Term Project: *ChocAn*

Design Document

Table of Contents

1 Introduction

This document is an overview of the design and its considerations for the database management system commissioned by the health insurance company ChocAn. The database will be used to track patient records, services performed, and costs. This document only covers the design of a portion of this full system. Functionality is outlined in the requirements document, and includes giving health care providers the ability to track and maintain patients (members), services, and associated costs.

This document includes examination of the target audience for this system, any necessary terms and definitions and general design considerations. Also a deeper view of the system design is given here, describing in detail the objects and entities that the system will consist of. Sample code is also given for some methods, to demonstrate how the method should behave. Several diagrams are included to show the system architecture from both a high-level perspective, down to the individual class and method interaction.

1.1 Purpose and Scope

This document serves as a central location for most design considerations of the ChocAn database system, from a high-level overview of the system's interaction with other existing ChocAn systems, to low-level descriptions of classes and their behavior when interacting with other classes in the system. Some things not included here are a full requirements overview (see requirements document) or unit testing design/implementation (see test plan document).

1.2 Target Audience

The target audience for this document are software engineers who are interested in the documentation, maintenance, or extension of the system. Engineers are expected to have a basic knowledge of Java, as that is the language of the system’s implementation. Other interested parties may include some stakeholders, however the low-level diagrams and descriptions may be too technical, as this document primary audience are software engineers.

1.3 Terms and Definitions

Term	Definition
Database	Group 4 has been tasked with the creation of a software system that can store ChocAn's partner and customer data, organize and edit said data, as well as use the data to generate reports about ChocAn's clients (providers of services or receivers of services) and ChocAn's financial status in terms of specific money owed to specific partners. In this document ChocAn's database is referred to as "database" or "the database."
Provider	A healthcare provider who uses ChocAn's database to deliver health services and their related expenses to customers. Note that ChocAn's database is to help providers send individualized member reports to the members they help, detailing the services they have acquired in the last week. Also, the database is to deliver provider reports to each individual provider detailing a week of provided services.
Member	An individual insured by ChocAn who receives medical assistance from a ChocAn provider. ChocAn's database contains a history of services, including the provider involved, for each ChocAn member. Records regarding payment status of members is kept although the database does not process individual member payments. Used interchangeably with "patient".
Manager	A ChocAn employee who is authorized by ChocAn to add members to ChocAn's database, remove members from ChocAn's database, and to otherwise edit member records. ChocAnalyst are also privy to the electronic funds transfer data file and are able to generate summary reports (see below). Used interchangeably with "ChocoAnalyst".
Member Report	A report the database generates on a provider's behalf and sends to a specific member detailing health services used in that week, if any used.
Provider Report	A report the database generates for each individual provider detailing a listing of services provided, a total count of consultations provided, and the total cost of services provided for that week.
Manager Report	A report the database generates for the manager detailing every provider to be paid for that week, the number of consultations each had, and the provider's total fee for that week. Also, the report details the total number of consultations provided

	that week as well as the total amount of money to be paid out.
Electronic Funds Transfer Report	A report detailing each provider’s name, number, and the funds to be transferred to that provider for that week. Used interchangeably with “EFT report”.
Services	In this context a service will often imply a ChocAn covered health care service.
Consults	Consults consist of a member, a provider, and a fee. A consult is not a service but an occurrence of a service for which ChocAn will be billed for.
Fees	In this context a fee refers to money ChocAn must pay to a provider for consults, a week's worth of consults, or some other grouping of consults. Fees members pay to ChocAn are handled by a third party entity. This database deals only with fees paid by ChocAn to providers.
Valid Member	A member status and a way of referring to a type of member. A manager determines a member to be valid if that member has paid their membership dues to ChocAn.
Invalid Member	A member status and a way of referring to a type of member. A manager determines a member to be invalid if that member has failed to pay their membership dues to ChocAn.
High-Level	High level aspects of this system are classes, functionality the classes are meant to provide, and public function (and their names) the classes are to provide that all members of Group 4 expect to exist. High level also refers to the control flow of the main class.
Mid-Level	Mid level aspects of this system are the return values (or lack thereof) and parameter lists of high level components that are crucial for interclass communication that all members of Group 4 will rely on in order to integrate the system into a whole.
Low-Level	Low level aspects of this system are the particular implementations of high level classes that are of concern only to the engineer or engineers tasked with the implementation of a specific class or group of dependent classes.
Login Terminal	The first control branch of the system as well as the first component of the system the user encounters. The login terminal is used to differentiate a manager from a provider and then to direct the user to the appropriate menu.
Group 4	A 3rd party software engineering firm. The engineers hired to implement ChocAn’s database.

2 Design Considerations

Listed here are constraints and dependencies group 4 has taken into consideration while designing this system. It includes both functional and nonfunctional requirements that were considered during the design of the system. This section also discusses Group 4's methodology behind the design stage, and going forward.

2.1 Constraints and Dependencies

Being a healthcare database, this program shall adhere to standards set forth by HIPAA (Health Insurance Portability and Accountability Act). Complying with these rules ensure that confidential customer information is kept secure, among other uptime constraints. In this vein, the system has been designed around providing different functionality based on the authorized credentials of the user.

Most other constraints placed on the system are based on the desired functionality of the system. While proper functionality will be delivered according to the requirements, the back-end mechanics of the system are left open to meet nonfunctional constraints in the best possible way. Good coding practice shall be used to protect the sensitive data.

A major dependency of this system is the data itself. This being a database application, functionality is built around stored data. However, future changes of the data format is possible, among other data additions or changes. This current snapshot of the system will operate around the data and its format as discussed in the Requirements Document.

This system shall be coded in Java, and therefore any necessary Java dependencies are applicable. The machines running the system should have any required libraries/dependencies installed to run a standard Java application.

2.2 Methodology

The engineering methodology for this project will be a hybrid of waterfall and incremental development. In compliance with the waterfall model, we work on the project in phases. These phases include the requirements definition, system and software design, implementation and unit testing, integration and system testing, and operation and maintenance. None of the phases are concurrent, since they only start when the previous phase has finished, cascading from one to the other.

In a traditional waterfall model, there is no backtracking between processes. However, we expect the implementation and unit testing stage to have some impact on the functionality of our system, prompting changes on the design and maybe the requirements. Therefore, at that stage we may have concurrent specification, development, and validation activities. Hence the implementation of a hybrid waterfall and incremental development models.

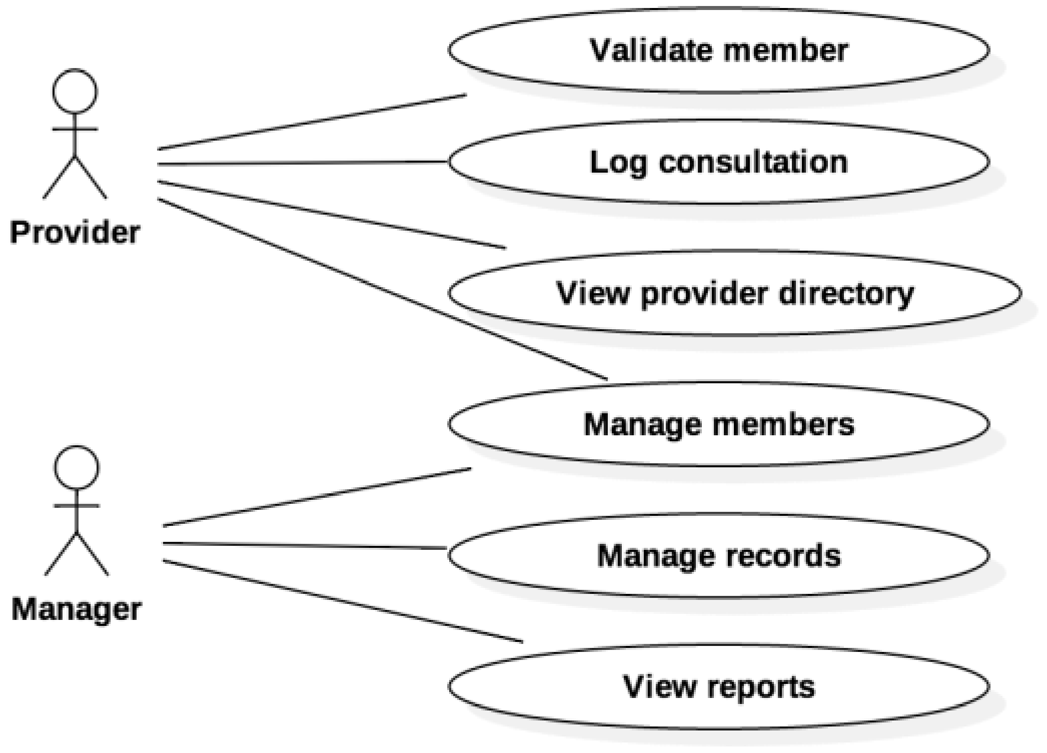
At the inception of this project, the customer presented us with an extensive plan of their needs and goals. Driven by that plan, and with the waterfall methodology in mind, we began the development process by defining the requirements, which produced the requirements document.

After the completion of the previous phase, we addressed system and software design, which is shown here in this design document. This document specifies the layout and design strategy of the project, and builds on the requirements defined on the preceding phase.

Finally, equipped with the specificity laid out in the design phase, we will move onto the integration and system testing phases. At this stage we will produce a test plan, while at the same time tackling the feasibility of the implementation of our design, and adapting our tests and design accordingly.

3 System Overview

The System Overview is a brief, high-level summary of the user base and functionality offered by the program. This section includes short descriptions of the intended users of the application and what they can use it for. The overview will be followed by an architectural overview and a detailed design of the system, showing how each of these high-level functions will be handled by the program.



Use Case Diagram

The System will interact with and be operated by two different user types: the Provider and the ChocAn Manager. ChocAn Members will never directly interact with the application. Reports shall be generated and sent automatically to Members, Providers, and Managers. A login screen will give a user the option to choose either a Provider or

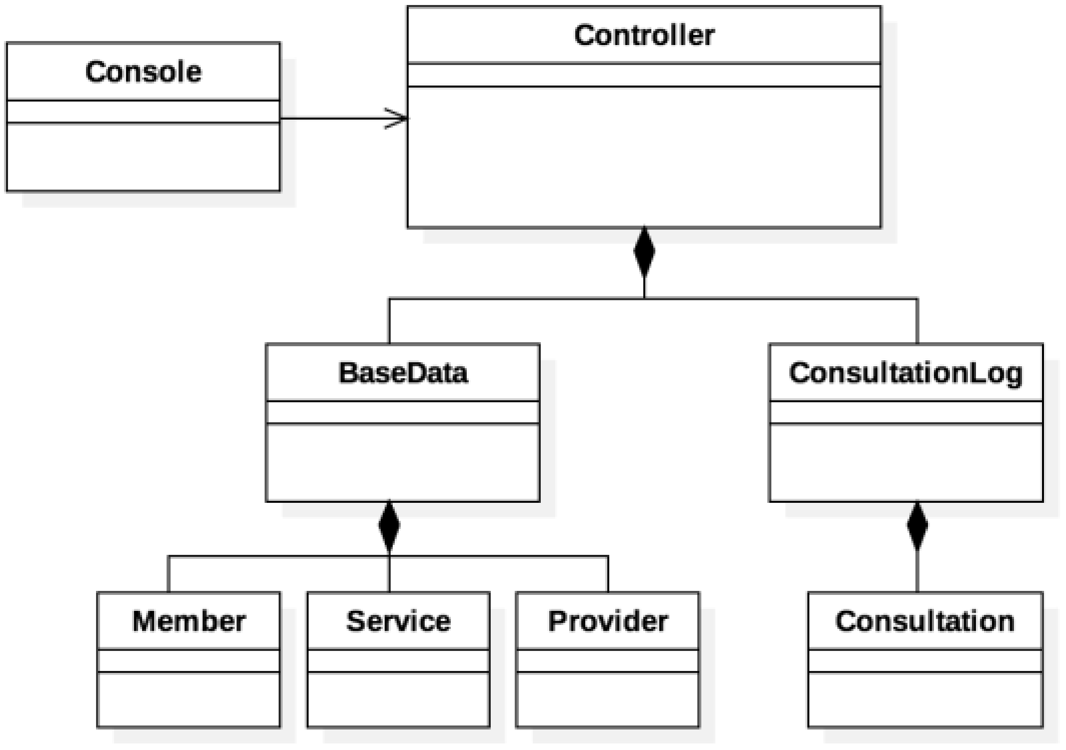
Manager login. Providers and Managers have different uses for the system, and upon logging in, the user will be presented with the options associated with their position.

The Provider’s functionality centers primarily around logging Services (consultations) provided for a Member. A Provider will have the option to validate the membership status of the Member they are currently providing a Service for. If the Member is validated by the system, the Provider will then have the option to log that consultation, which will be included in the various reports. If a Member is not valid, the system will inform the Provider, and logging a consultation for the invalid Member will not be an option. The Provider will have the option to view the Provider Directory, which is a list of every current service Provider registered with ChocAn. This will also be sent to the Provider’s email (output as a text file). Additionally, the Provider will be able to view a list of all current ChocAn Members. The Provider access will be a central functionality to the ChocAn terminal application.

The Manager’s functionality will be centered around managing and viewing various lists associated with the ChocAn service. Upon logging in, the Manager will be presented with a menu, where they will specify where to proceed. The Manager will have the option to view a complete list of current Members. The Manager will have the option to search this list for a specific member, either by name or by Member ID number. The Manager will be able to edit any entry on this Member list, with an additional option to completely delete a given Member from the list. The Manager will be able to add a new Member to the list. The same will be true of the Provider list that the Manager has access to: view all Providers, search by Provider ID number or name for a specific entry, edit an entry, and completely delete a Provider from the list. This will be the same list as the Provider Directory. The Manager will be able to view various reports generated by the system. These automatic reports are sent at predetermined times; however, the Manager will have access to view their current contents at any given time. The Manager access will be mainly for administrative and support purposes.

4 System Architecture

The system architecture contains eight subsystems, four of which are subsystems of the subsystems. These subsystems are Console, Controller, Database, Member, Service, Provider, ConsultationLog, and Consultation. Their relationship to each other is defined below.



Class Diagram

4.1 Console

The Console will encapsulate the user interface. The user logs on and all the user interactions will be done through console. Console provides a text based terminal interface for users to select options and input data.

4.2 Controller

The Controller initializes the database at startup and handle the interaction between Database and ConsultationLog.

4.3 ConsultationLog

The ConsultationLog will keep track of all services performed within ChocAn and handles the reporting. This represents all member provider interaction ever performed at ChocAn. This is effectively the database of the program and will be stored in a text file before the program shuts down.

4.3.1 Consultation

Consultation relates a member, provider, service, and key dates together. A consultation represents an interaction between a member and a provider. A single provider will perform a single service on a single member.

4.4 Database

This class will hold the instances of the three data classes: member, provider, and service. These represent all the entities that are required during a consultation with ChocAn.

4.4.1 Member

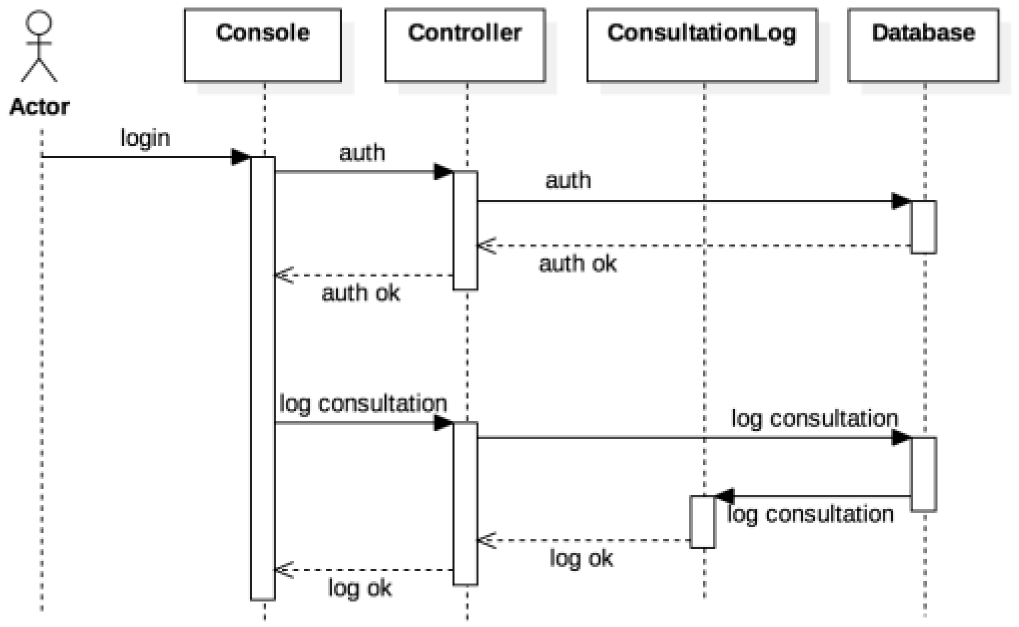
This is the person who comes to ChocAn looking for services who will be provided by a provider.

4.4.2 Provider

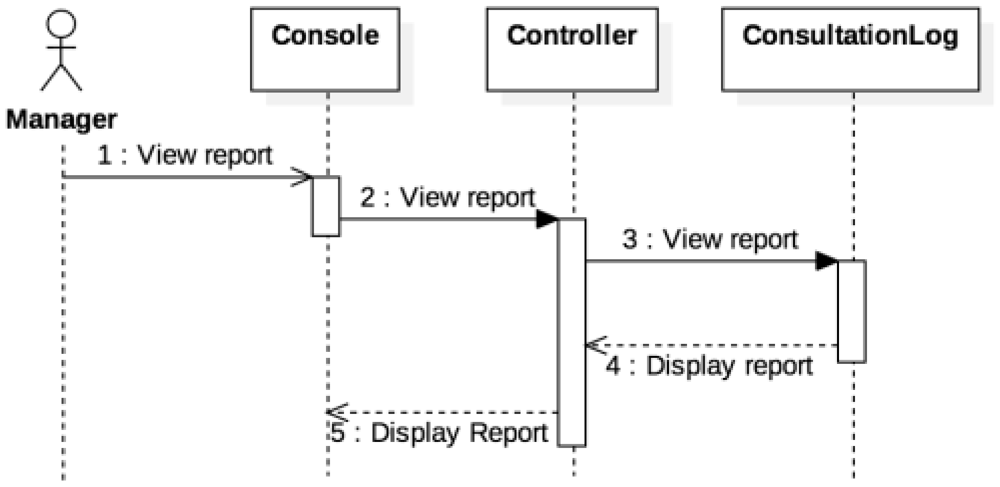
A provider provides a service to a member. A provider has elevated privileges that allows them to do reports, document and more.

4.4.3 Service

A service is an action that the provider provides to the member. The services are previously defined by ChocAn and will always be offered by a provider.



Sequence Diagram: Login and Log Consultation



Display Report

5 Detailed System Design

The following details all of the classes in the system and how they communicate with one another. Specific details regarding how the following functions are implemented are not included. Only to what class they belong, what their names are, the values they return, and what their parameters are.

5.1 Member Class

Name	**Member**
Data	String memberName String memberAddress int memberNumber int memberStatus
Public Methods	boolean isName (member) boolean isNumber (member) boolean printAddressFile (String) boolean printNumberFile (String) boolean printNameNumFile (String) boolean printNameFile (String) void printAddressScreen () void printNumberScreen () void printNameNumScreen () void printNameScreen ()

5.1.1 Member Data

The status integer is used to denote if the the member is an active member currently covered by ChocAn. Positive is current member, negative is former member, and 0 is an undefined status. The integer member number is a unique integer value assigned to each member, member numbers are greater than 0. The member number will be used by providers, managers, and the system's database to search for specific members. The member name is used by the system for searching purposes, as is the case with the member number. Also the member name will be used for identification purposes in several reports as is the case for the member number. Searching by member name shall be supported, though may not be required.

5.1.2 Member Methods

There are two sets of member methods, those that relate to searching, and those that relate to printing. Member searching methods that allow subsuming methods to search through list of members are `isName` and `isNumber`. Both of which take an instance of a member to be compared to the calling member and then signals through Boolean return value if the two have matching data.

There are two sets of member print methods. One set takes no parameters and prints the specified member data to the screen and returns nothing. The other set takes the name of the file (a string) to which the specified data will be set to. The print methods return a Boolean to check for valid file.

5.2 Provider Class

Name	**Provider**
Data	String providerName String providerAddress int providerNumber int totalConsultations int totalFee
Public Methods	Boolean isName (provider) Boolean isNumber (provider) void printNameScreen() void printNamNumScreen() void printAddressScreen() void printTotalConsultationsScreen() void printTotalFeeScreen() Boolean printNameFile (String) Boolean printNamNumFile (String) Boolean printAddressFile (String) Boolean printTotalConsultationsFile (String) Boolean printTotalFeeFile(String) int sumTotalConsultations (provider) int sumTotalFee(provider)

5.2.1 Provider Data

The provider class will have four data members: one string and three integers. The string will hold the provider’s name. One of the integers will be the provider’s number. Each provider is assigned a unique number to aid in identification, all numbers are greater than zero. The system will use both the provider name and the provider number to search through a list of providers for a specific provider. The remaining integer data members relate to billing. One tracks the total number of consultations a provider has had in the previous week, `totalConsultations`. The other tracks the total amount of money ChocAn owes to the provider for the serves the provider has given in the previous week, `totalFee`.

5.2.2 Provider Methods

The provider class has three sets of methods; the methods that relate to searching, those that relate to printing, and those that relate to calculating the total number of consultations within a group of providers and the total fee owed to a group of providers.

Search methods all take a provider instance as a parameter. They compare the passed provider to the calling provider's data and then return the results of the comparison in the form of a Boolean.

There are two sets of printing methods. The two sets are differentiated by where the print snippets of the provider's data. Methods that print the provider's data to an external file take the name (a string) of the file to which they will print data. They return true or false depending on whether the passed file name is a null string or not. The methods that print the member data to the screen take no parameters and return void.

Finally, to create the manager and EFT reports collections of providers will need their `totalConsultations` and `totalFee` data added to other providers. To facilitate that, the provider class will come with `sumConsultations` and `sumFee` methods that that both take a provider instance and return the sum of the passed provider and the calling providers data.

5.3 Service Class

Name	**Service**
Data	String serviceName int serviceNumber int serviceCost
Public Methods	Boolean isCode(service) void printNameScreen() void printCodeScreen() void printNameCodeScreen() void printCodeFeeScreen() Boolean printNameFile (String) Boolean printCodeFile (String) Boolean printNameCodeFile (String) Boolean printCodeFeeFile (String) int sumTotalFee (service)

5.3.1 Service Data

The service class will have three data members. The services name is represented as a string (`serviceName`) while the service code is represented as an integer (`serviceNumber`). The system will search for services by the service code and will identify particular services in reports by name, service code, or with both. The integer `serviceCost` is used to denote the cost of the service, that is, how much ChocAn owes a provider for giving the service.

5.3.2 Service Methods

Service methods are divided into three sets. Those sets are based on facilitation of searching, print, and determining total cost. There is one method that facilitates searching a list of services for a specific service, which is `isCode`. This method takes an instance of a service, compares the passed service to the calling service by service code and returns the result of the comparison as a Boolean. Similarly, there is one methods that supports summing the total fee of a list of services, `sumTotalFee`. This method operates similarly to `isCode`, differing in that the sum of the two services is returned.

The set of methods that support printing service data is divided into two based on if the method prints to screen or to an external file. Methods that print to screen take no parameters and return void. Methods that print to file take the name of the file (a string) to which the data in the service will be printed to. The methods will all return a Boolean, false if the passed string is a null reference.

5.4 Database Class

Class name	**Database**
Number of instances	One
Class(es) used by	Controller
Name of instantiated object(s)	liveDatabase;

Member data	TreeSet member_list;
	TreeSet service_list;
	TreeSet provider_list;

Public Methods	Boolean addMember (member) Boolean addProvider (provider) Boolean deleteMember (member) Boolean deleteProvider (provider) void printAllProvidersToScreen() void printAllMembersToScreen() Boolean printAllDataFile (String) Boolean isDuplicateMem (member) Boolean isDuplicateProv (provider) Boolean isDuplicateServ (service)

5.4.1 Database Data

The class has three data members, all of which are instances of TreeSet from Java's utility library. Each of the three sorted sets will be used to hold a list of providers, members, or services. Database will use searching and sorting methods built into TreeSet to implement its own methods.

5.4.2 Database Methods

There are 3 sets of methods in Database based on if the method prints data, adds/removes data, or searches data. Primarily for testing purposes, Database will be able to print all of its members and will be able to print all of its providers. It will also be able to print all of its data, including the service list, to an external file. That method, `printAllDataFile` will return false if the passed file is a null reference.

Another set of methods are methods that search for a particular entry in Database whether member, service, or provider. All of those methods are named after what they are searching for and have a single parameter, a reference to an object of the type the function is searching for. Database uses TreeSet functionality to have the passed object compare itself to each of the objects in the target set of objects. A true return from one of these methods signals that an object with identical *relevant* data has been found in Database, a false return signals to identical *relevant* data found. *Relevant* varies, for example members can have duplicate names but not have duplicate member numbers.

Finally, Database will be able to remove specific members and providers from itself, based on their unique identifying number. Each add and remove method deals with the addition or removal of an object of a specific type and so its name includes the type and if the type is to be removed or added. Each take a reference to an object of the type it is to remove or add and returns true or false based on whether the passed object has been successfully added, or if an object with information identical to the passed object has been deleted.

5.5 Consultation Class

Class name	**Consultation**
Number of instances	The number of consultations.
Name of instantiated object(s)	Objects will act as nodes, indexed by TreeSet.

Member data	member member_data;
	service service_data;
	provider provider_data;
	String datePerformed;
	String dateRecorded;

Public Methods	Boolean makeMemberReport (String) Boolean makeProviderReport (String) Boolean makeManagerReport (String) Boolean makeEftReport (String) int cmpDatePerf (consultation)

5.5.1 Consultation Data

A consultation is a digital representation of a physical consultation. As such it will have data members that represent each of the parties involved in a physical consultation: instances of a provider, a member, and a service. The class also has data members that represent when the consultation was recorded, `dateRecorded` and when the consultation was performed, `datePerformed`.

5.5.2 Consultation Methods

There are two sets of methods in the consultation class, those that generate reports, and a comparison method. The single comparison method, `cmpDatePerf` takes a reference to the object that will be compared to the calling object by their `datePerformed` data members. The method will return 1 if the passed object is older than the calling object, 0 if they the dates are the same, -1 if the passed object is younger, and -2 if the passed reference is null.

In addition to those methods, consultations will have functions to generate reports. Each of those classes will take the name of file their data is being printed to and return false if the passed file name is a null string. Each reporting method is named after the report it is to generate. The reporting methods will have the service, member, and provider print their own data to the passed file name.

5.6 ConsultationLog Class

Class name	**ConsultationLog**
Number of instances	One
Name of instantiated object(s)	ConsultationLog
Member data	TreeSet consultationList; int numConsultations int totalFee
Public Methods	Boolean addConsultation (consultation) Boolean printAllDataFile (String) Boolean makeMemberReport (String) Boolean makeProviderReport (String) Boolean makeMangReport (String) Boolean makeEftReport (String)

5.6.1 ConsultationLog Data

The ConsultationLog will have 1 data member, a TreeSet that it uses to store a set of consultations. ConsultationLog will use that TreeSet and the printing methods that the consultation class has, to perform its duties.

5.6.2 ConsultationLog Methods

As is the case for many of the other classes in ChocAn's system there are two sets of methods; methods that print, and methods that add data the ConsultationLog's list of consultations. All of the printing methods take a filename (a string) to which the data will be printed to. They all return false if the filename is not a null string and true otherwise. The methods are named after what they print, the report they make, or `printAllDataFile`, which prints all the logs data to the passed file, overwriting said file.

Also, the ConsultationLog has a method to add consultations to its list. It takes a reference to a consultation and returns false if that reference is null. If it is not, then the passed consultation is added to the the list of consultations that ConsultationLog contains.

5.7 Controller Class

Class name	**Controller**
Number of instances	One
Belongs to class	Console
Name of instantiated object(s)	controllerObj

Member data	Database theDataBase;
	ConsultationLog consultationRecord;

Public Methods	Boolean addService (service) Boolean makeConsultation (service, provider, member) Boolean addMember (member) Boolean addProvider (provider) void printAllServiceScreen () Boolean makeMemberReport (String) Boolean makeProviderReport (String) Boolean makeMangerReport (String) Boolean makeEftReport (String)

5.7.1 Controller Data

The controller will have 2 data members which are an instance of Database and an instance of ConsultationLog. The controller uses these data members to allow communication between Database and ConsultationLog while maintaining encapsulation. The console talks to the Controller in order to have an indirect line of communication to the data members.

5.7.2 Controller Methods

The methods within Controller will have three sets of methods. One set is for communicating with Database with methods that include creating new members or adding providers. Another set are those that will interact with the ConsultationLog to create a new consultation. Lastly, there will be a set of functions to handle all the reporting.

5.8 Console Class

Class name	**Console**

Data	controller theBoss;
	int managerPassword;

Public Methods	Boolean setMangerPass (String) void initialMenu () int memberOrManger () Boolean valiProvNum () Boolean valiMangPass () void mangMenu () Boolean changeMemStatus () Boolean addMem () Boolean removeMem () Boolean addProv () Boolean removeProv () Boolean makeMangReport (String) Boolean makeEFTReport (String) Boolean makeMemReport (String) void provMenu () Boolean makeConsult () Boolean makeProvReport () Boolean valiMemNum () void displayAllServices () int main ()

5.8.1 Console Data

The console will contain two private members. The controller instance is what allows Console to talk to the rest of the program. Through the methods provided in the Controller, Console will be able to perform actions on the database and allows for other functionality. In addition to a controller instance console will have an integer manager password that is used to differentiate manager access permissions from provider access permissions.

5.8.2 Console Methods

The console class has three main responsibilities. Those responsibilities are the implementation of the systems most general control flow, implementation of the menus that facilitate a change in control, and implementation of the ability to validate user input. The console class contains main to enable it to dictate the systems most general control flow. In addition to main, the console class has 3 sets of functions. The first set of functions are used to determine if a manager or a provider is accessing the system. Those functions include the systems initial menu and functions that validate a provider number or a manager password.

The system also has a set of functions to enable the manager to interact with the system as determined in the requirements documentation. Those functions include the manager’s menu and the functions that enable alteration to the system's database. The system also has a set of functions that are used to allow the provider to interact with the system as determined in the requirements documentation. Those functions include the provider’s menu, functions that enable the validation of member numbers, and functions that enable scheduling of consultations.