Questions to ask in lab:

- What classes do we need to write descriptions for?
- What information should we prioritize in the summary paragraph?

Updates needed:

- Project 1
    - [ ] Write includes/extends use case descriptions
    - [ ] Update use case descriptions to return to menu, not exit
- Project 2
    - [x] ~~Shorten summary paragraph~~
    - [x] ~~Fix MAD (Decision node labels, timer exit, login/decision menus)~~
    - [x] ~~Fix SCD (update actors & class types)~~

Project 3 Requirements:

- [ ] Class diagrams (all stereotypes + new ones)
- [ ] Sequence diagrams (all use cases)
- [x] ~~Argument paragraph~~
- [ ] HTML Reports

Use Cases:

- Validate Member (p) - Campbell DONE
- Request Provider Directory (p) - WES DONE
- Bill ChocAn (p) - Dylan
- Request reports (m) - Campbell DONE
- Update Member Records (o) - Sam
- Update Provider Records (o) - Sam
- Run Main Accounting Procedure (t)- Nicholas DONE
    - Includes request 4 reports
- Verify Operator - Nicholas DONE
- Verify Manager - SonnyDONE
- Verify Provider - SonnyDONE

Classes:

- Operator Controller
    - Void updateMemberRecords()
    - Void updateProviderRecords()
- Manager Controller
    - Void requestReports()
- Provider Controller

    - Void validateMember()
    - Void checkInMember()
    - String requestProviderDirectory()
- Account Verifier
    - Int verifyMemberAccount()
    - boolean verifyOperator()
    - Boolean verifyProvider()
    - boolean verifyManager()
- Account
    - String name
    - Int ID
    - Enum type
    - String streetAddress
    - Int phoneNumber
    - getters
- Operator Menu
    - Void activate(task)
    - Void navigateTo(newMenu)
    - displayValidityMessage() : void
- Manager Menu
    - DisplayValidityMessage() : void
    - Void activate(task)

Void navigateTo(newMenu)

- Provider Menu
    - Void activate(task)
    - Void navigateTo(newMenu)
    - displayValidityMessage() : void
- Member Report
    - Member member
    - generateReport()
- Provider Report
    - Provider provider
    - generateReport()
- Manager Report
    - ProviderList ArrayList<ProviderAccount>
    - Int totalConsultations
    - Double totalFee
    - generateReport()
- EFT Report
    - generateReport()

- Main Accounting Procedure
    - Void runMainAccountingProcedure()
    - Void requestReports()
    - DataCenter clearWeeklyData()
- Service
    - String name
    - Int codeNum
    - getters
- Service Log (composed by services)
    - Account provider
    - MemberAccount Member
    - Service service
    - String date
    - String time
    - Double fee
    - toString()
- DataCenter
    - Timer timer
    - providerDirectory : ArrayList <Services>
    - memberList : ArrayList <MemberAccount>
    - providerList : ArrayList <ProviderAccount>
    - userList : ArrayList <Account>
    - serviceLogList : <Service Log>
    - sendProviderDirectory() : void
    - promptMenu()

New Classes:

- Member Account extends Account
    - Int status
    - servicesAccessedThatWeek ArrayList<ServiceLog>
    - weeklyReset()
- Provider Account extends Account
    - servicesPerformedThatWeek ArrayList<ServiceLog>
    - weeklyReset()
    - Double weeklyFee
    - Int numConsultations
- ChocAnMain
    - assembly/runner

<u>Design Discussion:</u>

- Our design achieves healthy cohesion and coupling primarily through the use of modularity. The system consists of 20 different classes, many connected through inheritance or abstraction, each having attributes only accessible through getter methods, and very few attributes can be reset after their initial construction. We use procedural cohesion to clearly define the tasks each class may perform, as well as what information other classes may access. Unfortunately, Java does by nature require some higher coupling, which is primarily showcased in the DataCenter class. Because Java doesn't use pointers, the Data Center must be continuously copied and passed around as a class attribute in order to be efficiently updated, which may cause some security issues. However, our high cohesion and use of strictly private class attributes and very few update methods will hopefully protect the Data Center from any breaches of information, and allow updates to occur as efficiently as possible.

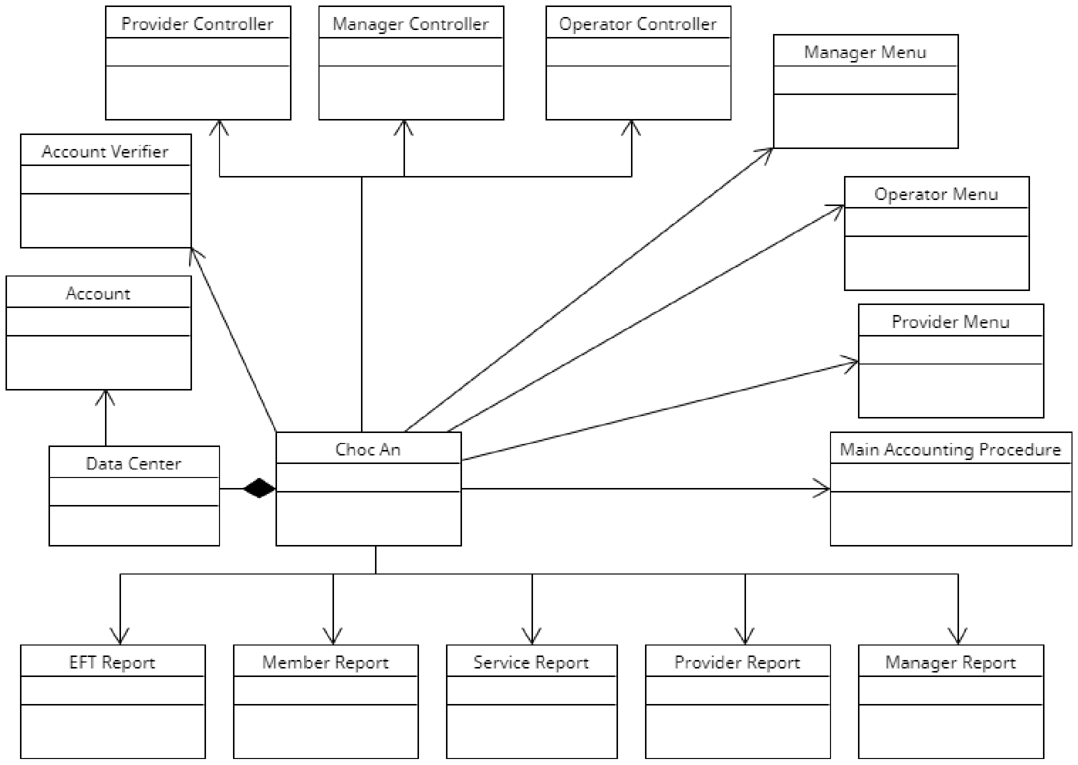
Names	**3/21**	**3/26**	**3/28**	**3/30**
Dylan Iovino	Y	Y	N	Y
Sonny Ngo	Y	Y	Y	Y
Campbell Thompson	Y	Y	Y	Y
Nicholas Seidl	Y	Y	Y	Y
Sam Wood				
Wesley Junkins	Y	Y	Y	Y

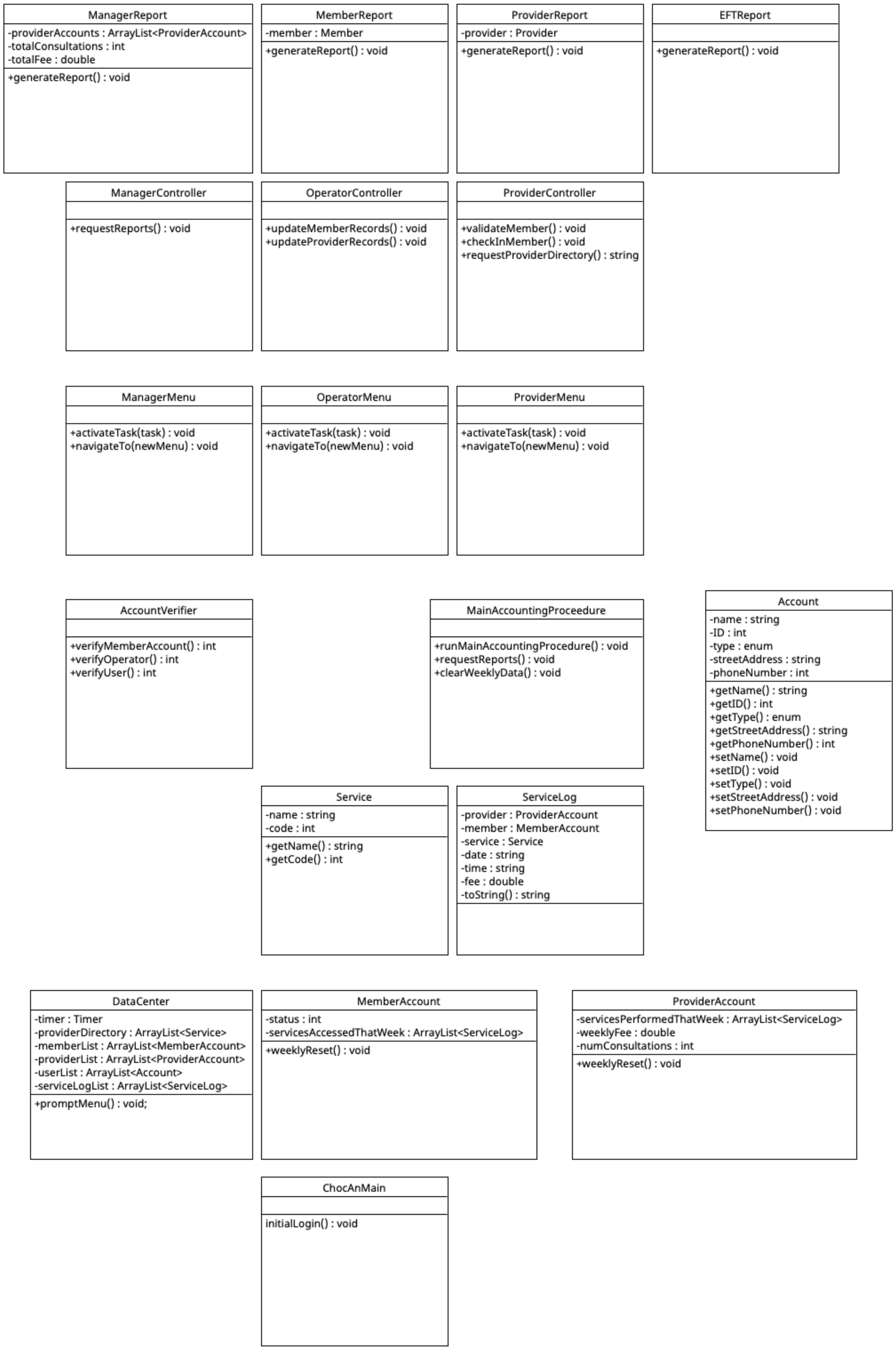
Name	CWID	Email	Hours
Dylan Iovino (Submitter)	12196331	djiovino@crimson.ua.edu	6.5
Sam Wood	12165183	scwood4@crimson.ua.edu	6.5
Campbell Thompson	12051119	crthompson10@crimson.ua.edu	7
Wesley Junkins	12048877	wcjunkins@crimson.ua.edu	6.5

Name	CWID	Email	Hours
Nicholas Seidl	12260915	naseidl@crimson.ua.edu	6.5
Sonny Ngo	12126308	smngo@crimson.ua.edu	6

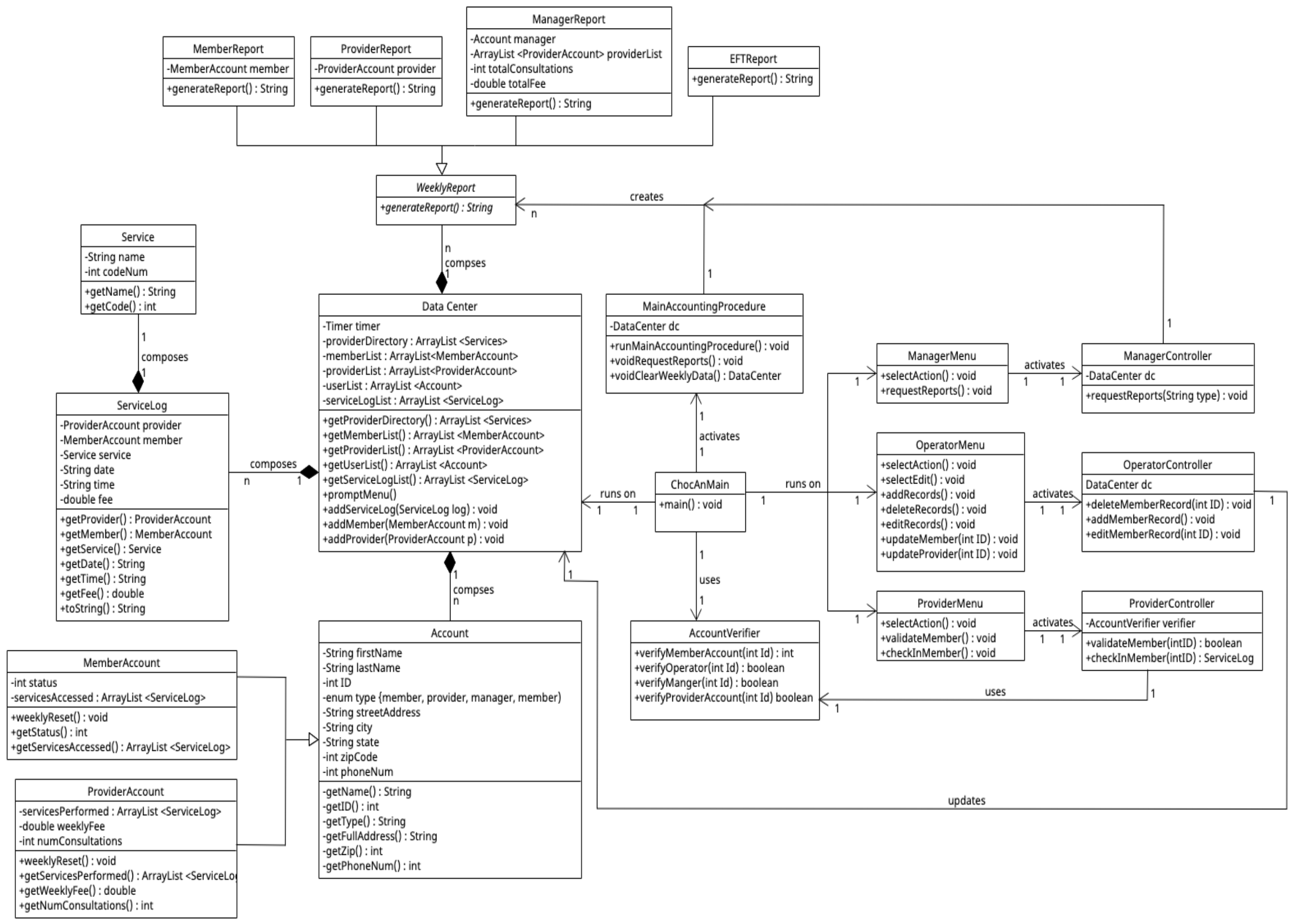
Name	Tasks Performed	% Contributed	Hours
Dylan Iovino	Individual class diagram Individual sequence diagram (Bill ChocAn) Individual html Fixed stereotype diagram	16.7%	5.5
Sonny Ngo	Individual class diagram and sequence diagrams (Verify Manager & Provider), Individual HTML report	16.6%	6
Campbell Thompson (Submitter)	Updated written portions of Project 1&2 Sequence Diagrams for Request Reports and Validate Member Finalized Class Diagram Wrote Design Discussion paragraph	16.6%	6.5
Nicholas Seidl	Created individual class diagram Created 2 sequence diagrams(Run Main Accounting Procedure, Verify Operator)	16.7%	6.5
Sam Wood	Individual main class diagram, two sequence diagrams (Update Provider/Member Records), HTML file, fixed activity diagram	16.7%	5.5
Wesley Junkins	Individual Main Class Diagram, Individual Sequence diagram,	16.8%	6.5

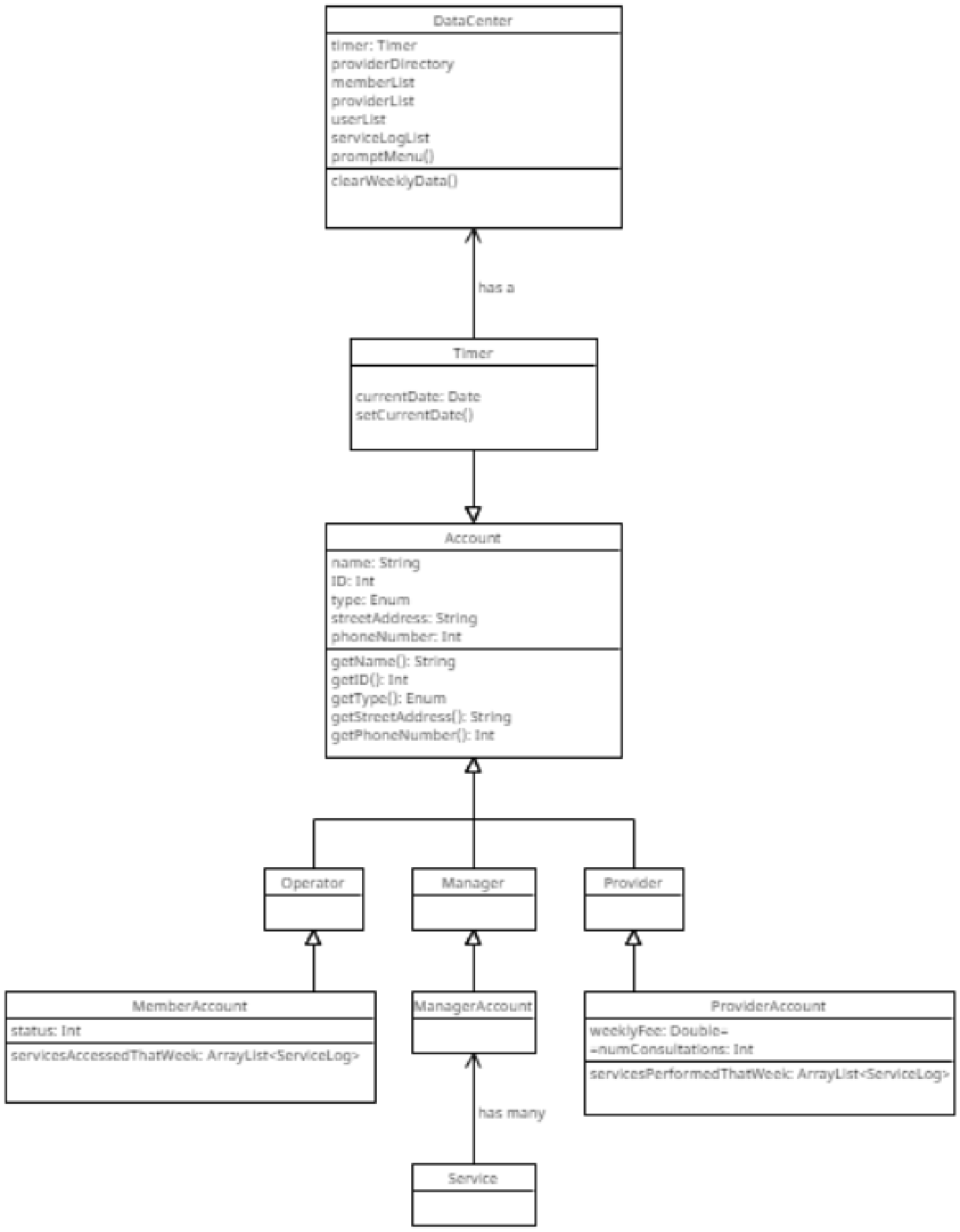
	personal HTML file, Designed and Developed Report.HTML, compiled individual HTML file.		


Provider Controller
Manager Controller
Operator Controller
Manager Menu
Account Verifier
Operator Menu
Account
Provider Menu
Data Center
Choc An
Main Accounting Procedure
EFT Report
Member Report
Service Report
Provider Report
Manager Report


ManagerReport
-providerAccounts : ArrayList<ProviderAccount>
-totalConsultations : int
-totalFee : double
+generateReport() : void
MemberReport
-member : Member
+generateReport() : void
ProviderReport
-provider : Provider
+generateReport() : void
EFTReport
+generateReport() : void
ManagerController
+requestReports() : void
OperatorController
+updateMemberRecords() : void
+updateProviderRecords() : void
ProviderController
+validateMember() : void
+checkInMember() : void
+requestProviderDirectory() : string
ManagerMenu
+activateTask(task) : void
+navigateTo(newMenu) : void
OperatorMenu
+activateTask(task) : void
+navigateTo(newMenu) : void
ProviderMenu
+activateTask(task) : void
+navigateTo(newMenu) : void
AccountVerifier
+verifyMemberAccount() : int
+verifyOperator() : int
+verifyUser() : int
MainAccountingProceedure
+runMainAccountingProcedure() : void
+requestReports() : void
+clearWeeklyData() : void
Account
-name : string
-ID : int
-type : enum
-streetAddress : string
-phoneNumber : int
+getName() : string
+getID() : int
+getType() : enum
+getStreetAddress() : string
+getPhoneNumber() : int
+setName() : void
+setID() : void
+setType() : void
+setStreetAddress() : void
+setPhoneNumber() : void
Service
-name : string
-code : int
+getName() : string
+getCode() : int
ServiceLog
-provider : ProviderAccount
-member : MemberAccount
-service : Service
-date : string
-time : string
-fee : double
-toString() : string
DataCenter
-timer : Timer
-providerDirectory : ArrayList<Service>
-memberList : ArrayList<MemberAccount>
-providerList : ArrayList<ProviderAccount>
-userList : ArrayList<Account>
-serviceLogList : ArrayList<ServiceLog>
+promptMenu() : void;
MemberAccount
-status : int
-servicesAccessedThatWeek : ArrayList<ServiceLog>
+weeklyReset() : void
ProviderAccount
-servicesPerformedThatWeek : ArrayList<ServiceLog>
-weeklyFee : double
-numConsultations : int
+weeklyReset() : void
ChocAnMain
initialLogin() : void

Campbell:


MemberReport
-MemberAccount member
+generateReport() : String
ProviderReport
-ProviderAccount provider
+generateReport() : String
ManagerReport
-Account manager
-ArrayList <ProviderAccount> providerList
-int totalConsultations
-double totalFee
+generateReport() : String
EFTReport
+generateReport() : String
WeeklyReport
+generateReport() : String
creates
n
1
1
n
compses
1
Service
-String name
-int codeNum
+getName() : String
+getCode() : int
1
composes
1
Data Center
-Timer timer
-providerDirectory : ArrayList <Services>
-memberList : ArrayList<MemberAccount>
-providerList : ArrayList<ProviderAccount>
-userList : ArrayList <Account>
-serviceLogList : ArrayList <ServiceLog>
+getProviderDirectory() : ArrayList <Services>
+getMemberList() : ArrayList <MemberAccount>
+getProviderList() : ArrayList <ProviderAccount>
+getUserList() : ArrayList <Account>
+getServiceLogList() : ArrayList <ServiceLog>
+promptMenu()
+addServiceLog(ServiceLog log) : void
+addMember(MemberAccount m) : void
+addProvider(ProviderAccount p) : void
MainAccountingProcedure
-DataCenter dc
+runMainAccountingProcedure() : void
+voidRequestReports() : void
+voidClearWeeklyData() : DataCenter
ManagerMenu
+selectAction() : void
+requestReports() : void
activates
1
1
ManagerController
-DataCenter dc
+requestReports(String type) : void
ServiceLog
-ProviderAccount provider
-MemberAccount member
-Service service
-String date
-String time
-double fee
+getProvider() : ProviderAccount
+getMember() : MemberAccount
+getService() : Service
+getDate() : String
+getTime() : String
+getFee() : double
+toString() : String
composes
n
1
1
activates
1
runs on
1
1
ChocAnMain
+main() : void
runs on
1
1
OperatorMenu
+selectAction() : void
+selectEdit() : void
+addRecords() : void
+deleteRecords() : void
+editRecords() : void
+updateMember(int ID) : void
+updateProvider(int ID) : void
activates
1
1
OperatorController
DataCenter dc
+deleteMemberRecord(int ID) : void
+addMember() : void
+editMemberRecord(int ID) : void
1
1
compses
n
1
1
uses
1
Account
-String firstName
-String lastName
-int ID
-enum type {member, provider, manager, member}
-String streetAddress
-String city
-String state
-int zipCode
-int phoneNum
-getName() : String
-getID() : int
-getType() : String
-getFullAddress() : String
-getZip() : int
-getPhoneNum() : int
AccountVerifier
+verifyMemberAccount(int Id) : int
+verifyOperator(int Id) : boolean
+verifyManger(int Id) : boolean
+verifyProviderAccount(int Id) boolean
1
ProviderMenu
+selectAction() : void
+validateMember() : void
+checkInMember() : void
activates
1
1
ProviderController
-AccountVerifier verifier
+validateMember(intID) : boolean
+checkInMember(intID) : ServiceLog
uses
1
1
MemberAccount
-int status
-servicesAccessed : ArrayList <ServiceLog>
+weeklyReset() : void
+getStatus() : int
+getServicesAccessed() : ArrayList <ServiceLog>
ProviderAccount
-servicesPerformed : ArrayList <ServiceLog>
-double weeklyFee
-int numConsultations
+weeklyReset() : void
+getServicesPerformed() : ArrayList <ServiceLo
+getWeeklyFee() : double
+getNumConsultations() : int
updates


DataCenter
timer: Timer
providerDirectory
memberList
providerList
userList
serviceLogList
promptMenu()
clearWeeklyData()
has a
Timer
currentDate: Date
setCurrentDate()
Account
name: String
ID: Int
type: Enum
streetAddress: String
phoneNumber: Int
getName(): String
getID(): Int
getType(): Enum
getStreetAddress(): String
getPhoneNumber(): Int
Operator
Manager
Provider
MemberAccount
status: Int
servicesAccessedThatWeek: ArrayList<ServiceLog>
ManagerAccount
ProviderAccount
weeklyFee: Double=
=numConsultations: Int
servicesPerformedThatWeek: ArrayList<ServiceLog>
has many
Service