**虚拟校园系统软件设计说明书**

组长：王新宇

组员：毛泊涵、刘晓臻、银雪岑 、胡嘉铭、万家铖、张浩飞

版本：v0.1

2019年8月24日星期六

目录

[1 引言 3](#_Toc17587759)

[1.1 编写目的 3](#_Toc17587760)

[1.2 背景 3](#_Toc17587761)

[1.3 定义 3](#_Toc17587762)

[1.4 参考资料 3](#_Toc17587763)

[2 程序系统的分析 3](#_Toc17587764)

[2.1 可行性分析 3](#_Toc17587765)

[2.2 需求分析 4](#_Toc17587766)

[2.3 开发设计环境： 5](#_Toc17587767)

[3 程序系统的结构 6](#_Toc17587768)

[4 用户管理模块设计说明 7](#_Toc17587769)

[4.1 模块背景 7](#_Toc17587770)

[4.2 需求分析 7](#_Toc17587771)

[4.3 系统设计 7](#_Toc17587772)

[4.3.1 界面设计 7](#_Toc17587773)

[4.3.2 模块流程图 8](#_Toc17587774)

[4.3.3 类分析 8](#_Toc17587775)

[5 公共模块设计说明 9](#_Toc17587776)

[5.1 Message 9](#_Toc17587777)

[5.2 User 9](#_Toc17587778)

[5.3 Book 10](#_Toc17587779)

[5.4 数据库连接 10](#_Toc17587780)

[6 网络模块设计说明 11](#_Toc17587781)

[6.1 客户端IMessageClientSrv 11](#_Toc17587782)

[6.2 服务器端IMessageServerSrv 11](#_Toc17587783)

[6.3 Socket 11](#_Toc17587784)

[6.3.1 客户端 11](#_Toc17587785)

[6.3.2 服务器端 11](#_Toc17587786)

[6.4 输入输出流 12](#_Toc17587787)

[6.4.1 读取输入 12](#_Toc17587788)

[6.4.2 发送输出 12](#_Toc17587789)

[7 多线程模块设计说明 12](#_Toc17587790)

[7.1.1 客户端 12](#_Toc17587791)

[7.1.2 服务器端 13](#_Toc17587792)

[8 数据库设计说明 14](#_Toc17587793)

[8.1 Users（用户） 14](#_Toc17587794)

[8.2 Book（书本） 14](#_Toc17587795)

[8.3 Course（课程） 14](#_Toc17587796)

[8.4 Goods（商品） 15](#_Toc17587797)

[9 其他 15](#_Toc17587798)

# 引言

## 编写目的

本文件作为东南大学计算机学院2017级大三短学期小组项目“虚拟校园系统”软件的设计说明书，详细阐述了此软件的需求、接口设计与实现方法，同时也阐述了软件的开发系统、系统结构。

本文件将从程序系统的分析、系统结构、管理模块设计说明、网络模块设计说明、多线程模块设计说明、工具模块设计说明与数据库设计说明等方面进行阐述。

## 背景

本软件全称叫做“虚拟校园系统”，是一款主要面向学生与教师的一套在线校园系统。同时也实现了管理员管理的功能。

在校园生活中，我们都希望能够拥有一款操作便捷使用方便的校园生活管理系统，而本软件正是基于这个大需求产生的。如果能够在线上管理自己的课程、查询成绩、续借图书，这对不论是学生还是教师都会带来极大的便利。作为均为大学生的软件设计小组，本小组主要从学生角度出发，基于java设计了这款“虚拟校园系统”。

## 定义

## 参考资料

# 程序系统的分析

## 可行性分析

由软件的设计要求可知，必做模块有登录模块、选课模块、教务模块、图书馆模块、商店模块，因为小组人数原因，我们决定再额外增设银行模块。

根据上述模块分类，我们可以单独列出每个模块需要的功能、将会进行的操作、涉及到的数据。根据每个模块的详细信息，我们可以大致明白顶层的数据元素与消息响应如何组织。

同时，因为涉及到远程通信与数据库操作，我们决定使用TCP/IP协议来完成网络通信，因此一个服务端的存在是必要的，服务端的功能便是与每个客户端建立连接并与其交互信息，并及时更新数据库或从数据库中拉取数据。其中，服务端与客户端的消息传递通过java自带的对象序列化功能来实现，每次通过TCP/IP协议传输的流实质都是一个对象，而传递不同类型的对象可以代表不同的操作信息，服务端便会对应不同的数据库操作。

而对于客户端来说，发送消息、接受消息并不是主要任务，它的任务难点在于更新本地的数据并及时向用户反馈，因此使用MCV框架就显得尤为重要。UI层与数据层会使用一个control层交互信息。

通过分析，软件实现的具体框架便浮现了出来。我们决定按照软件系统层次进行分工，三人合作完成底层的服务端与数据库，四人完成顶层的UI设计与客户端MCV框架。

## 需求分析

普通用户的需求是首要需求，每个子模块的需求如下表所示：

表1：普通用户需求分析表

|  |  |  |
| --- | --- | --- |
| **子模块名称** | **普通用户需求** | **注释** |
| 登录模块 | 可以选择用户登录，可以输入一卡通账号与密码，可以修改连接的服务端的IP地址； | 无； |
| 主界面模块 | 查看自己的基本信息（包括学号一卡通号一卡通余额等基本信息），有一些信息宣传，可以进入其他子模块； | 希望可以附加一些别的人性化功能（比如修改头像与昵称）； |
| 教务模块 | 可以查看自己课程成绩； | 无； |
| 图书馆模块 | 可以查看自己剩余借书数量，可以查看自己的借书详情，每本书的借还日期与剩余天数，可以查询馆藏； | 馆藏查询的功能为选做； |
| 选课模块 | 可以选课并查看课表； | 选课系统是直接关心到用户体验的功能，需要富有特色； |
| 银行模块 | 进入前需要先输入银行卡密码，可以充值一卡通，可以查看自己最近的流水消费，可以查看自己的银行卡余额； | 需要优先保证模块的安全性； |
| 商店模块 | 可以查看自己的一卡通余额，可以查看商店的货物与上架时间，可以购买商品。 | 无 |

由于本软件分为普通用户登录与管理员登录，管理员对应的模块需求与普通用户有所不同，且每个子模块都应有自己的管理员。其对应管理员对应的各个模块的需求如下表所示：

表2：管理员需求表

|  |  |  |
| --- | --- | --- |
| **子模块名称** | **管理员需求** | **注释** |
| 登录模块 | 可以选择管理员登录，可以输入管理员账号与密码，可以修改连接的服务端的IP地址； | 无； |
| 主界面 | 查看自己的基本信息（包括管理员账号，权限等级等基本信息），可以进入其他子模块； | 无 |
| 权限管理模块 | 更改用户的权限等级； | 仅最高权限管理员可以操作； |
| 图书馆模块 | 可以更改用户的所借书目以及还书日期，可以增加或删除馆藏； | 仅图书馆模块管理员可以操作； |
| 选课模块 | 可以增添新的课程，可以更改用户的选课，可以更改课程的用户上限； | 仅选课模块管理员可以操作； |
| 教务模块 | 可以增加或删除课程的考试信息，可以录入课程的成绩，可以编辑用户的考试成绩； | 仅教务模块管理员可以操作； |
| 商店模块 | 可以增加或删除商品信息，可以更改商品的价格与上架日期； | 仅商店模块管理员可以操作； |
| 银行模块 | 无 | 无管理员； |

## 开发设计环境：

集成开发环境：

IntelliJ IDEA 、Eclipse；

软件开发工具包：

jre1.8.0\_211；

数据库：

Microsoft Access 2013；

# 程序系统的结构

用一系列图表列出本程序系统内的每个程序（包括每个模块和子程序）的名称、标识符和它们之间的层次结构关系。



# 用户管理模块设计说明

## 模块背景

本模块作为用户管理模块，主要用途为管理用户的登入与登出、管理员的登入与登出。同时也当做整个程序启动器、与服务端的通信建立模块来运行。

## 需求分析

更新信息

建立通信

系统用户管理

登录

登出

本模块主要实现的功能是实现用户登入与登出，并在对应情况再后端同时与服务端建立socket连接。在登录成功后建立主界面窗口。

## 系统设计

## 界面设计

Client端：



## 模块流程图

建立通信

用户登出

用户登录

更新信息

## 类分析

**实体类**：登录账户(学生/教师/管理员)

User：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **类型** | **约束** | **备注** |
| 1 | EcardNumber | String |  | 登录名 |
| 2 | password | String | 6~16个字符 | 密码 |
| 3 | AuthorityNumber | Int | 0~6的整数 | 权限值 |

**服务类**：

Client端：IUserClientSrv

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **名称** | **方法** | **备注** |
| 1 | 登录 | Userlogin(User) |  |
| 2 | 注册 | Userregister(User) |  |
| 3 | 登出 | User logout(User) |  |
| 4 | … | … | … |

Server端：IUserServerSrv

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **名称** | **方法** | **备注** |
| 1 | 登录 | Userlogin(User) |  |
| 2 | 注册 | Userregister(User) |  |
| 3 | 登出 | User logout(User) |  |
| 4 | … | … | … |

# 公共模块设计说明

列出同时用于服务器端和客户端的接口、类或其他资源。

用于网络传输的类必须实现序列化接口：**java.io.Serializable。**

所有和通讯有关的类均继承Message，内部包含有Type数据成员。

**仅列出了部分类，还有其他类有待补全**

## Message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **类型** | **约束** | **备注** |
| 1 | EcardNumber | String | 唯一性 | 一卡通 |
| 2 | Type | MESSAGE\_TYPE | 6~16个字符 | 自定义的类型信息 |
| … | … | … | … | … |

## User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **类型** | **约束** | **备注** |
| 1 | id | String |  | 登录名 |
| 2 | Password | String | 6~16个字符 | 密码 |
| 3 | age | Integer | 非0 | 年龄 |
| 4 | AuthorityLevel | Int | 1~6整数 | 权限等级 |
| 5 | … | … | … | … |

## Book

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **类型** | **约束** | **备注** |
| 1 | id | String |  | 书本唯一id |
| 2 | IsLended | Bool |  | 是否被借取 |
| 3 | Lenddate | Date | 日期 | 借取日期 |
| 4 | … | … | … | … |

## 数据库连接

# 网络模块设计说明

## 客户端IMessageClientSrv

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **名称** | **方法** | **备注** |
| 1 | 发送 | writeObject (Message) | 向服务器发送message类 |
| 2 | 接收 | readObject(Message) | 从服务器接收message类 |

## 服务器端IMessageServerSrv

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **名称** | **方法** | **备注** |
| 1 | 发送 | writeObject (Message) | 向服务器发送message类 |
| 2 | 接收 | readObject(Message) | 从服务器接收message类 |

## Socket

## 客户端

// Code snippet::连接服务器Socket

socket = new Socket(IPAddress,Port);

## 服务器端

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 说明 | 备注 |
| 1 | Server | 服务器管理接口 | 提供新线程调用，线程删除等服务 |
| 2 | start() | 服务器运行函数 | 建立serversocket和socket，与客户端连接，调用线程 |

// Code snippet::创建服务器Socket

socketthread = new socketThread();

serverSocket = new ServerSocket(port);

## 输入输出流

## 读取输入

// Code snippet

is = socket.getInputStream(); //获得socket的输入流  
bis = new BufferedInputStream(is); //构建缓冲输入流  
ois = new ObjectInputStream(bis); //反序列化获得对象  
obtian = (Message) ois.readObject(); //获得message对象

## 发送输出

// Code snippet

ObjectOutputStream oos =

new ObjectOutputStream(cts.getOutputStream()); //建立输出流

Message msg = (Message) ois.writeObject(); //写入流

oos.flush(); //flush流

# 多线程模块设计说明

## 客户端

主程序调用：

Public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new MainFrame(title).setVisible(true);

}

});

}

线程函数：

class socketThread extends Thread{  
 @Override  
 public void run(){  
 StartButton.setEnabled(false);  
 StopButton.setEnabled(true);  
 mainTextArea.setText(mainTextArea.getText()+"正在启动");  
  
 try{  
 serverSocket = new ServerSocket(port);  
 mainTextArea.setText(mainTextArea.getText()+"\n"+"成功启动！");  
 }catch (IOException ioe){  
 mainTextArea.setText(mainTextArea.getText()+"\n"+ioe.toString());  
 }  
  
 int count = 0;  
 while (true){  
 try{  
 if(serverSocket.isClosed()){  
 return;  
 }  
 System.out.println("正在等待连接.........");  
 Socket socket = serverSocket.accept();  
 new ServerThread(socket).start();  
 mainTextArea.setText(mainTextArea.getText()+"\n"+"成功连接到第"+count+"个客户端");  
 }  
 catch (IOException ioe){  
 mainTextArea.setText(mainTextArea.getText()+"\n"+ioe.toString());  
 }  
  
 }  
 }  
}

## 服务器端

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 说明 | 备注 |
| 1 | ServerThread | 客户端线程类 | 为请求的每个客户端建立一个线程 |
| 2 | ServerUI | UI与主socketserver的建立 | 创建可视化GUI并建立socketserver为每一个客户端的连接建立socket |

# 数据库设计说明

数据库使用Microsoft Access建立。

以表格说明或E-R图直接体现。表设计

## Users（用户）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 类型 | 是否主键 | 备注 |
|  | ECardNumber | nvarchar | PRI | 登录ID |
|  | AuthorityNumber | int |  | 权限等级 |
|  | userName | varchar |  | 姓名 |
|  | Age | int |  | 年龄 |
|  | Sex | ENUM |  | 性别 |
|  | PassWord | nvarchar |  | 密码 |
|  | **…** | **…** | **…** | **…** |

## Book（书本）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 类型 | 是否主键 | 备注 |
| 1 | BID | nvarchar | PRI | 书本唯一编号 |
| 2 | bookName | varchar |  | 书本名称 |
| 3 | isLent | bool |  | 是否被借出 |
| 4 | lendDate | navrchar |  | 借出日期 |
| 5 | lenddays | Int |  | 已借出天数 |

## Course（课程）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 类型 | 是否主键 | 备注 |
|  | CID | nvarchar | PRI | 课程唯一编号 |
|  | courseName | varchar |  | 课程名称 |
|  | …… | …… | …… | …… |

## Goods（商品）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段 | 类型 | 是否主键 | 备注 |
|  | GID | nvarchar | PRI | 商品唯一编号 |
|  | goodsName | varchar |  | 商品名称 |
|  | Price | Double |  | 商品价格 |
|  | …… | …… | …… | …… |

# 其他