**EBU5304 Software Engineering**

**Group Coursework**

**Group x**

**Introduction**

The aim of this software engineering process is to develop a system for handling ordering of comic books, browsing of the catalogue of the store, and user subscriptions with rechargeable cards, by applying UML(unified Modeling Language).During the course of developing this system, the whole developing group is divided into 3 subgroups. Please see relative part for more details.

To develop this system, the development team began with a detailed requirements capture to identify the real needs of this system by analyzing the received specifications and investigating relative people. At the end of this stage, the use-case model, architecture description, user interface prototype, and glossary are obtained.

In the third stage, the team members started to transform the analysis model into a design model that serves as a blueprint for software construction. The following artifacts were obtained at the end of design: design model, architecture description, use-case realization-design, design classes, design subsystems, and interfaces.

In the fourth stage, the focus was shifted to implement the system in terms of components. The result of this stage is: implementation model, architecture description, integration build plan, components, implementation subsystems, and interfaces.

In the final stage, the team carried out tests to verify the results from the implementation stage by testing each software build. The result artifacts of this stage are: test model, test cases, test procedure, test evaluation, test plan, and test components.

Beside all the artifacts mentioned above, Java code is also generated as the required product for customer.

This report starts with introducing the structure and responsibility of the developing team. Then it reviews all stages of software engineering and shows

All the artifacts the team obtained at each stage. The final part of this report is summary and appendices.

**Requirement Analysis**

1. 捕获需求技术（Techniques for Finding Requirements）

为了精确定义和理解漫画店管理系统，我们试着去应用一些技术去捕获需求，如背景阅读、采访以及问卷。

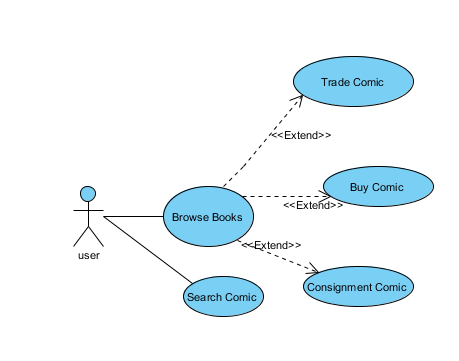
1. 对于背景捕获，我们小组查阅了一些书店管理系统以及图书馆管理系统，对比所要开发的系统，提取相似的功能与需求。
2. 用户采访：用户访谈是直接有效，形式灵活，是最基本的需求捕获工具。我们在线采访了书店老板，得出了以下的需求：
   1. 用户需要这个系统必须是独立的，即为了保护商店的账户安全以及用户的信息，该系统没有联网功能。
   2. 希望该系统软件简单容易上手，并且希望该软件的性价比比较高，即功能齐全价格实惠。
3. 问卷调查：对于问卷调查，我们主要统计一些数据的获取，可以帮助我们识别出系统的一些非功能性需求。比如：能最大接受输入密码错误的次数、系统内存的大小、书籍查询时间、账户更新时间等。
4. 描述需求细节（The detail description of the requirements）
   1. 功能需求（Functional requirements）
      1. 系统管理（System management）
         1. 管理账号
5. 该系统可以让直接接触该系统的人（经理、店员）进行注册，登陆。并且对他们的账号和密码进行管理（存在文件中）。
   * + 1. 管理订阅卡账号信息
6. 该系统管理者可以新增办理订阅卡用户，并将信息存入文件。
7. 该系统管理者可以查看该订阅卡中订阅者的信息以及订阅记录。
   * + 1. 管理漫画书信息
8. 管理者可以新增漫画。
9. 管理者可以键入漫画的打分，修改漫画信息。
10. 管理者可以浏览漫画的状态，如出租或已售。
11. 管理者可以查询漫画的库存量。
    * + 1. 订单管理
12. 用户和订阅者都可以购买漫画书，他们的数据会被保存在相关订单中。
13. 管理者可以查询相关的订单信息。
    * 1. 业务管理（Business management）
         1. 出售漫画
14. 书店职员处理订阅者和用户的购买请求。
15. 当购买行为发生时，系统自动更新商店账户。
    * + 1. 订阅漫画
16. 书店职员处理订阅者的租订漫画请求。
17. 当租订行为发生时，系统自动更新商店账户。
18. 该系统自动更新相对应的订阅卡信息。
    * + 1. 寄售漫画
19. 漫画书店处理顾客的寄售漫画请求，收取一定手续费，将顾客的漫画进行寄售。
    * + 1. 交换漫画
20. 漫画书店提供一个交换漫画的平台，允许打分相同的漫画在用户之间进行交换行为的发生。
    * + 1. 漫画进货
21. 在用户购买漫画时，如果某本书没有现货，店员则会向经销商下订单，为了订阅书籍，用户将付全款并计入商店账户，发票将和购物清单一起发给顾客。如果经销商有书籍现货，则全款将从商店账户中扣除支付给经销商，如果经销商没有现货，则全款则会从商店账户中扣除并返还给顾客，商店还会通知顾客此书无货。
    * 1. 商店账户管理
         1. 清除负值
22. 店主拥有权力清空订阅者卡里的负值。在这种情况下。卡里亏欠的金钱数额，将会从商店账户资金里转入亏欠者的卡中。
    * + 1. 订阅卡充值
23. 书店职员可以处理用户充值订阅卡需求。
24. 商店账户自动更新
    1. 非功能性需求（Non-functional requirements）
       1. The system must be developed as a standalone Java application with a simple Java GUI and should use J2SE 1.5 or above.
       2. The system should not use database.
       3. The system can run on command line
       4. Response time should be no more than 2 seconds
       5. Register shall be able to use the system after reading the readme file. This will cost no more than one hour. And after reading the file, the average number of errors by experienced user shall not exceed two per day
       6. The whole system should occupy no more than10M memory
       7. Only clerk and owner can manage the accounts.
25. 定义用例（Defining the Use Cases）
    1. 找到对象以及用例（Find Actors and Use Case for Them）

* User

书店顾客主要是购买漫画，并且他也能够办理订阅卡，成为订阅者。

* + User 的用例：

Browse Books, Search Comic, Consignment Comic, Buy Comic, Trade Comic.

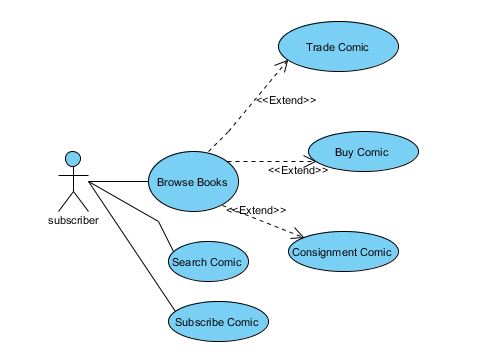


* Subscriber

订阅者可以用订阅卡订阅漫画，同样也能执行购买、寄售、交易漫画的功能

* + Subscriber的用例：

Browse Books, Search Comic, Consignment Comic, Buy Comic, Trade Comic, Subscribe Comic

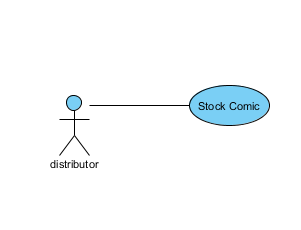


* Distributor

经销商负责给店里短缺的书进行供货

* + Subscriber的用例：

Stock Comic

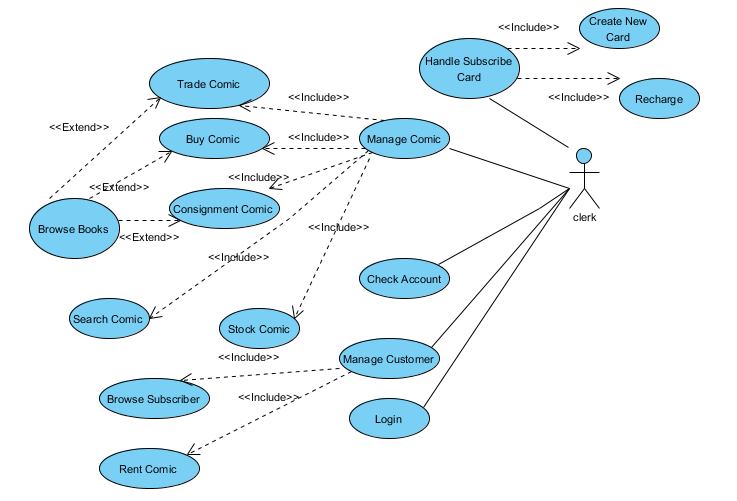


* Clerk

代表书店职员，是整个系统的直接使用者和维护者。

* + Clerk 的用例：

Handle Subscribe Card, Manage Comic, Check Account, Manage Customer, Login

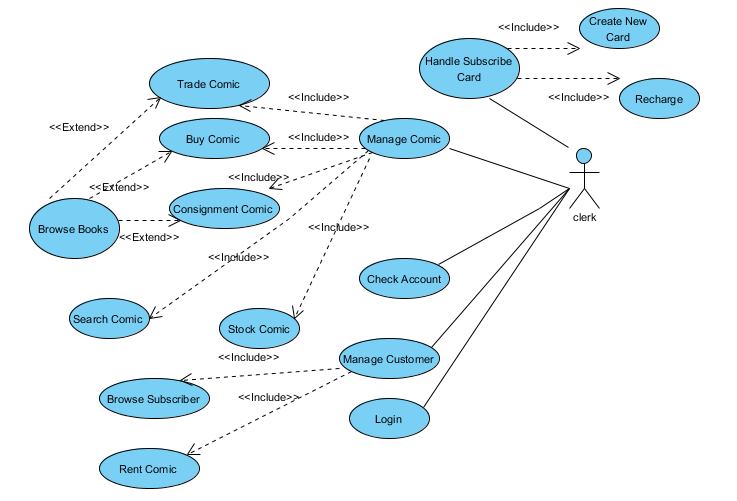


* Owner

书店所有者，也是整个系统的直接使用者，权力大于书店职员。

* + Owner的用例

Handle Subscribe Card, Manage Comic, Check Account, Manage Customer, Login, Remove Negative.



1. 描述用例(Detail Use Case Description)

|  |  |
| --- | --- |
| **用例名称** | Browse Books |
| **用例标号** | 1 |
| **参与者** | Subscriber、User、Clerk、Owner |
| **简要说明** | 订阅者和顾客可以浏览商店中的漫画书籍以及评分。书店职员可以查看商店漫画书籍的库存。 |
| **前置条件** | 书店职员登陆 |
| **事件基本流** | 1. 订阅者、顾客浏览书店中漫画书籍、评分标准、以及书籍的借售信息。 2. 书店职员可以查看商店漫画书籍的库存。 3. 用例终止 |
| **其他事件流** | 无 |
| **异常事件流** | 无 |
| **后置条件** | 无 |
| **注释** |  |

|  |  |
| --- | --- |
| **用例名称** | Search Comic |
| **用例标号** | 2 |
| **参与者** | Subscriber、User、Clerk、Owner |
| **简要说明** | 当用户向店员询问某一本特定书籍时，店员需要可以立即在系统中寻找到这本书籍并且知道是否可借阅和可售（是否有现货）。 |
| **前置条件** | 店员必须登录。 |
| **事件基本流** | 1. 顾客将要搜索的漫画名称告诉店员 2. 店员在系统中输入该书名 3. 显示该漫画的详细信息 4. 终止用例 |
| **其他事件流** | 在点击搜索之前，店员可以点击取消。 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 显示该漫画的详细信息。 |
| **注释** | 无 |

|  |  |
| --- | --- |
| **用例名称** | Rent Comic |
| **用例标号** | 3 |
| **参与者** | Subscriber、Clerk、Owner |
| **简要说明** | 订阅者可以允许用自己的卡租用漫画书，每张卡有其独特的条形码。 当客户订阅的时候，系统会收取15英镑的订阅费（以现金形式）。这些订阅费会归入订阅卡中，作为初始存款余额。  当用户进行租用操作时，订阅卡里的存款会进行更新。 这意味着，对每天来说（除了第一个免费日），R数量的金钱会从卡里扣除。当客户归还书时，扣费总量会被计算（即R\*天数）。如果卡里余额为负，订阅者将不能租用或购买任何书。  当用户取消订阅时，用户卡上的剩余的信贷则会从商店账户上扣除并返还给用户。 |
| **前置条件** | 店员必须登录，且用户必须持有订阅卡 |
| **事件基本流** | 1. 订阅者拿着订阅卡和要订阅的漫画到前台告诉书店职员自己的订阅请求。 2. 书店职员输入订阅卡条形码。 3. 书店职员收取15英镑订阅费。 4. 系统判断订阅卡的余额是否为负，为负终止用例。 5. 书店职员输入租用漫画ID或漫画名称，系统更新漫画状态。 6. 系统自动更新每天的租用费用。 7. 用户取消订阅，归还漫画。 8. 系统更新漫画状态。 9. 系统将订阅卡中剩余余额从商店账户中扣除，返还给用户。 10. 用例终止。 |
| **其他事件流** | 在点击订阅之前，店员可以点击取消。 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 商店账户、订阅卡余额以及漫画状态信息会更新。 |
| **注释** | 无 |

|  |  |
| --- | --- |
| **用例名称** | Trade Comic |
| **用例标号** | 1.1 |
| **参与者** | Subscriber、User、Clerk、Owner |
| **简要说明** | 用户可以和其他用户交换相似评分的漫画书。 |
| **前置条件** | 店员必须登录并且书库中有该用户漫画的评分标准。 |
| **事件基本流** | 1. 用户拿着自己的漫画到前台。 2. 书店职员输入该用户漫画的ID或名称。告知用户该漫画的打分。并给予该漫画一个打分标签，若书店内无该漫画的打分则终止用例。 3. 用户可以拿着具有打分标签的漫画去和书店中其他用户交换相似评分的漫画书。 4. 用例终止。 |
| **其他事件流** | 在点击搜索之前，店员可以点击取消。 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

|  |  |
| --- | --- |
| **用例名称** | Buy Comic |
| **用例标号** | 1.2 |
| **参与者** | Subscriber、User、Clerk、Owner |
| **简要说明** | 用户和订阅者都可以购买漫画书，他们的数据会被保存在相关订单中。 |
| **前置条件** | 店员登陆 |
| **事件基本流** | 1. 用户要购买的漫画到前台。 2. 书店职员输入该用户漫画的ID或名称。得到该漫画的价格信息以及库存信息，如果库存为0则转到进货用例。 3. 系统生成相关购买订单。 4. 用户支付现金。 5. 系统更新漫画库存，商店账户。 6. 系统打印发票和购物清单给用户。 7. 用例终止。 |
| **其他事件流** | 在点击支付之前，店员可以点击取消。 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

|  |  |
| --- | --- |
| **用例名称** | Consignment Comic |
| **用例标号** | 1.3 |
| **参与者** | Subscriber、User、Clerk、Owner |
| **简要说明** | 用户可以将漫画书给商店，商店可以代卖。一旦卖掉，商店收取售价的一定比例作为服务费，把剩下的钱给予用户。 |
| **前置条件** | 店员登陆 |
| **事件基本流** | 1. 用户要出售的漫画到前台。 2. 书店职员记录该用户编号、漫画名称、销售价格等信息到系统。 3. 系统生成相关寄售信息。 4. 系统将寄售单给用户。 5. 用户支付服务费用。 6. 系统更新商店账户。 7. 当用户漫画卖出后，将除服务费外的钱返回给用户。 8. 用例终止。 |
| **其他事件流** | 在点击寄售之前，店员可以点击取消。 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

|  |  |
| --- | --- |
| **用例名称** | Stock Comic |
| **用例标号** | 4 |
| **参与者** | Distributor、Clerk、Owner |
| **简要说明** | 在用户购买漫画时，如果某本书没有现货，店员则会向经销商下订单，为了订阅书籍，用户将付全款并计入商店账户，发票将和购物清单一起发给顾客。如果经销商有书籍现货，则全款将从商店账户中扣除支付给经销商，如果经销商没有现货，则全款则会从商店账户中扣除并返还给顾客，商店还会通知顾客此书无货。 |
| **前置条件** | 店员登陆 |
| **事件基本流** | 1. 用户要购买的漫画到前台。 2. 书店职员输入该漫画的ID或名称查询该漫画的库存信息。 3. 库存显示数量为0，则点击进货按钮，向经销商购买。 4. 用户支付该漫画的全额定金。 5. 系统更新商店账户。 6. 系统打印购物清单和发票给顾客。 7. 当经销商反馈有该存货时，系统扣除定金打入经销商账户中。并进货该漫画给用户。 8. 如果经销商无存货，则系统扣除定金返还给用户。 9. 用例终止。 |
| **其他事件流** | 在点击进货之前，店员可以点击取消。 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

|  |  |
| --- | --- |
| **用例名称** | Browse Subscriber |
| **用例标号** | 5 |
| **参与者** | Clerk、Owner |
| **简要说明** | 管理订阅者的概况。用户的基本信息可以被职员获知。 |
| **前置条件** | 店员登陆 |
| **事件基本流** | 1. 职员点击查看订阅者信息按钮。 2. 所有订阅卡持有者信息可以被职员浏览：订阅卡卡号、借阅记录、订阅卡余额、订阅卡持有人姓名。 3. 用例终止。 |
| **其他事件流** | 无 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

|  |  |
| --- | --- |
| **用例名称** | Check Account |
| **用例标号** | 6 |
| **参与者** | Clerk、Owner |
| **简要说明** | 书店职员可以查看商店的账户 |
| **前置条件** | 店员登陆 |
| **事件基本流** | 1. 职员点击查看商店的账户按钮。 2. 商店账户信息显示。 3. 用例终止。 |
| **其他事件流** | 无 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

|  |  |
| --- | --- |
| **用例名称** | Login |
| **用例标号** | 7 |
| **参与者** | Clerk、Owner |
| **简要说明** | 书店职员和持有者登陆系统 |
| **前置条件** | 店员和经理已经注册。 |
| **事件基本流** | 1. 职员和经理输入自己的账号密码登陆 2. 用例终止。 |
| **其他事件流** | 无 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 店员或经理成登陆状态 |
| **注释** | 无 |

|  |  |
| --- | --- |
|  | Register |
| **用例标号** | 8 |
| **参与者** | Clerk、Owner |
| **简要说明** | 书店职员和持有者注册 |
| **前置条件** | 系统开启 |
| **事件基本流** | 1. 注册者输入自己的用户名 2. 注册者输入密码。 3. 注册者再次确认密码。 4. 注册者成功注册。 5. 系统更新系统使用者库。 6. 用例终止。 |
| **其他事件流** | 无 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

|  |  |
| --- | --- |
|  | Create New Card |
| **用例标号** | 8 |
| **参与者** | Clerk、Owner、User |
| **简要说明** | 书店职员可以根据用户需求，给用户办理订阅卡 |
| **前置条件** | 书店职员登陆 |
| **事件基本流** | 1. 用户于前台提出办理订阅卡请求。 2. 书店职员点击按钮创建一张新的订阅卡 3. 系统自动生成订阅卡编号。 4. 用户填写订阅卡详细信息，如用户名等。 5. 系统更新订阅卡用户信息。 6. 用例终止。 |
| **其他事件流** | 无 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 新增订阅卡 |
| **注释** | 无 |

|  |  |
| --- | --- |
|  | Recharge |
| **用例标号** | 9 |
| **参与者** | Clerk、Owner、Subscriber |
| **简要说明** | 书店职员可以处理用户充值需求。 |
| **前置条件** | 书店职员登陆 |
| **事件基本流** | 1. 用户提出订阅卡充值请求。 2. 职员点击充值按钮，输入要充值的订阅卡号。 3. 用户通过支付现金方式充值。 4. 职员输入充值金额。 5. 系统更新商店账户和订阅卡账户。 6. 用例终止。 |
| **其他事件流** | 无 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

|  |  |
| --- | --- |
|  | Remove Negative |
| **用例标号** | 10 |
| **参与者** | Owner |
| **简要说明** | 店主拥有权力清空订阅者卡里的负值。在这种情况下。卡里亏欠的金钱数额，将会从商店账户资金里转入亏欠者的卡中。 |
| **前置条件** | 经理登陆 |
| **事件基本流** | 1. 店主点击情况订阅者卡里负值 2. 店主输入订阅卡编号。 3. 系统更新商店账户，扣除亏欠金额。 4. 该订阅卡更新余额。 5. 用例终止。 |
| **其他事件流** | 无 |
| **异常事件流** | 提示输入错误信息。 |
| **后置条件** | 无 |
| **注释** | 无 |

1. 整体用例以术语表（Describe the use case as a whole and produce the glossary）
   1. 整体用例图



* 1. 术语（Glossary）：

**Subscriber Card:**这里指的是在本书店租定漫画时，办理的租定卡，是租定漫画的凭证。

**Grade:**这里的打分是指根据不同的漫画“价值”，书店评定出一个等级，作为计算租金和交换漫画的依据。

**Java GUI:** Graphic User Interface of Java

**J2SE:** Java2 Standard Edition

**Response time**: the time that the system will need to process the user‟s instruction.

**System**

**Analysis**

1. 架构分析（Architecture analysis）
   1. Identifying analysis packages
      1. Analysis packages are identified by allocating the main portion of a number of use cases, which are required to support a specific actor of the system, to a specific package and then realize the corresponding functionality within the package. Therefore, we get the following packages:
2. Store(supports user, Subscriber actor, handling their request)
3. Administrator management(support Clerk and Owner actor ,who are the system direct user)
   * 1. Handling commonality among analysis packages
4. System user account management (required by Clerk’s management, Owner’s management).
5. Store account management.
   * 1. Identifying service packages

Service packages are identified for each optional service, or each service that could be made optional, even though every customer always wants it. The criteria could be described as: Identify one service package for each service provided by functionally related classes. According to the above criteria, the following service packages are identified:

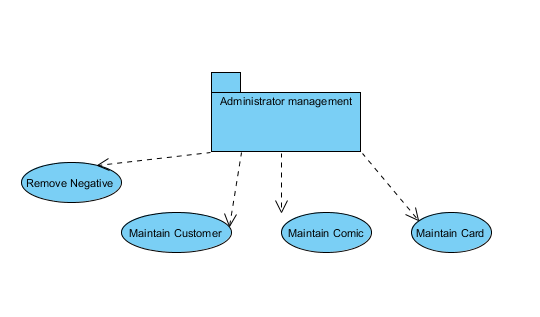
1. In Store package

Buy Comic, Rent Comic, Trade Comic, Consignment Comic, Browse Comic, Search Comic, Create New Card, Recharge.



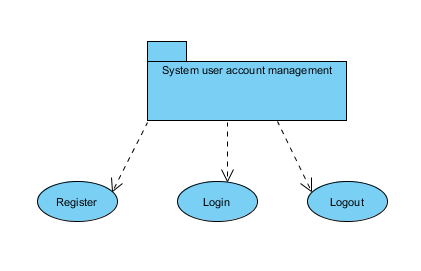
1. Administrator management

Maintain Customer, Maintain Comic, Maintain Card , Remove Negative



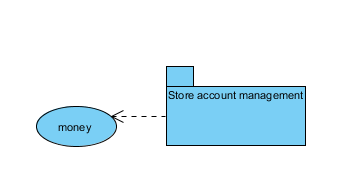
1. In System user account management package

Login, Register, Logout



1. In Store account management package

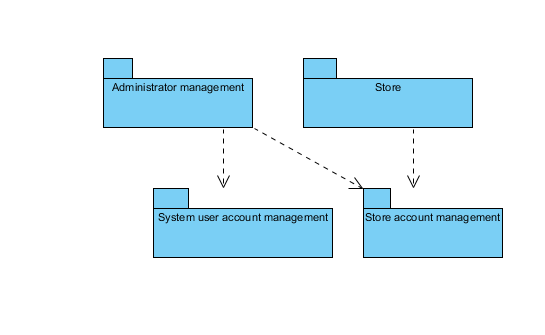
money



* 1. Identifying common special requirements

The common special requirements are listed as follow:

1. The common special requirements are listed as follow:
2. The time period an object typically needs to be kept persistent is no more than 1 second.
3. All the entities are updated only when modification occurs.
4. This system works without network.
5. This system works without database, only supports .txt file.
6. This system should be able to tolerate most user input faults, and perform effective validity check.
7. This system should not cause a crash or exceptional exit when wrong operation is performed.
8. All the users (administrator, chair, and reviewer) are identified by their respective user IDs and passwords.
9. The system should have friendly and coherent interfaces.
   1. Identifying Analysis Package Dependency



* 1. Domain Model

By analyzing the use case, we get the conceptual class and their relationship.

The basic steps of concept model:

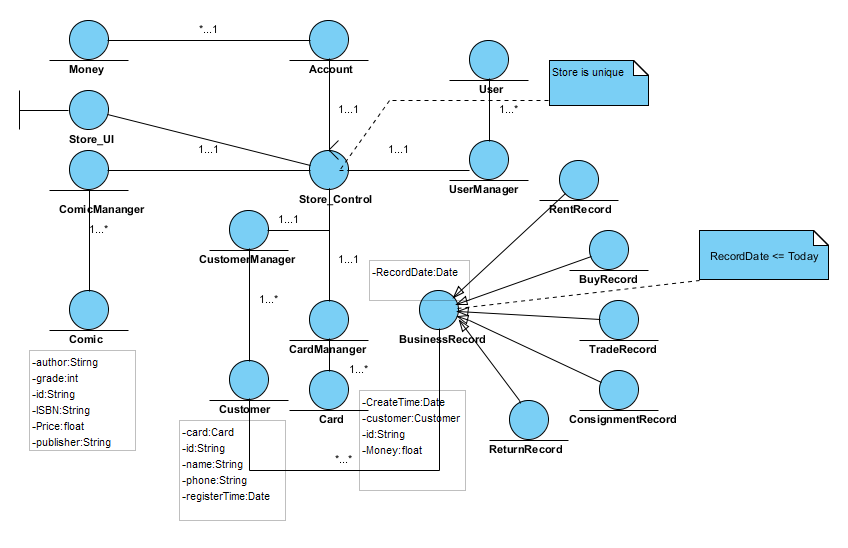
* + 1. Finding conceptual class
* Using the concept of class classification list to find out the concept of class

|  |
| --- |
| **Concept Class Name** |
| Comic |
| Card |
| Customer |
| Money |
| User |

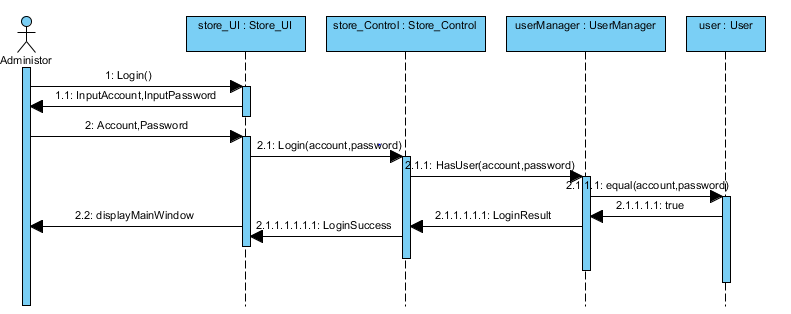
* According to concept of noun phrases to identify classes

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| BusinessRecord | An abstract class, which is used to record transactions |
| CardManager | Maintain Card function is supported by CardManager |
| CustomerManager | Maintain Customer function is supported by CustomerManager |
| StoreManager |  |
| UserManager | Login, Logout, Register function is supported by UserManager. |
| ComicManager | The Maintain Comic function is supported by ComicManager |
| StoreAccount | Record the store account. |
| ConsignmentRecord | Record the consignment transactions. |
| RentRecord | Record the Rent transactions |
| ReturnRecord | Record the Return transactions |
| TradeRecord | Record the Trade transactions |

* + 1. The relationship between them.

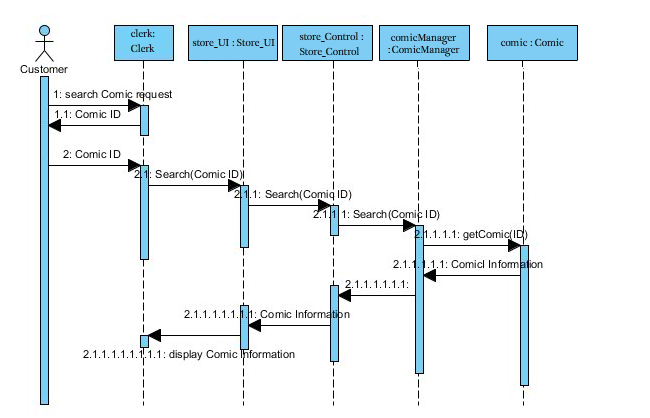


* 1. System sequence diagram
* Login



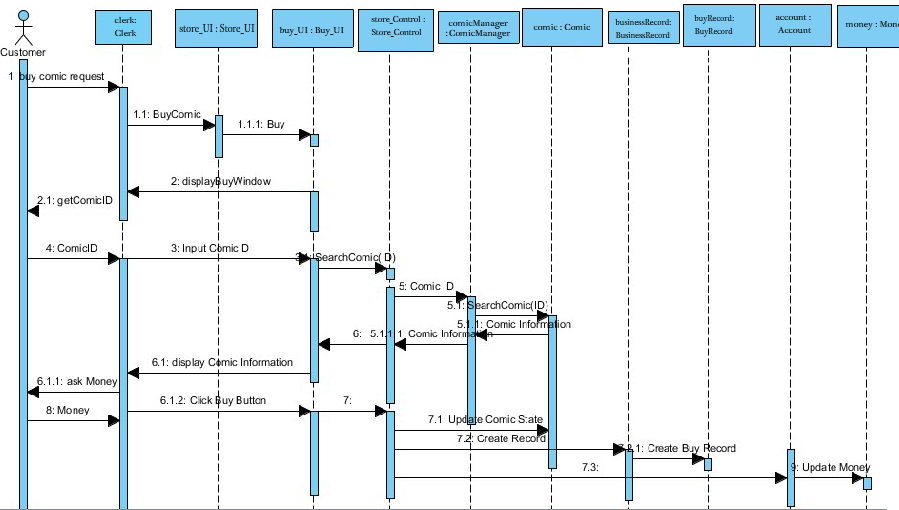
The Clerk and the Owner are regarded as Administer actor who is the system’s direct user.

* Search Comic

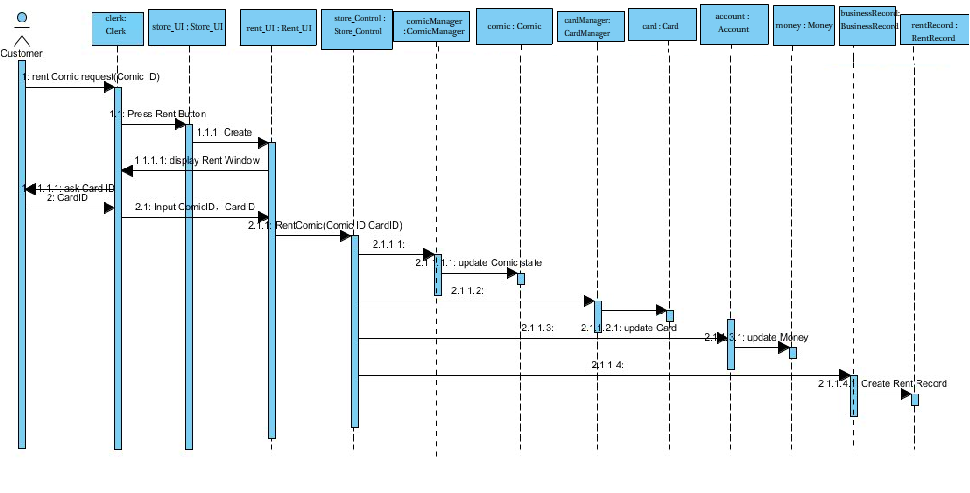


The User and the Subscriber are regarded as Customer actor.

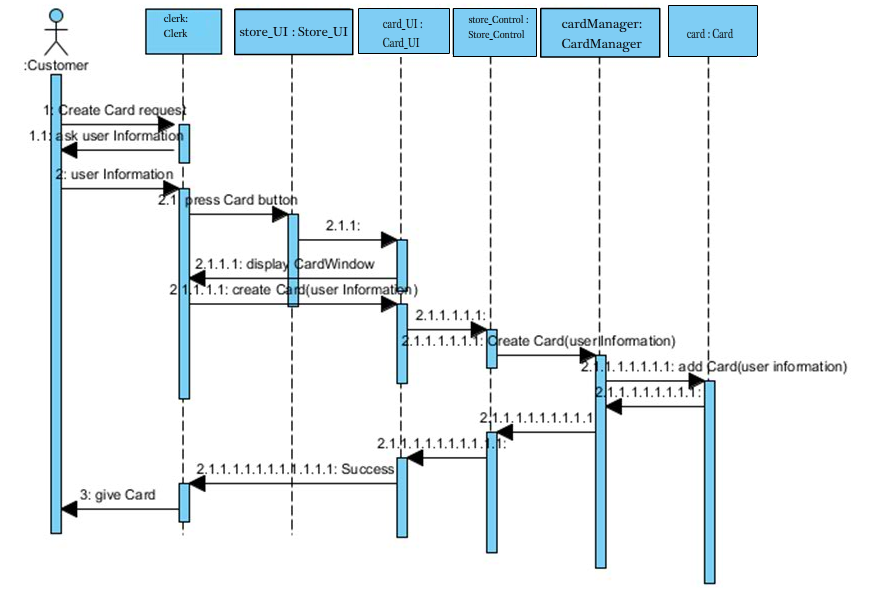
* Buy Comic



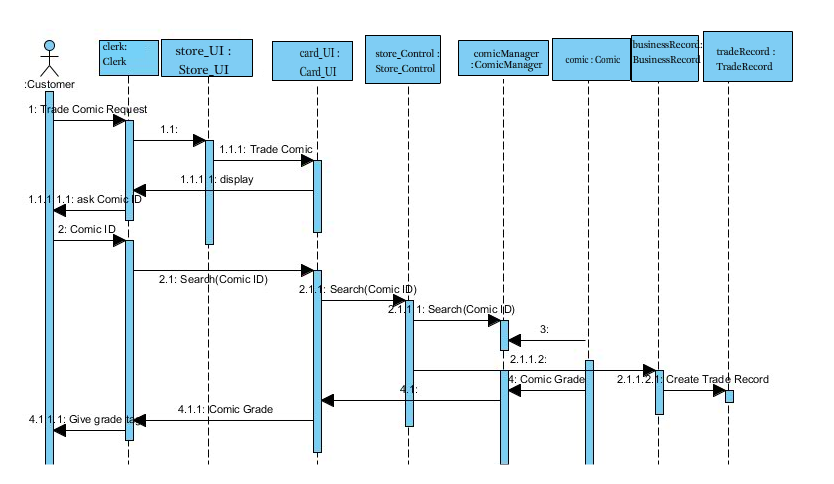
* Rent Comic



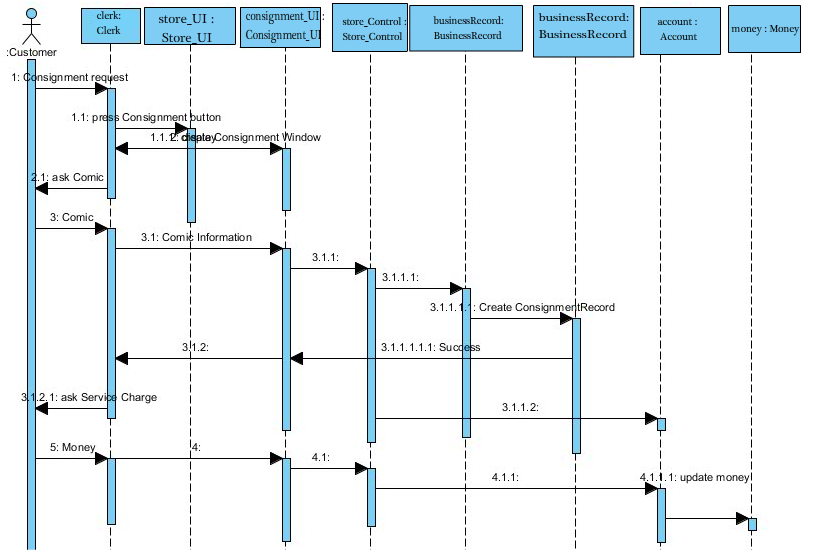
* Create New Card



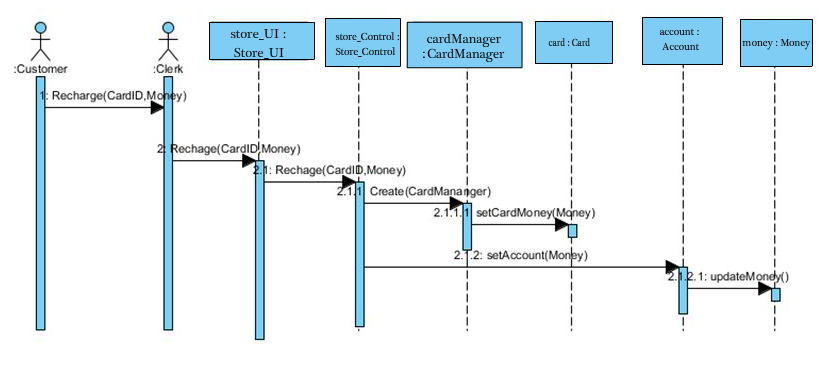
* Trade Comic



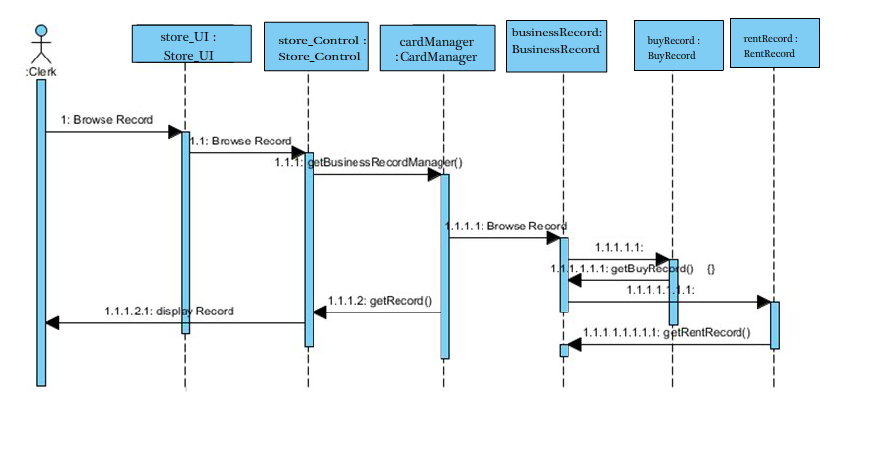
* Consignment Comic



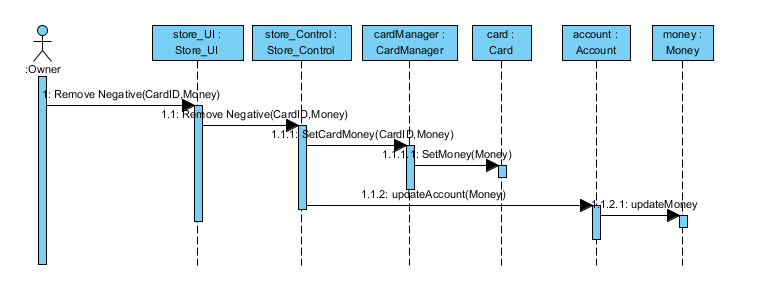
* Recharge



* Browse Record



* Remove Negative



* 1. System operation contract

在UP中，用例和系统特征是用来描述系统的主要行为方式，并且足以满足要求。有时需要对系统行为进行更为详细和精确得到描述。操作契约使用前置和后置条件的形式，描述领域模型里对象的详细变化，并作为系统的操作结果。

|  |  |
| --- | --- |
| **契约CO1** | SaleComplete(ComicID:String,u:Customer) |
| **操作** | 处理漫画销售 |
| **交叉引用** | 用例：Buy Comic |
| **Exception** | 提示输入错误信息。 |
| **Output** | 新增购买记录，购买漫画成功 |
| **前置条件** | 正在进行的销售中 |
| **后置条件** | * 假设用户不存在系统中，则创建新的用户u，假设用户存在于系统中，提取用户u * 创建购买记录R * R被关联到u中。 * R的属性被修改。 * 书本库存、状态更新 * 商店账户更新。 |

|  |  |
| --- | --- |
| **契约CO2** | RentComplete(ComicID:String,u:Customer,CardID:String) |
| **操作** | 处理漫画订阅 |
| **Exception** | 提示输入错误信息。 |
| **Output** | 用户租用漫画成功，创建租用记录 |
| **交叉引用** | 用例：Rent Comic |
| **前置条件** | 正在进行的订阅 |
| **后置条件** | * 假设用户不存在系统中，则创建新的用户u，假设用户存在于系统中，提取用户u * 假设卡不存在系统中，则创建新的卡c，假设卡存在于系统中，提取卡c * 将c关联到用户u上 * 创建订阅记录R * 书本库存、状态信息更新 * 商店账户更新 |

|  |  |
| --- | --- |
| **契约CO3** | ConsignmentComplete(ComicID:String,u:Customer) |
| **操作** | 处理漫画寄售 |
| **Exception** | 提示输入错误信息。 |
| **Output** | 漫画寄售成功，生成寄售记录 |
| **交叉引用** | 用例：Consignment Comic |
| **前置条件** | 正在进行的寄售 |
| **后置条件** | * 假设用户不存在系统中，则创建新的用户u，假设用户存在于系统中，提取用户u * 创建寄售记录R. * 书本库存、状态信息更新 * 商店账户更新 |

|  |  |
| --- | --- |
| **契约CO4** | TradeComplete(ComicID:String,u:Customer) |
| **操作** | 处理漫画交换 |
| **Exception** | 提示输入错误信息。 |
| **Output** | 漫画寄售成功，生成寄售记录 |
| **交叉引用** | 用例：Trade Comic |
| **前置条件** | 正在进行的交换 |
| **后置条件** | * 假设用户不存在系统中，则创建新的用户u，假设用户存在于系统中，提取用户u * 创建交换记录R * 生成漫画打分G。 |

|  |  |
| --- | --- |
| **契约CO5** | CreateNewCard() |
| **操作** | 处理办理订阅卡请求 |
| **Exception** | 提示输入错误信息。 |
| **Output** | 卡办理成功，生成新卡记录 |
| **交叉引用** | 用例：Create New Card |
| **前置条件** | 无 |
| **后置条件** | * 新建一个卡c，并生成特定的卡号。 * 假设用户不存在系统中，则创建新的用户u，假设用户存在于系统中，提取用户u * 将c关联到u中。 * 更新Card中信息。 |

|  |  |
| --- | --- |
| **契约CO6** | Recharge(CardID:String) |
| **操作** | 处理订阅卡充值请求 |
| **Exception** | 提示输入错误信息。 |
| **Output** | 用户卡充值成功 |
| **交叉引用** | 用例：Recharge |
| **前置条件** | 无 |
| **后置条件** | * 找到要充值的卡c * 更新c信息。 * 更新商店账户信息。 |

|  |  |
| --- | --- |
| **契约CO7** | Remove\_Nagative(CardID:String) |
| **操作** | 清除负值 |
| **Exception** | 提示输入错误信息。 |
| **Output** | 清楚负值成功 |
| **交叉引用** | 用例：Remove Negative |
| **前置条件** | Owner已经登陆 |
| **后置条件** | * 在系统中提取卡c * 将c中的属性Money修改 * 商店账户更新。 |

|  |  |
| --- | --- |
| **契约CO8** | SearchComic(ComicID:String) |
| **操作** | 处理漫画搜素 |
| **Exception** | 提示输入错误信息。 |
| **Output** | 漫画信息显示 |
| **交叉引用** | 用例：SearchComic |
| **前置条件** | 正在进行的销售中 |
| **后置条件** | * 假设漫画存在于系统中，则提取漫画Comic信息。假设漫画不在系统中，则返回none. |

|  |  |
| --- | --- |
| **契约CO9** | Login(u:User) |
| **操作** | 处理Owner/Clerk登陆 |
| **交叉引用** | 用例：Login |
| **前置条件** | none |
| **后置条件** | * 假设用户不存在系统中，则返回错误，假设用户存在于系统中，提取用户u * 将u关联到主界面中 |

|  |  |
| --- | --- |
| **契约CO10** | Browse\_Account() |
| **操作** | 查看商店账户 |
| **Exception** | 提示输入错误信息。 |
| **Output** | 商店账户显示 |
| **交叉引用** | 用例：Browse Account |
| **前置条件** | 无 |
| **后置条件** | * 提取商店账户信息a * a关联到主界面中 |

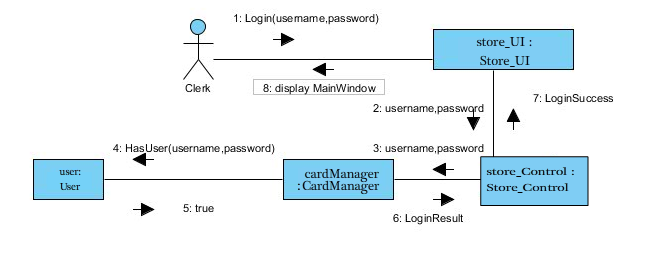
**System**

**Design**

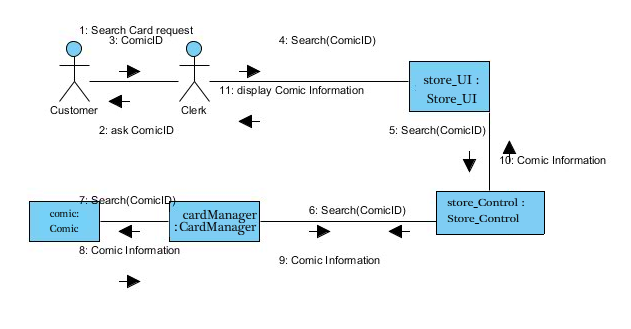
1. 架构设计（Architecture design）
   1. Communication Diagram

协作图用于表示对象间的消息往来。协作图明确的描述对象间的协作关系，是动态视图的另一种表现形式。因此根据系统分析中设计出来的时序图，我们可以得到如下的协作图：

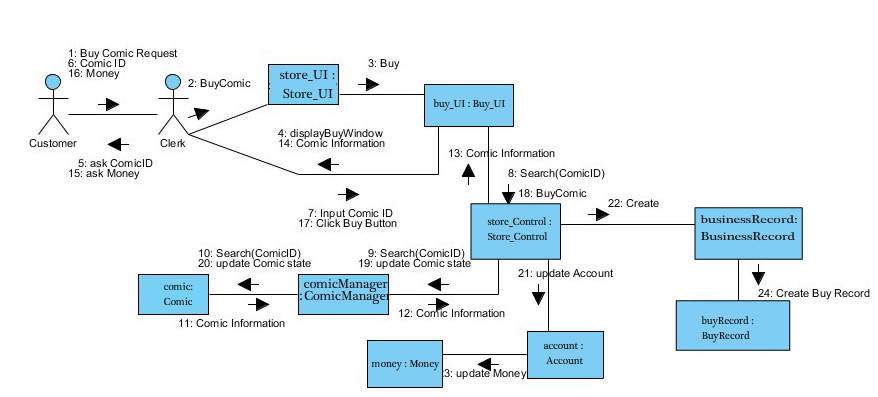
* Login



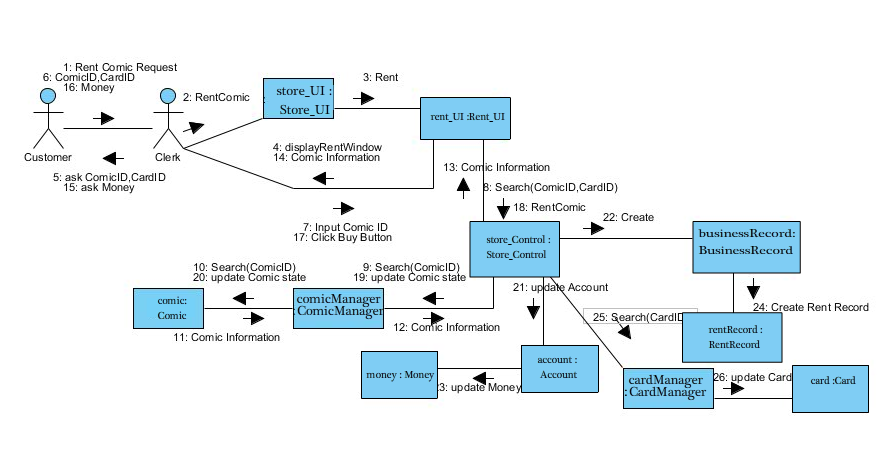
* Search Comic



* Buy Comic



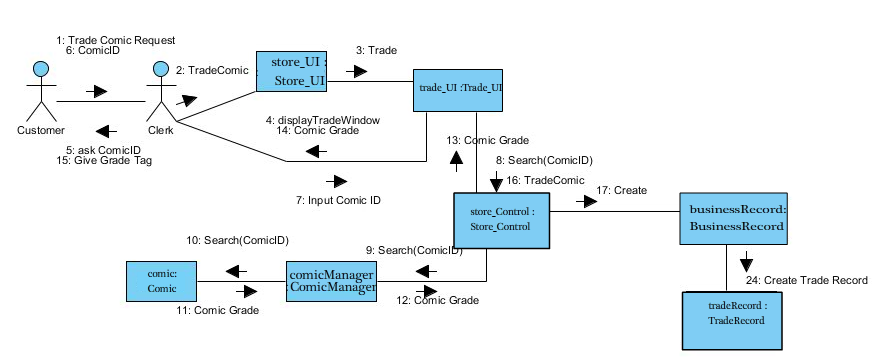
* Rent Comic



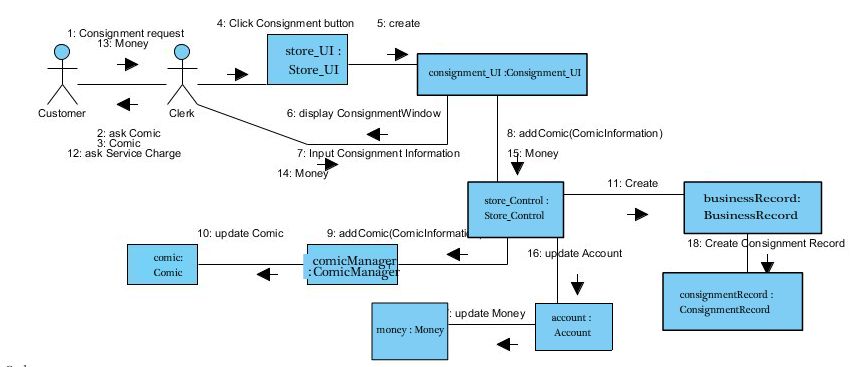
* Create New Card



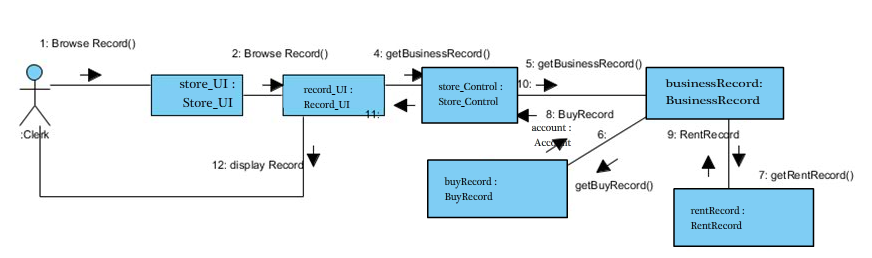
* Trade Comic



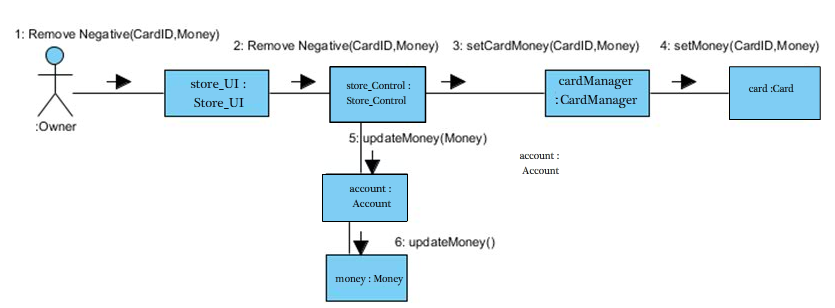
* Consignment Comic



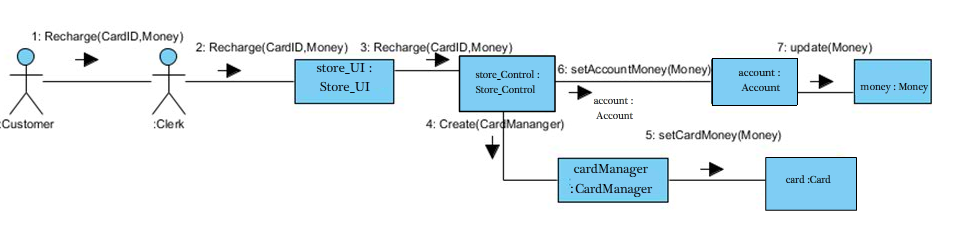
* Browse Record



* Remove Negative



* Recharge



* 1. Design classes
     1. Outline Boundary and Manager Classes:

In Login use case: UserManager, LoginWindow

In Trade Comic use case: ComicManager, TradeWindow

In Buy Comic use case: ComicManager, BuyWindow

In Rent Comic use case: ComicManager, RentWindow

In Stock Comic use case: ComicManager, StockWindow

In Consignment Comic use case: ComicManager, ConsignmentWindow

In Check Account use case: Account, StoreAccountWindow

In Handling Card use case: CardManager, HandlingCardWindow

In Remove Negative use case: Account, RemoveNegativeWindow

In Maintain Customer use case: CustomerManager

* + 1. Outline Entity

Entity classes:

User BusinessRecord

Customer RentRecord

Comic BuyRecord

Store ConsignmentRecord

Money ReturnRecord

TradeRecord

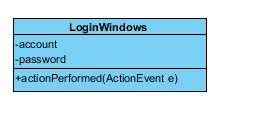
* + 1. Identifying attributes and operations
       - * Design LoginWindows Class
  + Attributes

-String account: record the user inputting ID

-String password: record the user inputing password

* + Operations

-public void actionPerformed(ActionEvent e): deal with the events of pressing different buttons.



* + - * + Design UserManager Class
  + Attributes

-User currentUser: record the current user’s login information

* + Operations

-public boolean add(User):add a new user into the system.

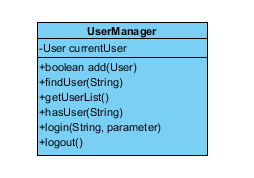
-public ArrayList <User> findUser(String):find a user according to the account.

-public ArrayList <User> getUserList(): get all user in the system.

-public boolean hasUser(String):Check if the system has the user according to the account.

-public boolean login(String, String): Confirm whether the account and the password match matched.

-public void logout():exit the system.



* + - * + Design ComicManager Class
  + Attributes

- ArrayList<ComicItem> comics

* + Operations

-public void addComicItem(Comic,int)

-public ArrayList<ComicItem> findComicItem(String): Find ComicItem.

-public ArrayList<ComicItem>findComicItemByID(String)

-public ComicItem removeComicItemByComicID(String)



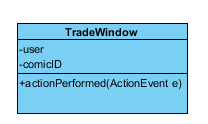
* + - * + Design TradeWindow Class
  + Attributes

-User u:record the user information

-String ComicID:record the Comic which the user want to trade.

* + Operations

-public void actionPerformed(ActionEvent e): deal with the events of pressing different buttons.



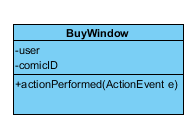
* + - * + Design BuyWindow Class
  + Attributes

-User u:record the user information

-String ComicID:record the ComicID which the user want to buy.

* + Operations

-public void actionPerformed(ActionEvent e): deal with the events of pressing different buttons.



* + - * + Design RentWindow Class
  + Attributes

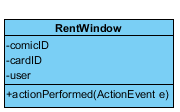
-User u:record the user information

-String CardID:record the subscriber’s card id.

-String ComicID:record the ComicID which the user want to rent.

* + Operations

-public void actionPerformed(ActionEvent e): deal with the events of pressing different buttons.

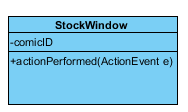


* + - * + Design StockWindow Class
  + Attributes

-String ComicID:record the ComicID which want to stock.

* + Operations

-public void actionPerformed(ActionEvent e): deal with the events of pressing different buttons.



* + - * + Design ConsignmentWindow Class
  + Attributes

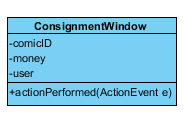
- String ComicID:record the ComicID which want to stock.

-Money m:record the service charge.

-User u:record the user information.

* + Operations

- public void actionPerformed(ActionEvent e): deal with the events of pressing different buttons.

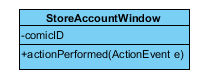


* + - * + Design StoreAccountWindow Class
  + Attributes

-Money m:record the store account.

* + Operations

- public void actionPerformed(ActionEvent e): deal with the events of pressing different buttons.

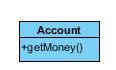


* + - * + Design Account Class
  + Attributes

-none

* + Operations

-public Money getMoney(): get the store account.



* + - * + Design CardManager Class
  + Attributes

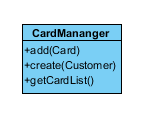
-none

* + Operations

-public boolean add(Card):add new card to the system.

-public Card create(Customer):create a new card.

-public ArrayList<Card> getCardList:show the cards in the system.

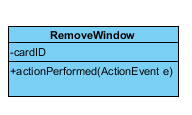


* + - * + Design RemoveWindow Class
  + Attributes

-String CardID:the Card wanted to remove it’s negative.

* + Operations

- public void actionPerformed(ActionEvent e): deal with the events of pressing different buttons.



* + - * + Design CustomerManager Class
  + Attributes

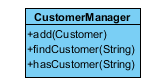
-none

* + Operations

-public boolean add(Customer):add customer into the system.

-public ArrayList<Customer> findCustomer(String):find Customer through some conditions.

-public boolean hasCustomer(String):Check whether the system has the customer.



* + - * + Design User Class
  + Attributes

-String name

-String password

-int permission: owner or clerk.

* + Operations

-public String getName()

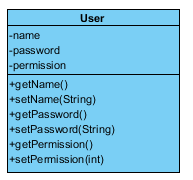
-public String getPassword()

-public int getPermission()

-public String setName(String)

-public String setPassword(String)

-public String setPermission(int)



* + - * + Design Customer Class
  + Attributes

-Card card

-String id

-String name

-String phone

-Date registerTime

* + Operations

-public Card getCard()

-public String getName()

-public String getId()

-public Date getRegisterTime ()

-public String getPhone()

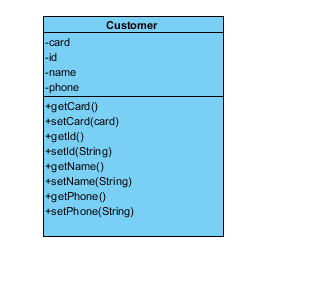
-public Card setCard(Card)

-public String setName(String)

-public String setId(String)

-public Date setRegisterTime (Date)

-public String setPhone(String)



* + - * + Design Comic Class
  + Attributes

-String author

-double grade

-String id

-String title

-String publisher

-String ISBN

- Money price

* + Operations

-public String getId()

-public String getTitle()

-public String getAuthor()

-public String getPublisher()

-public String getISBN()

-public Money getPrice()

-public double getGrade()

-public void setId(String id)

-public void setTitle(String title)

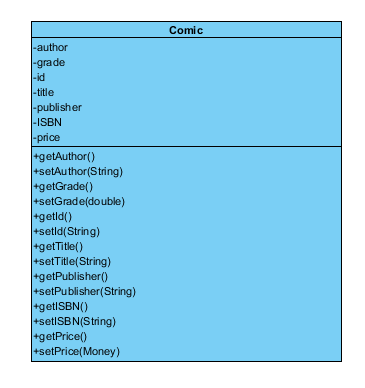
-public void setAuthor(String author)

-public void setPublisher(String publisher)

-public void setISBN(String iSBN)

-public void setPrice(Money price)

- public void setGrade(double grade)



* + - * + Design Store Class
  + Attributes

-BookStorage bookStorage;

- CardManager cardManager;

-UserManager userManager;-

-CustomerManager customerManager;

-BusinessRecordManager businessRecordManager

* + Operations

-public LoginResult login(String account, String password):Confirm whether the account and the password matched.

-public ArrayList<BookItem> getBookItemList()

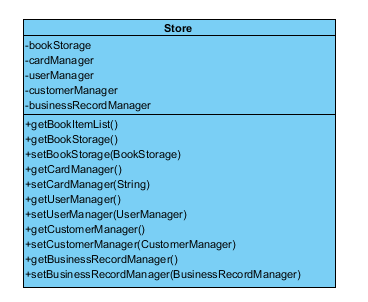
-public BookStorage getBookStorage()

-public CardManager getCardManager()

-public void setCardManager(CardManager cardManager)

-public UserManager getUserManager()

-public void setBookStorage(BookStorage bookStorage)



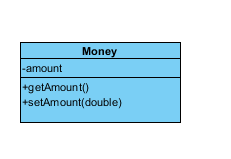
* + - * + Design Money Class
  + Attributes

-double amount

* + Operations

- public double getAmount()

- public void setAmount(double amount)



* + - * + Design Card Class
  + Attributes

-String id

-Customer customer

-Money money

-Date createdTime

* + Operations

- public Date getCreatedTime()

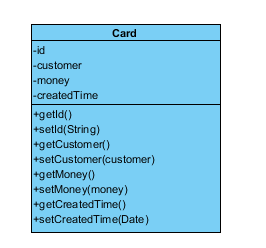
- public void setCreatedTime(Date createdTime)

- public Money getMoney()

- public void setMoney(Money money)

- public Customer getCustomer()

- public void setCustomer(Customer customer)

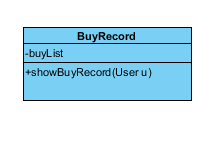


* + - * + Design BuyRecord Class
  + Attributes

- ArrayList<BuyItem> buyList :record the buy comic list.

* + Operations

-public String showBuyRecord( User u): return the user return buy record.



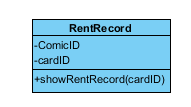
* + - * + Design RentRecord Class
  + Attributes

-String ComicID

-String cardID

* + Operations

- public String showRentRecord (String cardID):return the user rent record.

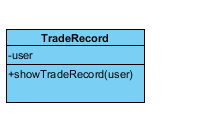


* + - * + Design TradeRecord Class
  + Attributes

-User u

* + Operations

-public String Trade(User u):return the user trade record.

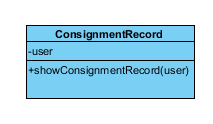


* + - * + Design ConsignmentRecord Class
  + Attributes

-User u

Operations

-public String showConsignmentRecord(User u):return the user Consignment record.

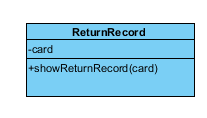


* + - * + Design ReturnRecord Class
  + Attributes

-Card card

* + Operations

-public String showReturnRecord(Card card): return the user return comic record.



* + - * + Design BusinessRecord Class
  + Attributes

-String id

- String customerId

- String action

- Money amount

- Date time

- String handlerId

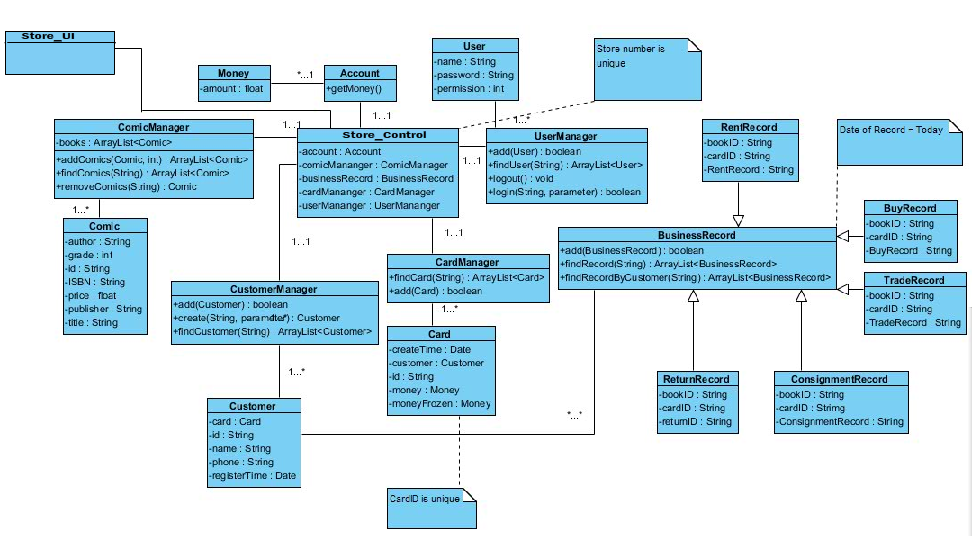
- String comment

* + Operations

- public static BusinessRecord createWithRecord(String record):Create the record according to the customer action.



Final class diagram



* 1. Identifying associations and aggregations

Association and aggregation are used to describe the relationships between design classes. (e.g. a “Card” class may have a single association with a “CardID” class) According to our generic design mechanism, we should minimize the number of relationships (associations) between classes. Hence, at the end of this activity, we manage to remove all of the associations between our entity classes. That is, searching or modifying any of the entity classes can be done without invoking other entity classes.

* 1. Identifying generalizations

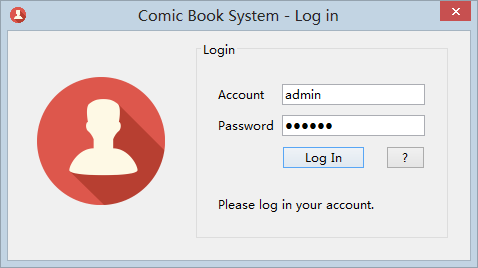
Generalization represents a relationship in which one class (the child) is derived from another class (the parent). For example, “BusinessRecord” class can be defined as the parent class while “RentRecord” class and “BuyRecord” class are defined as child class. In this system, no generalization relationship is applied.

* 1. Describing methods

Methods are used to specify how operations are realized. They will be generated using Java code directly at the next stage, i.e. implementation stage. Please see the next part of this report for more detail

**System**

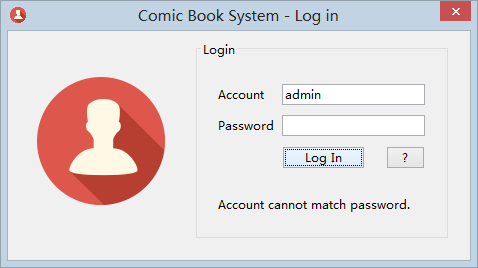
**Implementation**



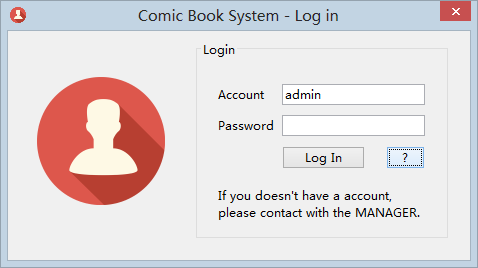
系统登录



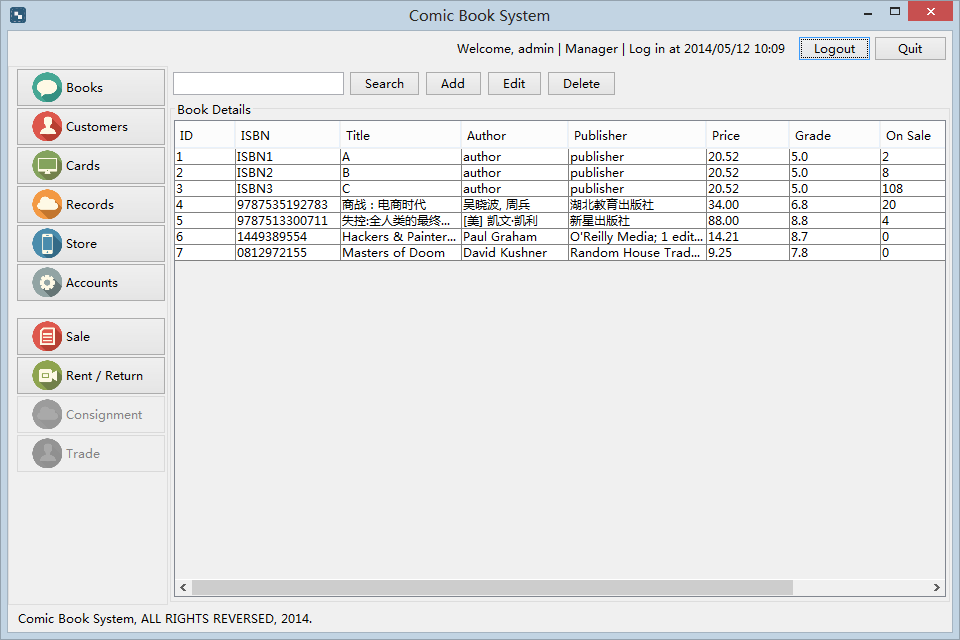
用户不存在



密码错误

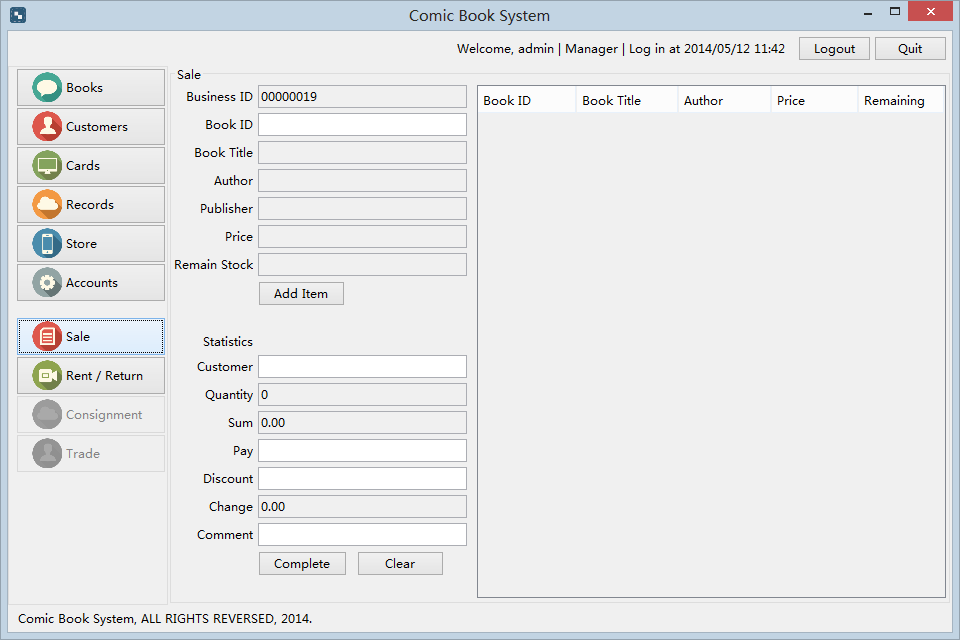


如果需要添加新用户，必须联系管理员。由管理员进行添加。

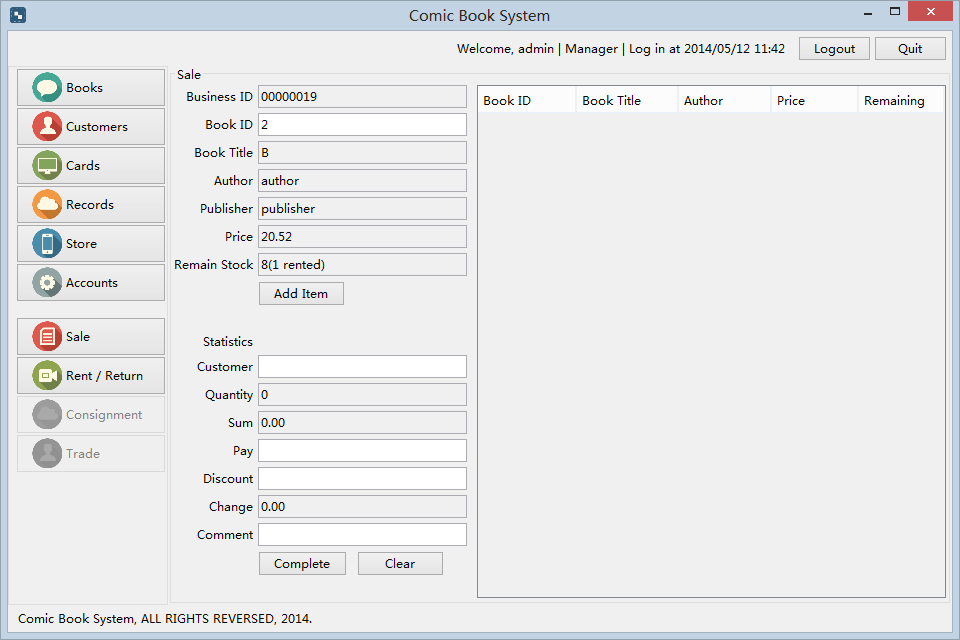


登录成功后，自动进入系统主界面。可以在右上角看到当前登录的用户名，角色（包括Clerk和Manager，和登录时间）默认界面是Book界面

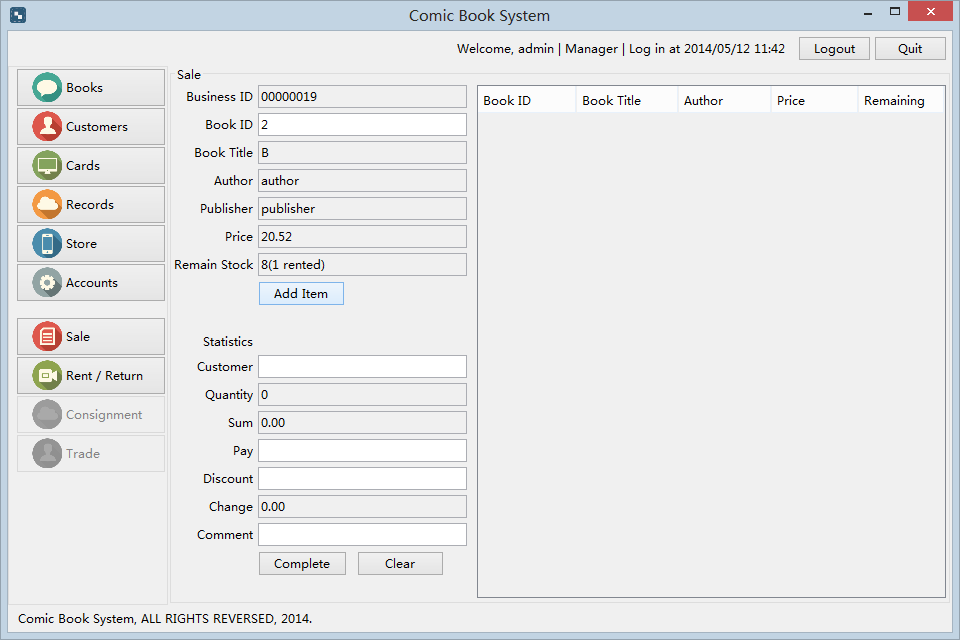
点击Sale可以转到卖书界面



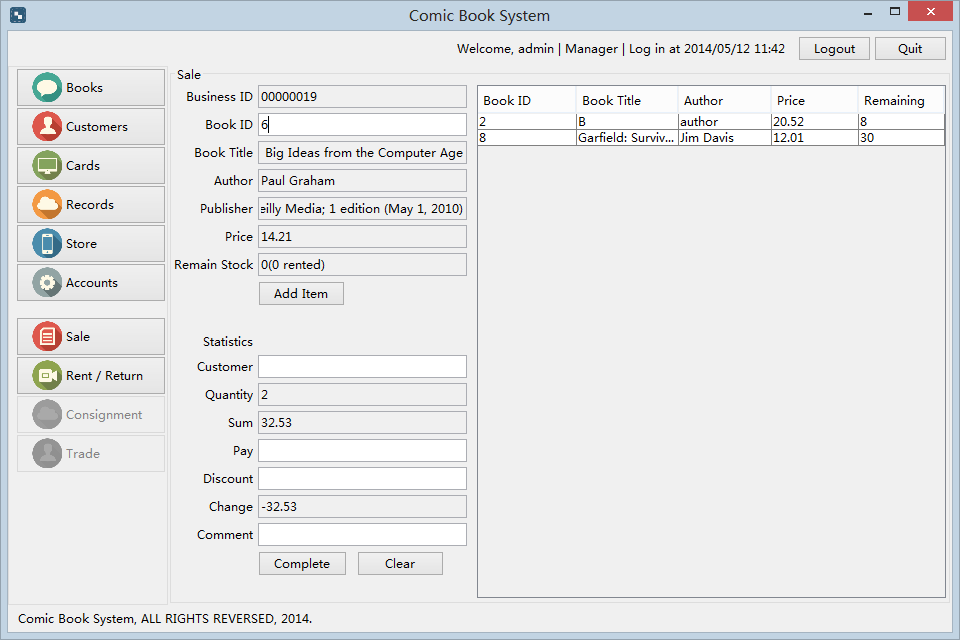
输入BookID，系统自动显示改书的详细信息



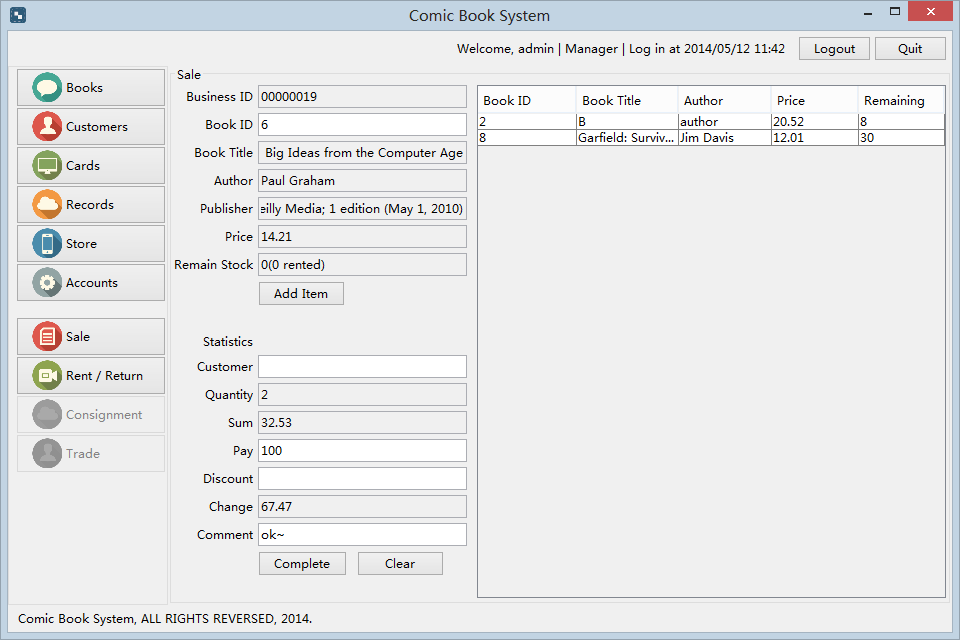
点击Add Item或者输入Enter 可以添加一项SalesItem



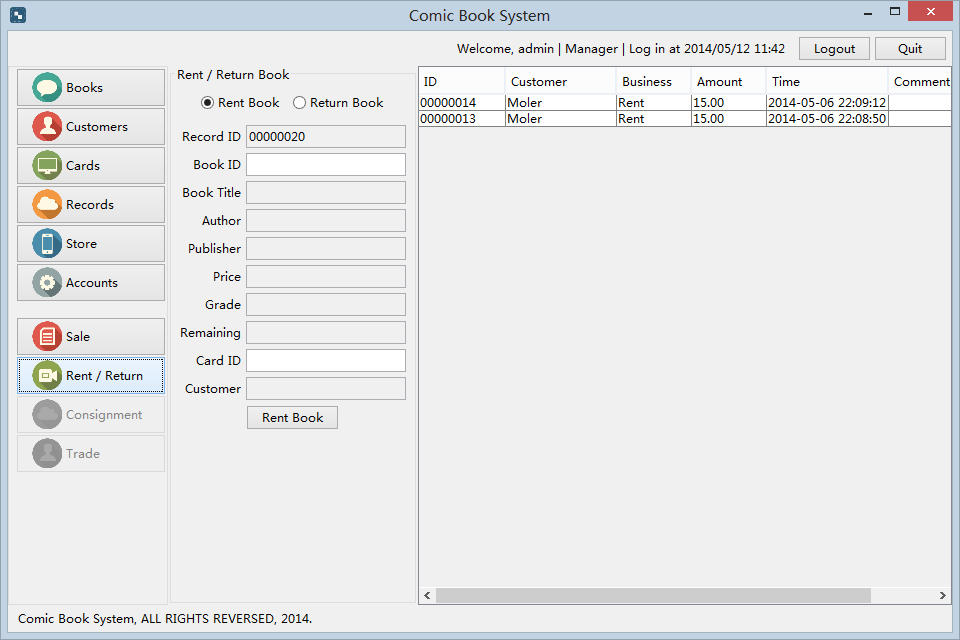
界面中下部分可以显示出本次销售的统计信息，包括Customer, Book qunatity, sum, pay discount, change 并且可以输入付款Pay，折扣Discount 以及备注



输入完所有信息并且 点击Complete完成这次购买



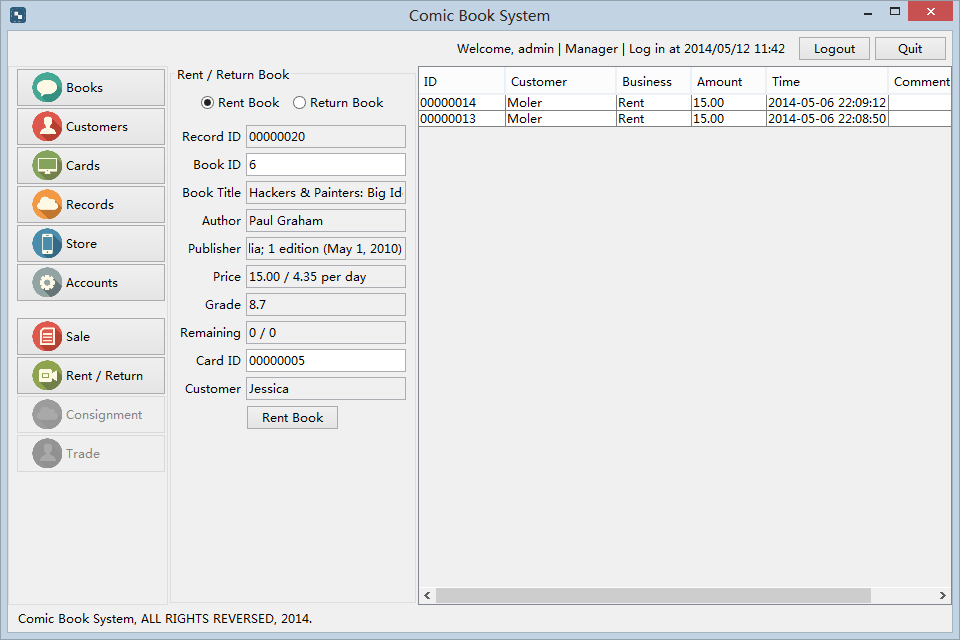
点击左侧的Rent / Return按钮，可以进行借阅和还书操作



点击中部面板的Rent Book进行借书操作：

输入Book ID 以及Card ID

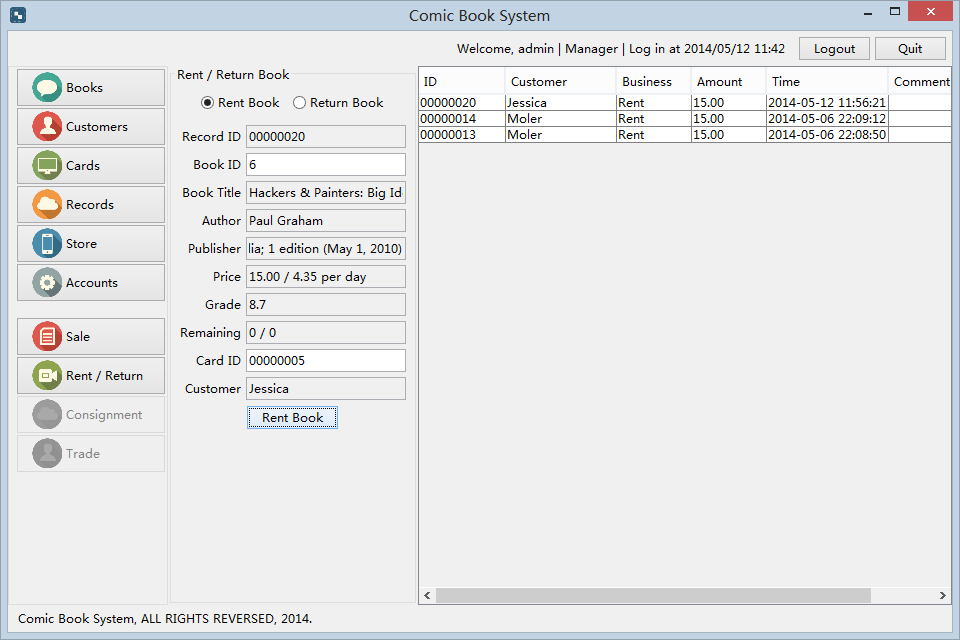
点击Rent Book，如果满足借阅条件（订阅卡借书少于2本， 订阅卡余额不为负值， 库存充足）则可以完成借阅功能。



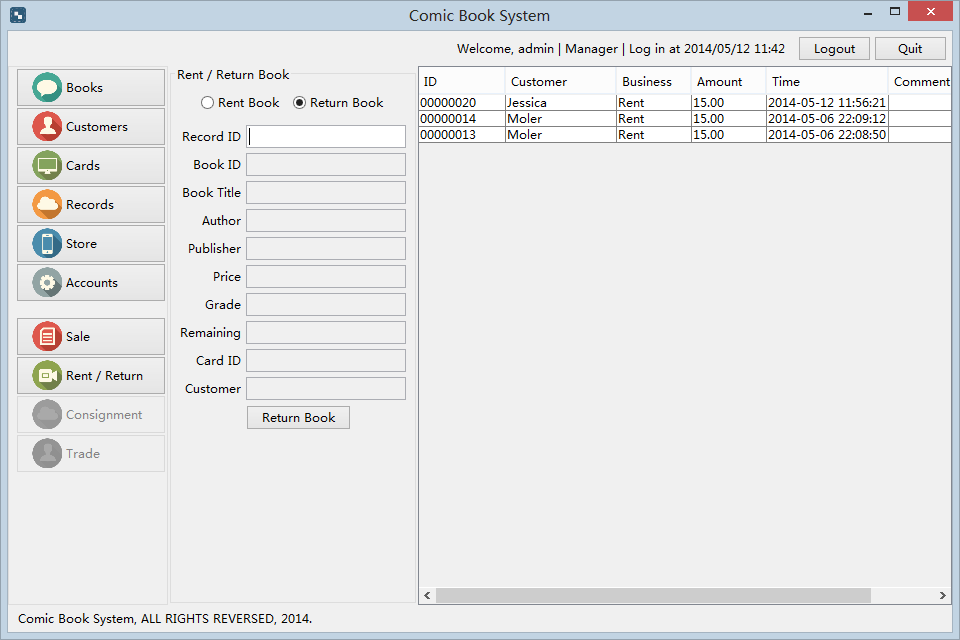
成功后会弹出提示窗口Rent record has been created

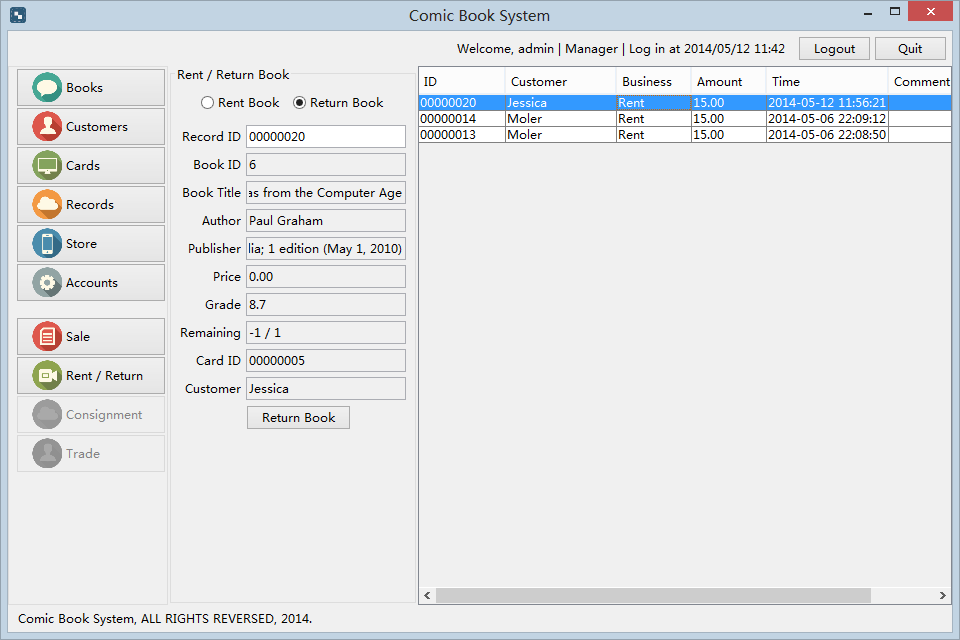


并且右侧借阅列表也会更新

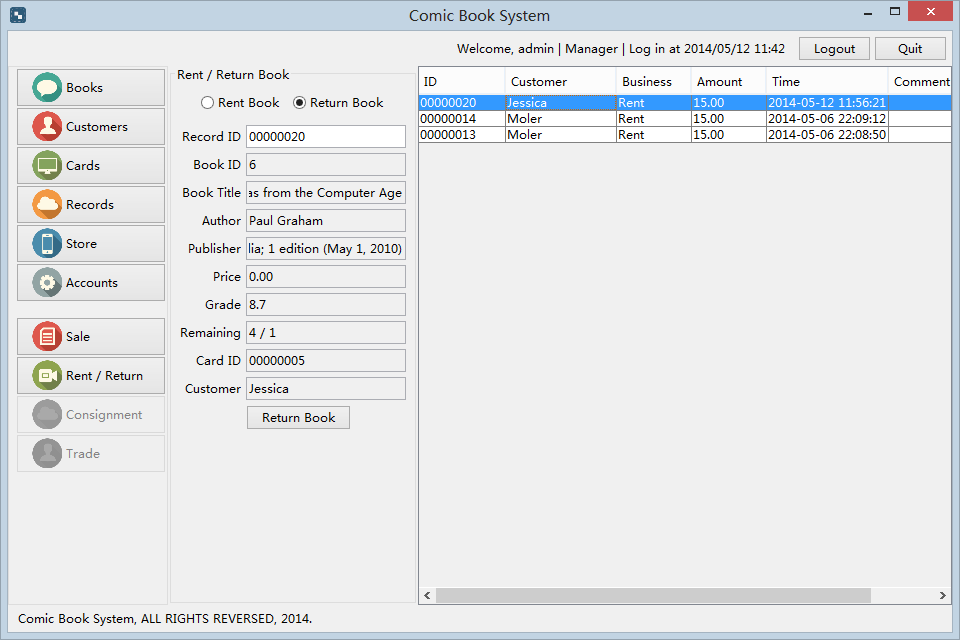


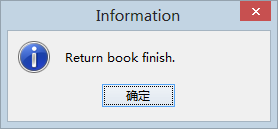
点击面板中部的Return Book进行归还操作，可以输入借阅ID或者双击右侧的借阅记录自动完成ID填写



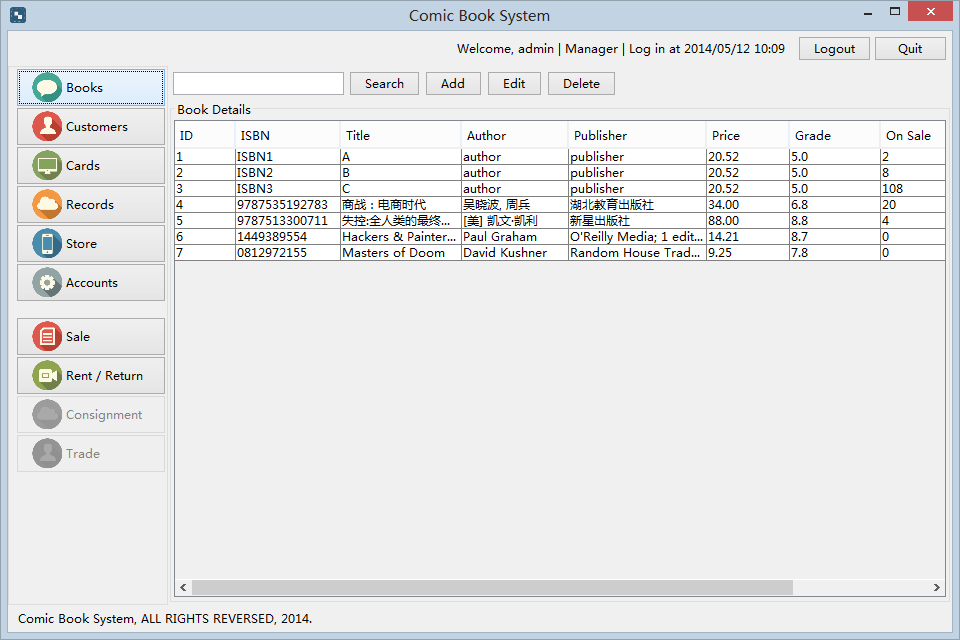


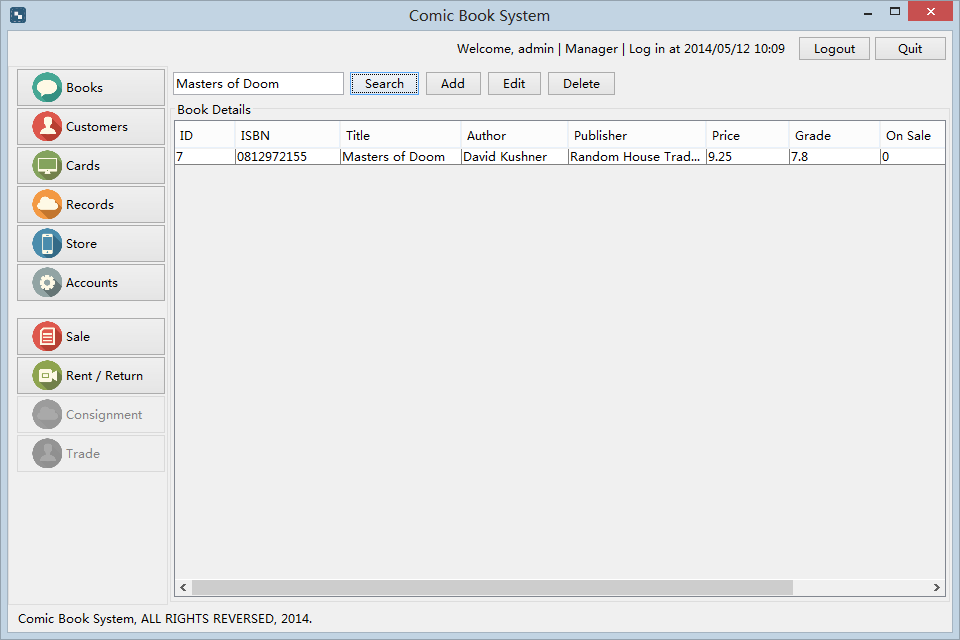
点击Return book即可完成归还操作





点击左侧Books按钮可以查看所有库存数据。右侧面板可以查看所有书籍的信息，包括书籍ID, ISBN, Title, Author, Publisher, Price, Grade and stock quantity

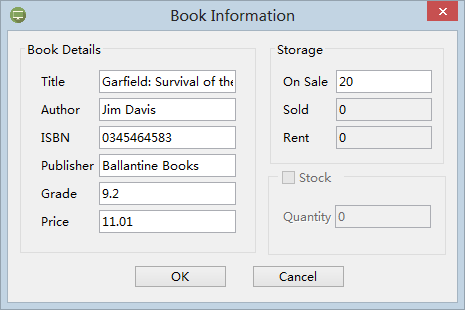




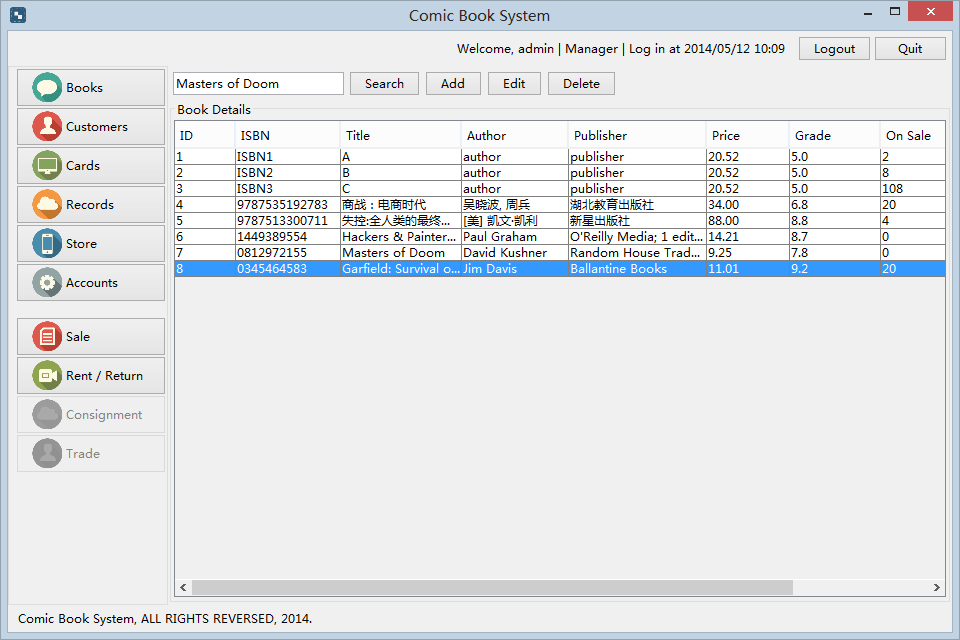
在中部的搜索输入框输入关键字，可以搜索所有匹配的书籍，方便查找

在Customers面板, Books面板, Cards面板, Records面板,Accounts面板中均有此项搜索功能

可以方便的检索数据量较多的系统

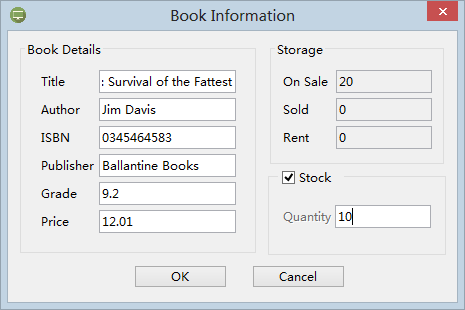


点击上方的Add按钮，可以添加一本新书。点击Add后，弹出Book Information窗口，输入漫画的Title, Author, ISBN, Publisher, Grade, Price 并且增加相应库存, 最后点击OK完成漫画的添加。

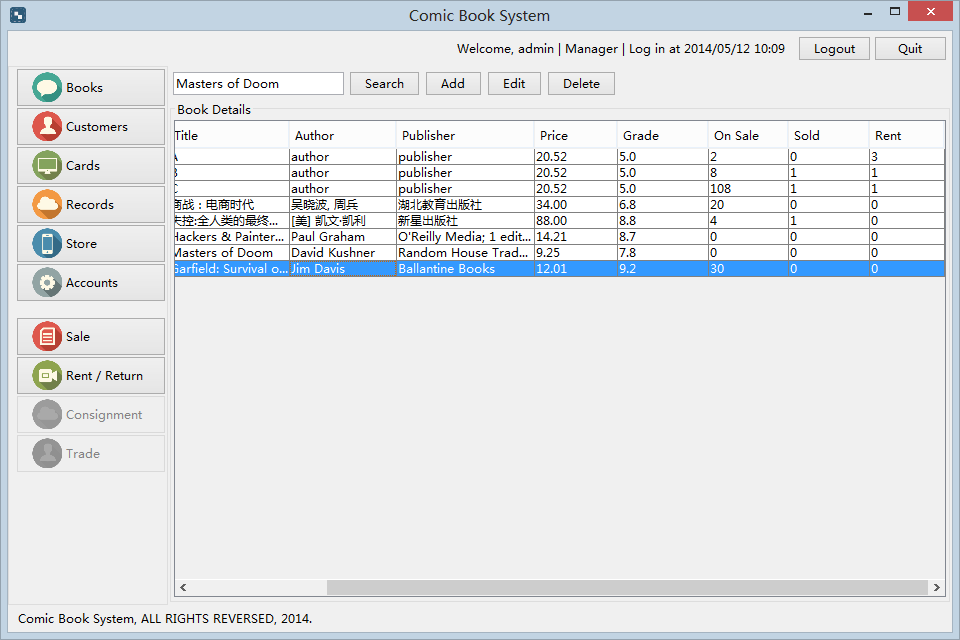


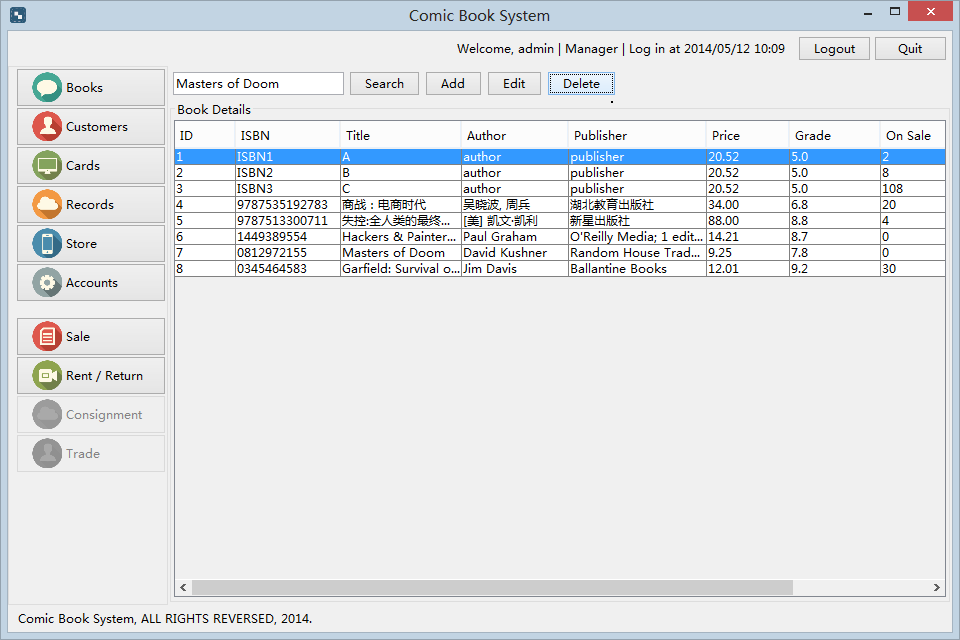
可以看到，刚刚添加的漫画《Garfield: Survival of the Fattest》已经被添加的Books书籍库存中。双击面板上的该书记录，再次弹出Book Information窗口，勾选右下角的Stock，可进行进货操作, 并且自动扣除Store account的余额。

左边的信息是可以编辑的

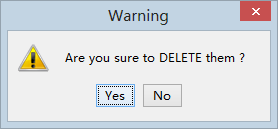


点击OK 书籍数据已经被更新





选中一本或多本书籍，点击Delete按钮，可以删除书籍

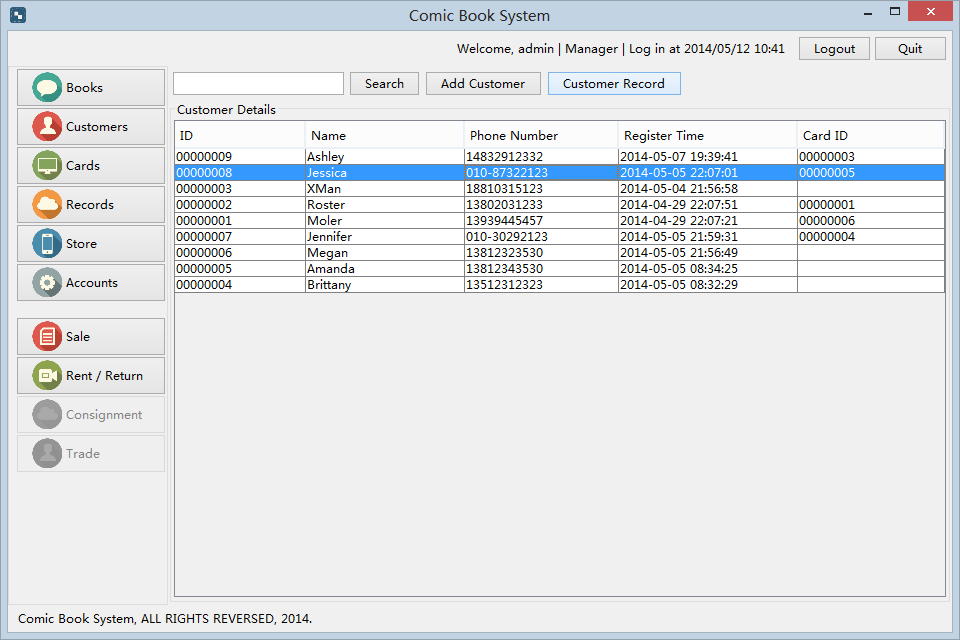


点击左侧的Customers，可以看到顾客列表

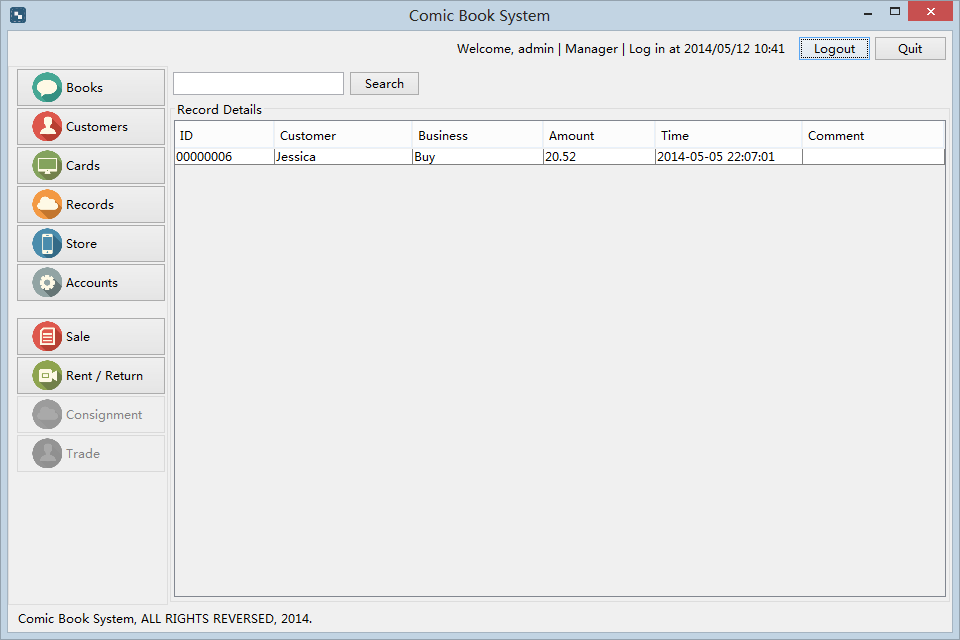


双击一个顾客记录，可以看到并且编辑顾客信息





选中一个顾客记录，并且点击中部的Customer Record按钮，可以看到该顾客的消费记录



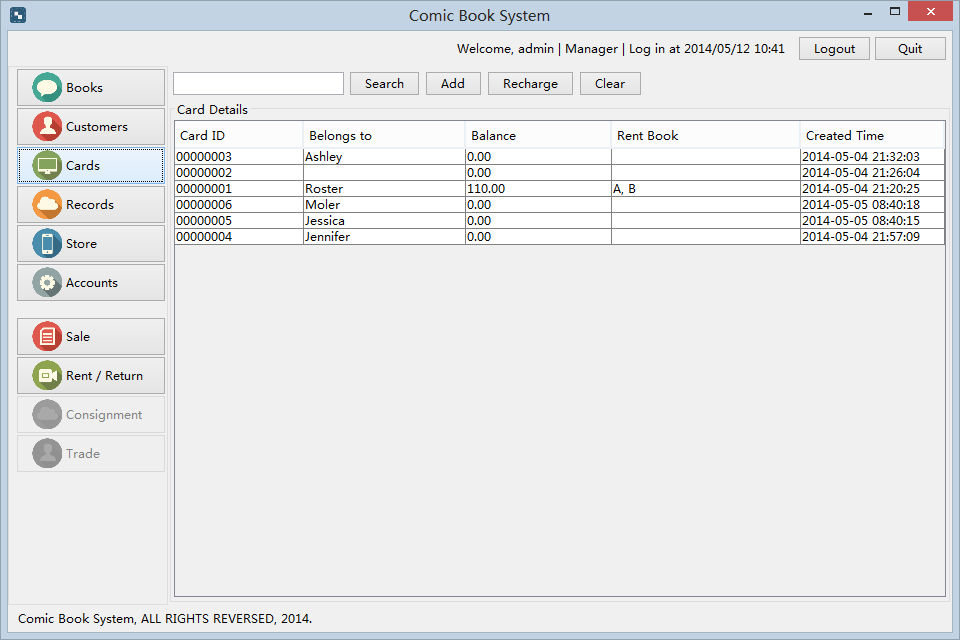
点击左侧的Cards按钮，可以进行订阅卡的管理。

操作有Add, Recharge, Clear

Add是添加操作可以创建一张新卡

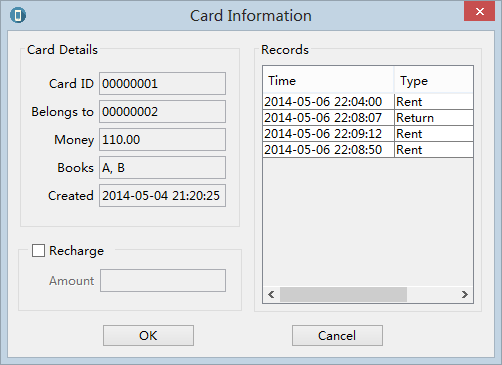
Recharge是充值操作

Clear是余额清零功能，可以将负债的订阅卡的余额清除。此项操作仅有Manager能使用

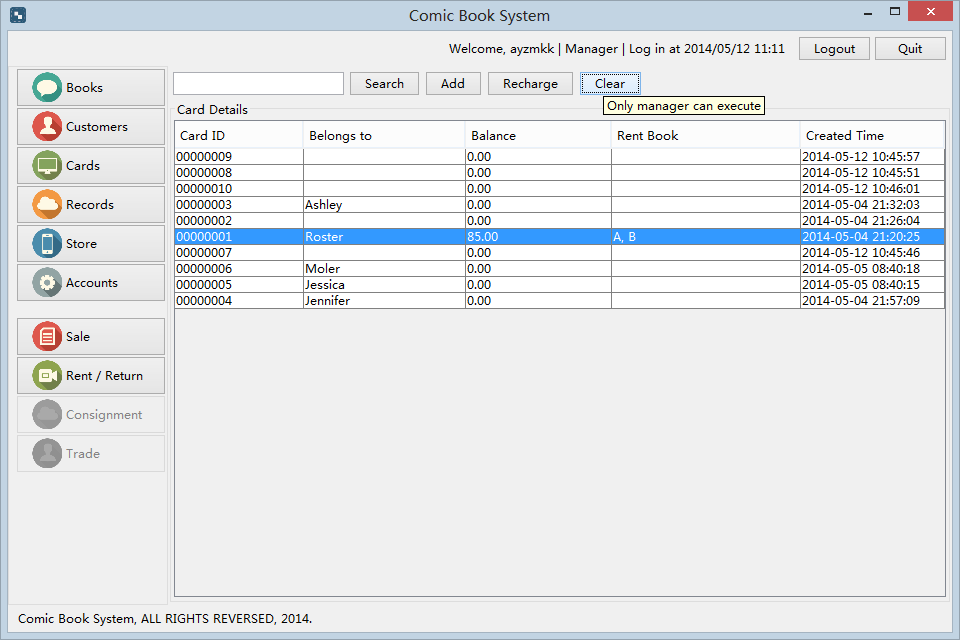


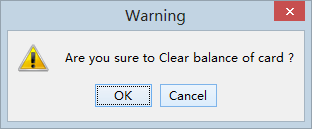
双击Card，可以进行充值以及查看Card的交易记录信息。

并且可以查看本卡已借阅多少本书

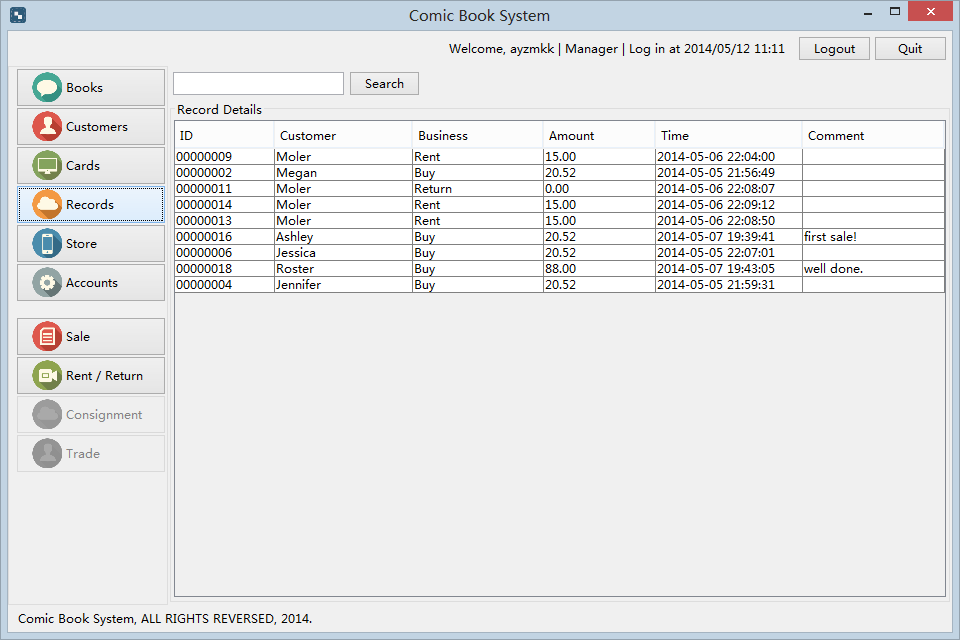


点击Clear按钮是余额清零功能，可以将负债的订阅卡的余额清除。

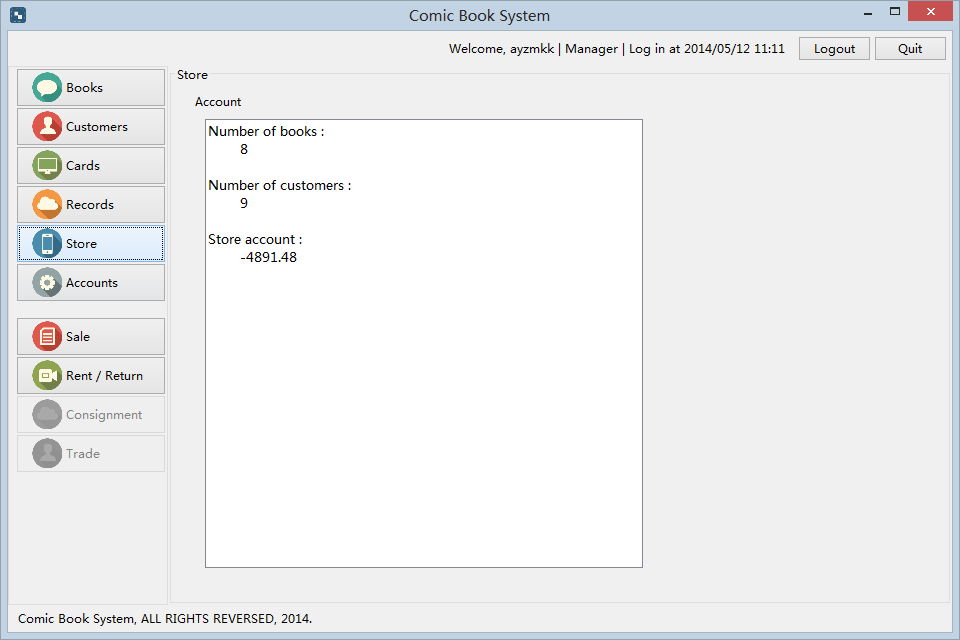




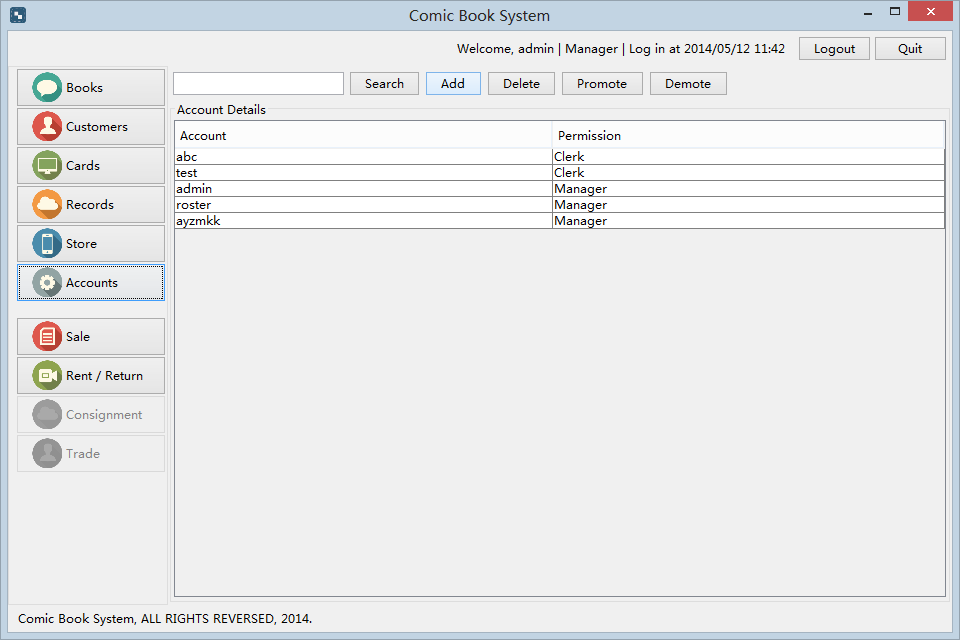
点击左侧的Records，可以查看业务记录列表，包括借阅购买返还等业务的记录



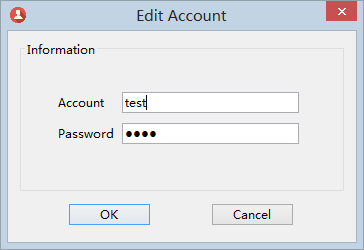
点击Store可以查看商店的统计信息，包括商店账户的余额，漫画书的数量，顾客的数量等



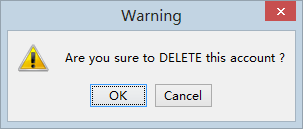
点击Accounts可以查看所有可以使用该系统的用户，包括Clerk和Manager。如果当前用户是Manager可以进行以下4个操作



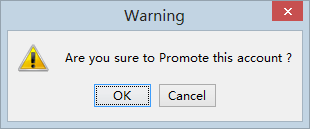
点击Add可以添加一个新的用户



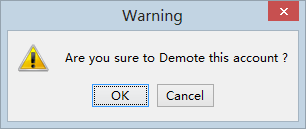
点击Delete可以删除一个用户



点击Promote可以将一个Clerk提升为Manager



点击Demote可以将一个Manager降级为Clerk



**Test**

1. Design Test

* Component testing
  1. Testing for Login
     1. Test case 1 for login (correct input)

-Input

account:a

password:123

-Result

Display the MainWindow

-Conditions

none

* + 1. Test case 2 for login (incorrect input)

-Input

account:aeee

password:123

-Result

Account does not exist.

-Conditions

none

* + 1. Test case 3 for login (incorrect input)

-Input

Account:a

Password:1233

-Result

Account does not match the password

-Conditions

none

* 1. Testing for add comic
     1. Test case 1 for add comic (correct input)

-Input

Title:蜡笔小新

Author: 臼井仪人

ISBN: 9789573420958

Publisher:东立

Grade:5

Price:100

-Result

See the new book in the MainWindow

-Conditions

none

* + 1. Test case 2 for add comic (incorrect input)

-Input

Title:蜡笔小新

Author: 臼井仪人

ISBN: 9789573420958

Publisher:东立

Grade:5

Price:

-Result

Please input a legal price!

-Conditions

Clerk is login

* + 1. Test case 3 for add comic (incorrect input)

-Input

Title:蜡笔小新

Author: 臼井仪人

ISBN: 9789573420958

Publisher:东立

Grade:20

Price:111

-Result

Grade must be 0<=grade <=10.

-Conditions

Clerk is login

* 1. Testing for add customer
     1. Test case 1 for add customer (correct input)

-Input

Name:Mocha

Phone:18810315099

-Result

A new customer in the MainWindow

-Conditions

Clerk is login

* + 1. Test case 2 for add customer (incorrect input)

-Input

Name:

Phone:18810315099

-Result

Please input name

-Conditions

Clerk is login

* + 1. Test case 3 for add customer (incorrect input)

-Input

Name:Mocha

Phone:

-Result

Please input phone.

-Conditions

Clerk is login

* System testing
  1. Testing for buy comic.
     1. **Test case 1 for buy comic (correct input)**

-Input

BookID:1

Customer:00000009

Pay:100

-Result

Sale record Create

-Conditions

Clerk is login

* + 1. Test case 2 for buy comic (incorrect input)

-Input

BookID:999

Customer:00000009

Pay:100

-Result

No response

-Conditions

Clerk is login

* + 1. Test case 3 for buy comic (incorrect input)

-Input

BookID:1

Customer:00000009

Pay:20

-Result

No response

-Conditions

Clerk is login

* 1. Testing for rent comic
     1. Test case 1 for rent comic (correct input)

-Input

BookID:1

CardID:00000001

-Result

Rent record has been created

-Conditions

Clerk is login

* + 1. Test case 2 for rent comic (incorrect input)

-Input

BookID:1

CardID:000000012

-Result

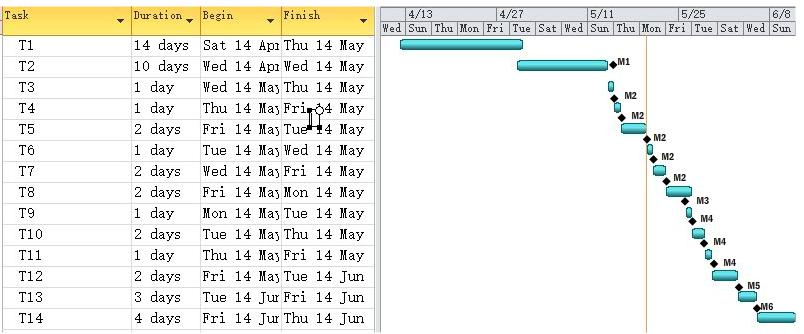
No response

-Conditions

Clerk is login

**Weekly**

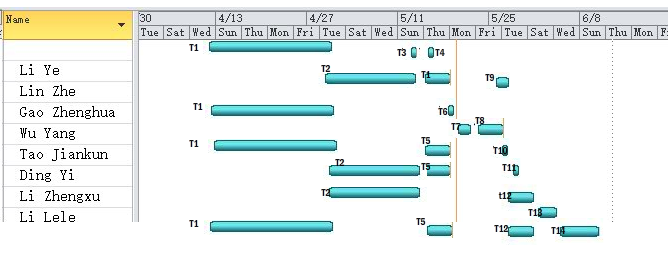
**Report**



|  |  |  |
| --- | --- | --- |
| **Task** | **Duration(days)** | **Dependencies** |
| **T1** | **14** |  |
| **T2** | **10** |  |
| **T3** | **1** | **T1,T2(M1)** |
| **T4** | **1** | **T3(M2)** |
| **T5** | **2** | **T3(M2)** |
| **T6** | **1** | **T3(M2)** |
| **T7** | **2** | **T3(M2)** |
| **T8** | **1** | **T7(M3)** |
| **T9** | **1** | **T8(M4)** |
| **T10** | **2** | **T8(M4)** |
| **T11** | **1** | **T8(M4)** |
| **T12** | **2** | **T8(M4)** |
| **T13** | **3** | **T12(M5)** |
| **T14** | **4** | **T13(M6)** |

|  |  |
| --- | --- |
| **Task** | **Description** |
| **T1** | **Capturing requirements** |
| **T2** | **Draw use case model** |
| **T3** | **Design entity classes** |
| **T4** | **Design subsystem--comic management** |
| **T5** | **Design subsystem-- User management** |
| **T6** | **Design subsystem--BusinessRecord** |
| **T7** | **Design subsystem:--Customer management** |
| **T8** | **Implement entity Class** |
| **T9** | **Implement subsystem--comic management** |
| **T10** | **Implement subsystem-- User management** |
| **T11** | **Implement subsystem--BusinessRecord** |
| **T12** | **Implement subsystem:--Customer management** |
| **T13** | **Component test** |
| **T14** | **System test** |

|  |  |
| --- | --- |
| **Milestone** | **Description** |
| **M1** | **Complete requirement capture and design the use case.** |
| **M2** | **Finish the entity class design** |
| **M3** | **Finish system design** |
| **M4** | **Finish the entity class implements** |
| **M5** | **Finish the system implements** |
| **M6** | **Finish the component test** |



**Group Number**: Group x

**Group Members**:

1

2

3

4

5

6

7

8

9

**Group Leaders Name**:

1

**Individual Member:**

2

3

4

5

6

7

8

9

**Individual Members Contribution**

Basic Contributions of Every Member:

1. Attending weekly meeting on time

2. Discussing problems with other members within the whole procedure of the project 3.Cooperating with each other in activities of work flows

4. Good Team Work Spirits and Serious Attitude toward tasks

Individual members contribution

**Subgroup 1**

|  |  |  |  |
| --- | --- | --- | --- |
|  | People1 | 2 | 3 |
| **Capturing requirements** | **1.Sum up the functional requirements of the System Management 2.Detail the use case description - 3. Draw use case model** | **1.Sum up the non-functional requirements 2.Detail the use case description 3.Draw use case model** | **1.Refine the requirement documentation**  **2. Detail the use case description**  **3.Draw use case model** |
| **Analysis** | **1.Detail the Analysis Packages**  **2.Identify Common Special Requirements 3.Analyze Use Case**  **4.Analyze the Class** | **1.Identify Common Special Requirements 2.Analyze Use Case**  **3.Analyze the Class** | **1.Modify the Analysis Packages 2.Identify Common Special Requirements**  **3.Analyze Use Case Initialization**  **4.Analyze the Class** |
| **Design** | **1.Architectural Design**  **2.Design use cases 3.Design classes**  **4.Design subsystem--Administrator‘s management subsystems** | **1.Architectural Design**  **2.Design use cases 3.Design classes 4.Design subsystem-- User account management subsystems** | **1. Identifying subsystem interfaces and design classes**  **2. Design use cases**  **3. Design classes-**  **4. Design subsystem interfaces** |
| **Implementation** | **1.Architectural Implementation**  **2.Integrate System 3.Implement Subsystem ---Administrator‘s management subsystems 4.Implement Class** | **1.Architectural Implementation 2.Integrate System 3.Implement a Subsystem of User account management subsystems 4.Implement Class** | **1.Architectural Implementation---- 2.Integrate System 3.modify Subsystem interfaces 4.Implement Class** |
| **Test** | **1.Plan the test for the system management 2.Design the test and write test components**  **3.Implement the test** | **1.Plan the test for the system management 2.Design the test and write test components**  **3. Implement the test Make integration test plan.** | **1.Design the test 2.Implement the designed test 3.Perform the integration test** |

**Subgroup 2**

|  |  |  |  |
| --- | --- | --- | --- |
|  | People1 | 2 | 3 |
| **Capturing requirements** | **1.Sum up the functional requirements of the System Management 2.Detail the use case description - 3. Draw use case model** | **1.Sum up the non-functional requirements 2.Detail the use case description 3.Draw use case model** | **1.Refine the requirement documentation**  **2. Detail the use case description**  **3.Draw use case model** |
| **Analysis** | **1.Detail the Analysis Packages**  **2.Identify Common Special Requirements**  **3.Analyze Use Case**  **4.Analyze the Class** | **1.Modify the Analysis Packages**  **2.Analyze Use Case**  **3.Analyze the Class** | **1.Identify Common Special Requirements**  **3.Analyze Use Case Initialization**  **4.Analyze the Class** |
| **Design** | **1.Architectural Design**  **2.Design use cases 3.Design classes**  **4.Design subsystem--comic‘s management subsystems** | **1.Design classes 2.Design subsystem-- comic management subsystems** | **1.Architectural Design**  **2.subsystem interfaces and design classes** |
| **Implementation** | **1.Architectural Implementation**  **2.Integrate System 3.Implement Subsystem ---comic‘s management subsystems 4.Implement Class** | **1.Architectural Implementation 2.Integrate System 3.Implement a Subsystem of comic management subsystems 4.Implement Class** | **1.Architectural Implementation---- 2.Integrate System 3.modify Subsystem interfaces 4.Implement Class** |
| **Test** | **1.Plan the test for the system management 2.Design the test and write test components**  **3.Implement the test** | **1.Plan the test for the system management 2.Design the test and write test components**  **3. Implement the test Make integration test plan.** | **1.Design the test 2.Implement the designed test 3.Perform the integration test** |

**Subgroup 3**

|  |  |  |  |
| --- | --- | --- | --- |
|  | People1 | 2 | 3 |
| **Capturing requirements** | 1. **Sum up the functional requirements of customer**   **Management 2.Find Actors and describe the actors**  **3.The glossary** | 1. **Catching detail functional requirements of customer**   **Management 2.Find Use Cases 3.Detail the glossary** | **1.Detail the use cases**  **2.Design the prototype of the system for different user interfaces** |
| **Analysis** | **1.Detail the Analysis Packages**  **2.Identify Common Special Requirements for customer management 3.Analyze Use Case 4.Analyze the Class** | **1.Modify the Analysis Packages**  **2.Identify Common Special Requirements 3.Analyze Use Case 4.Analyze the Class** | **1. Identify Common Special Requirements 2.Analyze Use 3.Analyze the Class** |
| **Design** | **1.Architectural Design**  **2.Design use cases 3.Design classes 4.Design subsystem—Customer management** | **1.Architectural Design**  **2.Design use cases 3.Design classes 4.Design subsystem:**  **BusinessRecord** | **1.Identifying subsystem interfaces and design classes 2.Design use cases 3.Design classes 4.Design subsystem interfaces** |
| **Implementation** | **1.Architectural Implementation 2.Integrate System 3.Implement Subsystem –customer management**  **4.Implement Class** | **1.Architectural Implementation 2.Integrate System**  **3.Implement Subsystem BusinessRecord subsystems 4.Implement Class** | **1.Architectural Implementation 2.Integrate System 3.modify Subsystem interfaces**  **4.Implement Class 5.system integration** |
| **Test** | **1.Plan the test for the system management 2.Design the test and write test components**  **3.Implement the test** | **1.Plan the test for the system management 2.Design the test and write test components**  **3. Implement the test Make integration test plan.** | **1.Design the test 2.Implement the designed test 3.Perform the integration test** |

**EBU5304 Software Engineering – Group Coursework**

**Weekly Report for**: Week 1<from 2014-04-22 to 2014-04-29 >

**Group Number**: <add number>

**Group Members**: <add names and student numbers>

**Group Leaders Name**: <add name and student number>

**Project Progress Summary**

**Group task**

**Subgroup 1:**

1. Do some background reading of the conference review system

2. Catching functional requirements of the System Management

3. Catching non-functional requirements

4. Plan for next week

**Subgroup 2:**

1. Do some background reading of the conference review system

2. Catching functional requirements of Paper Review Management

3. Design the interview and questionnaires

4. Do the interview and questionnaires and capture requirements from the result

5. Plan for next week

**Subgroup 3:**

1. Do some background reading of the conference review system

2. Catching functional requirements of Paper Voting Management

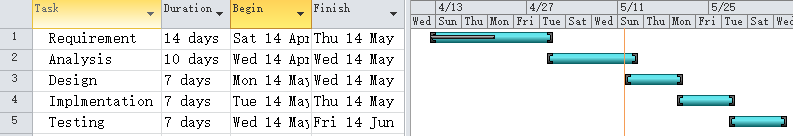
3. Find Actors and Use Cases

4. Plan for next week

**Milestone for week1**

* finished functional and nonfunctional requirements for the system
* description of actors and use cases.

**GANTT chart for week 1:**



**Week Summary**

Accomplishments:

1. Interviews and questionnaires to capture the requirements

2. Completing investigation

3. Functional and non-functional requirements of the system

4. Actors and general description of use cases

After the first meeting, we vote out the leader of the group and decided the three subgroups. Each group member finished the task on time with no delays. There are some problems when identifying the use cases but that are quickly solved after discussion among group members and suggestions from teacher.

**EBU5304 Software Engineering – Group Coursework**

**Weekly Report for**: Week 2<from 2014-04-29 to 2014-05-06 >

**Group Number**: <add number>

**Group Members**: <add names and student numbers>

**Group Leaders Name**: <add name and student number>

**Project Progress Summary**

**Group task**

**Subgroup 1:**

1. Detail the use case description

2. Build the use case model

3. Get familiar with all requirement and use cases ever found within one team

4. Discuss with other 2 subgroups for Identify Analysis Packages and Obvious Entity Classes

5. Identify Common Special Requirements within subgroup

6. Analyze Use Case ever found in the Requirement as use cases

**Subgroup 2:**

1. Detail the use case description

2. Draw the whole use case diagram

3. Get familiar with all requirement and use cases ever found within one team

4. Discuss with other 2 subgroups for Identify Analysis Packages and Obvious Entity Classes

5. Identify Common Special Requirements within subgroup

6. Analyze Use Case ever found in the Requirement as use cases

**Subgroup 3:**

1. Detail the use case description

2. The glossary of the system

3. Design the prototype of user interface

4. Get familiar with all requirement and use cases ever found within one team

5. Discuss with other 2 subgroups for Identify Analysis Packages and Obvious Entity Classes

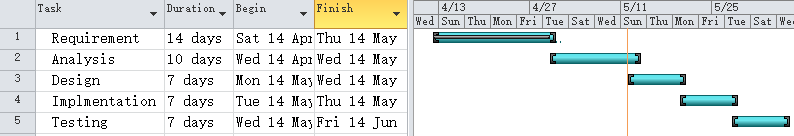
6. Identify Common Special Requirements within subgroup

7. Analyze Use Case ever found in the Requirement as use cases

**Milestone for week2**

* detailed dercription of use cases
* identified special requirments and packages
* finished requirement part description of actors and use cases.

**GANTT chart for week 2:**



**Week Summary**

Accomplishments:

1. Finished detail description of the use cases

2. Finished use case model and the whole use case diagram of the system

3. The prototypes of the system

4. Completed requirement capture

5. Special requirements for the system

6. Find out the classes for analysis part

After the meeting on Monday, we define the task for each member. For the work this week is not as less as the first week, there are some delay of the tasks like the prototypes of the system and the classes for analysis. There are some problems when descript the use cases in a much detail way. To solve some questions, we check some books and disuses together on Thursday‘s meeting. Then we finally come to the same opinion. From this week‘s work, I think the most important thing in the software engineering is the communication among different groups. Without a full understanding of each group, the project may be delay for misunderstanding.

**EBU5304 Software Engineering – Group Coursework**

**Weekly Report for**: Week 3<from 2014-05-07 to 2014-05-14>

**Group Number**: <add number>

**Group Members**: <add names and student numbers>

**Group Leaders Name**: <add name and student number>

**Project Progress Summary**

**Group task**

**Subgroup 1:**

1. Detail and modify the Analysis Packages and Obvious Entity Classes used in Administration management

2. Identify Common Special Requirements about Administration management within subgroup

3. Analyze Use Cases of Administration management ever found in the Requirement as use cases

4. Analyze the Class corresponding to Use Cases ever analyzed

5. Analyze a Package

6. Plan for next week

**Subgroup 2:**

1. Detail and modify the Analysis Packages and Obvious Entity Classes used in Paper review management

2. Identify Common Special Requirements about Paper review management within subgroup

3. Analyze Use Cases of Paper review management ever found in the Requirement as use cases

4. Analyze the Class corresponding to Use Cases ever analyzed

5. Analyze a Package

6. Plan for next week

**Subgroup 3:**

1. Detail and modify the Analysis Packages in Paper voting management

2. Identify Common Special Requirements about Paper voting management within subgroup

3. Analyze Use Cases of Paper voting management ever found in the Requirement as use cases

4. Analyze the Class corresponding to Use Cases ever analyzed

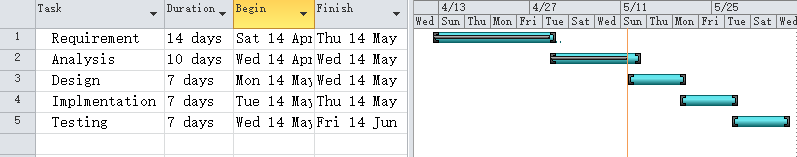
5. Analyze a Package

6. Plan for next week

**Milestone for week3**

* Finished prototypes of different user Interfaces

**GANTT chart for week 3:**



**Week Summary**

Accomplishments:

1. Detailed Analysis Packages

2. Common Special Requirements of the system

3. Analysis of the use cases and classes(boundary classes, control classes and entity classes)

During this week, we mainly focus on the analysis part of the project. The main problem we meet is in the analysis package part. We find it confuse to identify package especially service packages. After reading some books and ask about it on the message board, we finally finish our analysis. All works are done on time.

**EBU5304 Software Engineering – Group Coursework**

**Weekly Report for**: Week 4<from 2014-05-15 to 2014-05-22>

**Group Number**: <add number>

**Group Members**: <add names and student numbers>

**Group Leaders Name**: <add name and student number>

**Project Progress Summary**

**Group task**

**Subgroup 1:**

1. Architectural Design of user management

2. Design use cases in user management

3. Design classes related to the use cases with subgroup

4. Design subsystem user management subsystem, User account management subsystem

5. Plan for next week

**Subgroup 2:**

1. Architectural Design of Comic management

2. Design use cases in Business Record management

3. Design classes related to the use cases with subgroup

4. Design subsystem Card management subsystem

5. Plan for next week

**Subgroup 3:**

1. Architectural Design of Comic management

2. Design use cases in Business Record management

3. Design classes related to the use cases with subgroup

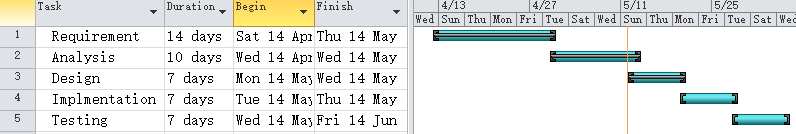
4. Design subsystem Card management subsystem

5. Plan for next week

**Milestone for week4**

* Finished design part

**GANTT chart for week 4:**



**Week Summary**

Accomplishments:

1. Designed use cases and classes

2. Designed subsystems

This week we mainly focus on the design part of the project. The problem we meet this week is major about the diagram of the design part. There is another kind of class diagram in the design part which is different from the one in the analysis part. And we at first not care that there is a new class in the diagram of the design part: the processing. The carelessness delays our plan for we have to redraw the diagram.

**EBU5304 Software Engineering – Group Coursework**

**Weekly Report for**: Week 5<from 2014-05-23 to 2014-05-29>

**Group Number**: <add number>

**Group Members**: <add names and student numbers>

**Group Leaders Name**: <add name and student number>

**Project Progress Summary**

**Group task**

**Subgroup 1:**

1. Modify design

2. Architectural Implementation

3. Integrate System

4. Implement Subsystems: user management subsystem, User account management subsystem

5. Implement Classes related to use cases login, initialization, maintain Comic and maintain Customer.

6. Plan for next week

**Subgroup 2:**

1. Modify design

2. Architectural Implementation

3. Integrate System

4. Implement Subsystems: Comic management subsystem, Customer subsystem.

5. Implement Classes related to use cases buy comic and rent comic

6. Plan for next week

**Subgroup 3:**

1. Modify design

2. Architectural Implementation

3. Integrate System

4. Implement Subsystems: Card management subsystem, Reviewer‘s management subsystem

5. Implement Classes related to use cases Create new card

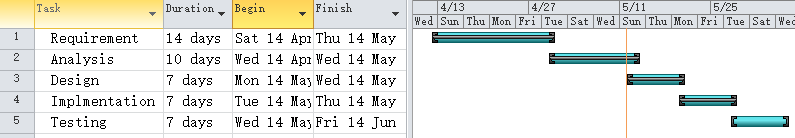
6. System integration

7. Plan for next week

**Milestone for week5**

* finished implementation of the system in JAVA

**GANTT chart for week 5:**



**Week Summary**

Accomplishments:

1. Implementation of the system in JAVA

2. Integrated system of several subsystems

This week we mainly focus on implementing the system in JAVA. For we have to review java when we really do coding. At first, the project move on very slow and we have to discuss a lot to find the solution to implement some functions. But late on, when we became familiar with coding with java, the project goes on quite well. And we finish this part on time with some simply test start.

**EBU5304 Software Engineering – Group Coursework**

**Weekly Report for**: Week 6<from 2014-05-30 to 2014-06-06>

**Group Number**: <add number>

**Group Members**: <add names and student numbers>

**Group Leaders Name**: <add name and student number>

**Project Progress Summary**

**Group task**

**Subgroup 1:**

1. Plan the test for the project

2. Design the test mainly for system management part of the system, writing test components

3. Implement the designed test and record the result

4. Find out the bugs, record the defects, discuss it with group members and solve them

5. Perform the integration test

**Subgroup 2:**

1. Plan the test for the project

2. Design the test mainly for paper review part of the system, writing test components

3. Implement the designed test and record the result

4. Find out the bugs, record the defects, discuss it with group members and solve them

5. Perform the system test

**Subgroup 3:**

1. Plan the test for the project

2. Design the test mainly for paper voting part of the system, writing test components

3. Implement the designed test and record the result

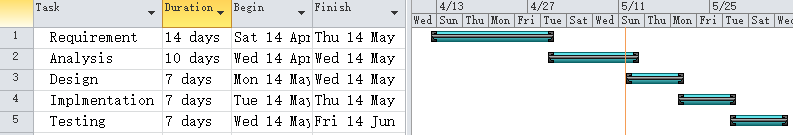
4. Find out the bugs, record the defects, discuss it with group members and solve them

5. Evaluate the test and refine the system

**Milestone for week6**

* Finished test of the system

**GANTT chart for week 6:**



**Week Summary**

Accomplishments:

1. Test plan for the system

2. Test implementation and solve several bugs which makes the system more robust

3. Refined system

4. Test report for the system

During the test, the most difficult thing is to find out bugs of the system and find ways to solve that problem. That always takes a long time for each bug and makes the test implementation delay in 2days. We also find out some problems we never think of before during the test. The system becomes more robust after the refinement.

**Java**

**Code**

* **DataFile.java**

**package** utils.data;

**public** **class** DataFile {

**public** **static** **final** **char** *SEPARATOR* = '\t';

}

* Persistence.java

package utils.data;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.PrintStream;

public abstract class Persistence {

protected String filePath;

protected PrintStream out;

protected BufferedReader in;

public Persistence() {

setFile();

}

public void beginPersist() throws FileNotFoundException {

out = new PrintStream(new FileOutputStream(filePath));

}

protected void persistNext(String record) {

out.println(record);

}

public void endPersist(){

if(out != null) {

out.close();

}

}

public void beginLoad() throws IOException {

try {

in = new BufferedReader(new FileReader(filePath));

} catch (FileNotFoundException e) {

File file = new File(filePath);

if(!file.exists()){

file.createNewFile();

}

in = new BufferedReader(new FileReader(filePath));

}

}

protected String loadNext() throws IOException {

if (in.ready()) {

String record = in.readLine();

return record;

}

return null;

}

public void endLoad() {

if(in != null) {

try {

in.close();

} catch (IOException e) {

} finally {

in = null;

}

}

}

public abstract void setFile();

public abstract void persist() throws Exception;

public abstract void load() throws Exception;

public String getFilePath() {

return filePath;

}

public void setFilePath(String filePath) {

this.filePath = filePath;

}

}

* AddBookDialog.java

package utils.data;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.PrintStream;

public abstract class Persistence {

protected String filePath;

protected PrintStream out;

protected BufferedReader in;

public Persistence() {

setFile();

}

public void beginPersist() throws FileNotFoundException {

out = new PrintStream(new FileOutputStream(filePath));

}

protected void persistNext(String record) {

out.println(record);

}

public void endPersist(){

if(out != null) {

out.close();

}

}

public void beginLoad() throws IOException {

try {

in = new BufferedReader(new FileReader(filePath));

} catch (FileNotFoundException e) {

File file = new File(filePath);

if(!file.exists()){

file.createNewFile();

}

in = new BufferedReader(new FileReader(filePath));

}

}

protected String loadNext() throws IOException {

if (in.ready()) {

String record = in.readLine();

return record;

}

return null;

}

public void endLoad() {

if(in != null) {

try {

in.close();

} catch (IOException e) {

} finally {

in = null;

}

}

}

public abstract void setFile();

public abstract void persist() throws Exception;

public abstract void load() throws Exception;

public String getFilePath() {

return filePath;

}

public void setFilePath(String filePath) {

this.filePath = filePath;

}

}

* DateFormatter.java

package ui;

import java.text.SimpleDateFormat;

import java.util.Date;

public class DateFormatter {

public static String format(Date date){

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

return df.format(date);

}

}

* DoubleLimitedDocument.java

package ui;

import javax.swing.text.AttributeSet;

import javax.swing.text.BadLocationException;

import javax.swing.text.PlainDocument;

public class DoubleLimitedDocument extends PlainDocument {

private static final long serialVersionUID = 2804111985157372467L;

private int maxLength = -1;

private double minValue = Double.MIN\_VALUE;

private double maxValue = Double.MAX\_VALUE;

public DoubleLimitedDocument() {

}

public DoubleLimitedDocument(int maxLength) {

this.maxLength = maxLength;

}

public DoubleLimitedDocument(int maxLength, double min\_value, double max\_value){

this.maxLength = maxLength;

this.minValue = min\_value;

this.maxValue = max\_value;

}

public void insertString(int offset, String str, AttributeSet attrSet)

throws BadLocationException {

if (str == null) {

return;

}

char[] charVal = str.toCharArray();

String strOldValue = getText(0, getLength());

char[] tmp = strOldValue.toCharArray();

String str\_inserted = strOldValue.substring(0, offset) + str +

strOldValue.substring(offset, strOldValue.length());

double result;

try {

result = Double.parseDouble(str\_inserted);

} catch (NumberFormatException e) {

return;

}

if(result > maxValue || result < minValue){

return;

}

if (maxLength != -1 && (tmp.length + charVal.length > maxLength)) {

return;

}

super.insertString(offset, str, attrSet);

}

public static void main(String[] args) {

}

}

* EditCardDialog.java

package ui;

import java.awt.Font;

import java.awt.Toolkit;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JPanel;

import javax.swing.SwingUtilities;

import javax.swing.UIManager;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.border.TitledBorder;

import system.\*;

import java.awt.GridBagLayout;

import java.awt.GridBagConstraints;

import java.awt.Insets;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

import javax.swing.ListSelectionModel;

import javax.swing.JCheckBox;

import javax.swing.SwingConstants;

public class EditCardDialog extends JDialog {

private static final long serialVersionUID = 1195788290600250698L;

private EditCardDialog dialog;

private JTextField textFieldID;

private JTextField textFieldCustomer;

private JTextField textFieldBooks;

private JTextField textFieldMoney;

private JTextField textFieldCreatedTime;

private MainWindow window;

private Card instance;

private JTable table;

private JTextField textFieldAmount;

private JLabel lblAmount;

private JCheckBox chckbxRecharge;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

try {

EditCardDialog dialog = new EditCardDialog(null, null);

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* Create the dialog.

\*/

@SuppressWarnings("serial")

public EditCardDialog(MainWindow jf, Card card) {

window = jf;

dialog = this;

this.instance = card;

setTitle("Card Information");

setResizable(false);

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (Exception e1) {

}

SwingUtilities.updateComponentTreeUI(this);

setIconImage(Toolkit.getDefaultToolkit()

.getImage(

EditCardDialog.class

.getResource("/resource/mobile-2-icon.png")));

setBounds(100, 100, 492, 355);

setBounds(

(Toolkit.getDefaultToolkit().getScreenSize().width - this

.getWidth()) / 2,

(Toolkit.getDefaultToolkit().getScreenSize().height - this

.getHeight()) / 2, getWidth(), getHeight());

getContentPane().setLayout(null);

JPanel panel = new JPanel();

panel.setBorder(new TitledBorder(null, "Card Details",

TitledBorder.LEADING, TitledBorder.TOP, new Font("微软雅黑",

Font.PLAIN, 12), null));

panel.setBounds(10, 10, 224, 188);

getContentPane().add(panel);

panel.setLayout(null);

JPanel panel\_1 = new JPanel();

panel\_1.setBounds(13, 25, 195, 152);

panel.add(panel\_1);

GridBagLayout gbl\_panel\_1 = new GridBagLayout();

gbl\_panel\_1.columnWidths = new int[] { 53, 0, 0 };

gbl\_panel\_1.rowHeights = new int[] {5, 17, 17, 17, 17};

gbl\_panel\_1.columnWeights = new double[] { 0.0, 1.0, Double.MIN\_VALUE };

gbl\_panel\_1.rowWeights = new double[] { 0.0, 0.0, 0.0, 0.0, 0.0 };

panel\_1.setLayout(gbl\_panel\_1);

JLabel lblNewLabel = new JLabel("Card ID");

GridBagConstraints gbc\_lblNewLabel = new GridBagConstraints();

gbc\_lblNewLabel.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel.gridx = 0;

gbc\_lblNewLabel.gridy = 0;

panel\_1.add(lblNewLabel, gbc\_lblNewLabel);

lblNewLabel.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldID = new JTextField();

textFieldID.setEditable(false);

textFieldID.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldID = new GridBagConstraints();

gbc\_textFieldID.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldID.fill = GridBagConstraints.BOTH;

gbc\_textFieldID.gridx = 1;

gbc\_textFieldID.gridy = 0;

panel\_1.add(textFieldID, gbc\_textFieldID);

textFieldID.setColumns(10);

JLabel lblNewLabel\_1 = new JLabel("Belongs to");

GridBagConstraints gbc\_lblNewLabel\_1 = new GridBagConstraints();

gbc\_lblNewLabel\_1.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_1.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel\_1.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_1.gridx = 0;

gbc\_lblNewLabel\_1.gridy = 1;

panel\_1.add(lblNewLabel\_1, gbc\_lblNewLabel\_1);

lblNewLabel\_1.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldCustomer = new JTextField();

textFieldCustomer.setEditable(false);

textFieldCustomer.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldCustomer = new GridBagConstraints();

gbc\_textFieldCustomer.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldCustomer.fill = GridBagConstraints.HORIZONTAL;

gbc\_textFieldCustomer.gridx = 1;

gbc\_textFieldCustomer.gridy = 1;

panel\_1.add(textFieldCustomer, gbc\_textFieldCustomer);

textFieldCustomer.setColumns(10);

JLabel lblNewLabel\_3 = new JLabel("Money");

GridBagConstraints gbc\_lblNewLabel\_3 = new GridBagConstraints();

gbc\_lblNewLabel\_3.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_3.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel\_3.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_3.gridx = 0;

gbc\_lblNewLabel\_3.gridy = 2;

panel\_1.add(lblNewLabel\_3, gbc\_lblNewLabel\_3);

lblNewLabel\_3.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldMoney = new JTextField();

textFieldMoney.setEditable(false);

textFieldMoney.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldMoney = new GridBagConstraints();

gbc\_textFieldMoney.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldMoney.fill = GridBagConstraints.HORIZONTAL;

gbc\_textFieldMoney.gridx = 1;

gbc\_textFieldMoney.gridy = 2;

panel\_1.add(textFieldMoney, gbc\_textFieldMoney);

textFieldMoney.setColumns(10);

JLabel lblNewLabel\_2 = new JLabel("Books");

GridBagConstraints gbc\_lblNewLabel\_2 = new GridBagConstraints();

gbc\_lblNewLabel\_2.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_2.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel\_2.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_2.gridx = 0;

gbc\_lblNewLabel\_2.gridy = 3;

panel\_1.add(lblNewLabel\_2, gbc\_lblNewLabel\_2);

lblNewLabel\_2.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldBooks = new JTextField();

textFieldBooks.setEditable(false);

textFieldBooks.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldBooks = new GridBagConstraints();

gbc\_textFieldBooks.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldBooks.fill = GridBagConstraints.HORIZONTAL;

gbc\_textFieldBooks.gridx = 1;

gbc\_textFieldBooks.gridy = 3;

panel\_1.add(textFieldBooks, gbc\_textFieldBooks);

textFieldBooks.setColumns(10);

JLabel lblNewLabel\_5 = new JLabel("Created");

lblNewLabel\_5.setLocation(20, 118);

GridBagConstraints gbc\_lblNewLabel\_5 = new GridBagConstraints();

gbc\_lblNewLabel\_5.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_5.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel\_5.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_5.gridx = 0;

gbc\_lblNewLabel\_5.gridy = 4;

panel\_1.add(lblNewLabel\_5, gbc\_lblNewLabel\_5);

lblNewLabel\_5.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldCreatedTime = new JTextField();

textFieldCreatedTime.setEditable(false);

textFieldCreatedTime.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldCreatedTime = new GridBagConstraints();

gbc\_textFieldCreatedTime.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldCreatedTime.fill = GridBagConstraints.HORIZONTAL;

gbc\_textFieldCreatedTime.gridx = 1;

gbc\_textFieldCreatedTime.gridy = 4;

panel\_1.add(textFieldCreatedTime, gbc\_textFieldCreatedTime);

textFieldCreatedTime.setColumns(10);

JButton btnNewButton = new JButton("OK");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(checkInput()){

dispose();

}

}

});

btnNewButton.setBounds(94, 293, 93, 23);

getContentPane().add(btnNewButton);

btnNewButton.setFont(new Font("微软雅黑", Font.PLAIN, 12));

JButton btnNewButton\_1 = new JButton("Cancel");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose();

}

});

btnNewButton\_1.setBounds(283, 293, 93, 23);

getContentPane().add(btnNewButton\_1);

btnNewButton\_1.setFont(new Font("微软雅黑", Font.PLAIN, 12));

JPanel panel\_2 = new JPanel();

panel\_2.setLayout(null);

panel\_2.setBorder(new TitledBorder(null, "Records",

TitledBorder.LEADING, TitledBorder.TOP, new Font("微软雅黑",

Font.PLAIN, 12), null));

panel\_2.setBounds(244, 10, 232, 273);

getContentPane().add(panel\_2);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(10, 24, 212, 239);

panel\_2.add(scrollPane);

table = new JTable();

table.setAutoResizeMode(JTable.AUTO\_RESIZE\_OFF);

table.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

table.setModel(new DefaultTableModel(new Object[][] { { null, null,

null }, }, new String[] { "Time", "Type", "Amount" }) {

boolean[] columnEditables = new boolean[] { false, false, false };

public boolean isCellEditable(int row, int column) {

return columnEditables[column];

}

});

table.setFont(new Font("微软雅黑", Font.PLAIN, 12));

table.getTableHeader().setFont(new Font("微软雅黑", Font.PLAIN, 12));

scrollPane.setViewportView(table);

chckbxRecharge = new JCheckBox("Recharge");

chckbxRecharge.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(chckbxRecharge.isSelected()){

lblAmount.setEnabled(true);

textFieldAmount.setEnabled(true);

}else{

lblAmount.setEnabled(false);

textFieldAmount.setEnabled(false);

}

}

});

chckbxRecharge.setFont(new Font("微软雅黑", Font.PLAIN, 12));

chckbxRecharge.setBounds(20, 208, 87, 23);

getContentPane().add(chckbxRecharge);

JPanel panel\_3 = new JPanel();

panel\_3.setLayout(null);

panel\_3.setFont(new Font("微软雅黑", Font.PLAIN, 12));

panel\_3.setBorder(new TitledBorder(UIManager.getBorder("TitledBorder.border"), "", TitledBorder.LEADING, TitledBorder.TOP, null, null));

panel\_3.setBounds(10, 218, 224, 62);

getContentPane().add(panel\_3);

textFieldAmount = new JTextField();

textFieldAmount.setText("0");

textFieldAmount.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldAmount.setEnabled(false);

textFieldAmount.setColumns(10);

textFieldAmount.setBounds(82, 20, 126, 23);

panel\_3.add(textFieldAmount);

textFieldAmount.setDocument(new DoubleLimitedDocument(10, 0.0, 999999.9));

lblAmount = new JLabel("Amount");

lblAmount.setHorizontalAlignment(SwingConstants.RIGHT);

lblAmount.setFont(new Font("微软雅黑", Font.PLAIN, 12));

lblAmount.setEnabled(false);

lblAmount.setBounds(20, 20, 56, 23);

panel\_3.add(lblAmount);

if (instance != null) {

textFieldID.setText(instance.getId());

if (instance.getCustomer() != null) {

textFieldCustomer.setText(instance.getCustomer().getId());

}

// textFieldBooks.setText(instance.getPhone());

textFieldMoney.setText(instance.getMoney().toString());

textFieldCreatedTime.setText(DateFormatter.format(instance

.getCreatedTime()));

}

}

protected boolean checkInput() {

if(chckbxRecharge.isSelected()){

double amount = 0;

try{

amount = Double.parseDouble(textFieldAmount.getText());

}catch(NumberFormatException e){

}

if(instance != null && amount > 0){

instance.addMoney(amount);

}

}

if(window != null){

window.fillCardTable(CardManager.getInstance().getCardList());

}

return true;

}

}

* EditCustomerDialog.java

package ui;

import java.awt.Font;

import java.awt.Toolkit;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.SwingUtilities;

import javax.swing.UIManager;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.border.TitledBorder;

import system.\*;

import java.awt.GridBagLayout;

import java.awt.GridBagConstraints;

import java.awt.Insets;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.util.ArrayList;

import java.util.Iterator;

import javax.swing.JScrollPane;

import javax.swing.SwingConstants;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

import javax.swing.ListSelectionModel;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class EditCustomerDialog extends JDialog {

/\*\*

\*

\*/

private static final long serialVersionUID = -6641536403824105277L;

private EditCustomerDialog dialog;

private JTextField textFieldID;

private JTextField textFieldName;

private JTextField textFieldPhone;

private JTextField textFieldTime;

private JTextField textFieldCardID;

private MainWindow window;

private Customer instance;

private ArrayList<Card> cardArrayList;

private JTable cardTable;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

try {

EditCustomerDialog dialog = new EditCustomerDialog(null, null);

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* Create the dialog.

\*/

@SuppressWarnings("serial")

public EditCustomerDialog(MainWindow jf, Customer customer) {

window = jf;

dialog = this;

this.instance = customer;

setTitle("Customer Information");

setResizable(false);

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (Exception e1) {

}

SwingUtilities.updateComponentTreeUI(this);

setIconImage(Toolkit.getDefaultToolkit().getImage(EditCustomerDialog.class.getResource("/resource/contact-icon.png")));

setBounds(100, 100, 594, 302);

setBounds((Toolkit.getDefaultToolkit().getScreenSize().width-this.getWidth())/2, (Toolkit.getDefaultToolkit().getScreenSize().height-this.getHeight())/2, getWidth(), getHeight());

getContentPane().setLayout(null);

JPanel panel = new JPanel();

panel.setBorder(new TitledBorder(null, "Customer Details", TitledBorder.LEADING, TitledBorder.TOP, new Font("微软雅黑", Font.PLAIN, 12), null));

panel.setBounds(10, 10, 224, 214);

getContentPane().add(panel);

panel.setLayout(null);

JPanel panel\_1 = new JPanel();

panel\_1.setBounds(13, 25, 195, 179);

panel.add(panel\_1);

GridBagLayout gbl\_panel\_1 = new GridBagLayout();

gbl\_panel\_1.columnWidths = new int[]{53, 0, 0};

gbl\_panel\_1.rowHeights = new int[]{17, 17, 17, 17, 17, 17, 0, 0};

gbl\_panel\_1.columnWeights = new double[]{0.0, 1.0, Double.MIN\_VALUE};

gbl\_panel\_1.rowWeights = new double[]{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, Double.MIN\_VALUE};

panel\_1.setLayout(gbl\_panel\_1);

JLabel lblNewLabel = new JLabel("ID");

GridBagConstraints gbc\_lblNewLabel = new GridBagConstraints();

gbc\_lblNewLabel.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel.gridx = 0;

gbc\_lblNewLabel.gridy = 0;

panel\_1.add(lblNewLabel, gbc\_lblNewLabel);

lblNewLabel.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldID = new JTextField();

textFieldID.setEditable(false);

textFieldID.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldID = new GridBagConstraints();

gbc\_textFieldID.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldID.fill = GridBagConstraints.BOTH;

gbc\_textFieldID.gridx = 1;

gbc\_textFieldID.gridy = 0;

panel\_1.add(textFieldID, gbc\_textFieldID);

textFieldID.setColumns(10);

JLabel lblNewLabel\_1 = new JLabel("Name");

GridBagConstraints gbc\_lblNewLabel\_1 = new GridBagConstraints();

gbc\_lblNewLabel\_1.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_1.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel\_1.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_1.gridx = 0;

gbc\_lblNewLabel\_1.gridy = 1;

panel\_1.add(lblNewLabel\_1, gbc\_lblNewLabel\_1);

lblNewLabel\_1.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldName = new JTextField();

textFieldName.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldName = new GridBagConstraints();

gbc\_textFieldName.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldName.fill = GridBagConstraints.HORIZONTAL;

gbc\_textFieldName.gridx = 1;

gbc\_textFieldName.gridy = 1;

panel\_1.add(textFieldName, gbc\_textFieldName);

textFieldName.setColumns(10);

JLabel lblNewLabel\_2 = new JLabel("Phone");

GridBagConstraints gbc\_lblNewLabel\_2 = new GridBagConstraints();

gbc\_lblNewLabel\_2.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_2.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel\_2.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_2.gridx = 0;

gbc\_lblNewLabel\_2.gridy = 2;

panel\_1.add(lblNewLabel\_2, gbc\_lblNewLabel\_2);

lblNewLabel\_2.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldPhone = new JTextField();

textFieldPhone.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldPhone = new GridBagConstraints();

gbc\_textFieldPhone.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldPhone.fill = GridBagConstraints.HORIZONTAL;

gbc\_textFieldPhone.gridx = 1;

gbc\_textFieldPhone.gridy = 2;

panel\_1.add(textFieldPhone, gbc\_textFieldPhone);

textFieldPhone.setColumns(10);

textFieldPhone.setDocument(new LimitedDocument(15, "1234567890-+"));

JLabel lblNewLabel\_3 = new JLabel("Join Date");

GridBagConstraints gbc\_lblNewLabel\_3 = new GridBagConstraints();

gbc\_lblNewLabel\_3.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_3.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel\_3.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_3.gridx = 0;

gbc\_lblNewLabel\_3.gridy = 3;

panel\_1.add(lblNewLabel\_3, gbc\_lblNewLabel\_3);

lblNewLabel\_3.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldTime = new JTextField();

textFieldTime.setEditable(false);

textFieldTime.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldTime = new GridBagConstraints();

gbc\_textFieldTime.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldTime.fill = GridBagConstraints.HORIZONTAL;

gbc\_textFieldTime.gridx = 1;

gbc\_textFieldTime.gridy = 3;

panel\_1.add(textFieldTime, gbc\_textFieldTime);

textFieldTime.setColumns(10);

JLabel lblNewLabel\_5 = new JLabel("Card ID");

GridBagConstraints gbc\_lblNewLabel\_5 = new GridBagConstraints();

gbc\_lblNewLabel\_5.anchor = GridBagConstraints.EAST;

gbc\_lblNewLabel\_5.fill = GridBagConstraints.VERTICAL;

gbc\_lblNewLabel\_5.insets = new Insets(0, 0, 5, 5);

gbc\_lblNewLabel\_5.gridx = 0;

gbc\_lblNewLabel\_5.gridy = 4;

panel\_1.add(lblNewLabel\_5, gbc\_lblNewLabel\_5);

lblNewLabel\_5.setFont(new Font("微软雅黑", Font.PLAIN, 12));

textFieldCardID = new JTextField();

textFieldCardID.setEditable(false);

textFieldCardID.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_textFieldCardID = new GridBagConstraints();

gbc\_textFieldCardID.insets = new Insets(0, 0, 5, 0);

gbc\_textFieldCardID.fill = GridBagConstraints.HORIZONTAL;

gbc\_textFieldCardID.gridx = 1;

gbc\_textFieldCardID.gridy = 4;

panel\_1.add(textFieldCardID, gbc\_textFieldCardID);

textFieldCardID.setColumns(10);

//textFieldGrade.setDocument(new LimitedDocument(10, "0123456789."));

JLabel lblNewLabel\_4 = new JLabel("Double Click a card on the right");

lblNewLabel\_4.setVerticalAlignment(SwingConstants.BOTTOM);

lblNewLabel\_4.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_lblNewLabel\_4 = new GridBagConstraints();

gbc\_lblNewLabel\_4.insets = new Insets(0, 0, 5, 0);

gbc\_lblNewLabel\_4.anchor = GridBagConstraints.SOUTHEAST;

gbc\_lblNewLabel\_4.gridwidth = 2;

gbc\_lblNewLabel\_4.gridx = 0;

gbc\_lblNewLabel\_4.gridy = 5;

panel\_1.add(lblNewLabel\_4, gbc\_lblNewLabel\_4);

JLabel lblNewLabel\_6 = new JLabel(" list to choose it.");

lblNewLabel\_6.setVerticalAlignment(SwingConstants.BOTTOM);

lblNewLabel\_6.setFont(new Font("微软雅黑", Font.PLAIN, 12));

GridBagConstraints gbc\_lblNewLabel\_6 = new GridBagConstraints();

gbc\_lblNewLabel\_6.gridwidth = 2;

gbc\_lblNewLabel\_6.anchor = GridBagConstraints.WEST;

gbc\_lblNewLabel\_6.gridx = 0;

gbc\_lblNewLabel\_6.gridy = 6;

panel\_1.add(lblNewLabel\_6, gbc\_lblNewLabel\_6);

JButton btnNewButton = new JButton("OK");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(checkInput()){

dispose();

}

}

});

btnNewButton.setBounds(77, 234, 93, 23);

getContentPane().add(btnNewButton);

btnNewButton.setFont(new Font("微软雅黑", Font.PLAIN, 12));

JButton btnNewButton\_1 = new JButton("Cancel");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose();

}

});

btnNewButton\_1.setBounds(247, 234, 93, 23);

getContentPane().add(btnNewButton\_1);

btnNewButton\_1.setFont(new Font("微软雅黑", Font.PLAIN, 12));

JPanel panel\_2 = new JPanel();

panel\_2.setBorder(new TitledBorder(null, "Card List", TitledBorder.LEADING, TitledBorder.TOP, new Font("微软雅黑", Font.PLAIN, 12), null));

panel\_2.setBounds(244, 10, 334, 214);

getContentPane().add(panel\_2);

panel\_2.setLayout(null);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(10, 24, 314, 180);

panel\_2.add(scrollPane);

cardTable = new JTable();

cardTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

if(e.getClickCount() == 2){

if(instance != null && cardTable.getSelectedRow() >= 0){

Card selectedCard = cardArrayList.get(cardTable.getSelectedRow());

instance.setCard(selectedCard);

if(selectedCard != null){

selectedCard.setCustomer(instance);

}

refresh();

}

}

}

});

cardTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

cardTable.setAutoResizeMode(JTable.AUTO\_RESIZE\_OFF);

cardTable.setModel(new DefaultTableModel(

new Object[][] {

{null, null, null, null, null},

},

new String[] {

"Card ID", "Belongs to", "Balance", "Rent Book", "Created Time"

}

) {

boolean[] columnEditables = new boolean[] {

false, false, false, false, false

};

public boolean isCellEditable(int row, int column) {

return columnEditables[column];

}

});

cardTable.getColumnModel().getColumn(0).setPreferredWidth(80);

cardTable.getColumnModel().getColumn(1).setPreferredWidth(100);

cardTable.getColumnModel().getColumn(2).setPreferredWidth(69);

cardTable.getColumnModel().getColumn(3).setPreferredWidth(122);

cardTable.getColumnModel().getColumn(4).setPreferredWidth(157);

scrollPane.setViewportView(cardTable);

cardTable.setFont(new Font("微软雅黑", Font.PLAIN, 12));

cardTable.getTableHeader().setFont(new Font("微软雅黑", Font.PLAIN, 12));

scrollPane.setViewportView(cardTable);

JButton btnNewCard = new JButton("New Card");

btnNewCard.setFont(new Font("微软雅黑", Font.PLAIN, 12));

btnNewCard.setBounds(417, 234, 93, 23);

getContentPane().add(btnNewCard);

// if(instance != null){

// textFieldID.setText(instance.getId());

// textFieldName.setText(instance.getName());

// textFieldPhone.setText(instance.getPhone());

// textFieldTime.setText(DateFormatter.format(instance.getRegisterTime()));

// textFieldCardID.setText((instance.getCard() == null) ? "" : instance.getCard().getId());

// }

//

// fillCardTable(CardManager.getInstance().getCardList());

refresh();

}

protected void refresh(){

if(instance != null){

textFieldID.setText(instance.getId());

textFieldName.setText(instance.getName());

textFieldPhone.setText(instance.getPhone());

textFieldTime.setText(DateFormatter.format(instance.getRegisterTime()));

textFieldCardID.setText((instance.getCard() == null) ? "[None]" : instance.getCard().getId());

}

fillCardTable(CardManager.getInstance().getCardList());

}

protected boolean checkInput() {

String message = null;

if(this.textFieldName.getText().length() == 0){

message = "Please input Name";

}else if(this.textFieldPhone.getText().length() == 0){

message = "Please input Phone";

}

if(message != null){

JOptionPane.showOptionDialog(dialog, message, "Notice",

JOptionPane.DEFAULT\_OPTION, JOptionPane.INFORMATION\_MESSAGE,

null, new Object[] { "OK" } , "OK");

return false;

}

if(instance == null){

instance = CustomerManager.getInstance().create(textFieldName.getText(), textFieldPhone.getText());

instance.setCard(CardManager.getInstance().get(textFieldCardID.getText()));

}else{

instance.setName(textFieldName.getText());

instance.setPhone(textFieldPhone.getText());

instance.setCard(CardManager.getInstance().get(textFieldCardID.getText()));

}

if(window != null){

window.fillCustomerTableModel(CustomerManager.getInstance().getCustomerList());

}

return true;

}

public void fillCardTable(ArrayList<Card> list){

cardArrayList = list;

JTable table = cardTable;

DefaultTableModel tableModel =(DefaultTableModel) table.getModel();

tableModel.setRowCount(0);

if(list == null){

return;

}

Iterator<Card> iter = list.iterator();

while(iter.hasNext()){

Card a = iter.next();

String[] arr = new String[] {

a.getId(),

(a.getCustomer() == null) ? "" : a.getCustomer().getName(),

a.getMoney().toString(),

//TODO

//BusinessRecordManager.getInstance().getRentBook,

"[TODO]",

DateFormatter.format(a.getCreatedTime())

};

tableModel.addRow(arr);

}

table.validate();

table.repaint();

}

}

* LimitedDocument.java

package ui;

import javax.swing.text.AttributeSet;

import javax.swing.text.BadLocationException;

import javax.swing.text.PlainDocument;

public class LimitedDocument extends PlainDocument {

private static final long serialVersionUID = -4388652889449772549L;

private int maxLength = -1;// 允许的最大长度

private String allowCharAsString = null;// 允许的字符串格式（0123456789）

public LimitedDocument() {

super();

}

public LimitedDocument(int maxLength) {

super();

this.maxLength = maxLength;

}

public LimitedDocument(int maxLength, String chars){

this(maxLength);

this.setAllowChar(chars);

}

public void insertString(int offset, String str, AttributeSet attrSet)

throws BadLocationException {

if (str == null) {

return;

}

if (allowCharAsString != null && str.length() == 1) {

if (allowCharAsString.indexOf(str) == -1) {

return;// 不是所要求的字符格式，就直接返回，不进行下面的添加

}

}

char[] charVal = str.toCharArray();

String strOldValue = getText(0, getLength());

char[] tmp = strOldValue.toCharArray();

if (maxLength != -1 && (tmp.length + charVal.length > maxLength)) {

//Toolkit.getDefaultToolkit().beep();// 发出一个警告声

return;// 长度大于指定的长度maxLength，也直接返回，不进行下面的添加

}

super.insertString(offset, str, attrSet);

}

public void setAllowChar(String str) {

allowCharAsString = str;

}

}

* LoginWindow.java

package ui;

import java.awt.EventQueue;

import java.awt.Toolkit;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JPanel;

import javax.swing.SwingUtilities;

import javax.swing.UIManager;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

import javax.swing.JLabel;

import javax.swing.border.TitledBorder;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.Font;

import javax.swing.SwingConstants;

import javax.swing.ImageIcon;

import system.Store;

import system.Store.LoginResult;

public class LoginWindow extends JDialog {

private static final long serialVersionUID = 3110240126605474382L;

private JPasswordField password;

private JTextField account;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

try {

LoginWindow dialog = new LoginWindow();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* Create the dialog.

\*/

public LoginWindow() {

getContentPane().setFont(new Font("微软雅黑", Font.PLAIN, 12));

setTitle("Comic Book System - Log in");

setResizable(false);

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (Exception e1) {

}

SwingUtilities.updateComponentTreeUI(this);

setIconImage(Toolkit.getDefaultToolkit().getImage(LoginWindow.class.getResource("/resource/contact-icon.png")));

setBounds(100, 100, 468, 258);

setBounds((Toolkit.getDefaultToolkit().getScreenSize().width-this.getWidth())/2, (Toolkit.getDefaultToolkit().getScreenSize().height-this.getHeight())/2, getWidth(), getHeight());

getContentPane().setLayout(null);

JPanel panel = new JPanel();

panel.setBorder(new TitledBorder(null, "Login", TitledBorder.LEADING, TitledBorder.TOP, new Font("微软雅黑", Font.PLAIN, 12), null));

panel.setBounds(186, 10, 256, 199);

getContentPane().add(panel);

panel.setLayout(null);

JLabel lblNewLabel\_1 = new JLabel("Password");

lblNewLabel\_1.setFont(new Font("微软雅黑", Font.PLAIN, 12));

lblNewLabel\_1.setBounds(24, 77, 68, 15);

panel.add(lblNewLabel\_1);

JLabel lblNewLabel = new JLabel("Account");

lblNewLabel.setFont(new Font("微软雅黑", Font.PLAIN, 12));

lblNewLabel.setBounds(24, 46, 54, 15);

panel.add(lblNewLabel);

account = new JTextField();

account.setFont(new Font("微软雅黑", Font.PLAIN, 12));

account.setBounds(88, 43, 143, 21);

panel.add(account);

account.setColumns(10);

password = new JPasswordField();

password.setFont(new Font("微软雅黑", Font.PLAIN, 12));

password.setBounds(88, 74, 143, 21);

panel.add(password);

JButton helpButton = new JButton("?");

helpButton.setToolTipText("");

helpButton.setFont(new Font("微软雅黑", Font.PLAIN, 12));

helpButton.setBounds(192, 105, 39,23);

panel.add(helpButton);

final JLabel messageLabel = new JLabel("Please log in your account.");

messageLabel.setHorizontalAlignment(SwingConstants.LEFT);

messageLabel.setFont(new Font("微软雅黑", Font.PLAIN, 12));

messageLabel.setBounds(24, 138, 204, 51);

panel.add(messageLabel);

JLabel lblNewLabel\_2 = new JLabel("");

lblNewLabel\_2.setIcon(new ImageIcon(LoginWindow.class.getResource("/resource/contact-icon.png")));

lblNewLabel\_2.setBounds(29, 34, 135, 153);

getContentPane().add(lblNewLabel\_2);

JButton loginButton = new JButton("Log In");

loginButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(account.getText() == null || account.getText().length() == 0){

return;

}

LoginResult result = Store.getInstance().login(account.getText(), new String(password.getPassword()));

password.setText("");

messageLabel.setText(result.getMessage());

if(result.success){

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

MainWindow window = new MainWindow();

window.show();

} catch (Exception e) {

}

}

});

dispose();

}

}

});

loginButton.setFont(new Font("微软雅黑", Font.PLAIN, 12));

loginButton.setBounds(88, 105, 83, 23);

panel.add(loginButton);

loginButton.setActionCommand("OK");

getRootPane().setDefaultButton(loginButton);

helpButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

messageLabel.setText("<html>If you doesn't have a account, please contact with the MANAGER.</html>");

}

});

account.setText("a");

password.setText("123");

}

}

* MainWindow.java

package ui;

import java.awt.EventQueue;

import java.awt.Toolkit;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JPanel;

import javax.swing.SwingUtilities;

import javax.swing.UIManager;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

import javax.swing.JLabel;

import javax.swing.border.TitledBorder;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.Font;

import javax.swing.SwingConstants;

import javax.swing.ImageIcon;

import system.Store;

import system.Store.LoginResult;

public class LoginWindow extends JDialog {

private static final long serialVersionUID = 3110240126605474382L;

private JPasswordField password;

private JTextField account;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

try {

LoginWindow dialog = new LoginWindow();

dialog.setDefaultCloseOperation(JDialog.DISPOSE\_ON\_CLOSE);

dialog.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* Create the dialog.

\*/

public LoginWindow() {

getContentPane().setFont(new Font("微软雅黑", Font.PLAIN, 12));

setTitle("Comic Book System - Log in");

setResizable(false);

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

} catch (Exception e1) {

}

SwingUtilities.updateComponentTreeUI(this);

setIconImage(Toolkit.getDefaultToolkit().getImage(LoginWindow.class.getResource("/resource/contact-icon.png")));

setBounds(100, 100, 468, 258);

setBounds((Toolkit.getDefaultToolkit().getScreenSize().width-this.getWidth())/2, (Toolkit.getDefaultToolkit().getScreenSize().height-this.getHeight())/2, getWidth(), getHeight());

getContentPane().setLayout(null);

JPanel panel = new JPanel();

panel.setBorder(new TitledBorder(null, "Login", TitledBorder.LEADING, TitledBorder.TOP, new Font("微软雅黑", Font.PLAIN, 12), null));

panel.setBounds(186, 10, 256, 199);

getContentPane().add(panel);

panel.setLayout(null);

JLabel lblNewLabel\_1 = new JLabel("Password");

lblNewLabel\_1.setFont(new Font("微软雅黑", Font.PLAIN, 12));

lblNewLabel\_1.setBounds(24, 77, 68, 15);

panel.add(lblNewLabel\_1);

JLabel lblNewLabel = new JLabel("Account");

lblNewLabel.setFont(new Font("微软雅黑", Font.PLAIN, 12));

lblNewLabel.setBounds(24, 46, 54, 15);

panel.add(lblNewLabel);

account = new JTextField();

account.setFont(new Font("微软雅黑", Font.PLAIN, 12));

account.setBounds(88, 43, 143, 21);

panel.add(account);

account.setColumns(10);

password = new JPasswordField();

password.setFont(new Font("微软雅黑", Font.PLAIN, 12));

password.setBounds(88, 74, 143, 21);

panel.add(password);

JButton helpButton = new JButton("?");

helpButton.setToolTipText("");

helpButton.setFont(new Font("微软雅黑", Font.PLAIN, 12));

helpButton.setBounds(192, 105, 39,23);

panel.add(helpButton);

final JLabel messageLabel = new JLabel("Please log in your account.");

messageLabel.setHorizontalAlignment(SwingConstants.LEFT);

messageLabel.setFont(new Font("微软雅黑", Font.PLAIN, 12));

messageLabel.setBounds(24, 138, 204, 51);

panel.add(messageLabel);

JLabel lblNewLabel\_2 = new JLabel("");

lblNewLabel\_2.setIcon(new ImageIcon(LoginWindow.class.getResource("/resource/contact-icon.png")));

lblNewLabel\_2.setBounds(29, 34, 135, 153);

getContentPane().add(lblNewLabel\_2);

JButton loginButton = new JButton("Log In");

loginButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(account.getText() == null || account.getText().length() == 0){

return;

}

LoginResult result = Store.getInstance().login(account.getText(), new String(password.getPassword()));

password.setText("");

messageLabel.setText(result.getMessage());

if(result.success){

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

MainWindow window = new MainWindow();

window.show();

} catch (Exception e) {

}

}

});

dispose();

}

}

});

loginButton.setFont(new Font("微软雅黑", Font.PLAIN, 12));

loginButton.setBounds(88, 105, 83, 23);

panel.add(loginButton);

loginButton.setActionCommand("OK");

getRootPane().setDefaultButton(loginButton);

helpButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

messageLabel.setText("<html>If you doesn't have a account, please contact with the MANAGER.</html>");

}

});

account.setText("a");

password.setText("123");

}

}

* BusinessRecord.java

package system.record;

import java.util.Date;

import system.\*;

import utils.data.DataFile;

public abstract class BusinessRecord implements IBusinessRecord {

public static final String ACTION\_BUY = "Buy";

public static final String ACTION\_RENT = "Rent";

public static final String ACTION\_RETURN = "Return";

public static final String ACTION\_TRADE = "Trade";

public static final String ACTION\_CONSIGNMENT = "Consignment";

protected String id;

protected String customerId;

protected String action;

protected Money amount;

protected Date time;

protected String handlerId;

protected String comment;

public BusinessRecord() {

}

public BusinessRecord(String id, String customerID, String action,

Money amount, Date time, String handlerId, String comment) {

super();

this.id = id;

this.customerId = customerID;

this.action = action;

this.amount = amount;

this.time = time;

this.handlerId = handlerId;

this.comment = comment;

}

public String getCustomerId() {

return customerId;

}

public void setCustomerId(String customerId) {

this.customerId = customerId;

}

public String getHandlerId() {

return handlerId;

}

public void setHandlerId(String handlerId) {

this.handlerId = handlerId;

}

public Money getAmount() {

return amount;

}

public void setAmount(Money amount) {

this.amount = amount;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getAction() {

return action;

}

public void setAction(String action) {

this.action = action;

}

public Date getTime() {

return time;

}

public void setTime(Date time) {

this.time = time;

}

public String getComment() {

return comment;

}

public void setComment(String comment) {

this.comment = comment;

}

protected static void baseParser(BusinessRecord br, String[] arr){

if(arr.length >= 7){

br.id = arr[0];

br.customerId = arr[1];

br.action = arr[2];

br.amount = new Money(arr[3]);

br.time = new Date(Long.parseLong(arr[4]));

br.handlerId = arr[5];

br.comment = arr[6];

}

}

protected String baseToString(){

return

this.id + DataFile.SEPARATOR +

this.customerId + DataFile.SEPARATOR +

this.action + DataFile.SEPARATOR +

this.amount.toString() + DataFile.SEPARATOR +

this.time.getTime() + DataFile.SEPARATOR +

this.handlerId + DataFile.SEPARATOR +

this.comment;

}

public static BusinessRecord createWithRecord(String record) {

String sep = String.valueOf(DataFile.SEPARATOR);

String[] arr = record.split(sep);

BusinessRecord br = null;

if(arr.length >= 7){

br = createInstance(arr[2]);

br.initWithRecord(arr);

}

return br;

}

public static BusinessRecord createInstance(String action){

BusinessRecord br = null;

if(BusinessRecord.ACTION\_BUY.equals(action)){

br = new BuyRecord();

}else if(BusinessRecord.ACTION\_CONSIGNMENT.equals(action)){

br = new ConsignmentRecord();

}else if(BusinessRecord.ACTION\_RENT.equals(action)){

br = new RentRecord();

}else if(BusinessRecord.ACTION\_RETURN.equals(action)){

br = new ReturnRecord();

}else if(BusinessRecord.ACTION\_TRADE.equals(action)){

br = new TradeRecord();

}

return br;

}

public String toString(){

return baseToString();

}

public static void main(String[] args) {

BusinessRecord br = new BuyRecord("123", "123", new Money(213.0), new Date(), "123", "123");

String str = br.convertToString();

System.out.println(str);

BusinessRecord br2 = BusinessRecord.createWithRecord(str);

System.out.println(br2.convertToString());

}

}

* BuyItem.java

package system.record;

import java.util.Date;

import system.\*;

import utils.data.DataFile;

public abstract class BusinessRecord implements IBusinessRecord {

public static final String ACTION\_BUY = "Buy";

public static final String ACTION\_RENT = "Rent";

public static final String ACTION\_RETURN = "Return";

public static final String ACTION\_TRADE = "Trade";

public static final String ACTION\_CONSIGNMENT = "Consignment";

protected String id;

protected String customerId;

protected String action;

protected Money amount;

protected Date time;

protected String handlerId;

protected String comment;

public BusinessRecord() {

}

public BusinessRecord(String id, String customerID, String action,

Money amount, Date time, String handlerId, String comment) {

super();

this.id = id;

this.customerId = customerID;

this.action = action;

this.amount = amount;

this.time = time;

this.handlerId = handlerId;

this.comment = comment;

}

public String getCustomerId() {

return customerId;

}

public void setCustomerId(String customerId) {

this.customerId = customerId;

}

public String getHandlerId() {

return handlerId;

}

public void setHandlerId(String handlerId) {

this.handlerId = handlerId;

}

public Money getAmount() {

return amount;

}

public void setAmount(Money amount) {

this.amount = amount;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getAction() {

return action;

}

public void setAction(String action) {

this.action = action;

}

public Date getTime() {

return time;

}

public void setTime(Date time) {

this.time = time;

}

public String getComment() {

return comment;

}

public void setComment(String comment) {

this.comment = comment;

}

protected static void baseParser(BusinessRecord br, String[] arr){

if(arr.length >= 7){

br.id = arr[0];

br.customerId = arr[1];

br.action = arr[2];

br.amount = new Money(arr[3]);

br.time = new Date(Long.parseLong(arr[4]));

br.handlerId = arr[5];

br.comment = arr[6];

}

}

protected String baseToString(){

return

this.id + DataFile.SEPARATOR +

this.customerId + DataFile.SEPARATOR +

this.action + DataFile.SEPARATOR +

this.amount.toString() + DataFile.SEPARATOR +

this.time.getTime() + DataFile.SEPARATOR +

this.handlerId + DataFile.SEPARATOR +

this.comment;

}

public static BusinessRecord createWithRecord(String record) {

String sep = String.valueOf(DataFile.SEPARATOR);

String[] arr = record.split(sep);

BusinessRecord br = null;

if(arr.length >= 7){

br = createInstance(arr[2]);

br.initWithRecord(arr);

}

return br;

}

public static BusinessRecord createInstance(String action){

BusinessRecord br = null;

if(BusinessRecord.ACTION\_BUY.equals(action)){

br = new BuyRecord();

}else if(BusinessRecord.ACTION\_CONSIGNMENT.equals(action)){

br = new ConsignmentRecord();

}else if(BusinessRecord.ACTION\_RENT.equals(action)){

br = new RentRecord();

}else if(BusinessRecord.ACTION\_RETURN.equals(action)){

br = new ReturnRecord();

}else if(BusinessRecord.ACTION\_TRADE.equals(action)){

br = new TradeRecord();

}

return br;

}

public String toString(){

return baseToString();

}

public static void main(String[] args) {

BusinessRecord br = new BuyRecord("123", "123", new Money(213.0), new Date(), "123", "123");

String str = br.convertToString();

System.out.println(str);

BusinessRecord br2 = BusinessRecord.createWithRecord(str);

System.out.println(br2.convertToString());

}

}

* BuyRecord.java

package system.record;

import java.util.ArrayList;

import java.util.Date;

import system.Money;

import utils.data.DataFile;

public class BuyRecord extends BusinessRecord {

private ArrayList<BuyItem> buyList = new ArrayList<BuyItem>();;

public BuyRecord(String id, String customerID, Money amount,

Date time, String handlerId, String comment) {

super(id, customerID, BusinessRecord.ACTION\_BUY, amount, time, handlerId, comment);

}

public BuyRecord() {

action = BusinessRecord.ACTION\_BUY;

}

public ArrayList<BuyItem> getBuyList() {

return buyList;

}

public void setBuyList(ArrayList<BuyItem> buyList) {

this.buyList = buyList;

}

@Override

public void initWithRecord(String[] arr) {

BusinessRecord.baseParser(this, arr);

if(arr.length >= 8){

int c = 7;

int size = Integer.parseInt(arr[c++]);

for(int i = 0; i < size; i++){

String bid = arr[c++];

int quantity = Integer.valueOf(arr[c++]);

buyList.add(new BuyItem(bid, quantity));

}

}

}

@Override

public String convertToString() {

StringBuilder sb = new StringBuilder();

sb.append(baseToString() + DataFile.SEPARATOR);

sb.append("" + buyList.size() + "" + DataFile.SEPARATOR);

for(BuyItem b : buyList){

sb.append(b.getBookId()); sb.append(DataFile.SEPARATOR);

sb.append(b.getQunatity()); sb.append(DataFile.SEPARATOR);

}

return sb.toString();

}

}

* ConsignmentRecord.java

package system.record;

import java.util.ArrayList;

import java.util.Date;

import system.Money;

import utils.data.DataFile;

public class BuyRecord extends BusinessRecord {

private ArrayList<BuyItem> buyList = new ArrayList<BuyItem>();;

public BuyRecord(String id, String customerID, Money amount,

Date time, String handlerId, String comment) {

super(id, customerID, BusinessRecord.ACTION\_BUY, amount, time, handlerId, comment);

}

public BuyRecord() {

action = BusinessRecord.ACTION\_BUY;

}

public ArrayList<BuyItem> getBuyList() {

return buyList;

}

public void setBuyList(ArrayList<BuyItem> buyList) {

this.buyList = buyList;

}

@Override

public void initWithRecord(String[] arr) {

BusinessRecord.baseParser(this, arr);

if(arr.length >= 8){

int c = 7;

int size = Integer.parseInt(arr[c++]);

for(int i = 0; i < size; i++){

String bid = arr[c++];

int quantity = Integer.valueOf(arr[c++]);

buyList.add(new BuyItem(bid, quantity));

}

}

}

@Override

public String convertToString() {

StringBuilder sb = new StringBuilder();

sb.append(baseToString() + DataFile.SEPARATOR);

sb.append("" + buyList.size() + "" + DataFile.SEPARATOR);

for(BuyItem b : buyList){

sb.append(b.getBookId()); sb.append(DataFile.SEPARATOR);

sb.append(b.getQunatity()); sb.append(DataFile.SEPARATOR);

}

return sb.toString();

}

}

* IBusinessRecord.java

package system.record;

import java.util.ArrayList;

import java.util.Date;

import system.Money;

import utils.data.DataFile;

public class BuyRecord extends BusinessRecord {

private ArrayList<BuyItem> buyList = new ArrayList<BuyItem>();;

public BuyRecord(String id, String customerID, Money amount,

Date time, String handlerId, String comment) {

super(id, customerID, BusinessRecord.ACTION\_BUY, amount, time, handlerId, comment);

}

public BuyRecord() {

action = BusinessRecord.ACTION\_BUY;

}

public ArrayList<BuyItem> getBuyList() {

return buyList;

}

public void setBuyList(ArrayList<BuyItem> buyList) {

this.buyList = buyList;

}

@Override

public void initWithRecord(String[] arr) {

BusinessRecord.baseParser(this, arr);

if(arr.length >= 8){

int c = 7;

int size = Integer.parseInt(arr[c++]);

for(int i = 0; i < size; i++){

String bid = arr[c++];

int quantity = Integer.valueOf(arr[c++]);

buyList.add(new BuyItem(bid, quantity));

}

}

}

@Override

public String convertToString() {

StringBuilder sb = new StringBuilder();

sb.append(baseToString() + DataFile.SEPARATOR);

sb.append("" + buyList.size() + "" + DataFile.SEPARATOR);

for(BuyItem b : buyList){

sb.append(b.getBookId()); sb.append(DataFile.SEPARATOR);

sb.append(b.getQunatity()); sb.append(DataFile.SEPARATOR);

}

return sb.toString();

}

}

* RentRecord.java

package system.record;

import java.util.ArrayList;

import java.util.Date;

import system.Money;

import utils.data.DataFile;

public class BuyRecord extends BusinessRecord {

private ArrayList<BuyItem> buyList = new ArrayList<BuyItem>();;

public BuyRecord(String id, String customerID, Money amount,

Date time, String handlerId, String comment) {

super(id, customerID, BusinessRecord.ACTION\_BUY, amount, time, handlerId, comment);

}

public BuyRecord() {

action = BusinessRecord.ACTION\_BUY;

}

public ArrayList<BuyItem> getBuyList() {

return buyList;

}

public void setBuyList(ArrayList<BuyItem> buyList) {

this.buyList = buyList;

}

@Override

public void initWithRecord(String[] arr) {

BusinessRecord.baseParser(this, arr);

if(arr.length >= 8){

int c = 7;

int size = Integer.parseInt(arr[c++]);

for(int i = 0; i < size; i++){

String bid = arr[c++];

int quantity = Integer.valueOf(arr[c++]);

buyList.add(new BuyItem(bid, quantity));

}

}

}

@Override

public String convertToString() {

StringBuilder sb = new StringBuilder();

sb.append(baseToString() + DataFile.SEPARATOR);

sb.append("" + buyList.size() + "" + DataFile.SEPARATOR);

for(BuyItem b : buyList){

sb.append(b.getBookId()); sb.append(DataFile.SEPARATOR);

sb.append(b.getQunatity()); sb.append(DataFile.SEPARATOR);

}

return sb.toString();

}

}

* ReturnRecord.java

package system.record;

import java.util.Date;

import system.Money;

import utils.data.DataFile;

public class ReturnRecord extends BusinessRecord {

private String cardID;

private String bookID;

private String rentRecordID;

public ReturnRecord() {

super();

action = BusinessRecord.ACTION\_RETURN;

}

public ReturnRecord(String id, String customerID, Money amount, Date time,

String handlerId, String comment, String cardID, String bookID,

String rentRecordID) {

super(id, customerID, BusinessRecord.ACTION\_RETURN, amount, time, handlerId, comment);

this.cardID = cardID;

this.bookID = bookID;

this.rentRecordID = rentRecordID;

}

public ReturnRecord(String id, RentRecord rentRecord, Money money, String handler, String comment){

super(id, rentRecord.getCustomerId(), BusinessRecord.ACTION\_RETURN, money, new Date(), handler, comment);

this.cardID = rentRecord.getCardID();

this.bookID = rentRecord.getBookID();

this.rentRecordID = rentRecord.getId();

}

@Override

public void initWithRecord(String[] record) {

BusinessRecord.baseParser(this, record);

if(record.length >= 10){

this.cardID = record[7];

this.bookID = record[8];

this.rentRecordID = record[9];

}

}

@Override

public String convertToString() {

StringBuilder sb = new StringBuilder();

sb.append(baseToString() + DataFile.SEPARATOR);

sb.append(cardID); sb.append(DataFile.SEPARATOR);

sb.append(bookID); sb.append(DataFile.SEPARATOR);

sb.append(rentRecordID);

return sb.toString();

}

public String getCardID() {

return cardID;

}

public void setCardID(String cardID) {

this.cardID = cardID;

}

public String getBookID() {

return bookID;

}

public void setBookID(String bookID) {

this.bookID = bookID;

}

public String getRentRecordID() {

return rentRecordID;

}

public void setRentRecordID(String rentRecordID) {

this.rentRecordID = rentRecordID;

}

}

* TradeRecord.java

package system.record;

import java.util.Date;

import system.Money;

import utils.data.DataFile;

public class TradeRecord extends BusinessRecord {

public static final String STATE\_WAIT = "wait";

public static final String STATE\_CLOSE = "close";

public static final String STATE\_SUCCESS = "success";

private String bookID;

private String tradeState;

private String secordCustomerID;

public TradeRecord() {

super();

action = BusinessRecord.ACTION\_TRADE;

}

public TradeRecord(String id, String customerID,

Money amount, Date time, String handlerId, String comment, String bookID, String tradeState, String secordCustomer) {

super(id, customerID, BusinessRecord.ACTION\_TRADE, amount, time, handlerId, comment);

this.bookID = bookID;

this.tradeState = tradeState;

this.secordCustomerID = secordCustomer;

}

@Override

public void initWithRecord(String[] record) {

BusinessRecord.baseParser(this, record);

if(record.length >= 10){

this.bookID = record[7];

this.tradeState = record[8];

this.secordCustomerID = record[9];

}

}

@Override

public String convertToString() {

StringBuilder sb = new StringBuilder();

sb.append(baseToString() + DataFile.SEPARATOR);

sb.append(bookID); sb.append(DataFile.SEPARATOR);

sb.append(tradeState); sb.append(DataFile.SEPARATOR);

sb.append(secordCustomerID);

return sb.toString();

}

public String getBookID() {

return bookID;

}

public void setBookID(String bookID) {

this.bookID = bookID;

}

public String getTradeState() {

return tradeState;

}

public void setTradeState(String tradeState) {

this.tradeState = tradeState;

}

public String getSecordCustomerID() {

return secordCustomerID;

}

public void setSecordCustomerID(String secordCustomerID) {

this.secordCustomerID = secordCustomerID;

}

}

* Book.java

package system;

import utils.data.DataFile;

public class Book {

private String id;

private String title;

private String author;

private String publisher;

private String ISBN;

private Money price;

private double grade;

public Book(String id, String title, String author, String publisher, String ISBN,

double price, double grade) {

this.id = id;

this.title = title;

this.author = author;

this.publisher = publisher;

this.ISBN = ISBN;

this.price = new Money(price);

this.grade = grade;

}

public Book(String record){

String sep = String.valueOf(DataFile.SEPARATOR);

String[] arr = record.split(sep);

if(arr.length >= 7){

this.id = arr[0];

this.title = arr[1];

this.author = arr[2];

this.publisher = arr[3];

this.ISBN = arr[4];

this.price = new Money(Double.valueOf(arr[5]));

this.grade = Double.valueOf(arr[6]);

}

}

public String toString(){

StringBuilder s = new StringBuilder();

s.append(this.id);s.append(DataFile.SEPARATOR);

s.append(this.title);s.append(DataFile.SEPARATOR);

s.append(this.author);s.append(DataFile.SEPARATOR);

s.append(this.publisher);s.append(DataFile.SEPARATOR);

s.append(this.ISBN);s.append(DataFile.SEPARATOR);

s.append(this.price);s.append(DataFile.SEPARATOR);

s.append(this.grade);

return s.toString();

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getPublisher() {

return publisher;

}

public void setPublisher(String publisher) {

this.publisher = publisher;

}

public String getISBN() {

return ISBN;

}

public void setISBN(String iSBN) {

ISBN = iSBN;

}

public Money getPrice() {

return price;

}

public void setPrice(Money price) {

this.price = price;

}

public double getGrade() {

return grade;

}

public void setGrade(double grade) {

this.grade = grade;

}

public boolean equals(Book b){

return

this.id != null && this.id.equals(b.getId()) &&

this.title != null && this.title.equals(b.getTitle()) &&

this.author != null && this.author.equals(b.getAuthor()) &&

this.ISBN != null && this.ISBN.equals(b.getISBN()) &&

this.publisher != null && this.publisher.equals(b.publisher);

}

}

* BookStorage.java

package system;

import utils.data.DataFile;

public class Book {

private String id;

private String title;

private String author;

private String publisher;

private String ISBN;

private Money price;

private double grade;

public Book(String id, String title, String author, String publisher, String ISBN,

double price, double grade) {

this.id = id;

this.title = title;

this.author = author;

this.publisher = publisher;

this.ISBN = ISBN;

this.price = new Money(price);

this.grade = grade;

}

public Book(String record){

String sep = String.valueOf(DataFile.SEPARATOR);

String[] arr = record.split(sep);

if(arr.length >= 7){

this.id = arr[0];

this.title = arr[1];

this.author = arr[2];

this.publisher = arr[3];

this.ISBN = arr[4];

this.price = new Money(Double.valueOf(arr[5]));

this.grade = Double.valueOf(arr[6]);

}

}

public String toString(){

StringBuilder s = new StringBuilder();

s.append(this.id);s.append(DataFile.SEPARATOR);

s.append(this.title);s.append(DataFile.SEPARATOR);

s.append(this.author);s.append(DataFile.SEPARATOR);

s.append(this.publisher);s.append(DataFile.SEPARATOR);

s.append(this.ISBN);s.append(DataFile.SEPARATOR);

s.append(this.price);s.append(DataFile.SEPARATOR);

s.append(this.grade);

return s.toString();

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getPublisher() {

return publisher;

}

public void setPublisher(String publisher) {

this.publisher = publisher;

}

public String getISBN() {

return ISBN;

}

public void setISBN(String iSBN) {

ISBN = iSBN;

}

public Money getPrice() {

return price;

}

public void setPrice(Money price) {

this.price = price;

}

public double getGrade() {

return grade;

}

public void setGrade(double grade) {

this.grade = grade;

}

public boolean equals(Book b){

return

this.id != null && this.id.equals(b.getId()) &&

this.title != null && this.title.equals(b.getTitle()) &&

this.author != null && this.author.equals(b.getAuthor()) &&

this.ISBN != null && this.ISBN.equals(b.getISBN()) &&

this.publisher != null && this.publisher.equals(b.publisher);

}

}

* BusinessRecordManager.java

package system;

import utils.data.DataFile;

public class Book {

private String id;

private String title;

private String author;

private String publisher;

private String ISBN;

private Money price;

private double grade;

public Book(String id, String title, String author, String publisher, String ISBN,

double price, double grade) {

this.id = id;

this.title = title;

this.author = author;

this.publisher = publisher;

this.ISBN = ISBN;

this.price = new Money(price);

this.grade = grade;

}

public Book(String record){

String sep = String.valueOf(DataFile.SEPARATOR);

String[] arr = record.split(sep);

if(arr.length >= 7){

this.id = arr[0];

this.title = arr[1];

this.author = arr[2];

this.publisher = arr[3];

this.ISBN = arr[4];

this.price = new Money(Double.valueOf(arr[5]));

this.grade = Double.valueOf(arr[6]);

}

}

public String toString(){

StringBuilder s = new StringBuilder();

s.append(this.id);s.append(DataFile.SEPARATOR);

s.append(this.title);s.append(DataFile.SEPARATOR);

s.append(this.author);s.append(DataFile.SEPARATOR);

s.append(this.publisher);s.append(DataFile.SEPARATOR);

s.append(this.ISBN);s.append(DataFile.SEPARATOR);

s.append(this.price);s.append(DataFile.SEPARATOR);

s.append(this.grade);

return s.toString();

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getPublisher() {

return publisher;

}

public void setPublisher(String publisher) {

this.publisher = publisher;

}

public String getISBN() {

return ISBN;

}

public void setISBN(String iSBN) {

ISBN = iSBN;

}

public Money getPrice() {

return price;

}

public void setPrice(Money price) {

this.price = price;

}

public double getGrade() {

return grade;

}

public void setGrade(double grade) {

this.grade = grade;

}

public boolean equals(Book b){

return

this.id != null && this.id.equals(b.getId()) &&

this.title != null && this.title.equals(b.getTitle()) &&

this.author != null && this.author.equals(b.getAuthor()) &&

this.ISBN != null && this.ISBN.equals(b.getISBN()) &&

this.publisher != null && this.publisher.equals(b.publisher);

}

}

* Card.java

package system;

import java.util.Date;

import utils.data.DataFile;

public class Card {

private String id;

private Customer customer;

private Money money;

private Date createdTime;

private Money moneyFrozen = new Money(0);

public Card(Customer user, double money) {

this.id = "";

this.customer = user;

this.money = new Money(money);

this.createdTime = new Date();

}

public Card(String id, Customer user, Money money, Date createdTime) {

super();

this.id = id;

this.customer = user;

this.money = money;

this.createdTime = createdTime;

}

public Card(String id, Customer user, double money, Date createdTime) {

this(id, user, new Money(money), createdTime);

}

public Card(String record){

String sep = String.valueOf(DataFile.SEPARATOR);

String[] arr = record.split(sep);

if(arr.length >= 4){

this.id = arr[0];

this.customer = CustomerManager.getInstance().get(arr[1]);

this.money = new Money(arr[2]);

this.createdTime = new Date(Long.parseLong(arr[3]));

}

}

@Override

public String toString() {

return

this.id + DataFile.SEPARATOR +

((this.customer == null) ? "" : this.customer.getId()) + DataFile.SEPARATOR +

this.money + DataFile.SEPARATOR +

this.createdTime.getTime();

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public Money getMoney() {

return money;

}

public Money getActualMoney(){

return new Money(money.getAmount() - moneyFrozen.getAmount());

}

public void setMoney(Money money) {

this.money = money;

}

public Customer getCustomer() {

return customer;

}

public void setCustomer(Customer customer) {

this.customer = customer;

}

public Date getCreatedTime() {

return createdTime;

}

public void setCreatedTime(Date createdTime) {

this.createdTime = createdTime;

}

public void addMoney(double val){

this.money.add(new Money(val));

}

public Money getMoneyFrozen() {

return moneyFrozen;

}

public void setMoneyFrozen(Money moneyFrozen) {

this.moneyFrozen = moneyFrozen;

}

}

* CardManager.java

package system;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import utils.data.Persistence;

public class CardManager extends Persistence {

private static CardManager instance = new CardManager();

public static CardManager getInstance(){

return instance;

}

//Card ID <==> Card

private HashMap<String, Card> list;

//8 digits

private IDGenerator idGenerator;

private CardManager(){

list = new HashMap<String, Card>();

idGenerator = new IDGenerator();

}

public void clear(){

list.clear();

}

public boolean add(Card card){

if(list.containsKey(card.getId())){

return false;

}

list.put(card.getId(), card);

idGenerator.remove(card.getId());

return true;

}

public Card create(Customer customer){

if(customer.getCard() == null){

Card card = new Card(idGenerator.next(), customer, 0, new Date());

if(add(card)){

customer.setCard(card);

return card;

}

}

return null;

}

public Card create(){

Card card = new Card(idGenerator.next(), null, 0, new Date());

if(add(card)){

return card;

}

return null;

}

public ArrayList<Card> getCardList(){

ArrayList<Card> arr = new ArrayList<Card>();

Iterator<Entry<String, Card>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, Card> entry = (Map.Entry<String, Card>) iter.next();

Card val = (Card) entry.getValue();

arr.add(val);

}

return arr;

}

public ArrayList<Card> findCard(String key){

if(key == null || key.length() == 0){

return null;

}

String findKey = key.toLowerCase();

ArrayList<Card> arr = new ArrayList<Card>();

Iterator<Entry<String, Card>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Entry<String, Card> entry = iter.next();

Card val = entry.getValue();

if(val != null && val.toString().toLowerCase().contains(findKey)){

arr.add(val);

}

}

return arr;

}

public Card remove(String id){

return list.remove(id);

}

public Card get(String id){

return list.get(id);

}

public HashMap<String, Card> getCards() {

return list;

}

public void setCards(HashMap<String, Card> cards) {

this.list = cards;

}

@Override

public void setFile() {

this.setFilePath("Card.dat");

}

@Override

public void persist() throws FileNotFoundException {

beginPersist();

Iterator<Entry<String, Card>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, Card> entry = (Map.Entry<String, Card>) iter.next();

Card val = (Card) entry.getValue();

persistNext(val.toString());

}

endPersist();

}

@Override

public void load() throws IOException {

beginLoad();

list.clear();

String record = loadNext();

while(record != null){

Card card = new Card(record);

this.add(card);

record = loadNext();

}

endLoad();

}

public static void main(String[] args) {

CardManager a = CardManager.getInstance();

Customer c1 = new Customer("id", "name", "phone", new Date(), null);

Card card = a.create(c1);

System.out.println(c1);

System.out.println(card);

}

}

* Customer.java

package system;

import java.util.Date;

import utils.data.DataFile;

public class Customer {

private String id;

private String name;

private String phone;

private Date registerTime;

private Card card;

public Customer(){

}

public Customer(String id, String name, String phone, Date registerTime, Card card) {

this.id = id;

this.name = name;

this.phone = phone;

this.card = card;

this.registerTime = registerTime;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public Date getRegisterTime() {

return registerTime;

}

public void setRegisterTime(Date registerTime) {

this.registerTime = registerTime;

}

public Card getCard() {

return card;

}

public void setCard(Card card) {

this.card = card;

}

public Customer(String record){

String sep = String.valueOf(DataFile.SEPARATOR);

String[] arr = record.split(sep);

if(arr.length >= 4){

this.id = arr[0];

this.name = arr[1];

this.phone = arr[2];

this.registerTime = new Date(Long.parseLong(arr[3]));

this.card = CardManager.getInstance().get(arr[4]);

if(this.card != null){

this.card.setCustomer(this);

}

}

}

@Override

public String toString() {

return

id + DataFile.SEPARATOR +

name + DataFile.SEPARATOR +

phone + DataFile.SEPARATOR +

registerTime.getTime() + DataFile.SEPARATOR +

(card != null ? card.getId() : "0");

}

public static void main(String[] args) {

Customer a = new Customer("id", "name", "phone", new Date(), null);

String ser = a.toString();

System.out.println(ser);

System.out.println(a.getRegisterTime());

Customer b = new Customer(ser);

System.out.println(b.toString());

System.out.println(b.getRegisterTime());

}

}

* CustomerManager.java

package system;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import utils.data.Persistence;

public class CustomerManager extends Persistence {

private static CustomerManager instance = new CustomerManager();

public static CustomerManager getInstance() {

return instance;

}

private HashMap<String, Customer> list;

private IDGenerator idGenerator;

private CustomerManager() {

list = new HashMap<String, Customer>();

idGenerator = new IDGenerator();

}

public void clear(){

list.clear();

}

public boolean add(Customer customer){

if(list.containsKey(customer.getName())){

return false;

}

list.put(customer.getId(), customer);

idGenerator.remove(customer.getId());

return true;

}

public Customer create(String name, String phone){

Customer customer = new Customer(idGenerator.next(), name, phone, new Date(), null);

list.put(customer.getId(), customer);

return customer;

}

public boolean hasCustomer(String customerID){

return list.containsKey(customerID);

}

public Customer get(String customerID){

return list.get(customerID);

}

public HashMap<String, Customer> getList() {

return list;

}

public void setList(HashMap<String, Customer> list) {

this.list = list;

}

public ArrayList<Customer> getCustomerList(){

ArrayList<Customer> arr = new ArrayList<Customer>();

Iterator<Entry<String, Customer>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, Customer> entry = (Map.Entry<String, Customer>) iter.next();

Customer val = (Customer) entry.getValue();

arr.add(val);

}

return arr;

}

public ArrayList<Customer> findCustomer(String key){

if(key == null || key.length() == 0){

return null;

}

String findKey = key.toLowerCase();

ArrayList<Customer> arr = new ArrayList<Customer>();

Iterator<Entry<String, Customer>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Entry<String, Customer> entry = iter.next();

Customer val = entry.getValue();

if(val != null && val.toString().toLowerCase().contains(findKey)){

arr.add(val);

}

}

return arr;

}

@Override

public void setFile() {

this.setFilePath("Customer.dat");

}

@Override

public void persist() throws FileNotFoundException {

beginPersist();

Iterator<Entry<String, Customer>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, Customer> entry = (Map.Entry<String, Customer>) iter.next();

Customer val = (Customer) entry.getValue();

persistNext(val.toString());

}

endPersist();

}

@Override

public void load() throws IOException {

beginLoad();

list.clear();

String record = loadNext();

while(record != null){

Customer customer = new Customer(record);

this.add(customer);

record = loadNext();

}

endLoad();

}

public static void main(String args[]) throws Exception{

CustomerManager cm = CustomerManager.getInstance();

cm.setFile();

cm.persist();

cm.load();

}

}

* IDGenerator.java

package system;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import utils.data.Persistence;

public class CustomerManager extends Persistence {

private static CustomerManager instance = new CustomerManager();

public static CustomerManager getInstance() {

return instance;

}

private HashMap<String, Customer> list;

private IDGenerator idGenerator;

private CustomerManager() {

list = new HashMap<String, Customer>();

idGenerator = new IDGenerator();

}

public void clear(){

list.clear();

}

public boolean add(Customer customer){

if(list.containsKey(customer.getName())){

return false;

}

list.put(customer.getId(), customer);

idGenerator.remove(customer.getId());

return true;

}

public Customer create(String name, String phone){

Customer customer = new Customer(idGenerator.next(), name, phone, new Date(), null);

list.put(customer.getId(), customer);

return customer;

}

public boolean hasCustomer(String customerID){

return list.containsKey(customerID);

}

public Customer get(String customerID){

return list.get(customerID);

}

public HashMap<String, Customer> getList() {

return list;

}

public void setList(HashMap<String, Customer> list) {

this.list = list;

}

public ArrayList<Customer> getCustomerList(){

ArrayList<Customer> arr = new ArrayList<Customer>();

Iterator<Entry<String, Customer>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, Customer> entry = (Map.Entry<String, Customer>) iter.next();

Customer val = (Customer) entry.getValue();

arr.add(val);

}

return arr;

}

public ArrayList<Customer> findCustomer(String key){

if(key == null || key.length() == 0){

return null;

}

String findKey = key.toLowerCase();

ArrayList<Customer> arr = new ArrayList<Customer>();

Iterator<Entry<String, Customer>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Entry<String, Customer> entry = iter.next();

Customer val = entry.getValue();

if(val != null && val.toString().toLowerCase().contains(findKey)){

arr.add(val);

}

}

return arr;

}

@Override

public void setFile() {

this.setFilePath("Customer.dat");

}

@Override

public void persist() throws FileNotFoundException {

beginPersist();

Iterator<Entry<String, Customer>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, Customer> entry = (Map.Entry<String, Customer>) iter.next();

Customer val = (Customer) entry.getValue();

persistNext(val.toString());

}

endPersist();

}

@Override

public void load() throws IOException {

beginLoad();

list.clear();

String record = loadNext();

while(record != null){

Customer customer = new Customer(record);

this.add(customer);

record = loadNext();

}

endLoad();

}

public static void main(String args[]) throws Exception{

CustomerManager cm = CustomerManager.getInstance();

cm.setFile();

cm.persist();

cm.load();

}

}

* Money.java

package system;

import java.math.BigDecimal;

import java.math.RoundingMode;

public class Money {

private double amount;

public Money(){

amount = 0;

}

public Money(String amount){

this.amount = Double.parseDouble(amount);

roundUp();

}

public Money(double amount){

this.amount = amount;

roundUp();

}

private void roundUp(){

BigDecimal bd = new BigDecimal(amount);

bd = bd.setScale(2, RoundingMode.HALF\_UP);

amount = bd.doubleValue();

}

public Money plus(Money m){

amount += m.amount;

roundUp();

return this;

}

public Money add(Money m){

amount += m.amount;

roundUp();

return this;

}

public Money minus(Money m){

amount -= m.amount;

roundUp();

return this;

}

public Money multiply(Money m){

amount \*= m.amount;

roundUp();

return this;

}

public Money divide(Money m){

amount /= m.amount;

roundUp();

return this;

}

public double getAmount() {

return amount;

}

public void setAmount(double amount) {

this.amount = amount;

}

public String toString(){

return String.format("%.2f", amount);

}

public boolean equals(Money m){

if(m == null) return false;

return m.amount - this.amount < 0.00001 && m.amount - this.amount > -0.00001;

}

}

* Store.java

package system;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import system.BookStorage.BookItem;

import system.record.\*;

public class Store {

private BookStorage bookStorage;

private CardManager cardManager;

private UserManager userManager;

private CustomerManager customerManager;

private BusinessRecordManager businessRecordManager;

private StoreAccount account;

private static Store instance = new Store();

private Store(){

bookStorage = BookStorage.getInstance();

cardManager = CardManager.getInstance();

userManager = UserManager.getInstance();

customerManager = CustomerManager.getInstance();

businessRecordManager = BusinessRecordManager.getInstance();

account = StoreAccount.getInstance();

try {

loadAllData();

} catch (FileNotFoundException e) {

} catch (IOException e) {

}

}

public static Store getInstance(){

return instance;

}

public void loadAllData() throws FileNotFoundException, IOException {

bookStorage.load();

cardManager.load();

userManager.load();

customerManager.load();

businessRecordManager.load();

account.load();

refreshMoneyFrozen();

}

public void persistAll() throws FileNotFoundException{

bookStorage.persist();

cardManager.persist();

userManager.persist();

customerManager.persist();

businessRecordManager.persist();

account.persist();

}

public LoginResult login(String account, String password){

LoginResult result = new LoginResult();

result.success = userManager.login(account, password);

result.user = userManager.getCurrentUser();

if(!result.success){

if(userManager.hasUser(account)){

result.setMessage("Account cannot match password.");

}else{

result.setMessage("Account does not exist.");

}

}else{

result.setMessage("Log in successful.");

}

return result;

}

public class LoginResult extends Result {

public boolean success;

public User user;

}

public ArrayList<BookItem> getBookItemList(){

return bookStorage.getBooks();

}

public class Result {

private String message;

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

}

public void buy(BuyRecord buyRecord){

businessRecordManager.add(buyRecord);

}

public void rent(RentRecord rentRecord){

BookItem bi = bookStorage.findBookItemByID(rentRecord.getBookID());

bi.setQuantity\_on\_sale(bi.getQuantity\_on\_sale() - 1);

bi.setQuantity\_rented(bi.getQuantity\_rented() + 1);

businessRecordManager.add(rentRecord);

}

/\*\*

\* return the balance that cleared

\*/

public Money clear(Card card){

double card\_actual = card.getActualMoney().getAmount();

if(card\_actual < 0){

card.getMoney().plus(new Money(-card\_actual));

StoreAccount.getInstance().getMoney().plus(new Money(card\_actual));

}

return new Money(-card\_actual);

}

/\*\*

\* money should be > 0

\*/

public void pay(Card card, Money money){

if(card != null){

card.getMoney().minus(money);

}

StoreAccount.getInstance().getMoney().plus(money);

}

public void refreshMoneyFrozen(){

HashMap<String, Card> list = cardManager.getCards();

Iterator<Entry<String, Card>> iter = list.entrySet().iterator();

Date current = new Date();

while(iter.hasNext()){

Map.Entry<String, Card> entry = (Map.Entry<String, Card>) iter.next();

Card val = (Card) entry.getValue();

ArrayList<RentRecord> rentList = businessRecordManager.getRentRecordByCard(val.getId());

Money money = val.getMoneyFrozen();

for(RentRecord i : rentList){

Book book = bookStorage.findBookItemByID(i.getBookID()).getBook();

if(book != null){

money.minus(calculateReturnPrice(i.getTime(), current, book.getGrade()));

}

}

}

}

public Money calculateReturnPrice(Date date\_rent, Date date\_return, double grade){

long time\_diff = date\_return.getTime() - date\_rent.getTime();

int days = 0;

if(time\_diff > 0){

days = (int) (time\_diff / (1000 \* 60 \* 60 \* 24));

}

Money money = new Money(days \* grade \* 0.5);

return money;

}

public void returnBook(ReturnRecord returnRecord){

RentRecord rentRecord = (RentRecord) BusinessRecordManager.getInstance().get(returnRecord.getRentRecordID());

rentRecord.setReturnID(returnRecord.getId());

BookItem bi = bookStorage.findBookItemByID(returnRecord.getBookID());

bi.setQuantity\_on\_sale(bi.getQuantity\_on\_sale() + 1);

bi.setQuantity\_rented(bi.getQuantity\_on\_sale() - 1);

businessRecordManager.add(returnRecord);

}

public ArrayList<Book> getRentBookByCard(String id){

Card card = cardManager.get(id);

ArrayList<Book> books = new ArrayList<Book>();

if(card != null){

}

return books;

}

public BookStorage getBookStorage() {

return bookStorage;

}

public void setBookStorage(BookStorage bookStorage) {

this.bookStorage = bookStorage;

}

public CardManager getCardManager() {

return cardManager;

}

public void setCardManager(CardManager cardManager) {

this.cardManager = cardManager;

}

public UserManager getUserManager() {

return userManager;

}

public void setUserManager(UserManager userManager) {

this.userManager = userManager;

}

public CustomerManager getCustomerManager() {

return customerManager;

}

public void setCustomerManager(CustomerManager customerManager) {

this.customerManager = customerManager;

}

public BusinessRecordManager getBusinessRecordManager() {

return businessRecordManager;

}

public void setBusinessRecordManager(BusinessRecordManager businessRecordManager) {

this.businessRecordManager = businessRecordManager;

}

public StoreAccount getAccount() {

return account;

}

public void setAccount(StoreAccount account) {

this.account = account;

}

}

* StoreAccount.java

package system;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import system.BookStorage.BookItem;

import system.record.\*;

public class Store {

private BookStorage bookStorage;

private CardManager cardManager;

private UserManager userManager;

private CustomerManager customerManager;

private BusinessRecordManager businessRecordManager;

private StoreAccount account;

private static Store instance = new Store();

private Store(){

bookStorage = BookStorage.getInstance();

cardManager = CardManager.getInstance();

userManager = UserManager.getInstance();

customerManager = CustomerManager.getInstance();

businessRecordManager = BusinessRecordManager.getInstance();

account = StoreAccount.getInstance();

try {

loadAllData();

} catch (FileNotFoundException e) {

} catch (IOException e) {

}

}

public static Store getInstance(){

return instance;

}

public void loadAllData() throws FileNotFoundException, IOException {

bookStorage.load();

cardManager.load();

userManager.load();

customerManager.load();

businessRecordManager.load();

account.load();

refreshMoneyFrozen();

}

public void persistAll() throws FileNotFoundException{

bookStorage.persist();

cardManager.persist();

userManager.persist();

customerManager.persist();

businessRecordManager.persist();

account.persist();

}

public LoginResult login(String account, String password){

LoginResult result = new LoginResult();

result.success = userManager.login(account, password);

result.user = userManager.getCurrentUser();

if(!result.success){

if(userManager.hasUser(account)){

result.setMessage("Account cannot match password.");

}else{

result.setMessage("Account does not exist.");

}

}else{

result.setMessage("Log in successful.");

}

return result;

}

public class LoginResult extends Result {

public boolean success;

public User user;

}

public ArrayList<BookItem> getBookItemList(){

return bookStorage.getBooks();

}

public class Result {

private String message;

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

}

public void buy(BuyRecord buyRecord){

businessRecordManager.add(buyRecord);

}

public void rent(RentRecord rentRecord){

BookItem bi = bookStorage.findBookItemByID(rentRecord.getBookID());

bi.setQuantity\_on\_sale(bi.getQuantity\_on\_sale() - 1);

bi.setQuantity\_rented(bi.getQuantity\_rented() + 1);

businessRecordManager.add(rentRecord);

}

/\*\*

\* return the balance that cleared

\*/

public Money clear(Card card){

double card\_actual = card.getActualMoney().getAmount();

if(card\_actual < 0){

card.getMoney().plus(new Money(-card\_actual));

StoreAccount.getInstance().getMoney().plus(new Money(card\_actual));

}

return new Money(-card\_actual);

}

/\*\*

\* money should be > 0

\*/

public void pay(Card card, Money money){

if(card != null){

card.getMoney().minus(money);

}

StoreAccount.getInstance().getMoney().plus(money);

}

public void refreshMoneyFrozen(){

HashMap<String, Card> list = cardManager.getCards();

Iterator<Entry<String, Card>> iter = list.entrySet().iterator();

Date current = new Date();

while(iter.hasNext()){

Map.Entry<String, Card> entry = (Map.Entry<String, Card>) iter.next();

Card val = (Card) entry.getValue();

ArrayList<RentRecord> rentList = businessRecordManager.getRentRecordByCard(val.getId());

Money money = val.getMoneyFrozen();

for(RentRecord i : rentList){

Book book = bookStorage.findBookItemByID(i.getBookID()).getBook();

if(book != null){

money.minus(calculateReturnPrice(i.getTime(), current, book.getGrade()));

}

}

}

}

public Money calculateReturnPrice(Date date\_rent, Date date\_return, double grade){

long time\_diff = date\_return.getTime() - date\_rent.getTime();

int days = 0;

if(time\_diff > 0){

days = (int) (time\_diff / (1000 \* 60 \* 60 \* 24));

}

Money money = new Money(days \* grade \* 0.5);

return money;

}

public void returnBook(ReturnRecord returnRecord){

RentRecord rentRecord = (RentRecord) BusinessRecordManager.getInstance().get(returnRecord.getRentRecordID());

rentRecord.setReturnID(returnRecord.getId());

BookItem bi = bookStorage.findBookItemByID(returnRecord.getBookID());

bi.setQuantity\_on\_sale(bi.getQuantity\_on\_sale() + 1);

bi.setQuantity\_rented(bi.getQuantity\_on\_sale() - 1);

businessRecordManager.add(returnRecord);

}

public ArrayList<Book> getRentBookByCard(String id){

Card card = cardManager.get(id);

ArrayList<Book> books = new ArrayList<Book>();

if(card != null){

}

return books;

}

public BookStorage getBookStorage() {

return bookStorage;

}

public void setBookStorage(BookStorage bookStorage) {

this.bookStorage = bookStorage;

}

public CardManager getCardManager() {

return cardManager;

}

public void setCardManager(CardManager cardManager) {

this.cardManager = cardManager;

}

public UserManager getUserManager() {

return userManager;

}

public void setUserManager(UserManager userManager) {

this.userManager = userManager;

}

public CustomerManager getCustomerManager() {

return customerManager;

}

public void setCustomerManager(CustomerManager customerManager) {

this.customerManager = customerManager;

}

public BusinessRecordManager getBusinessRecordManager() {

return businessRecordManager;

}

public void setBusinessRecordManager(BusinessRecordManager businessRecordManager) {

this.businessRecordManager = businessRecordManager;

}

public StoreAccount getAccount() {

return account;

}

public void setAccount(StoreAccount account) {

this.account = account;

}

}

* User.java

package system;

import utils.data.DataFile;

public class User {

public static final int PERMISSION\_CLERK = 0;

public static final int PERMISSION\_MANAGER = 1;

private String name;

private String password;

private int permission;

public User(String name, String password, int permission) {

super();

this.name = name;

this.password = password;

this.permission = permission;

}

public User(String record) {

String sep = String.valueOf(DataFile.SEPARATOR);

String[] arr = record.split(sep);

if(arr.length >= 3){

this.name = arr[0];

this.password = arr[1];

this.permission = Integer.valueOf(arr[2]);

}

}

@Override

public String toString() {

return

this.name + DataFile.SEPARATOR +

this.password + DataFile.SEPARATOR +

this.permission;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public int getPermission() {

return permission;

}

public void setPermission(int permission) {

this.permission = permission;

}

}

* UserManager.java

package system;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import utils.data.Persistence;

public class UserManager extends Persistence {

private static UserManager instance = new UserManager();

public static UserManager getInstance(){

return instance;

}

private HashMap<String, User> list;

private User currentUser;

private UserManager(){

list = new HashMap<String, User>();

currentUser = null;

}

public void clear(){

list.clear();

}

public boolean add(User user){

if(list.containsKey(user.getName())){

return false;

}

list.put(user.getName(), user);

return true;

}

public boolean hasUser(String name){

return list.containsKey(name);

}

public boolean login(String name, String password){

User u = list.get(name);

if(u != null && u.getName().equals(name) && u.getPassword().equals(password)){

currentUser = u;

return true;

}

return false;

}

public void logout(){

currentUser = null;

}

public User getCurrentUser() {

return currentUser;

}

public HashMap<String, User> getList() {

return list;

}

public ArrayList<User> getUserList(){

ArrayList<User> arr = new ArrayList<User>();

Iterator<Entry<String, User>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, User> entry = iter.next();

User val = entry.getValue();

arr.add(val);

}

return arr;

}

public ArrayList<User> findUser(String key){

if(key == null || key.length() == 0){

return null;

}

String findKey = key.toLowerCase();

ArrayList<User> arr = new ArrayList<User>();

Iterator<Entry<String, User>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Entry<String, User> entry = iter.next();

User val = entry.getValue();

if(val != null && val.toString().toLowerCase().contains(findKey)){

arr.add(val);

}

}

return arr;

}

public static void main(String[] args) throws FileNotFoundException {

UserManager a = UserManager.getInstance();

a.clear();

a.add(new User("rainvision", "123", User.PERMISSION\_CLERK));

a.add(new User("ayzmkk", "123", User.PERMISSION\_CLERK));

a.add(new User("a", "123", User.PERMISSION\_CLERK));

System.out.println(a.hasUser("c"));

System.out.println(a.hasUser("b"));

System.out.println(a.hasUser("a"));

System.out.println(a.login("a", "123"));

a.persist();

}

@Override

public void setFile() {

this.setFilePath("User.dat");

}

@Override

public void persist() throws FileNotFoundException {

beginPersist();

Iterator<Entry<String, User>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, User> entry = (Map.Entry<String, User>) iter.next();

User val = (User) entry.getValue();

persistNext(val.toString());

}

endPersist();

}

@Override

public void load() throws IOException {

beginLoad();

list.clear();

String record = loadNext();

while(record != null){

User user = new User(record);

this.add(user);

record = loadNext();

}

endLoad();

}

}

* FocusTraversalOnArray.java

package system;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import utils.data.Persistence;

public class UserManager extends Persistence {

private static UserManager instance = new UserManager();

public static UserManager getInstance(){

return instance;

}

private HashMap<String, User> list;

private User currentUser;

private UserManager(){

list = new HashMap<String, User>();

currentUser = null;

}

public void clear(){

list.clear();

}

public boolean add(User user){

if(list.containsKey(user.getName())){

return false;

}

list.put(user.getName(), user);

return true;

}

public boolean hasUser(String name){

return list.containsKey(name);

}

public boolean login(String name, String password){

User u = list.get(name);

if(u != null && u.getName().equals(name) && u.getPassword().equals(password)){

currentUser = u;

return true;

}

return false;

}

public void logout(){

currentUser = null;

}

public User getCurrentUser() {

return currentUser;

}

public HashMap<String, User> getList() {

return list;

}

public ArrayList<User> getUserList(){

ArrayList<User> arr = new ArrayList<User>();

Iterator<Entry<String, User>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, User> entry = iter.next();

User val = entry.getValue();

arr.add(val);

}

return arr;

}

public ArrayList<User> findUser(String key){

if(key == null || key.length() == 0){

return null;

}

String findKey = key.toLowerCase();

ArrayList<User> arr = new ArrayList<User>();

Iterator<Entry<String, User>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Entry<String, User> entry = iter.next();

User val = entry.getValue();

if(val != null && val.toString().toLowerCase().contains(findKey)){

arr.add(val);

}

}

return arr;

}

public static void main(String[] args) throws FileNotFoundException {

UserManager a = UserManager.getInstance();

a.clear();

a.add(new User("rainvision", "123", User.PERMISSION\_CLERK));

a.add(new User("ayzmkk", "123", User.PERMISSION\_CLERK));

a.add(new User("a", "123", User.PERMISSION\_CLERK));

System.out.println(a.hasUser("c"));

System.out.println(a.hasUser("b"));

System.out.println(a.hasUser("a"));

System.out.println(a.login("a", "123"));

a.persist();

}

@Override

public void setFile() {

this.setFilePath("User.dat");

}

@Override

public void persist() throws FileNotFoundException {

beginPersist();

Iterator<Entry<String, User>> iter = list.entrySet().iterator();

while(iter.hasNext()){

Map.Entry<String, User> entry = (Map.Entry<String, User>) iter.next();

User val = (User) entry.getValue();

persistNext(val.toString());

}

endPersist();

}

@Override

public void load() throws IOException {

beginLoad();

list.clear();

String record = loadNext();

while(record != null){

User user = new User(record);

this.add(user);

record = loadNext();

}

endLoad();

}

}

* BusinessController.java

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Copyright (c) 2011 Google, Inc.

\* All rights reserved. This program and the accompanying materials

\* are made available under the terms of the Eclipse Public License v1.0

\* which accompanies this distribution, and is available at

\* http://www.eclipse.org/legal/epl-v10.html

\*

\* Contributors:

\* Google, Inc. - initial API and implementation

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

package org.eclipse.wb.swing;

import java.awt.Component;

import java.awt.Container;

import java.awt.FocusTraversalPolicy;

/\*\*

\* Cyclic focus traversal policy based on array of components.

\* <p>

\* This class may be freely distributed as part of any application or plugin.

\*

\* @author scheglov\_ke

\*/

public class FocusTraversalOnArray extends FocusTraversalPolicy {

private final Component m\_Components[];

////////////////////////////////////////////////////////////////////////////

//

// Constructor

//

////////////////////////////////////////////////////////////////////////////

public FocusTraversalOnArray(Component components[]) {

m\_Components = components;

}

////////////////////////////////////////////////////////////////////////////

//

// Utilities

//

////////////////////////////////////////////////////////////////////////////

private int indexCycle(int index, int delta) {

int size = m\_Components.length;

int next = (index + delta + size) % size;

return next;

}

private Component cycle(Component currentComponent, int delta) {

int index = -1;

loop : for (int i = 0; i < m\_Components.length; i++) {

Component component = m\_Components[i];

for (Component c = currentComponent; c != null; c = c.getParent()) {

if (component == c) {

index = i;

break loop;

}

}

}

// try to find enabled component in "delta" direction

int initialIndex = index;

while (true) {

int newIndex = indexCycle(index, delta);

if (newIndex == initialIndex) {

break;

}

index = newIndex;

//

Component component = m\_Components[newIndex];

if (component.isEnabled() && component.isVisible() && component.isFocusable()) {

return component;

}

}

// not found

return currentComponent;

}

////////////////////////////////////////////////////////////////////////////

//

// FocusTraversalPolicy

//

////////////////////////////////////////////////////////////////////////////

public Component getComponentAfter(Container container, Component component) {

return cycle(component, 1);

}

public Component getComponentBefore(Container container, Component component) {

return cycle(component, -1);

}

public Component getFirstComponent(Container container) {

return m\_Components[0];

}

public Component getLastComponent(Container container) {

return m\_Components[m\_Components.length - 1];

}

public Component getDefaultComponent(Container container) {

return getFirstComponent(container);

}

}

**Reference**

**References**

[1] Ivar Jacobson, Grady Booch, James Rumbaugh, *The UNIFIED SOFTWARE DEVELOPMENT PROCESS,* Chapter 8, Chapter 9, Chapter 10, Qinghua university Press,January,2005

[2] en.wikipedia.org/wiki/Unified\_Modeling\_Language

[3] http://publib.boulder.ibm.com/infocenter/rsmhelp/v7r0m0/index.jsp?topic=/com. ibm.xtools.modeler.doc/ topics/ cassn.html

[4] en.wikipedia.org/wiki/Use\_case

[5]en.wikipedia.org/wiki/Use\_case\_diagram