**The ChocAn Simulator**

by Team 7, LLC

Portland State University

Science Research and Teaching Center

1719 SW 10th Ave., Suite 247

Portland, Oregon 97201

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Author | Changes |
| 2017-10-28 | 0.1 | Brent McManus | Data Structures Architecture/Design |
| 2017-10-28 | 0.2 | Edward Koroteev | Data Structures Architecture/Design |
| 2017-10-29 | 0.3 | Ray Emory | Introduction, Considerations, Overview. |
| 2017-11-01 | 0.4 | Lance Booth | CAC Class, Architecture/Design |
| 2017-11-02 | 1.0 | Ray Emory | Additions to System Architecture and Design |
| 2017-11-02 | 1.1 | Ray Emory | Team Review Modifications |
|  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc497413687)

[1.1 Purpose and Scope 5](#_Toc497413688)

[1.2 Target Audience 5](#_Toc497413689)

[1.3 Definitions 6](#_Toc497413690)

[1.3.1 Conformance Levels 6](#_Toc497413691)

[1.3.2 Acronyms, Abbreviations, and Initialisms 6](#_Toc497413692)

[1.3.3 Glossary of Terms 7](#_Toc497413693)

[2 Design Considerations 8](#_Toc497413694)

[2.1 Constraints and Dependencies 8](#_Toc497413695)

[2.2 Methodology 8](#_Toc497413696)

[3 System Overview 9](#_Toc497413697)

[4 System Architecture 11](#_Toc497413698)

[4.1 Person subsystem and Object 11](#_Toc497413699)

[4.1.1 Provider Object 11](#_Toc497413700)

[4.1.1.1 Provider Identification Number Validation 11](#_Toc497413701)

[4.1.1.2 Provider Login Validation 12](#_Toc497413702)

[4.1.2 Member Object 13](#_Toc497413703)

[4.1.2.1 Member Identification Number Validation 13](#_Toc497413704)

[4.1.2.2 Member Status Validation 14](#_Toc497413705)

[4.2 Administrator Object 15](#_Toc497413706)

[4.3 Service Object 15](#_Toc497413707)

[4.4 List Template Object 16](#_Toc497413708)

[4.5 Report Object 17](#_Toc497413709)

[4.6 Report List Object 17](#_Toc497413710)

[4.7 CAC Object 17](#_Toc497413711)

[5 Detailed System Design 22](#_Toc497413712)

[5.1 Person Subsystem and Object 22](#_Toc497413713)

[5.1.1 Provider Object 23](#_Toc497413714)

[5.1.2 Member Object 25](#_Toc497413715)

[5.2 Service Object 27](#_Toc497413716)

[5.3 Template List Object 28](#_Toc497413717)

[5.4 Report Object 29](#_Toc497413718)

[5.5 Report List Object 30](#_Toc497413719)

[5.6 CAC Object 31](#_Toc497413720)

Table of Figures

[Figure 3‑1: Database Management Software Overview 9](#_Toc497412871)

[Figure 4‑1: Provider PIN Verification 12](#_Toc497412872)

[Figure 4‑2: Provider Login 13](#_Toc497412873)

[Figure 4‑3: Member PIN Verification 14](#_Toc497412874)

[Figure 4‑4: Member Status Validation 14](#_Toc497412875)

[Figure 4‑5: Administrator Login 15](#_Toc497412876)

[Figure 4‑6: Service Fee Request 16](#_Toc497412877)

[Figure 4‑7: List Template Overview 17](#_Toc497412878)

[Figure 4‑8: CAC Object Overview 18](#_Toc497412879)

[Figure 4‑9: Member Creation 18](#_Toc497412880)

[Figure 4‑10: Provider Creation 19](#_Toc497412881)

[Figure 4‑11: Member Reports 20](#_Toc497412882)

[Figure 4‑12: Provider Reports 20](#_Toc497412883)

# Introduction

## Purpose and Scope

The purpose of this document is to layout the technical design of the Chocoholics Anonymous Management Program System software.

It is intended to provide guidance for the development of the Chocoholics Anonymous data processing software to both managerial and technical staff.

The system design outlined in this document builds upon the requirements document generated during the requirements phase.

## Target Audience

This document is intended for use by the Chocoholics Anonymous Corporation and the development group working at Team 7, who have been contracted by Chocoholics Anonymous to deliver the data management software.

The design outlined within this document is to be followed by the developers at Team 7 in the implementation of the data management software for Chocoholics Anonymous.

## Definitions

### Conformance Levels

|  |  |
| --- | --- |
|  |  |
| **Expected** | A keyword used to describe the behavior of the software in the design models assumed by this requirements document. |
| **May** | A keyword that indicates flexibility of choice with no implied preference. |
| **Shall** | A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements. |
| **Should** | A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.” |

### Acronyms, Abbreviations, and Initialisms

|  |  |
| --- | --- |
|  |  |
| **CAC** | Chocoholics Anonymous Control Center. Refers to the primary control interface to be used by actors upon the data management software |
| **ChocAn** | Chocoholics Anonymous |
| **EFT** | Electronic Funds Transfer |
| **M-PIN** | Member Personal Identification Number |
| **P-PIN** | Provider Personal Identification Number |
|  |  |
|  |  |

### Glossary of Terms

|  |  |
| --- | --- |
|  |  |
| **Administrator (Admin)** | An employee of Chocoholics Anonymous. Has specific permission for managing and controlling the system. |
| **Database** | In the context of this document and software, a database refers to a localized text file that stores information relating to the software. |
| **IT Staff** | Support staff at the Chocoholics Anonymous Data Center. |
| **Member** | A member of Chocoholics Anonymous |
| **Provider** | A healthcare provider for Chocoholics Anonymous. |
| **Service Code** | A 6 digit code corresponding to a service available from a Provider |
| **Service(s)** | Refers to the service provided by a Provider to a Member. |
| **Software** | Unless otherwise noted, refers to the Chocoholics Anonymous Data Processing software. To be created based off of the requirements outlined within this document. |
| **Terminal** | A terminal interface with which to control the software. |
| **User** | A user of the software. Can refer to either an Administrator or a Provider. |
| **Validated** | An M-PIN is recognized as valid by the software. Meaning that the member exists and is current on their payments to ChocAn |

# Design Considerations

Considerations of the project, as it relates to goals, constraints, and engineering practices.

## Constraints and Dependencies

The goal of the software system is to provide a scalable database management program for use by ChocAn Administrators and Providers. To that end, the software must be reliable with minimal downtime or loss of data. Should any changes be made to data within the database, the database must be updated; so as to avoid risk of any data loss, should an unexpected outage occur.

Access to the software must be gated by through security measures. Access shall not be granted without authentication of the user.

## Methodology

Team 7 will be using the waterfall method of software development. This particular methodology was selected due to the nature of the project, with full requirements gathering and design work done up front. Work on the database software shall be divided among the Team 7 developers in encapsulated sections.

Code shall be developed, by the Team 7 developers, in a test driven manner, with unit tests created alongside each function. This shall minimize the validation testing needed at the end of the project.

By combining waterfall with a test driven development, Team 7 will thus be able to create dependable database management software for Chocoholics Anonymous by the expected delivery date of 2017-11-30.

# System Overview

The ChocAn database management software shall have its primary functionality contained within the ChocAn Control Center software (CAC). The CAC shall manage the collections of ChocAn Administrators, ChocAn Providers, ChocAn Members, Services, and Reports.

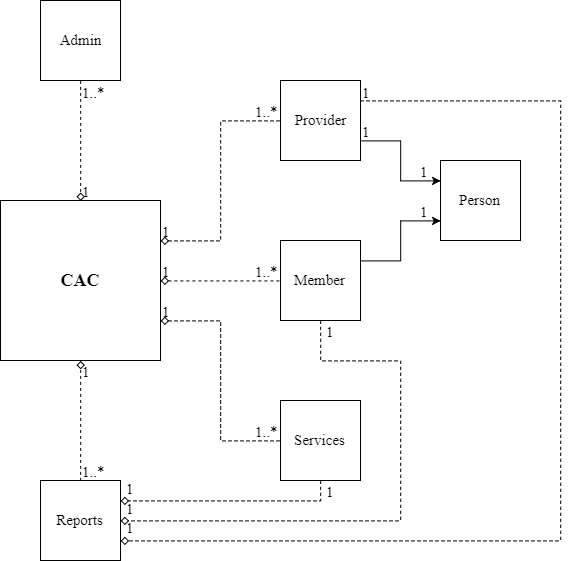


Figure ‑: Database Management Software Overview

The CAC will contain a collection of authorized ChocAn Administrators. Should an Administrator wish to perform any actions within the CAC, authentication via a username and password is required. Actions available to Administrators include the management of Members and Providers, management being the addition of a new Member or Provider or the editing of data or removal of an existing Member or Provider. Administrators shall also have the ability to generate the following reports, Member Reports, Provider Reports, EFT Reports, and Summary Reports.

The CAC will store the collection of ChocAn Members, organized alphabetically by name, and without any duplication of Member information. Members shall have no direct interaction with the CAC software.

The CAC will contain the collection of ChocAn providers, organized alphabetically by name, and without any duplication of Provider information. Should a Provider wish to provide a service to a Member, the CAC shall allow the Provider to login to the database system using their P-PIN and password. Once logged in, a Provider can validate that a Member is a current ChocAn subscriber before rendering service. When a Provider logs a service rendered to a Member through the CAC, a service report is generated within the CAC.

Reports shall be stored as a collection of an instance of a Provider, a Member and a Service, along with two dates, when the service was rendered and when the service was logged. Should a ChocAn Administrator wish to generate a report, be it a Provider Report, Member Report, EFT Report or Summary Report, the pertinent data shall be extracted from the collection of reports.

# System Architecture

Object oriented design will be the focus of our architecture. All data will be contained in class objects. Class objects will interact with each other to accomplish the tasks required of the system in an efficient manner.

## Person subsystem and Object

The base object for all members and providers is the person object class. All personal data for providers and members will be contained in this class and are part of the person subsystem. The member and provider classes inherit from the person class. The person class will contain; a name (either for an individual or an organization), street address, city, state and zip code. This class shall be an abstract base class and not allow for base class creation.

### Provider Object

The provider class shall be responsible for the creation of person and provider objects. The provider object is the data contained in its person parent class, plus a field for a password and a second field for a provider identification number. This class shall primarily verify information about the object and allow administrators of ChocAn to modify the provider objects data. The provider class shall also have the capacity to display all its data to the terminal screen, and to facilitate the storage of its information within the database. Information shall be written to the database using a single delimited string. The class shall also allow for the copying of contained data to another provider object for the purpose of storing service reports.

An object of this class shall be used in the generation of member reports, provider reports, EFT reports, and summary reports.

#### Provider Identification Number Validation

The Provider class shall have a method to validate whether an entered provider number matches the provider number contained within the object. The number shall be passed as an argument for a public class function. The function shall verify whether a passed provider number is of the appropriate length.

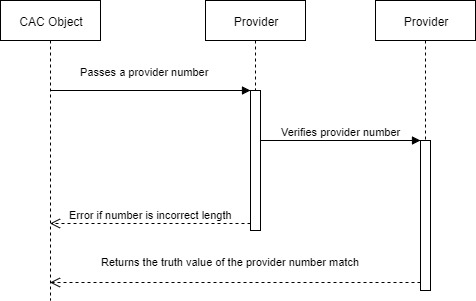


Figure ‑: Provider PIN Verification

This method shall be used when a ChocAn Provider is logging into a terminal, when a ChocAn Administrator is selecting a provider whose information is to be updated, and in the generation of member reports, provider reports, EFT reports, and summary reports.

#### Provider Login Validation

The provider class shall verify whether a password entered by a ChocAn provider into the terminal matches the password stored within the provider object. The password to be checked shall be passed as an argument to a public class function.

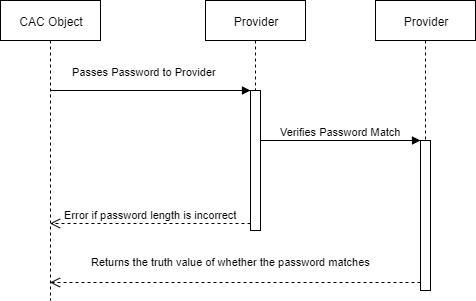


Figure ‑: Provider Login

### Member Object

The member class shall be responsible for the creation of person and member objects. The member object is the data contained in its person parent class, plus a field for member status and a second field for member number. This class shall primarily verify information about the object and allow administrators of ChocAn to modify the member objects data. The member class shall also have the capacity to display all of its data to the terminal screen and facilitate the storage of its information within the database. Information shall be written to the database using a single delimited string. This class shall also allow for the copying of contained data to another member object for the purpose of storing service reports.

An object of this class shall be used in the generation of member reports, provider reports, EFT reports, and summary reports.

#### Member Identification Number Validation

The member class shall have a method to validate whether an entered member number matches the member number contained within the object. The number shall be passed as an argument for a public class function. The function shall verify whether the passed member number is of the appropriate length.

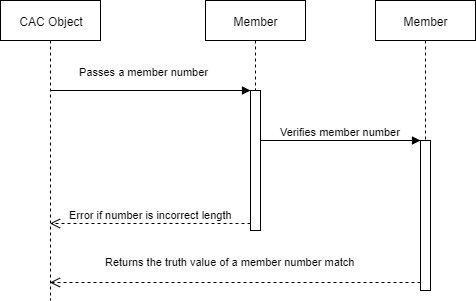


Figure ‑: Member PIN Verification

#### Member Status Validation

The member function shall allow the CAC to query the status of members. The member object will return a Boolean value of whether a member’s status is active.

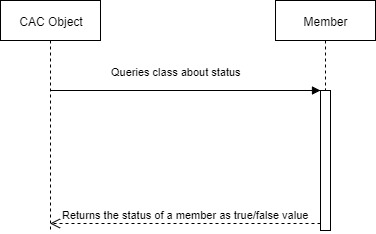


Figure ‑: Member Status Validation

## Administrator Object

The administrator class shall hold information pertaining to a ChocAn administrator; fields for a username and a password. The administrator class shall verify whether a combination of username and password matches the username and password stored within the administrator object. The username and password to be checked shall be passed as an argument to a public class function.

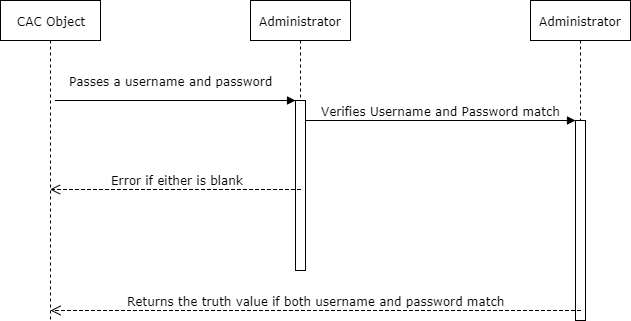


Figure ‑: Administrator Login

## Service Object

The service class shall hold all the information about an individual service that ChocAn Providers can render; the service name, the service cost, and the service code. The class shall be able to display the service information and return the fee of a service when queried by the CAC.

An object of this class shall be used in the generation of member reports, provider reports, EFT reports, and summary reports.

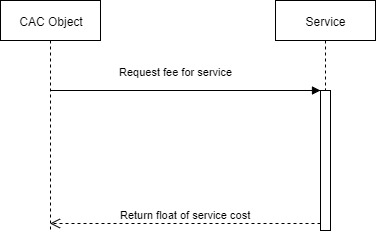


Figure ‑: Service Fee Request

## List Template Object

This List class object shall be a template class. This class should be the backbone of all storage required for the CAC software. The list class shall be instantiated with one of the following: provider object, member object, service object or administrator object. The list class will take the instantiated argument and derive from the argument. Upon its creation, the list class will only accept arguments that match the instantiated argument. The class will not allow for duplicate objects in the same list. The list template shall add object in sorted order based on a shared field called name. The list shall allow for retrieval of items from the list to be utilized by other objects. The list template shall have the ability to display all object in the list on request. The list shall be responsible for writing and reading information within the list to the database upon request of the CAC.

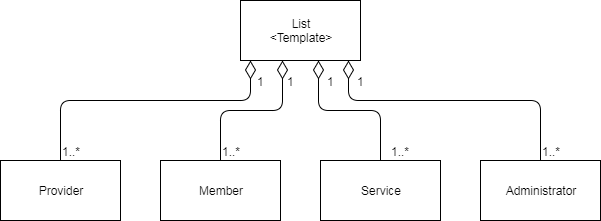


Figure ‑: List Template Overview

## Report Object

The Report class shall hold the information pertaining to a given service rendered by a ChocAn Provider to a ChocAn Member: the Provider’s information, the Member’s information, the Service information, a comment from the Provider, the date the service was rendered and the date the service was logged.

The Report class shall also facilitate the storage of its information within the database. Information shall be written to the database using a single delimited string. The information stored shall be the complete member information, complete provider information, complete service information, and dates for service rendered and service logged.

## Report List Object

The Report List object shall contain only Report objects. The reports list shall add report objects sorted in order of the date of service logged.

The report list class shall be capable of writing data contained within report object both to a file and to the terminal display. The report class shall separate contain methods for generating a Provider Report and a Member Report from the reports contained within. The functions to generate these reports shall take Provider PIN and a Member PIN respectively. The report list class shall have a method for generating an EFT report from the reports contained within. The Report list class shall have a method for generating a Summary report from the reports contained within.

## CAC Object

The CAC class shall be the interface for the rest of the software. It shall coordinate logins of Administrators or Providers, generate a service log when a Provider renders service to a Member, facilitate the management of both ChocAn Members and Providers by an Administrator, and allow for the generation of reports, either on a schedule or on demand. The CAC class shall contain a List Template object for each of the following; member class, provider class, administrator class, and service class. This class shall also contain a Report List object, for the storage of logged services and generation of reports.

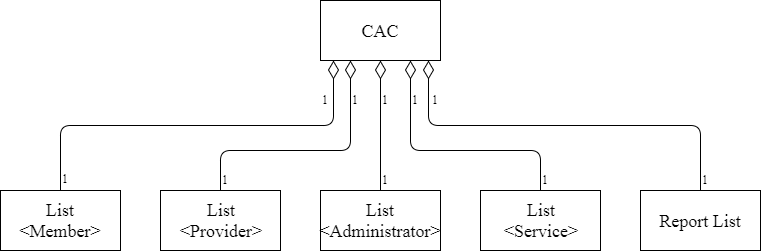


Figure ‑: CAC Object Overview

The CAC class shall enable an Administrator to create a new member. The Administrator shall enter the information for the new member; the member object shall be added to the list of members.

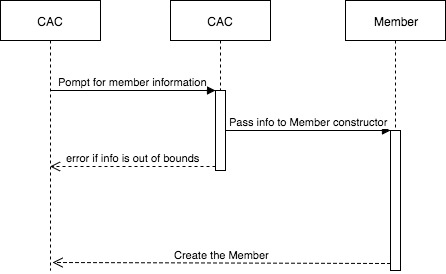


Figure ‑: Member Creation

The CAC class shall enable an Administrator to create a new provider. The Administrator shall enter the information for the new provider; the provider object shall be added to the list of providers

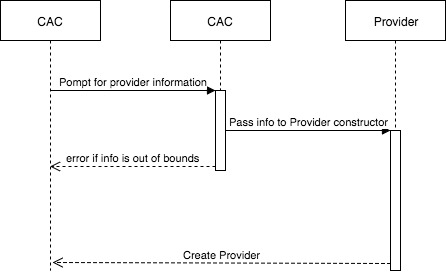


Figure ‑: Provider Creation

.

The CAC class shall control the interface in which a Provider logs a service rendered to a Member. The CAC shall provide an interface for a Provider to validate an M-PIN and to log a service, while providing commentary. When a service is logged, the CAC shall generate a report object and add it to the Report List.

The CAC class shall enable an Administrator to generate a Member report. Taking an M-PIN input from the user and passing it to the report list.

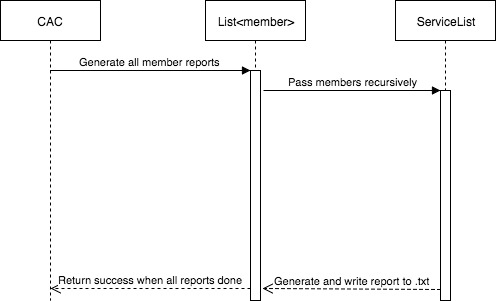


Figure ‑: Member Reports

The CAC class shall enable an Administrator to generate a Provider report. Taking a P-PIN input from the user and passing it to the report list.

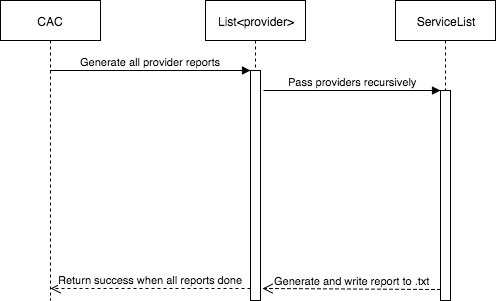


Figure ‑: Provider Reports

The CAC class shall be responsible for reading from and writing to the database. On initialization of the CAC class, Member Data, Provider Data, Service Data, Administrator Data and Report Data shall be read from their respective databases and populated in their respective data structures. On modification of any existing data, or creation of new data, the CAC shall be responsible for updating the appropriate database.

# Detailed System Design

## Person Subsystem and Object

The person class shall consist of five data elements. All data elements will be of type string.

* string name; //Maximum acceptable length of 20 characters
* string streetAddress; //Maximum acceptable length of 25 characters
* string city; //Maximum acceptable length of 14 characters
* string state; //Length must be equal to two characters
* string zip; //Length must be equal to five characters

The person class will have five functions that will be used by all derived classes. This structure will limit our lines of code while increasing the functionality of the class.

* create(string & name, string & street, string & city, string & state, string & zip)
  + This shall fill in the fields of the calling object with their respective passed arguments. This function shall provide the added functionality of the constructor without the need for redundant code. This function shall be called by the copy constructor and the parameterized constructor functions.
* person(string & name, string & street, string & city, string & state, string & zip)
  + This is the parameterized constructor, it shall call the create function to populate the member data with the appropriate arguments.
  + Length of the arguments should be validated by the CAC class before calling this function, as the function assumes that the arguments are of an appropriate length for the field.
* copy(person & to\_copy)
  + A function that shall copy the data from one person object into the calling object. This shall call the create function by parsing out the data members into the five individual parameters as arguments.
* person(person & to\_copy)
  + This copy constructor shall allow for the copying of all data from one person object to another with a single argument. It shall call the copy function.
* display()
  + A purely virtual function. Derived classes (member & provider) must have a display() function.

### Provider Object

The provider class is a class derived from the person class with two additional data members.

* string password; //Maximum acceptable length of 20 characters
* string id\_number; //Length must be equal to nine characters. Must start with a 9.

The provider class shall have the following member functions.

* display()
  + This function shall output relevant information about the provider to the terminal display. There shall be no return type. The format of the information output shall be:

Provider #: 000000000

Provider Name: John Smith

Provider Address: 123 Sesame Lane

Portland, OR 97217

* prepare\_write()
  + This function shall be called by the list class. It shall facilitate the writing of individual provider data to the database of providers. There are no arguments and this function will return a string containing all of the provider’s data delimited by the ‘|’ symbol.

Ex: 000000000|John Smith|123 Sesame Lane|Portland|OR|97217|password

* is\_password(string & to\_check)
  + This function shall confirm that a person object’s password matches the argument passed in. The function will first check that the string is within the password length constraints of the system. The class will then compare the argument to the object’s password data member and return a Boolean value.
* equals(string & number)
  + This function shall be used by the list class in determining whether a provider object matches a specified provider number. The string argument shall first be checked to ensure its length is equal to nine. The string shall then be compared to the id\_number field of the calling object. A Boolean value shall be returned with regards to a match.
* modify()
  + This function shall be used by the CAC. This class object shall output a menu of options for changing any of the seven object parameters, and an option to exit the menu.
    1. Edit identification number.
    2. Edit name
    3. Edit street address
    4. Edit city
    5. Edit state
    6. Edit zip code
    7. Edit password
    8. Exit
  + If a parameter is selected from the menu, a prompt will display for the user to enter a new value for that parameter. The function will catch the user input and verify that its length is valid. If the length is invalid, an error message shall be displayed asserting such, the user should be prompted again for input.
  + Once a valid string has been entered for a field, the calling object will replace the appropriate field with the input.
  + The menu shall allow for the modification of none to all of the fields with a single function call and will terminate once the ‘exit’ option is selected from the menu.
  + Upon exit of the function, the updated provider information shall be written to the provider database.
* create(string & name, string & street, string & city, string & state, string & zip, string & password, string & number)
  + This shall invoke the parent person class create function and will fill the remaining fields of the calling object with their respective passed arguments.
  + This function will provide the added functionality of the constructor without the need for redundant code. This shall be called by the copy and parameterized constructor functions, and will return a Boolean value communicating its success or failure.
* provider(string & name, string & street, string & city, string & state, string & zip, string & password, string & number)
  + This parameterized constructor shall invoke the parent person constructor and then fill the remaining data fields of the calling object appropriately.
  + Length of the arguments should be validated by the CAC class before calling this function, as the function assumes that the arguments are of an appropriate length for the field.
* copy(provider & to\_copy)
  + This function shall copy the data from a provider object to the calling object. This will call the create function by parsing out the member data within the provider object into seven individual parameters and passing them as arguments.
* provider(provider & to\_copy)
  + This copy constructor shall allow for the copying of all data from one provider object to another with a single argument. It shall call the copy function.

### Member Object

The member class is a class derived from the person class with two additional data members.

* bool status; //Indicates if the member is in good standing with ChocAn
* string id\_number; //Length must be equal to nine characters. Must start with a 1-8.

The provider class shall have the following member functions.

* display()
  + This function shall output relevant information about the provider to the terminal display. There shall be no return type. The format of the information output shall be:

Member #: 000000000

Member Name: John Smith

Member Address: 123 Sesame Lane

Portland, OR 97217

Member Status: Active/Inactive

* prepare\_write()
  + This function shall be called by the list class. It shall facilitate the writing of individual member data to the database of members. There are no arguments and this function will return a string containing all of the member’s data delimited by the ‘|’ symbol.

Ex: 000000000|John Smith|123 Sesame Lane|Portland|OR|97217|1

* equals(string & number)
  + This function shall be used by the list class in determining whether a member object matches a specified member number. The string argument shall first be checked to ensure its length is equal to nine. The string shall then be compared to the id\_number field of the calling object. A Boolean value shall be returned with regards to a match.
* modify()
  + This function shall be used by the CAC. This class object shall output a menu of options for changing any of the seven object parameters, and an option to exit the menu.
    1. Edit identification number.
    2. Edit name
    3. Edit street address
    4. Edit city
    5. Edit state
    6. Edit zip code
    7. Update Status
    8. Exit
  + If a parameter is selected from the menu, a prompt will display for the user to enter a new value for that parameter. The function will catch the user input and verify that its length is valid. If the length is invalid, an error message shall be displayed asserting such, the user should be prompted again for input.
  + Once a valid string has been entered for a field, the calling object will replace the appropriate field with the input.
  + The menu shall allow for the modification of none to all of the fields with a single function call and will terminate once the ‘exit’ option is selected from the menu.
  + Upon exit of the function, the updated member information shall be written to the member database.
* is\_valid()
  + This is a getter function that shall return the Boolean value of the member’s status.
* create(string & name, string & street, string & city, string & state, string & zip, bool status, string & number)
  + This shall invoke the parent person class create function and will fill the remaining fields of the calling object with their respective passed arguments.
  + This function will provide the added functionality of the constructor without the need for redundant code. This shall be called by the copy and parameterized constructor functions, and will return a Boolean value communicating its success or failure.
* member(string & name, string & street, string & city, string & state, string & zip, bool status, string & number)
  + This parameterized constructor shall invoke the parent person constructor and then fill the remaining data fields of the calling object appropriately.
  + Length of the arguments should be validated by the CAC class before calling this function, as the function assumes that the arguments are of an appropriate length for the field.
* copy(member & to\_copy)
  + This function shall copy the data from a member object to the calling object. This will call the create function by parsing out the member data within the member object into seven individual parameters and passing them as arguments.
* member(member & to\_copy)
  + This copy constructor shall allow for the copying of all data from one member object to another with a single argument. It shall call the copy function.

## Service Object

The service class shall consist of three data elements. Two strings and one float.

* string name; //Description. Maximum acceptable length of 20 characters
* string service\_code; //Length must be equal to six characters
* float fee; //Must be between 0.00 and 999.99.

The service class shall have the following member functions.

* display()
  + Outputs all the information about an individual service. Takes no arguments and returns nothing. Displayed in the following format:

000000 Description $000.00

* prepare\_write()
  + This function shall be called by the list class. It shall facilitate the writing of individual service data to the database of services. There are no arguments and this function will return a string containing all of the service object data delimited by the ‘|’ symbol.

Ex: 000000|Description|00.00

* equals(string & number)
  + This function shall be used by the list class to determine whether a service matches a service code. The string argument should be checked to ensure length is equal to six. The argument shall then be compared to the service\_code. A Boolean value will be returned with regards to a match.
* get\_fee()
  + A getter function. Returns the float value of the fee data member.
* create(string & name, string & number, float fee)
  + This function shall provide the added functionality of the constructor without the need for redundant code. This shall be called by the copy and parameterized constructor functions, and will return a Boolean value communicating its success or failure.
* service(string & name, string & number, float fee)
  + This parameterized constructor shall call the create function and pass the arguments given appropriately.
  + Length of the arguments should be validated by the CAC class before calling this function, as the function assumes that the arguments are of an appropriate length for the field.
* copy(service & to\_copy)
  + This function shall copy the data from a service object to the calling object. This will call the create function by parsing out the member data within the service object into three individual parameters and passing them as arguments.
* service(service & to\_copy)
  + This copy constructor shall allow for the copying of all data from one service object to another with a single argument. It shall call the copy function.

## Template List Object

The list class is a template class that contains data of the same type as the instantiating argument. The class will have a pointer to the template object and a pointer to the next list item. This will be a sorted LLL.

* List(<T>)
  + An parameterized constructor that in instantiate the list to hold objects of type <T>. The constructor will point to an object of type <T> copying the information using the <T> copy constructor. The pointer for the next item in the list is set to NULL.
  + All objects <T> must contain a name field.
* Add(<T>)
  + This will add an item of type <T> to the list of objects; <T> can only be added to the list if the list was initialized with an object of type <T>.
  + The function shall recursively move through the list to ensure that an identical object does not already exist in the list.
  + If no identical object <T> is found, on the return call the function shall insert the object <T> in sorted order by the name field.
  + The function shall return a Boolean denoting success or failure on the insertion of <T> into the list.
* Remove(string & number)
  + This function shall recursively move through a list of objects removing any object whose number field matches the number argument.
  + The function shall return a Boolean regarding the removal of an object from the list.
* find(string & number)
  + This function shall iterate through the list comparing each object’s number field to the argument.
  + If the number’s match, then the matching object is returned to the calling object.
* display()
  + This function will recursively iterate through the list of <T> object’s, calling their common display() functions respectively.
  + This function has no arguments and has a void return type.
* write(string & filename)
  + This function shall write the contents of the list to the database. In this and all cases, the database is a text file.
  + This function expects as an argument the name of the text file to write. If the file does not exist, then the file will be created and opened.
  + The data written to the file shall be the string returned by the individual objects prepare\_write() function. The strings shall be separated by a new line.
  + The function will recur through the list writing the data for each object to the text file.
  + The function will return a Boolean value communicating the success or failure of the write.

## Report Object

The service class shall consist of five data elements. A member object, a provider object, a service object and three strings.

* member a\_member;
* provider a\_provider;
* service a\_service;
* string comment; //Comment provided by the Provider
* string date\_logged; //Date of service logged, must be format YYYY-MM-DD-HH:MM
* string date\_rendered; //Date of service rendered, must be format YYYY-MM-DD

This report class shall have the following member functions:

* prepare\_write()
  + This function shall be called by the list class. It shall facilitate the writing of a service report data to the database of reports. There are no arguments and this function will return a string containing all of the service object data delimited by the ‘|’ symbol.

Ex: 000000000|John Smith|123 Sesame Lane|Portland|OR|97217|password |000000000|John Smith|123 Sesame Lane|Portland|OR|97217|1|000000|Description|00.00|This is a comment|YYYY-MM-DD-HH:MM|YYYY-MM-DD

* has\_member(string & number)
  + This function shall call the is\_valid function on the contained member object.
* has\_provider(string & number)
  + This function shall call the is\_valid function on the contained provider object.
* prepare\_member\_report()
  + This function shall return a string formatted for a member report. Containing provider, member and date information.
* prepare\_provider\_report()
  + This function shall return a string formatted for a provider report. Containing provider, member and date information.
* prepare\_eft\_report(float & passback)
  + This function shall return a string formatted for an EFT report. Containing provider information.
  + The fee owed for the service shall be passed back to the calling function via the passback argument.

## Report List Object

The Report List class will contain a Report Object and a pointer to the next Report List item. This will be a sorted LLL.

* add(report & to\_add)
  + This will add a report object to the list.
  + The function shall recursively move through the list to insert the report object based on date of service rendered.
* member\_report(string & number)
  + This function shall iterate through the list of report objects, searching for reports with a member matching the string argument.
  + This function shall generate a member report, writing to a text file.
* provider\_report(string & number)
  + This function shall iterate through the list of report objects, searching for reports with a provider matching the string argument.
  + This function shall generate a provider report, writing to a text file.
* eft\_report(string & number, float & total)
  + This function shall iterate through the list of report objects, searching for reports with a provider matching the string argument.
  + This function shall generate an eft report for that Provider, writing to a text file.
* summary\_report(float & total)
  + This function shall iterate through the list of reports, generating an EFT report for each provider.
  + This function shall keep track of the total of all fees owed with the passed argument.
  + This function shall generate a summary report for all providers, writing to a text file.
* weekly\_report()
  + This function shall generate all Member and Provider reports and a Summary report.

## CAC Object

The CAC class shall consist of 5 elements, a Template List of Members, a Template List of Providers, a Template List of Services, a Template List of Administrators, and a Report List.

The CAC class shall have the following member functions:

* login\_menu()
  + This function shall enable an Administrator or a Provider to login to the CAC interface.
  + A user cannot login without proper credentials.
* provider\_menu()
  + This function shall provide the interface for a Provider using the software.
  + This menu shall give the Provider options for validating a Member, logging a Service, requesting the services directory and logging out.
* validate\_member(string & number)
  + This function shall be used when a Provider wishes to validate a member in provider menu.
  + This function will pass the number argument to the find function of the Template List containing member information. It shall return a Boolean value communicating the success or failure.
* log\_service()
  + This function shall be used when a Provider wishes to log a service rendered to a Member.
  + This function shall prompt the Provider for an M-PIN, passing that number to the validate\_member function, and capturing the returned member object if it exists.
  + Should the member be validated, this function will prompt the provider for the service code, passing the number entered to the find function of the Template List containing the service information, and capturing the returned service object if it exists.
  + This function shall prompt the Provider for the date of services rendered.
  + This function shall prompt the Provider for a comment on the service rendered.
  + Using the information gathered from the find function, this function shall create a Report object and pass it to the Reports List object.
* display\_services()
  + This function shall be used when a Provider wishes to view the service Directory.
  + This function shall call the display function for the Template List containing the service objects.
* administrator\_menu()
  + This function shall provide the interface for an Administrator using the software.
  + This menu shall give the Administrator the options for managing members, managing providers, generating reports and logging out.
* add\_member()
  + This function shall be used when an Administrator wishes to add a Member to the database.
  + This function shall prompt the user for a Member PIN for the new Member. Passing that provided number to the find function for the Template List containing the member objects. If no existing Member is found with that PIN, the Administrator is allowed to continue entering Member information.
  + This function shall prompt the user for relevant information to create a member object. The function shall then add the member object to the Template list containing members.
* modify\_member()
  + This function shall be used when an Administrator wishes to modify the data of an existing member.
  + This function shall prompt the user for a Member PIN for the new Member. Passing that provided number to the find function for the Template List containing the member objects. If a matching Member is found with that PIN, the administrator is then enabled to modify their existing data.
* remove\_member()
  + This function shall be used when an Administrator wishes to delete a member from the database.
  + This function shall prompt the user for a Member PIN for the new Member. Passing that provided number to the find function for the Template List containing the member objects. If a matching Member is found with that PIN, the administrator is prompted for confirmation of deletion.
* add\_provider()
  + This function shall be used when an Administrator wishes to add a Provider to the database.
  + This function shall prompt the user for a Provider PIN for the new Provider. Passing that provided number to the find function for the Template List containing the provider objects. If no existing Provider is found with that PIN, the Administrator is allowed to continue entering Provider information.
  + This function shall prompt the user for relevant information to create a provider object. The function shall then add the provider object to the Template list containing providers.
* modify\_provider()
  + This function shall be used when an Administrator wishes to modify the data of an existing provider.
  + This function shall prompt the user for a Provider PIN for the new Provider. Passing that provided number to the find function for the Template List containing the provider objects. If a matching Provider is found with that PIN, the administrator is then enabled to modify their existing data.
* remove\_provider()
  + This function shall be used when an Administrator wishes to delete a provider from the database.
  + This function shall prompt the user for a Provider PIN for the new Provider. Passing that provided number to the find function for the Template List containing the provider objects. If a matching Provider is found with that PIN, the administrator is prompted for confirmation of deletion.
* report\_menu()
  + This function shall provide the interface for an Administrator to generate reports.
  + This menu shall provide options for generating a Member Report, a Provider Report, an EFT Report, a Summary Report, and the weekly reports (which is all Member and Provider Reports and the Summary Report).
* generate\_member\_report()
  + This function shall be used to generate a member report.
  + This function shall prompt the provider for an M-PIN, passing that number to the member\_report function of the Report List class.
* generate\_provider\_report()
  + This function shall be used to generate a provider report.
  + This function shall prompt the provider for a P-PIN, passing that number to the provider\_report function of the Report List class.
* generate\_EFT\_report()
  + This function shall be used to generate an EFT report.
  + This function shall prompt the provider for a P-PIN, passing that number to the eft\_report function of the Report List class.
* generate\_summary\_report()
  + This function shall be used to generate a Summary report.
  + This function shall call the summary\_report function of the Report List class.
* generate\_weekly()
  + This function shall be used to generate the weekly reports.
  + This function shall call the weekly\_report function of the Report List class.