**CLASS: ManageReport**

**Summary:**

**Elements:**

1. private MemberList memberList
2. private ProviderList providerList
3. private ServiceList serviceList
4. private ScheduleList scheduleList

**Functions:**

1. public List<MemberReport> GetWeeklyMemberReports( )
2. public List<ProviderReport> GetWeeklyProviderReports( )
3. public List<EFTReport> GetWeeklyEFTReports( )
4. public MemberReport GetMemberReport(integer memberID, date startDate, date endDate)
5. public EFTReport GetEFTReport(integer providerID, date startDate, date endDate)
6. public boolean UpdateMemberReportSchedule(integer weekDay, time time)
7. public boolean UpdateProviderReportSchedule(integer weekDay, time time)
8. public boolean UpdateEFTReportSchedule(integer weekDay, time time)
9. public boolean VerifyWeeklyProviderReportServices(  
   integer providerID, date startDate, date endDate)
10. public boolean VerifyWeeklyProviderReportFee(integer providerID, date startDate, date endDate)

**Detail:**

1. **public List<MemberReport> GetWeeklyMemberReports( )**

Module name : **GetWeeklyMemberReports**

Module type : Function

Return type : **List<MemberReport>**

Input arguments : None

Output arguments : None

Error messages : error compiling reports

Files accessed : None

Files changed : None

Modules called : MemberList.GetAllMembers( )

ProviderList.GetProvider(integer providerID)

MemberReport.SetMemberInformation(Member member)

MemberReport.AddService(Service service, Provider provider)

Member.GetServices( )

Service.GetProviderID( )

Narrative : This method returns all member reports for the week.

public List<MemberReport> GetWeeklyMemberReports()

{

List<MemberReport> memberReports = new List<MemberReport>()

try

{

var schedule = ScheduleList.GetSchedule(

ReportType.MemberReportType);

var startDate = //calculate start date from schedule;

var endDate = //calculate end date from schedule;

var members = memberList.GetAllMembers();

foreach (var member in members)

{

var memberReport = new MemberReport();

memberReport.SetMemberInformation(member);

foreach (var service in member.GetServices()

.Where(s => s.GetDate() <= endDate && s.GetDate() > startDate)

{

memberReport.AddService(service,

ProviderList.GetProvider(service.GetProviderID()));

}

memberReports.Add(memberReport);

}

}

catch (Exception e)

{

//record exception

memberReports = null;

}

return memberReports;

}

1. **public List<ProviderReport> GetWeeklyProviderReports( )**

Module name : **GetWeeklyProviderReports**

Module type : Function

Return type : **List<ProviderReport>**

Input arguments : None

Output arguments : None

Error messages : error compiling reports

Files accessed : None

Files changed : None

Modules called : ProviderList.GetAllProviders( )

MemberList.GetMember(integer memberID)

ServiceList.GetServicesByProviderID(integer providerID)

ProviderReport.SetProviderInformation(Member member)

ProviderReport.AddService(Service service, Member member)

Provider.GetProviderID( )

Service.GetProviderID( )

Narrative : This method returns all provider reports for the week.

public List<ProviderReport> GetWeeklyProviderReports()

{

List<ProviderReport> providerReports = new List<ProviderReport>();

try

{

var schedule = ScheduleList.GetSchedule(

ReportType.ProviderReportType);

var startDate = //calculate start date from schedule;

var endDate = //calculate end date from schedule;

var providers = providerList.GetAllProviders();

foreach (var provider in providers)

{

var providerReport = new ProviderReport();

providerReport.SetProviderInformation(provider);

foreach (var service in ServiceList.GetServicesByProviderID

(provider.GetProviderID())

.Where(s => s.GetDate() <= endDate && s.GetDate() > startDate)

{

providerReport.AddService(service,

ProviderList.GetProvider(service.GetMemberID()));

}

providerReports.Add(providerReport);

}

}

catch (Exception e)

{

//record exception

providerReports = null;

}

return providerReports;

}

1. **public List<EFTReport> GetWeeklyEFTReports( )**

Module name : **GetWeeklyEFTReports**

Module type : Function

Return type : **List<EFTReport>**

Input arguments : None

Output arguments : None

Error messages : error compiling reports

Files accessed : None

Files changed : None

Modules called : ProviderList.GetAllProviders( )

ServiceList.GetServicesByProviderID(integer providerID)

EFTReport.SetProviderID(integer providerID)

EFTReport.SetProviderName(string providerName)

EFTReport.SetTotalFee(decimal fee)

Provider.GetProviderID( )

Service.GetFee()

Narrative : This method returns all EFT reports for the week.

public List<EFTReport> GetWeeklyEFTReports()

{

List<EFTReport> eftReports = new List<EFTReport>();

try

{

var schedule = ScheduleList.GetSchedule(

ReportType.EFTReportType);

var startDate = //calculate start date from schedule;

var endDate = //calculate end date from schedule;

var providers = providerList.GetAllProviders();

foreach (var provider in providers)

{

var eftReport = new EFTReport();

eftReport.SetProviderID(provider.GetProviderID());

eftReport.SetProviderName(provider.GetProviderName());

var totalFee = 0.0;

foreach (var service in ServiceList.GetServicesByProviderID

(provider.GetProviderID())

.Where(s => s.GetDate() <= endDate && s.GetDate() > startDate)

{

totalFee += service.GetFee();

}

eftReports.Add(eftReport);

}

}

catch (Exception e)

{

//record exception

eftReports = null;

}

return providerReports;

}

1. **public MemberReport GetMemberReport  
   (integer memberID, date startDate, date endDate)**

Module name : **GetMemberReport**

Module type : Function

Return type : **MemberReport**

Input arguments : memberID : integer

startDate : date

endDate : date

Output arguments : None

Error messages : invalid member

Files accessed : None

Files changed : None

Modules called : MemberList.GetMember(integer memberID)

ProviderList.GetProvider(integer providerID)

MemberReport.SetMemberInformation(Member member)

MemberReport.AddService(Service service, Provider provider)

Member.GetServices( )

Service.GetProviderID( )

Narrative : This method returns member report for a specific member.

public MemberReport GetMemberReport

(integer memberID, date startDate, date endDate)

{

MemberReport memberReport = new MemberReport();

try

{

member = MemberList.GetMember(memberID);

if (null != member)

{

memberReport.SetMemberInformation(member);

foreach (var service in member.GetServices()

.Where(s => s.GetDate() <= endDate && s.GetDate() > startDate)

{

memberReport.AddService(service,

ProviderList.GetProvider(service.GetProviderID()));

}

}

Else throw new Exception(“member not found”);

}

catch (Exception e)

{

//record exception

memberReport = null;

}

return memberReport;

}

1. **public EFTReport GetEFTReport  
   (integer providerID, date startDate, date endDate)**

Module name : **GetEFTReport**

Module type : Function

Return type : **EFTReport**

Input arguments : providerID : integer

startDate : date

endDate : date

Output arguments : None

Error messages : invalid member

Files accessed : None

Files changed : None

Modules called : ProviderList.GetAllProviders( )

ServiceList.GetServicesByProviderID(integer providerID)

EFTReport.SetProviderID(integer providerID)

EFTReport.SetProviderName(string providerName)

EFTReport.SetTotalFee(decimal fee)

Provider.GetProviderID( )

Service.GetFee()

Narrative : This method returns EFT report for a specific provider.

public MemberReport GetMemberReport

(integer memberID, date startDate, date endDate)

{

EFTReport eftReport = new EFTReport();

try

{

provider = ProviderList.GetProvider(providerID);

if (null != provider)

{

eftReport.SetProviderID(provider.GetProviderID());

eftReport.SetProviderName(provider.GetProviderName());

var totalFee = 0.0;

foreach (var service in ServiceList.GetServicesByProviderID

(provider.GetProviderID())

.Where(s => s.GetDate() <= endDate && s.GetDate() > startDate)

{

totalFee += service.GetFee();

}

}

Else throw new Exception(“member not found”);

}

catch (Exception e)

{

//record exception

eftReport = null;

}

return eftReport;

}

1. **public boolean UpdateMemberReportSchedule  
   (integer weekDay, time time)**

Module name : **UpdateMemberReportSchedule**

Module type : Function

Return type : **boolean**

Input arguments : weekDay : integer

time : time

Output arguments : None

Error messages : None

Files accessed : None

Files changed : None

Modules called : ScheduleList.UpdateSchedule(Schedule updatedSchedule)

Schedule.SetReportType(integer reportType)

Schedule.SetWeekDay(integer weekDay)

Schedule.SetTime(time time)

Narrative : This method updates member report schedule.

public boolean UpdateMemberReportSchedule

(integer weekday, time time)

{

boolean success = false;

try

{

updatedSchedule = new Schedule();

updatedSchedule.SetReportType = ReportType.MemberReportType;

updatedSchedule.SetWeekDay = weekDay;

updatedSchedule.SetTime = time;

success = ScheduleList.UpdateSchedule(updatedSchedule);

}

catch (Exception e)

{

//record exception

success = false;

}

Return success;

}

1. **public boolean UpdateProviderReportSchedule  
   (integer weekDay, time time)**

Module name : **UpdateProviderReportSchedule**

Module type : Function

Return type : **boolean**

Input arguments : weekDay : integer

time : time

Output arguments : None

Error messages : None

Files accessed : None

Files changed : None

Modules called : ScheduleList.UpdateSchedule(Schedule updatedSchedule)

Schedule.SetReportType(integer reportType)

Schedule.SetWeekDay(integer weekDay)

Schedule.SetTime(time time)

Narrative : This method updates provider report schedule.

public boolean UpdateProviderReportSchedule

(integer weekday, time time)

{

boolean success = false;

try

{

updatedSchedule = new Schedule();

updatedSchedule.SetReportType = ReportType.ProviderReportType;

updatedSchedule.SetWeekDay = weekDay;

updatedSchedule.SetTime = time;

success = ScheduleList.UpdateSchedule(updatedSchedule);

}

catch (Exception e)

{

//record exception

success = false;

}

Return success;

}

1. **public boolean UpdateEFTReportSchedule  
   (integer weekDay, time time)**

Module name : **UpdateEFTReportSchedule**

Module type : Function

Return type : **boolean**

Input arguments : weekDay : integer

time : time

Output arguments : None

Error messages : None

Files accessed : None

Files changed : None

Modules called : ScheduleList.UpdateSchedule(Schedule updatedSchedule)

Schedule.SetReportType(integer reportType)

Schedule.SetWeekDay(integer weekDay)

Schedule.SetTime(time time)

Narrative : This method updates EFT report schedule.

public boolean UpdateEFTReportSchedule

(integer weekday, time time)

{

boolean success = false;

try

{

updatedSchedule = new Schedule();

updatedSchedule.SetReportType = ReportType.EFTReportType;

updatedSchedule.SetWeekDay = weekDay;

updatedSchedule.SetTime = time;

success = ScheduleList.UpdateSchedule(updatedSchedule);

}

catch (Exception e)

{

//record exception

success = false;

}

Return success;

}

1. **public boolean VerifyProviderReportServices  
   (integer providerID, date startDate, date endDate)**

Module name : **VerifyProviderReportServices**

Module type : Function

Return type : **boolean**

Input arguments : providerID : integer

startDate : date

endDate : date

Output arguments : None

Error messages : invalid provider

Files accessed : None

Files changed : None

Modules called : ProviderList.GetProvider(integer providerID)

ServiceList.GetServicesByProviderID(integer providerID)

ServiceList.UpdateService(Service updatedService)

Service.SetServiceVerified(boolean status)

Narrative : This method verifies all weekly services for a provider.

public boolean VerifyProviderReportServices

(integer providerID, date startDate, date endDate)

{

boolean success = false;

try

{

if (null != providerList.GetProvider(providerID))

{

var services = serviceList.GetServicesByProviderID(providerID)

.Where(s => s.GetDate() <= endDate

&& s.GetDate() > startDate);

foreach (var service in services)

{

service.SetServiceVerified(true);

ServiceList.UpdateService(service);

}

}

Else throw new Exception(“invalid provider”);

}

catch (Exception e)

{

//record exception

success = false;

}

Return success;

}

1. **public boolean VerifyProviderReportFees  
   (integer providerID, date startDate, date endDate)**

Module name : **VerifyProviderReportServices**

Module type : Function

Return type : **boolean**

Input arguments : providerID : integer

startDate : date

endDate : date

Output arguments : None

Error messages : invalid provider

Files accessed : None

Files changed : None

Modules called : ProviderList.GetProvider(integer providerID)

ServiceList.GetServicesByProviderID(integer providerID)

ServiceList.UpdateService(Service updatedService)

Service.SetFeeVerified(boolean status)

Narrative : This method verifies all weekly fees for a provider.

public boolean VerifyProviderReportFees

(integer providerID, date startDate, date endDate)

{

boolean success = false;

try

{

if (null != providerList.GetProvider(providerID))

{

var services = serviceList.GetServicesByProviderID(providerID)

.Where(s => s.GetDate() <= endDate

&& s.GetDate() > startDate);

foreach (var service in services)

{

service.SetFeeVerified(true);

ServiceList.UpdateService(service);

}

}

Else throw new Exception(“invalid provider”);

}

catch (Exception e)

{

//record exception

success = false;

}

Return success;

}