**Linux系统与开源软件设计**



巧克力爱好者匿名系统

**学 院 ：计算机学院**

**专 业 名 称 ：软件工程**

**姓 名 ：胡祥瑞**

**学 号 ：2018282110373**

# 项目说明

## 1、项目背景

巧克力是一种糖类美食，在美洲和欧洲有非常多的人喜欢吃巧克力，它似乎有中可以使人上瘾的魔力，有些人一吃巧克力就停不住嘴，那么控制不好吃巧克力的量就会引发一些问题，比如吃多了也会发胖，巧克力是一种能量密度很大的食物,黑巧克力中钠和糖的含量尽管低于牛奶巧克力、白巧克力,但脂肪含量(尤其是饱和脂肪的含量)还是比较高的。每天吃一些黑巧克力并不会让你增肥,且可以很好的控制食欲,但吃多了还是会发胖的,尤其是吃加工后添加糖、油脂、全脂奶等成分的黑巧克力。吃多会导致营养不良，单纯只吃黑巧克力减肥以至于代替了正餐,会使体内缺乏相应的营养元素,导致营养不良。减肥很重要的一项就是营养均衡,如果不遵循反倒会不利于减肥瘦身。吃多会引发头痛，巧克力并不是让所有人都幸福的美食,巧克力虽然滋味迷人,但它内含的酪胺却是引人头疼的罪魁祸首。所以,吃多了很容易引发头痛。因此,经常头痛的人更要少吃巧克力,而最好是别吃。

那么如何既能放心大胆的吃美味的巧克力，又不会因为吃巧克力而产生相应的问题呢！在这种情况下，我们的巧克力爱好者联盟就产生了，巧克力爱好者匿名(ChocAn)是一个致力于帮助各种吃巧克力上瘾者的组织。该组织的会员每月向ChocAn付费，然后他们就有权利向保健专家，如营养学家、内科医师和运动专家要求得到不受限制的资讯和治疗。

## 2、项目需求

（1）会员可进行网上付费，付费后可得到一张有会员姓名，9位号码以及磁条编码的塑料卡。每个会员得到一个塑料卡，上面刻有会员名字以及一个9位数的成员编号，同时卡中含有一个磁条，上面有编码信息。

（2）提供者拥有一台终端在开机后输入提供者号码。向ChocAn成员提供服务的每个保健专家(提供者)有一台专门设计的ChocAn计算机终端，它类似于一个商店里的信用卡设备。当一个服务提供者的终端开机时，要求该提供者输入他的提供者号码。

（3）服务开始前会员卡交给提供者进行刷卡后，计算机会对会员状态进行验证，若状态有效则通过相应人员为会员提供服务。服务结束后提供者再次为会员刷卡，提供者输入服务日期。

（4）为了接受来自ChocAn的保健服务，会员将他的卡交给提供者，由提供者在终端读卡器上刷一下卡。然后终端拨打ChocAn数据中心，ChocAn数据中心，ChocAn数据中心计算机验证该成员号码，如果该号码是有效的，Validated（有效）一词出现在线路另一方的显示器上；如果该号码是无效的，其原因也显示出来，如Invalid number（无效号码）或Member suspended（成员暂停）。后一条消息指示欠费（即该成员至少一个月没交会费了），并且成员状态被置为suspended（暂停）。

（5）当向成员提供保健服务后，提供者要为ChocAn记账，这是提供者再次通过读卡器刷卡，或者键入该成员号码。当出现Validated字样时，提供者按照MM-DD-YYYY格式键入服务提供的日期。服务提供的日期是必要的，因为硬件或其他困难可能会阻止提供者在服务提供后立即为ChocAn记账。

（6）提供者可根据“提供者目录”输入相应服务的6位代码对所产生的服务名称进行核对，并可输入相应服务的注释信息，根据上述输入可产生本次服务的具体信息。

（7）提供者使用“提供者目录”查找对应于所提供服务的适当的六位代码。比如，598470是与一个营养师建立会话的代码，而883948是一个有氧锻炼会话的代码。然后提供者键入服务代码。为了核对已经正确的查找到和键入了该服务代码，软件产品随后显示相应于该代码的服务名称（最多20个字符），并且请求提供者核实这确实是所提供的服务，如果提供者输入了一个不存在的代码，则打印出一个错误消息。提供这也可以输入关于所提供服务的注释。根据具体信息可以得到会员应付费用并显示（可由提供者进行人工核对）。

（8）该软件产品接下来查找要为该服务付的费用并在提供者终端上显示出来。为了便于核对，为提供者设计了一个表格，可以在表格上输入当前的日期和时间、提供服务的日期、会员名字和号码、服务代码，以及要付的费用。到周末时提供者进行费用合计，以核对该周ChocAn应付给提供者的钱数。电子资金由银行转账给提供者，并向经理打印概要报告。由电子资金转账数据组成的记录被写入到磁盘，银行计算机稍后将确保向每个提供者的银行账户支付适当的钱数。

（9）为了进行账户支付，需要向经理提供一份概要报告。该报告列出了该周要支付的提供者名单。每个服务者所做的咨询数量，以及他该周的总酬金。最后，打印出提供服务的提供者总数、咨询总数以及总的支付费用。

（10）周末前会为管理人员打印一周的服务报告，对会员打印相应的服务报告。

在星期五午夜，ChocAn数据中心运行主计算机程序。它读取一周提供的服务文件并且打印一些报告。在这一周的任何时间内在ChocAn管理员请求的情况下，每个报告可以单独打印出来。在该周内向ChocAn提供者进行咨询的每个会员都收到一份提供给该会员的一份清单，该清单按服务日期的顺序分类。该报告也以电子邮件的附件形式发送

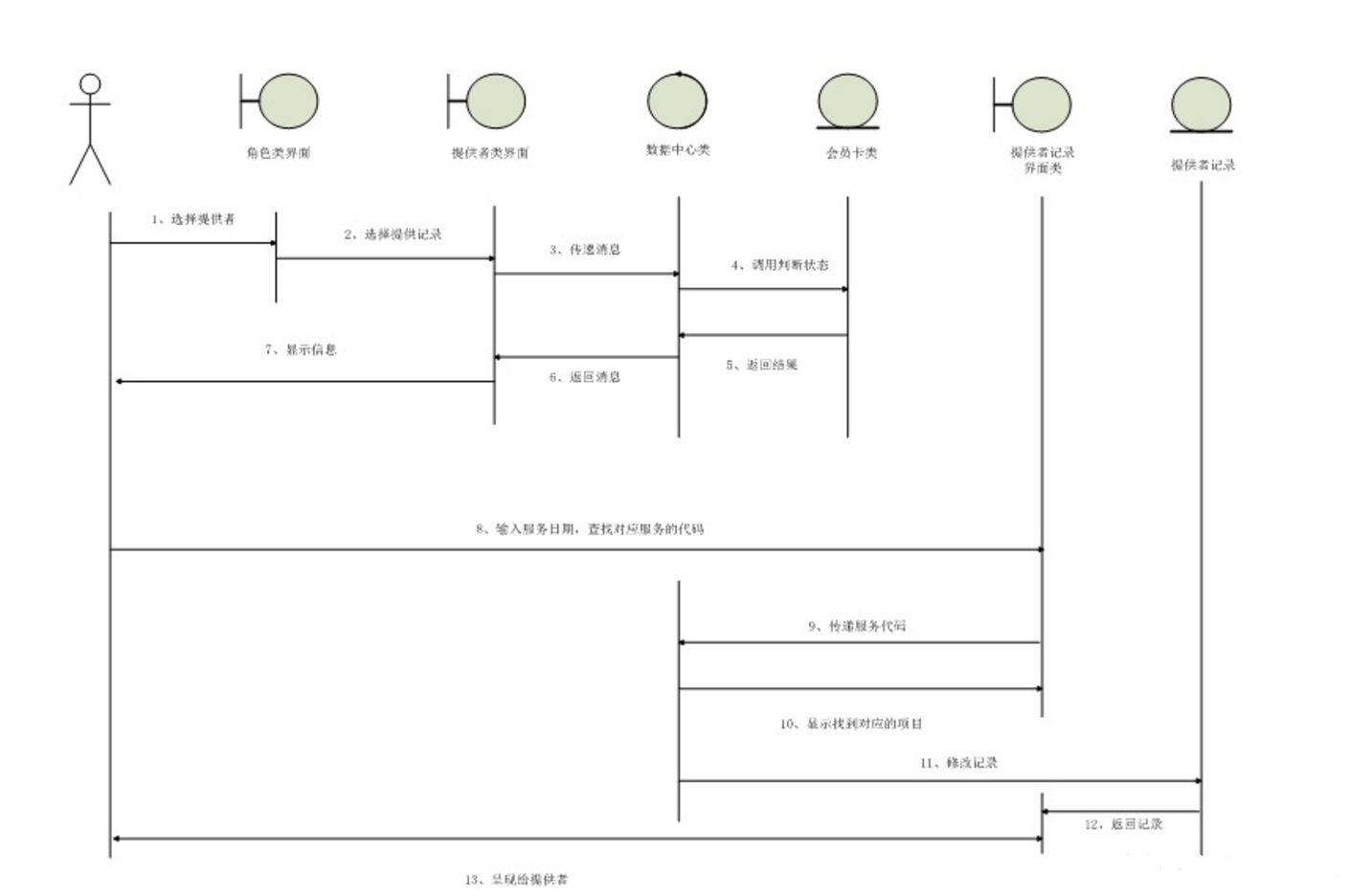
（11）工作日可以对会员进行增删改。在工作日，ChocAn数据中心的软件以交互模式运行，允许操作员向ChocAn加入新的会员，删除退出的会员，以及更新会员记录。类似的，提供者的就可以增加删除修改。

（12）任何时候提供者可以请求获得“提供者目录”。在任何时候，提供者可以请求软件产品给出“提供者目录”，它是一个按字母顺序列出名称的服务和相应服务代码以及费用的清单。“提供者目录”作为电子邮件的附件发送给提供服务者。

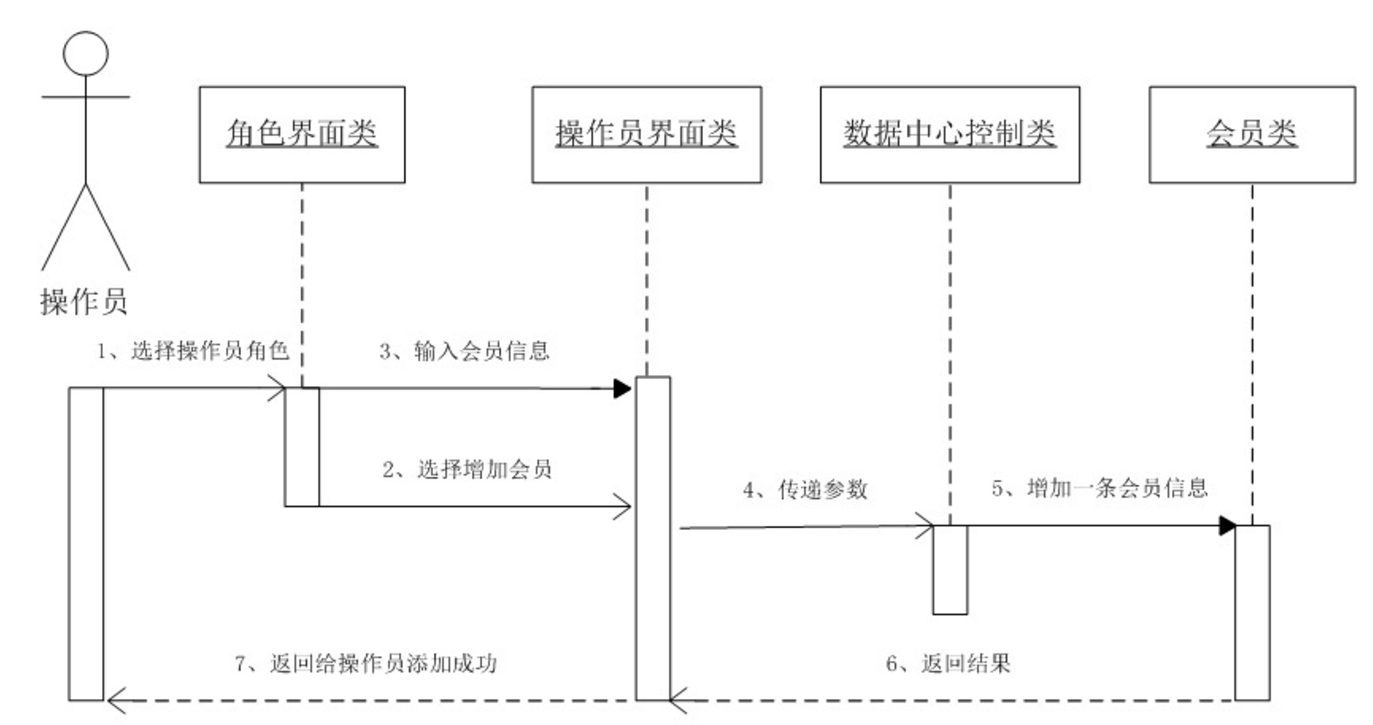
## 3、项目设计

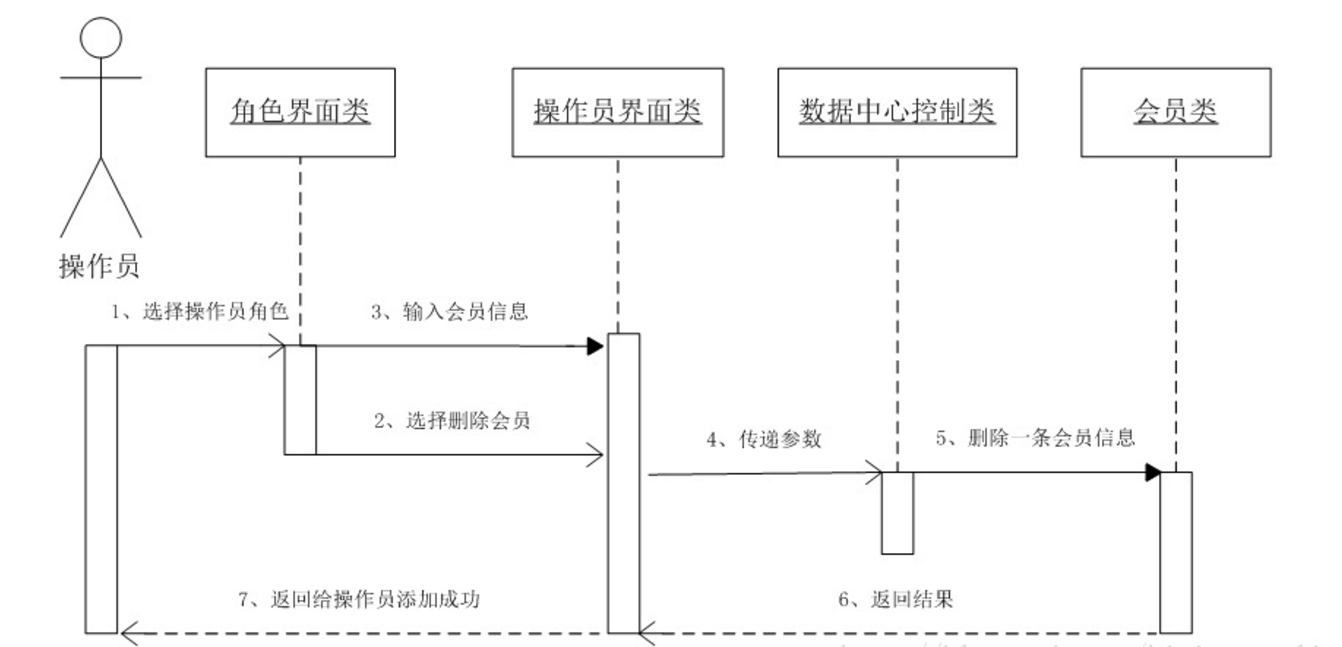
### 3.1角色顺序图

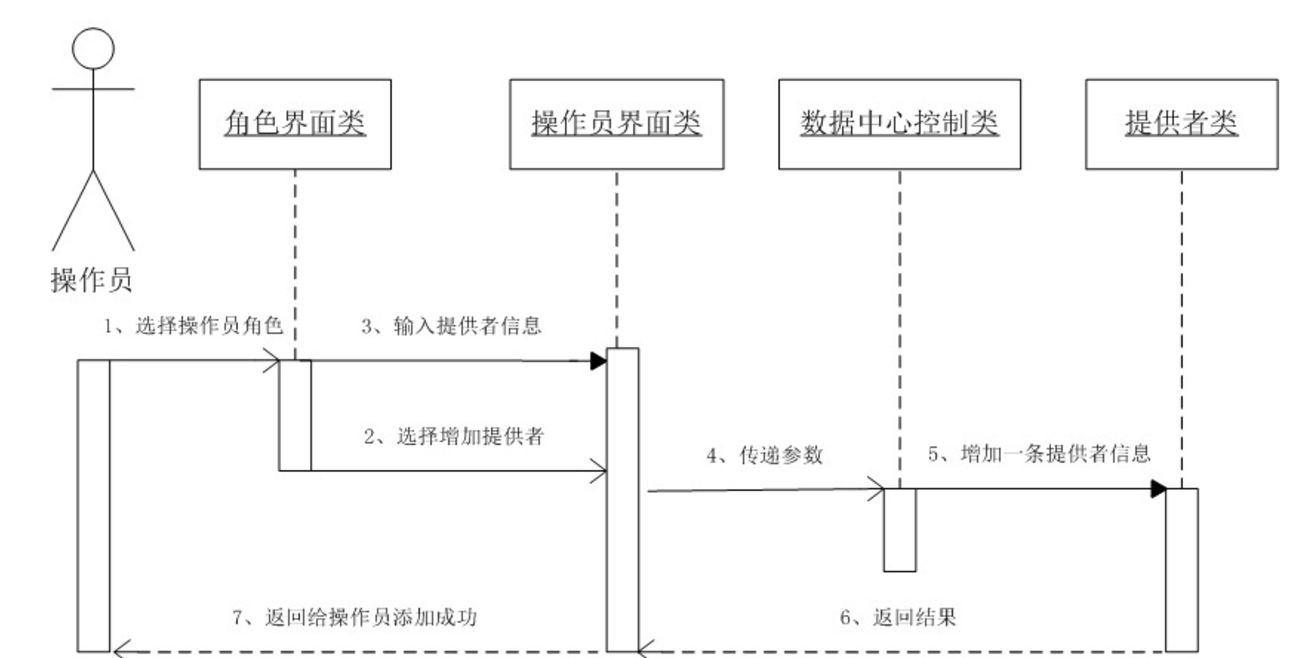
#### 3.1.1 提供者

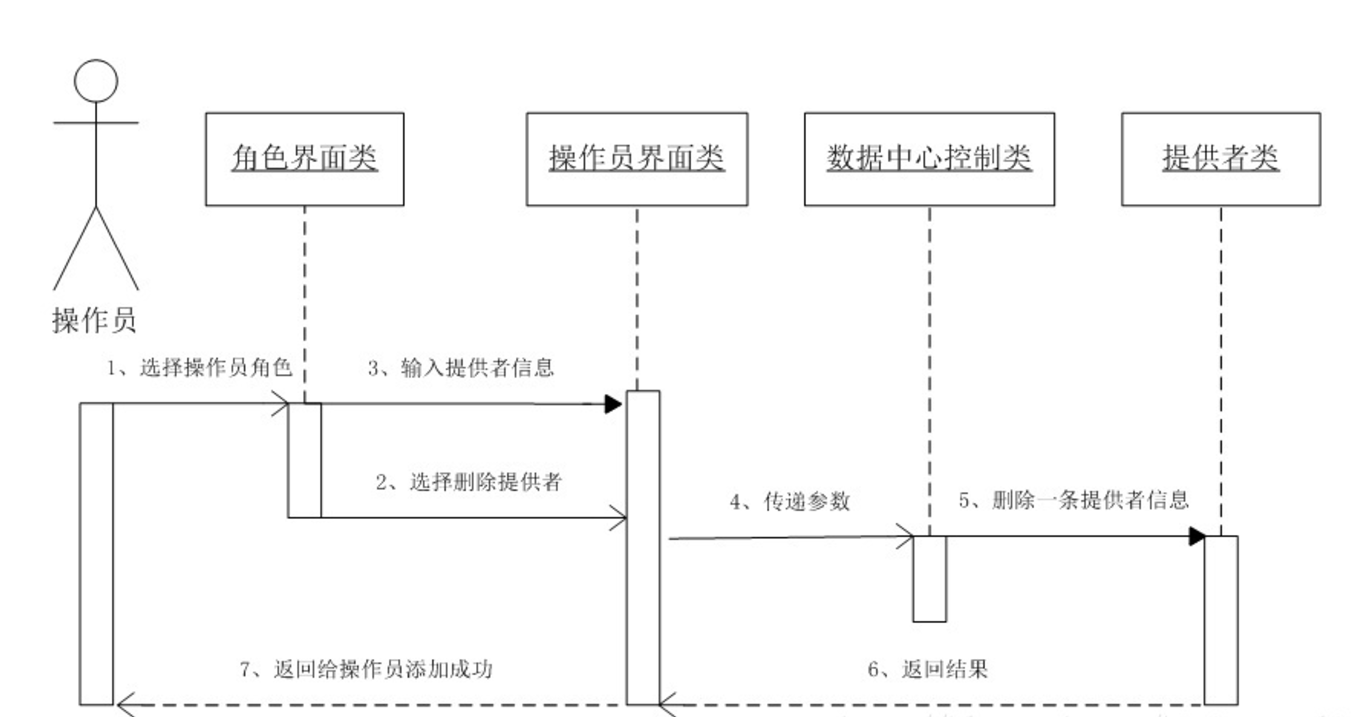


#### 3.1.2 操作员

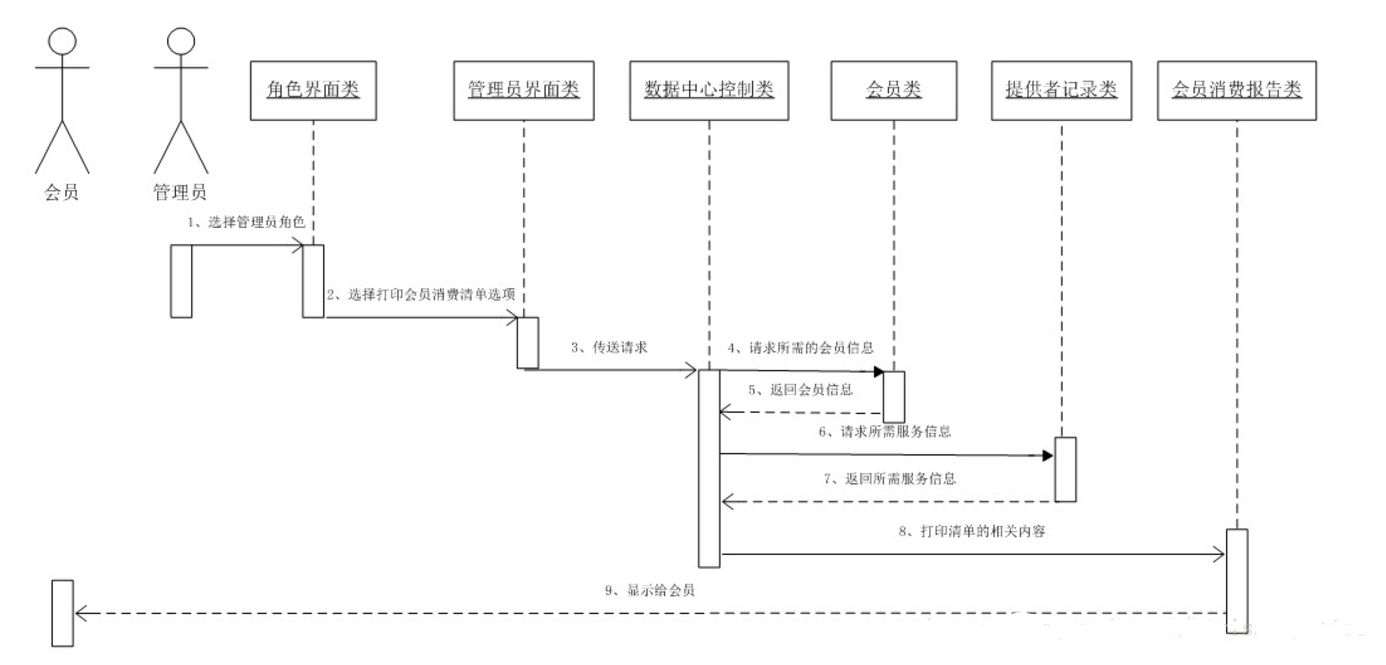


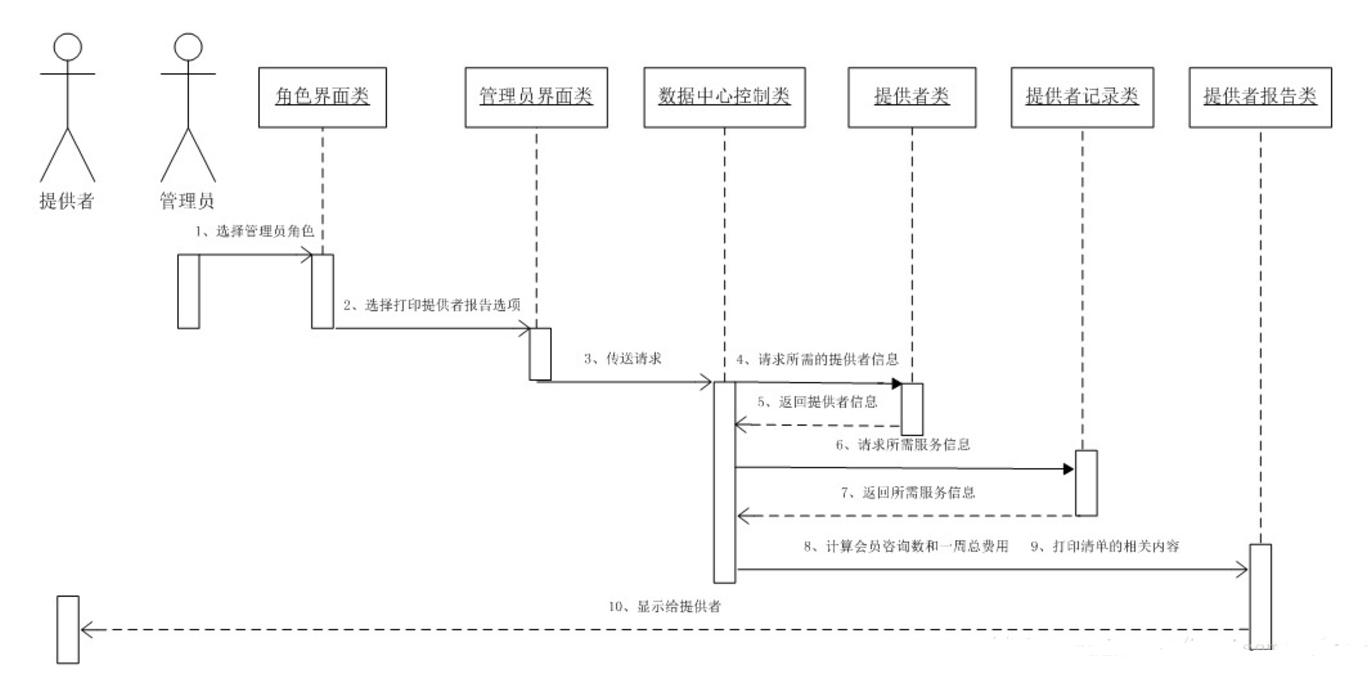


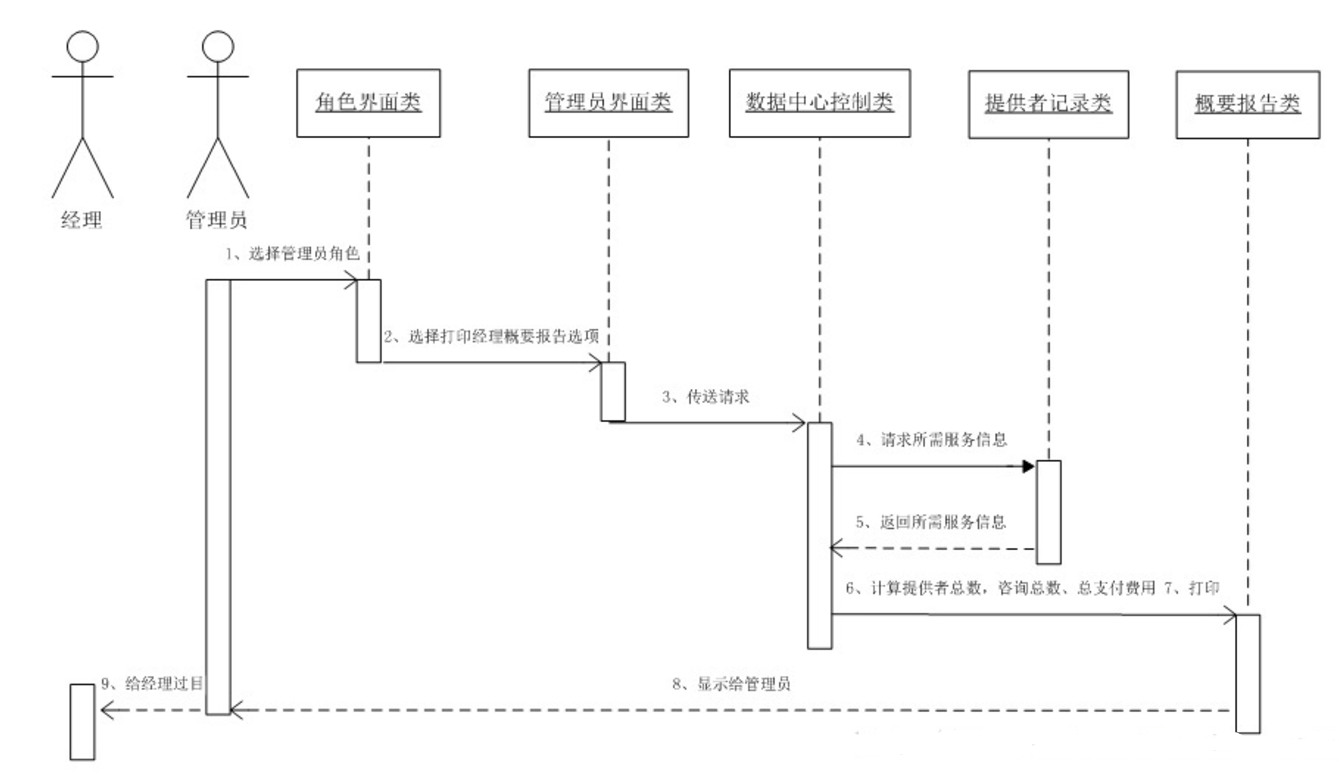




#### 3.1.3 管理员

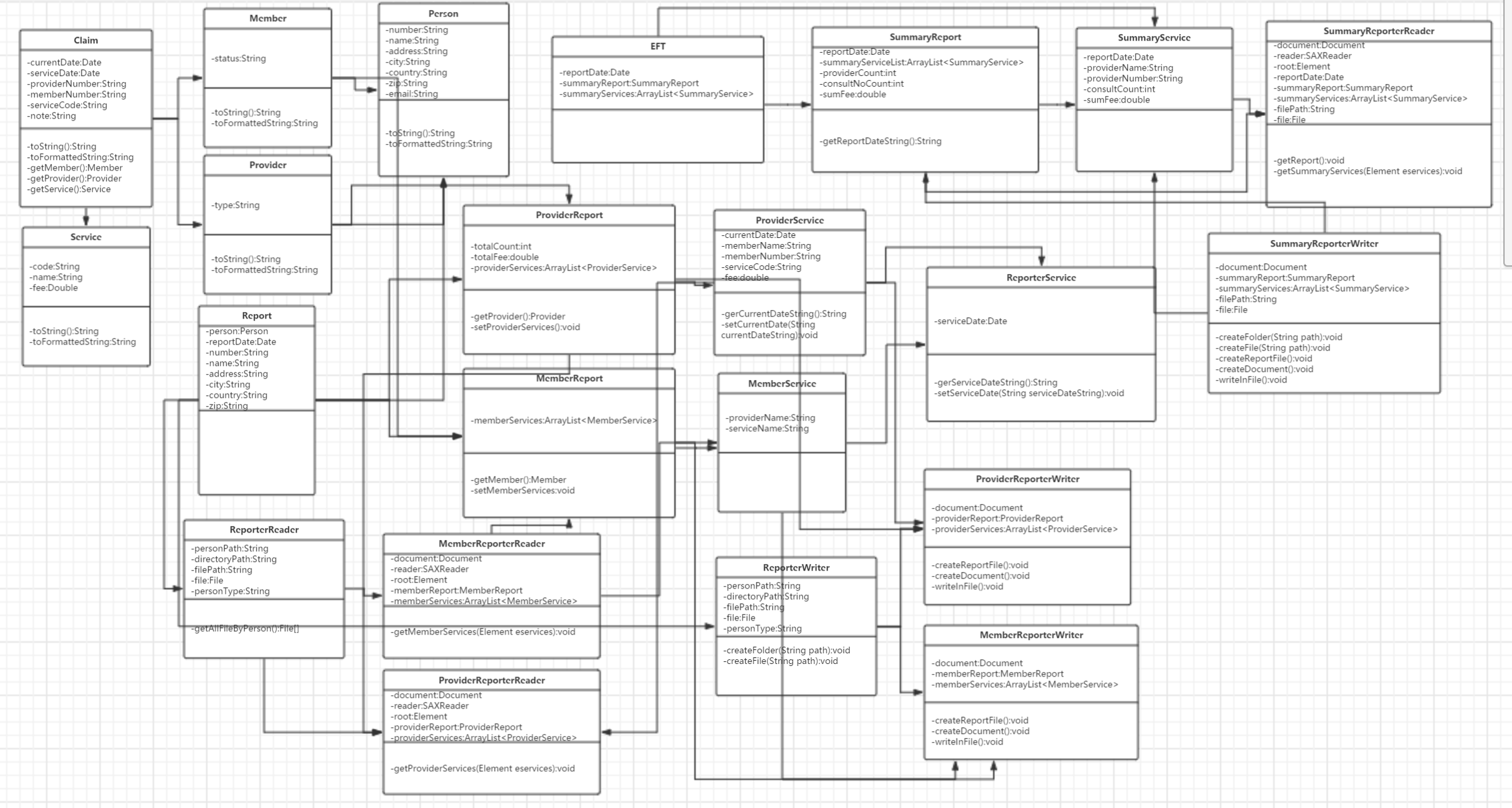






### 3.2 类和接口设计

#### 3.2.1 类的设计与实现



#### 3.2.2 外部接口

由于本系统仅实现了ChocAn数据处理的功能，所以需要提供、访问一些外部的接口，以实现ChocAn数据处理系统与ChocAn提供者终端、Acme会计服务公司通信软件、EFT等组件的系统交互。

### 3.3 数据库设计

#### 3.3.1 数据库系统的选择

ChocAn数据量不大，对数据要求不高，所以采用了xml的格式，并将数据存储于txt文件中。

#### 3.3.2 数据存储设计

















### 3.4出错处理设计

#### 3.4.1 出错输出信息

在业务处理流程中，若出现了异常错误，需要在屏幕上呈现相应的错误信息，让使用者得到接收到相应的反馈，提高系统的交互能力。

#### 3.4.2 数据库操作出错处理

由于本系统采用xml的格式存储数据，这样的方式较为简单，不能提供回滚处理，所以我们实现系统时如果在进行数据库操作时出现错误，要注意对数据进行手动的回滚处理。

# README文件

## 1、项目来源：

武汉大学开源软件工程

## 2、环境设置

* jdk1.7以上
* Tomcat7以上
* jquery3以上
* dom4j

## 3、项目简介

本项目实现了一个针对巧克力爱好者的管理软件，用于巧克力爱好者协会的管理，包括人员管理、账单管理以及报表生成。

项目分为前端用户界面和后端信息处理两个部分，前端与后台的交互采用jsp与ajax结合的形式进行交互， 数据存储在xml文件之中，并且采用XML Schema的文件对每一个数据的数据结构进行约束。在对XML操作上，本项目用dom4j作为操作库。

## 4、代码目录结构

* WebChatOn
  + src --后端代码
    - object --基础数据对象
    - path --获取程序地址
    - report --报表格式
    - reportreader --从生成报表
    - reportservice --报表子部分
    - reportwriter --将报表数据写入存储文件
    - ui --起始界面交互
    - uiEFT --金融报表界面交互
    - uimanager --管理者界面交互
    - uioperator --操作员界面交互
    - user --基础数据存储
  + WebRoot --前端代码
    - js --前端依赖的js包
    - jsp --前端页面
    - report --报表存储文件
    - user --基础数据存储文件

## 5、安装说明

本软件可直接用eclipse导入，但是要修改项目配置中的jdk版本

# 源代码文档

## 1、数据对象代码（以“Person.java”为例）

package object;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public abstract class Person {

//成员变量

private String number;

private String name;

private String address;

private String city ;

private String country;

private String zip;

private String email;

protected static final char SEPARATOR = '#';

public static final int NAME\_LENGTH = 25;

public static final int STREET\_LENGTH = 25;

public static final int CITY\_LENGTH = 14;

public static final int STATE\_LENGTH = 2;

public static final int ZIP\_LENGTH = 5;

public Person(String number,String name,String address,String city,String country,String zip,String email){

this.number=number;

this.name=name;

this.address=address;

this.city=city;

this.country=country;

this.zip=zip;

this.email=email;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*accessor methods

/\*\* Returns the person's unique number

\* @return the person's unique number

\*/

public String getNumber()

{

return number;

}//getNumber

/\*\*

\* Returns the person's name

\* @return the person's name

\*/

public String getName()

{

return name;

}//getName

/\*\*

\* Returns the person's street address

\* @return the person's street address

\*/

public String getAddress()

{

return address;

}//getStreet

/\*\*

\* Returns the person's city

\* @return the person's city

\*/

public String getCity()

{

return city;

}//getCity

/\*\*

\* Returns the person's state

\* @return the person's state

\*/

public String getCountry()

{

return country;

}//getState

/\*\*

\* Returns the person's zip code

\* @return the person's zip code

\*/

public String getZip()

{

return zip;

}//getZip

/\*\*

\* Returns the person's email address

\* @return the person's email address

\*/

public String getEmail()

{

return email;

}//getEmail

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*mutator methods

public void setNumber(String aNumber){

if (aNumber == null || aNumber.length() == 0)

throw new IllegalArgumentException("A number is required");

String regEx = "^[1-9]\*$";

Pattern pattern = Pattern.compile(regEx);

Matcher matcher = pattern.matcher(aNumber);

if(matcher.matches()&&aNumber.length()==9){

number=aNumber;

}

else{

throw new IllegalArgumentException("Number code must be exactly "

+ ZIP\_LENGTH + " digits");

}

}

/\*\* Changes the person's name

\* @param aName the new name

\* @throws IllegalArgumentException if aName is null, an empty string

\* or longer than NAME\_LENGTH characters

\*/

public void setName(String aName)

{

if (aName == null || aName.length() == 0)

throw new IllegalArgumentException("A name is required");

else if (aName.length() > NAME\_LENGTH)

throw new IllegalArgumentException("Name may not be more than "

+ NAME\_LENGTH + " characters");

name = aName;

}//setName

/\*\* Changes the person's street address

\* @param aStreet the new street address

\* @throws IllegalArgumentException if aStreet is not null

\* and is longer than STREET\_LENGTH characters

\*/

public void setAddress(String aAddress)

{

if (aAddress == null) aAddress = "";

else if (aAddress.length() > STREET\_LENGTH) //street is not required

throw new IllegalArgumentException("Street may not be more than "

+ STREET\_LENGTH + " characters");

address = aAddress;

}//setStreet

/\*\* Changes the person's city

\* @param aCity the new city

\* @throws IllegalArgumentException if aCity is not null

\* and is longer than CITY\_LENGTH characters

\*/

public void setCity(String aCity)

{

if (aCity == null) aCity = "";

else if (aCity.length() > CITY\_LENGTH) //city is not required

throw new IllegalArgumentException("City may not be more than "

+ CITY\_LENGTH + " characters");

city = aCity;

}//setCity

/\*\* Changes the person's state

\* @param aState the new state

\* @throws IllegalArgumentException if aState is not

\* exactly STATE\_LENGTH letters, null or an empty string

\*/

public void setCountry(String aCountry)

{

if (aCountry == null) aCountry = "";

else if (aCountry.length() > 0) //state is not required

{

if (aCountry.length() != STATE\_LENGTH)

throw new IllegalArgumentException("State must be exactly "

+ STATE\_LENGTH + " letters");

//test whether each character is a letter

for (int i = 0; i < STATE\_LENGTH; i++)

if(! Character.isLetter(aCountry.charAt(i)))

throw new IllegalArgumentException("State must be "

+ STATE\_LENGTH + " letters only");

}

country = aCountry;

}//setState

/\*\* Changes the person's zip code

\* @param aZip the new zip code

\* @throws IllegalArgumentException if aZip

\* is not null or an empty string and

\* is not composed entirely of digits or

\* is longer than ZIP\_LENGTH digits

\*/

public void setZip(String aZip)

{

if (aZip == null) aZip = ""; //Replace null with an empty string

else if (aZip.length() > 0) //if given, zip must be 5 digits

{

//test for correct length

if (aZip.length() != ZIP\_LENGTH)

throw new IllegalArgumentException("Zip code must be exactly "

+ ZIP\_LENGTH + " digits");

//test whether each character is a digit

for (int i = 0; i < ZIP\_LENGTH; i++)

if (!Character.isDigit(aZip.charAt(i)))

throw new IllegalArgumentException("Zip code must be "

+ ZIP\_LENGTH + " digits only");

}

zip = aZip;

}//setZip

/\*\* Changes the person's email address. No validation is done.

\* @param anEmail the new email address

\*/

public void setEmail (String anEmail)

{

if (anEmail == null) anEmail = "";

email = anEmail;

}//setEmail

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*utility methods

/\*\* Returns a string representation of the person consisting of the values,

\* converted to strings, of all the instance variables separated by

\* the character SEPARATOR.

\* @return the string representation of the person.

\*/

public String toString()

{

return "" + number + SEPARATOR + name + SEPARATOR

+ address + SEPARATOR + city + SEPARATOR + country + SEPARATOR

+ zip + SEPARATOR + email;

}//toString

/\*\* Changes all the instance variables to the values given by the string

\* representation of the person.

\* @param data the string representation of the person

\* @throws NumberFormatException if the person number

\* is not a valid integer

\* @throws IllegalArgumentException if any of the values are

\* invalid.

\* @throws IndexOutOfBoundsException if there are

\* not enough values in the string

\*/

/\*\* Returns a string representation of the person in a format that is

\* suitable for text display.

\* @return a formatted string representation of the person

\*/

public String toFormattedString()

{

String personString = "Number:" + number

+ "\nName:" + name

+ "\nAddress:" + address

+ "\nCity:" + city

+ "\nCountry:" + country

+ "\nZip Code:" + zip

+ "\nEmail:" + email;

return personString;

}

}

## 2、数据处理代码（以“Members.java”为例）

package user;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.util.ArrayList;

import java.util.List;

import org.dom4j.Document;

import org.dom4j.DocumentException;

import org.dom4j.DocumentHelper;

import org.dom4j.Element;

import org.dom4j.Node;

import org.dom4j.io.OutputFormat;

import org.dom4j.io.SAXReader;

import org.dom4j.io.XMLWriter;

import object.Member;

import path.Path;

public class Members {

private String filepath=Path.filepath+"/WebRoot/user/Member/Members.xml";

private String personType="Member";

//成员变量

private Document document;

private Element root;

private ArrayList<Member> memberList;

private ArrayList<String> numberList;

public Members()

{

memberList = new ArrayList<Member>();

numberList = new ArrayList<String>();

init();

}

@SuppressWarnings("unchecked")

private void init(){

try {

SAXReader reader = new SAXReader();

document = reader.read(filepath);

root=document.getRootElement();

List<Element> elementList=root.elements();

if(elementList.size()>0){

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

Element emember = elementList.get(i);

String number=emember.attributeValue("number");

String name=emember.element("name").getText();

String address=emember.element("address").getText();

String city=emember.element("city").getText();

String country=emember.element("country").getText();

String zip=emember.element("zip").getText();

String email=emember.element("email").getText();

String status=emember.element("status").getText();

Member member=new Member(number,name,address,city,country,zip,email,status);

memberList.add(member);

numberList.add(number);

}

}

} catch (DocumentException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

private void writeInFile(){

try {

FileOutputStream out= new FileOutputStream(filepath);

OutputFormat format = OutputFormat.createPrettyPrint();

format.setEncoding("utf-8");

XMLWriter writer;

writer = new XMLWriter(out,format);

writer.write(document);

writer.close();

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public Member search(String number){

Member member = null;

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

if(numberList.get(i).equals(number)){

member=memberList.get(i);

}

}

return member;

}

public ArrayList<Member> getAllMember(){

return memberList;

}

public boolean add(Member member){

String number=member.getNumber();

if(numberList.contains(number)){

return update(member);

}

String name=member.getName();

String address=member.getAddress();

String city=member.getCity();

String country=member.getCountry();

String zip=member.getZip();

String email=member.getEmail();

String status=member.getStatus();

Element emember = DocumentHelper.createElement(personType);

emember.addAttribute("number", String.valueOf(number));

Element ename=emember.addElement("name");

Element eaddress=emember.addElement("address");

Element ecity=emember.addElement("city");

Element ecountry=emember.addElement("country");

Element ezip=emember.addElement("zip");

Element eemail=emember.addElement("email");

Element estatus=emember.addElement("status");

ename.setText(name);

eaddress.setText(address);

ecity.setText(city);

ecountry.setText(country);

ezip.setText(zip);

eemail.setText(email);

estatus.setText(status);

root.add(emember);

writeInFile();

init();

return true;

}

public boolean remove(String number){

if(!numberList.contains(number)){

return false;

}

else{

String xpath="/Members/Member[@number="+number+"]";

Node member = document.selectSingleNode(xpath);

root.remove(member);

writeInFile();

init();

return true;

}

}

public boolean remove(Member member){

String number=member.getNumber();

return remove(number);

}

public boolean update(Member member){

if(remove(member)){

return add(member);

}

else{

return false;

}

}

}

## 3、报表生成代码（以“MemberReportReader.java”为例）

package reportreader;

import java.io.File;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import org.dom4j.Document;

import org.dom4j.DocumentException;

import org.dom4j.Element;

import org.dom4j.io.SAXReader;

import object.Member;

import report.MemberReport;

import reportservice.MemberService;

import user.Members;

public class MemberReportReader extends ReportReader{

private static final String personType="member";

private static final String DATE\_FORMAT = "MM-dd-yyyy";

private static SimpleDateFormat dateFormatter = new SimpleDateFormat();

//成员变量

private Document document;

private SAXReader reader;

private Element root;

private MemberReport memberReport;

private ArrayList<MemberService> memberServices;

public MemberReportReader(Member member,Date reportDate){

super(member,reportDate,personType);

memberServices=new ArrayList<MemberService>();

try {

this.reader = new SAXReader();

this.document = reader.read(getFilePath());

this.root=document.getRootElement();

} catch (DocumentException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public MemberReportReader(File file){

super(file);

memberServices=new ArrayList<MemberService>();

try {

this.reader = new SAXReader();

this.document = reader.read(getFilePath());

this.root=document.getRootElement();

} catch (DocumentException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public MemberReportReader(String filepath){

super(filepath);

memberServices=new ArrayList<MemberService>();

memberServices=new ArrayList<MemberService>();

try {

this.reader = new SAXReader();

this.document = reader.read(getFilePath());

this.root=document.getRootElement();

} catch (DocumentException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public MemberReport getMemberReport(){

try {

Element enumber=root.element("MemberNumber");

Element ereportDate=root.element("ReportDate");

String number=enumber.getText();

String reportDateString= ereportDate.getText();

Members members=new Members();

Member member=members.search(number);

dateFormatter = new SimpleDateFormat(DATE\_FORMAT);

Date reportDate = dateFormatter.parse(reportDateString);

Element eservices=root.element("Services");

getMemberServices(eservices);

memberReport=new MemberReport(member,reportDate,memberServices);

return memberReport;

} catch (ParseException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return memberReport;

}

private void getMemberServices(Element eservices){

List<?> eserviceList=eservices.elements();

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

try {

Element eservice=(Element)eserviceList.get(i);

Element eserviceDate=eservice.element("ServiceDate");

Element eproviderName=eservice.element("ProviderName");

Element eserviceName=eservice.element("ServiceName");

String serviceDateString=eserviceDate.getText();

String providerName=eproviderName.getText();

String serviceName=eserviceName.getText();

dateFormatter = new SimpleDateFormat(DATE\_FORMAT);

Date serviceDate = dateFormatter.parse(serviceDateString);

MemberService memberService=new MemberService(serviceDate,providerName,serviceName);

memberServices.add(memberService);

} catch (ParseException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

}

## 4、Servlet交互代码（以“OperatorChoose.java”为例）

package uioperator;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.ServletConfig;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import object.Member;

import object.Provider;

import object.Service;

import user.Members;

import user.Providers;

import user.Services;

/\*\*

\* Servlet implementation class OperatorChoose

\*/

@WebServlet("/OperatorChoose")

public class OperatorChoose extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public OperatorChoose() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see Servlet#init(ServletConfig)

\*/

public void init(ServletConfig config) throws ServletException {

// TODO Auto-generated method stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

ArrayList<String> providerNumberList=new ArrayList<String>();

Providers providers=new Providers();

ArrayList<Provider> providerList=providers.getAllProvider();

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

providerNumberList.add(providerList.get(i).getNumber());

}

ArrayList<String> memberNumberList=new ArrayList<String>();

Members members=new Members();

ArrayList<Member> memberList= members.getAllMember();

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

memberNumberList.add(memberList.get(i).getNumber());

}

ArrayList<String> serviceCodeList=new ArrayList<String>();

Services services=new Services();

ArrayList<Service> serviceList=services.getAll();

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

serviceCodeList.add(serviceList.get(i).getCode());

}

String choose=request.getParameter("option");

switch (choose){

case "option1":

request.getSession().setAttribute("providerNumberList",providerNumberList);

request.getRequestDispatcher("ProviderCreate.jsp").forward(request,response);

break;

case "option2":

request.getSession().setAttribute("providerList", providerList);

request.getRequestDispatcher("ProviderDelete.jsp").forward(request,response);

break;

case "option3":

request.getSession().setAttribute("providerList", providerList);

request.getRequestDispatcher("ProviderChange.jsp").forward(request,response);

break;

case "option4":

request.getSession().setAttribute("providerList", providerList);

request.getRequestDispatcher("ProviderShow.jsp").forward(request,response);

break;

case "option5":

request.getSession().setAttribute("memberNumberList",memberNumberList);

request.getRequestDispatcher("MemberCreate.jsp").forward(request,response);

break;

case "option6":

request.getSession().setAttribute("memberList", memberList);

request.getRequestDispatcher("MemberDelete.jsp").forward(request,response);

break;

case "option7":

request.getSession().setAttribute("memberList", memberList);

request.getRequestDispatcher("MemberChange.jsp").forward(request,response);

break;

case "option8":

request.getSession().setAttribute("memberList", memberList);

request.getRequestDispatcher("MemberShow.jsp").forward(request,response);

break;

case "option9":

request.getSession().setAttribute("serviceCodeList",serviceCodeList);

request.getRequestDispatcher("SerivceCreate.jsp").forward(request,response);

break;

case "option10":

request.getSession().setAttribute("serviceList", serviceList);

request.getRequestDispatcher("ServiceDelete.jsp").forward(request,response);

break;

case "option11":

request.getSession().setAttribute("serviceList", serviceList);

request.getRequestDispatcher("ServiceChange.jsp").forward(request,response);

break;

case "option12":

request.getSession().setAttribute("serviceList", serviceList);

request.getRequestDispatcher("ServiceShow.jsp").forward(request,response);

break;

}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

## 5、前端代码（以“MemberReportShow.jsp”为例）

<%@ page language="java" import="java.util.\*" import="object.Member" pageEncoding="UTF-8"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPE HTML>

<html>

<head>

<base href="<%=basePath%>">

<title>My JSP 'MemberShow.jsp' starting page</title>

<script type="text/javascript" src=<%=basePath+"js/jquery-3.2.1.min.js" %>></script>

</head>

<body>

<%

Object omemberList=session.getAttribute("memberList");

ArrayList<Member> memberList=new ArrayList<Member>();

if(omemberList!=null){

memberList=(ArrayList<Member>)omemberList;

}

%>

<table id="table" border="1">

<tr>

<th>会员编号</th>

<th>会员姓名</th>

<th>会员地址</th>

<th>会员城市</th>

<th>会员国家</th>

<th>会员zip码</th>

<th>会员邮箱</th>

<th>会员状态</th>

</tr>

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

Member member=memberList.get(i); %>

<tr>

<td><%=member.getNumber() %></td>

<td><%=member.getName() %></td>

<td><%=member.getAddress() %></td>

<td><%=member.getCity() %> </td>

<td><%=member.getCountry() %> </td>

<td><%=member.getZip() %></td>

<td><%=member.getEmail() %></td>

<td><%=member.getStatus() %> </td>

</tr>

<% } %>

</table>

<form action="jsp/OperatorChoose.jsp" method="get">

<input type="submit" value="确认">

</form>

</body>

</html>

# 测试文档

这里测试主要以前端界面方式展示。

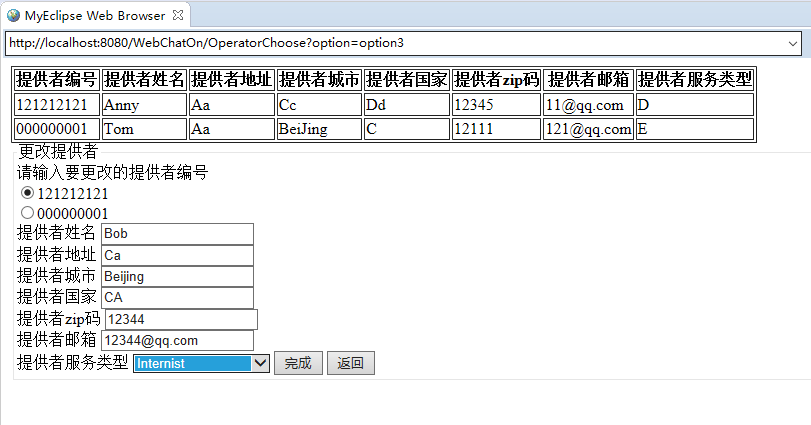
## 1、统一认证界面

## 2、提供者界面—查看创建的记录

## 3、提供者界面—查看提供者目录

## 4、操作员界面—主界面

## 5、操作员界面—更改提供者



## 6、操作员界面—查看提供者列表

## 7、管理员界面—显示提供者报告

## 8、EFT界面—打印选择的EFT数据