辆的操作，观察是否出现意料之外的结果。

接下来点击要被删除的车辆，选择后再点击确定按钮

（2）集成测试小结

本次集成测试的结果显示本系统在当前数据下的表现正常，没有发现能够显著引起错误的地方，系统在当前数据下，添加车辆模块，调整用户信息模块，生成订单模块都能够正确运行并输出正确的数据。

（3）白盒测试，单元测试（数据2，动态分析，基本路径测试，统计分析模块）

这一小结采用动态分析的白盒测试使用数据2对统计分析模块进行测试。

根据系统实现，当数据2被正确插入后将会显示在车辆信息查询栏中，此时可以选择对其进行统计分析，分析结果将会显示在统计分析界面中。

（4）单元，白盒测试的结果显示本系统在当前数据下的表现正常，没有发现能够显著引起错误的地方，系统在当前数据下，统计分析模块能够正确运行并输出正确的数据。

### 5. 软件测试总结

测试总结的是问题，是风险，是经验，是如何改进，是分析测试过程中的问题，并针对项目进行分析从而找出解决方案。也可以说用于分析并总结这些缺陷，将总结出来的经验用于指导下一次的测试用例设计。在每个版本测试完毕后，要进行测试总结，做一些比例的总结、缺陷严重级别及比例的总结，单车工作效率的总结，还有最重要的是风险的平复，对下一测试版本的建议等。 本系统在软件编写代码时已经经过了多次实验，导致在软件测试时几乎找不出什么错误。在本章的软件测试中已经对几个主要的模块进行了测试，没有找出什么错误，表明在现阶段中的软件应用没有太大的问题需要去修改。

## 7.2 软件维护

软件维护活动类型总起来大概有四种：纠错性维护（校正性维护）、适应性维护、完善性维护或增强、预防性维护或再工程。除此四类维护活动外，还有一些其它类型的维护活动，如：支援性维护（如用户的培训等）。

软件维护的成本一般远超软件前期开发的成本，考虑到本系统的大小，后期软件的维护主要在以下方面

1、维护开发文档，使其他人能够根据文档理解项目代码，在修改代码后及时修改文档

2、进行回归测试，修复潜在的BUG

3、留出为未来扩展项目的空间，保证未来软件的可扩展性

# 8. 思考与反思

## 8.1 面向对象的分析

在边界类和控制类方面，没有对其进行很好的设计，导致边界类和控制类耦合度较高，被封装在了同一个类中，最终导致本项目中的图形化界面类的耦合度很高，集成了多种类的功能，这是本项目一个有待解决的问题。

## 8.2 面向对象的实现

在图形化界面的设计中，没有将各个类抽象出共同的基类，导致每个界面都重写了一个类，最后写了一共有11个界面类，最后导致类过多，这是值得反思的，导致这个的原因有以下几个：

（1）Java程序设计的能力不强，对Java中面向对象设计的掌握不足

（2）没有很好地理解面向对象的思想，没能将各个界面的共同特点抽象出来