Лабораторная работа № 2

*Задание*: Разработать диаграммы прецедентов и диаграмму классов UML

по техническому заданию 1-й работы.

*Содержание отчета:* Текст задания, части технического задания,

титульный лист проекта, диаграммы с описанием, таблицы описания

прецедентов и классов.

*Замечания:* Для каждого класса указать атрибуты, выполняемые

операции, зависимости. На диаграммах классов должны быть реализованы

различные типы отношений. Диаграммы прецедентов должны описывать

весь основной функционал программного средства. Все классы и прецеденты

должны быть описаны в поясняющих таблицах.

*Задание на дополнительные баллы:*

* использование наследования: 1 балл;
* использование интерфейсов: 1 балл;
* использование абстрактных классов: 1 балл;
* использование статических методов и/или классов: 1 балл;
* использование коллекций: 1 балл;
* использование перегрузки операций (в одном классе): 1 балл;
* использование переопределения (между классами): 1 балл.

Диаграммы отдельными файлами в архиве.

**Use case diagram**

Все прецеденты добавления, изменения и удаления работают схожим образом для всех подсистем:

|  |  |
| --- | --- |
| Имя прецедента | Добавление |
| Краткое описание | Прецедент позволяет добавлять объекты |
| Акторы | Все акторы, кроме считывателя карт |
| Предусловия | Клиент должен кликнуть по кнопке добавления |
| Основной поток | Открывается форма добавления, которую нужно заполнить актору |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Изменение |
| Краткое описание | Прецедент позволяет изменять объекты |
| Акторы | Все акторы, кроме считывателя карт |
| Предусловия | Должен существовать хотя бы один объект |
| Основной поток | Открывается карточка необходимого объекта и его полями, которые можно изменить |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Удаление |
| Краткое описание | Прецедент позволяет удалять объекты |
| Акторы | Все акторы, кроме считывателя карт |
| Предусловия | Должен существовать хотя бы один объект. В карточки объекта есть кнопка удаления |
| Основной поток | При нажатии соответствующей кнопки удаляется объект |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Все прецеденты фильтрации, сортировки работают схожим образом для всех подсистем:

|  |  |
| --- | --- |
| Имя прецедента | Фильтрация списка |
| Краткое описание | Прецедент позволяет фильтровать списки |
| Акторы | Все акторы, кроме считывателя карт |
| Предусловия | Должен существовать список объектов |
| Основной поток | Открывается форма, которую пользователь должен заполнить. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Сортировка списка |
| Краткое описание | Прецедент позволяет сортировать списки |
| Акторы | Все акторы, кроме считывателя карт |
| Предусловия | Должен существовать список объектов |
| Основной поток | Открывается форма, которую пользователь должен заполнить. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Оплата услуг |
| Краткое описание | Прецедент позволяет оплачивать услуги |
| Акторы | Клиент |
| Предусловия | Клиент должен выбрать услуги или абонемент |
| Основной поток | Выводится форма, которую клиента заполняет |
| Постусловия | Если прецедент был успешно завершен, то в базе данные появиться новый объект «Payment». |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Вывод информации  об оплате |
| Краткое описание | Прецедент позволяет выводить информацию об оплате |
| Акторы | Клиент |
| Предусловия | Клиент должен выбрать интересующую оплату |
| Основной поток | Выводится карточка, содержащее информацию о платеже |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | История платежей |
| Краткое описание | Прецедент позволяет список платежей клиента |
| Акторы | Клиент |
| Предусловия | Должен существовать хотя бы один платеж |
| Основной поток | Выводится список платежей |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр расписания |
| Краткое описание | Прецедент позволяет расписания |
| Акторы | Клиент, тренер, менеджер |
| Предусловия | Должен существовать хотя бы одно расписание |
| Основной поток | Выводится расписание |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Выбор расписание |
| Краткое описание | Прецедент позволяет выбирать расписание тренировок и расписание рабочих смен |
| Акторы | Клиент, тренер, менеджер |
| Предусловия | Должен существовать хотя бы один объект расписание тренировок и расписание рабочих смен |
| Основной поток | Выводится нужное расписание |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Изменить расписание |
| Краткое описание | Прецедент позволяет изменять элементы расписания |
| Акторы | Тренер, менеджер |
| Предусловия | Должен существовать хотя бы один элемент расписания. А также пользователь должен выбрать нужный ему элемент расписания |
| Основной поток | Выводится нужный элемент, который можно редактировать |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка клиентов |
| Краткое описание | Прецедент позволяет выводить список клиентов |
| Акторы | Тренер, менеджер |
| Предусловия | Пользователь должен заполнить поля фильтрации, сортировки |
| Основной поток | Выводится список клиентов |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр данных о конкретном клиенте |
| Краткое описание | Прецедент позволяет выводить карточку необходимого клиента |
| Акторы | Тренер, менеджер |
| Предусловия | Должен существовать хотя бы один объект |
| Основной поток | Выводится карточка клиента |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр посещений |
| Краткое описание | Прецедент позволяет выводить список посещений клиента |
| Акторы | Менеджер |
| Предусловия | Пользователь должен заполнить поля фильтрации, сортировки |
| Основной поток | Выводится нужное количество посещений конкретного клиента |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Отметить вход клиента |
| Краткое описание | Прецедент позволяет считывателю карт отмечать вход клиента |
| Акторы | Считыватель карт |
| Предусловия | Клиент должен провести картой по считывателю при входе и выходе из клуба |
| Основной поток | Считыватель карт отправляет время входа, время выхода, номер карты, филиал |
| Постусловия | Создается объект посещения |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка уведомлений |
| Краткое описание | Прецедент позволяет выводить список уведомлений |
| Акторы | Менеджер |
| Предусловия | Пользователь должен заполнить поля фильтрации, сортировки |
| Основной поток | Выводится список уведомлений |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Отправка уведомлений |
| Краткое описание | Прецедент позволяет отправлять уведомления пользователям |
| Акторы | Менеджер |
| Предусловия | Пользователь должен заполнить поля (количество клиентов, текст) |
| Основной поток | Уведомления отправляются выбранным пользователям |
| Постусловия | Если прецедент был успешно завершен, то создается объект уведомления |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка акций |
| Краткое описание | Прецедент позволяет выводить список акций |
| Акторы | Клиент, менеджер |
| Предусловия | Должен существовать хотя бы одно уведомления |
| Основной поток | Выводятся акции за последний месяц |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка акций  конкретного клиента |
| Краткое описание | Прецедент позволяет выводить список акций конкретного клиента |
| Акторы | Клиент |
| Предусловия | Должен существовать хотя бы одно уведомления |
| Основной поток | Выводятся акции клиента за последний месяц |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка работников |
| Краткое описание | Прецедент позволяет выводить список работников |
| Акторы | Директор филиала |
| Предусловия | Должен существовать хотя бы один работник. Пользователь должен заполнить поля фильтрации и сортировки |
| Основной поток | Выводится список работников |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр данных  о конкретном работнике |
| Краткое описание | Прецедент позволяет выводить карточку работника |
| Акторы | Директор филиала |
| Предусловия | Должен существовать хотя бы один работник. Актор должен выбрать работника |
| Основной поток | Выводится карточка работника |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка залов |
| Краткое описание | Прецедент позволяет выводить список залов |
| Акторы | Тренер, директора филиала |
| Предусловия | Актор должен выбрать карточку филиала, в которой есть функция просмотра залов |
| Основной поток | Выводится список залов в рамках одного филиала |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка персонала, который имеет доступ в зал |
| Краткое описание | Прецедент позволяет выводить список персонала и список залов |
| Акторы | Тренер, директора филиала |
| Предусловия | Актор должен выбрать карточку филиала, в которой есть функция просмотра залов |
| Основной поток | Выводится список персонала в рамках выбранного зала |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр данных  о конкретном зале |
| Краткое описание | Прецедент позволяет выводить данные зала |
| Акторы | Директор филиала |
| Предусловия | Должен существовать хотя бы один зал. Актор должен выбрать зал из списка |
| Основной поток | Выводятся данные о зале |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка абонементов |
| Краткое описание | Прецедент позволяет выводить список абонементов |
| Акторы | Менеджер, директора филиала |
| Предусловия | Актор должен выбрать филиал, в рамках которого он желает увидеть список абонементов |
| Основной поток | Выводится список абонементов в филиале |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр конкретного абонемента |
| Краткое описание | Прецедент позволяет выводить карточку абонементов |
| Акторы | Менеджер, директора филиала |
| Предусловия | Актор должен выбрать абонемент из списка |
| Основной поток | Выводится карточка абонемента |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр содержимого  абонемента |
| Краткое описание | Прецедент позволяет список услуг абонемента |
| Акторы | Менеджер, директора филиала |
| Предусловия | Актор должен выбрать абонемент из списка |
| Основной поток | Выводится список услуг в рамках выбранного абонемента |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка контактов |
| Краткое описание | Прецедент позволяет выводить список контактов |
| Акторы | Тренер, директора филиала |
| Предусловия | Актор должен заполнить поля фильтрации, сортировки |
| Основной поток | Выводится список необходимых контактов |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Договор у конкретного клиента |
| Краткое описание | Прецедент позволяет выводить список |
| Акторы | Клиент, Менеджер |
| Предусловия | Актор должен выбрать карточку клиента и нажать на соответствующую кнопку. |
| Основной поток | Список договор в рамках одного клиента |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка услуг |
| Краткое описание | Прецедент позволяет выводить список услуг |
| Акторы | Клиент, менеджер, директора филиала |
| Предусловия | Актор должен нажать на соответствующую кнопку |
| Основной поток | Выводится список услуг в рамках филиала, в котором содержится |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр конкретной услуги |
| Краткое описание | Прецедент позволяет выводить информацию об услуги |
| Акторы | Клиент, менеджер, директора филиала |
| Предусловия | Актор должен выбрать услугу из списка |
| Основной поток | Выводится карточка услуги |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка отчетов |
| Краткое описание | Прецедент позволяет выводить список отчетов |
| Акторы | Директор компании, директора филиала |
| Предусловия | Актор должен нажать на соответствующую кнопку |
| Основной поток | Выводится список услуг отчетов |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр конкретного отчета |
| Краткое описание | Прецедент позволяет карточку отчета |
| Акторы | Директор компании, директора филиала |
| Предусловия | Актор должен выбрать отчет из списка |
| Основной поток | Выводится карточка отчета |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр конкретного отчета |
| Краткое описание | Прецедент позволяет выводить карточку отчета |
| Акторы | Директор компании, директора филиала |
| Предусловия | Актор должен выбрать отчет из списка |
| Основной поток | Выводится карточка отчета |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка филиалов |
| Краткое описание | Прецедент позволяет выводить список филиалов |
| Акторы | Директор компании |
| Предусловия | Актор должен нажать на соответствующую кнопку |
| Основной поток | Выводится список филиалов |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр данных конкретного филиала |
| Краткое описание | Прецедент позволяет выводить карточку филиала |
| Акторы | Директор компании |
| Предусловия | Актор должен выбрать филиал из списка |
| Основной поток | Выводится карточка филиала |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка работников в рамках одного филиала |
| Краткое описание | Прецедент позволяет список персонала в филиале |
| Акторы | Директор компании |
| Предусловия | Актор должен выбрать филиал из списка |
| Основной поток | Выводится список персонала |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

**Class diagram**

**Entity**

Абстрактный класс, являющий родителей для всех классов

*Атрибуты:*

# ID: int – поле, содержащее идентификатор объекта;

**User**

Абстрактный класс и родитель для других классов, таких как: Client, Employee.

*Атрибуты:*

+ Name: string – поле, которая содержит имя пользователя;

+ BirthDay: string – поле, которая содержит дату рождения пользователя;

+ PhoneNum: string – поле, которая содержит номер телефона пользователя;

# Login: string – поле, которая содержит логин пользователя для входа в систему;

#Password: string – поле , которая содержит пароль пользователя для входа в систему;

*Функции:*

+ VerifyLogin() : bool – метод, отвечающий за проверку введенных данных для входа в систему. Возвращает bool переменную, отвечающую за успех входа.

+ GetNotification(in filter: HashTable): List<Notification> - метод, позволяющий получить список уведомлений.

**Employee**

Реализует такие интерфейсы: IWorkout, IWorkShift, IService

*Атрибуты:*

+ PassportDate: string – поле, содержит паспортные данные работника;

+ Insurance\_ID: string – поле, содержит страховой номер работника;

+ Taxpayer\_ID: string – поле, содержит номер налогоплательщика работника;

+ Branch\_ID: string – поле, содержит номер филиала;

+ Post: PostType – поле, содержащее должность работника.

*Функции описаны в соответствующих интерфейсах.*

**Client**

Реализует такие интерфейсы: IWorkout, IContract

*Атрибуты:*

+ ClientCards: List<ClientCard> - поле, содержащее список клиентских карт;

*Функции:*

+ GetPayments(in filter: HashTable): List<Payment> – метод, выводящий список платежей клиента;

+ GetVisits(in filter: HashTable): List<Visit> - метод, выводящий список посещений клиента.

**ClientCard**

+ Client\_ID: int – поле, содержащее идентификатор клиента;

+ Name: string – поле, содержащее название карты;

**Visit**

*Атрибуты:*

+ Entrance: DateTime – поле, содержащее в себе время входа в один из филиалов клуба;

+ Exit: DateTime – поле, содержащее в себе время выхода из филиала клуба;

+ Branch\_ID: int – поле, содержащее идентификатор филиала, в котором был осуществлён вход;

+ ClientCard\_ID: int – поле, содержащее идентификатор клиентской карты, которую использовали при входе.

*Функции:*

+ GetClient(): Client – метод, который по id карты находит объект типа «Client»;

+ GetBranch(): Branch – метод, который по id находит объект типа «Branch».

**Payment**

*Атрибуты:*

+ Time: DateTime – поле, содержащее дату и время платежа;

+ Client\_ID: int – поле, содержащее идентификатор клиента

+ Amount: int – поле, содержащее количество потраченных денег за платеж;

+ Contract: Contract – поле, содержащее объект «Contract», который является целью платежа;

**Contract**

Класс реализует такие интерфейсы: IService, IClient

*Атрибуты:*

+ BeginningTime: DateTime – поле, содержащее информацию о начале действия контракта;

+ EndTime: DateTime – поле, содержащее информацию о конце действия контракта;

+ Client\_ID: int – поле, содержащее идентификатор клиента;

+ Client\_Name: string – поле, содержащее имя клиента, владельца клиента;

+ Pass: Pass – поле, содержащее абонемент(если клиент покупал абонемент);

+ Service: Service – поле, содержащее услугу(если клиент покупал отдельно услугу);

**Service**

Класс реализует интерфейс IClient

*Атрибуты:*

+ Name: string – поле, содержащее название услуги;

+ Description: string – поле, содержащее описание услуги;

+ AvTime: string – поле, содержащее среднее время услуги;

+ Price: decimal – поле, содержащее цену услуги;

**Pass**

Класс реализует интерфейс IClient

*Атрибуты:*

+ Service: List<Service> - поле, содержащее список услуг абонемента;

+ Type: PassType – поле, содержащее вид абонемента;

+ FreezingDays: int – поле, содержащее возможное время заморозки абонемента;

+ AmountDays: int – поле, содержащее количество дней в абонементе;

+ Price: decimal – поле, содержащее цену услуги;

**ElementTimeTable**

Абстрактный класс, родитель для классов: Workout, WorkShift.

*Атрибуты*:

+ BeginnigTime: DateTime – поле, содержащее время начала тренировки/рабочей смены;

+ EndTime: DateTime – поле, содержащее время окончания тренировки/рабочей смены;

**Workout**

Класс является наследников абстрактного класса «ElementTimeTable»

*Атрибуты:*

+ Hall: Hall – поле, содержащая зал, в котором будет проводиться тренировка;

+ TypeTraining: bool – поле, отвечающее за вид тренировка (true - персональные, false – групповые);

+ Trainer: Trainer – поле, содержащее за тренера, который будет проводить тренировку;

+ Commentary: string – поле, содержащее описание тренировки;

+ Clients: List<Client> - поле, содержащее список клиентов, участвующих в тренировке;

**WorkShift**

*Атрибуты:*

+ Employee: Employee – поле, содержащее работника;

**TimeTableWorkout**

*Атрибуты*:

+ Workouts: List<Workout> - поле, содержащее список тренировок;

**TimeTableWorkShift**

*Атрибуты*:

+ WorkShifts: List<WorkShift> - поле, содержащее список рабочих смен;

**Notification**

Класс реализует интерфейс IClient

*Атрибут:*

+ Inform: string – поле, отвечающее за информацию в уведомлении;

+ CreationDate: DateTime – поле, содержащее дату создания;

+ Manager\_ID: int – поле, содержащее идентификатор менеджера;

+ Client\_ID: List<int> - поле, содержащее список идентификаторов клиентов;

*Функции:*

+ SetClients(in clients\_id: List<int>): bool – метод, отвечающий за выбор клиентов, которым будут отправлены уведомления;

**Branch**

*Атрибуты:*

+ Name: string – поле, содержащее название филиала;

+ Address:string – поле, содержащее адрес филиала;

*Функции:*

+ GetHalls (in filter: HashTable): List<Hall> - метод, который выводит список залов в филиале;

+ GetEmployees (in filter: HashTable): List<Employee> - метод, который выводит список работников в филиале;

+ AddHall (in hall: Hall): bool – метод, позволяющий добавлять зал в филиал;

+ AddHall (): bool – метод, позволяющий добавлять зал в филиал;

+ DelHall (in id: int): bool – метод, позволяющий удалять по id зал.

**Hall**

Класс реализует интерфейс IClient

*Атрибуты:*

+ Number: string – поле, содержащее номер зала;

+ Branch\_ID: int – поле, содержащее идентификатор филиала

*Функции:*

+ GetTrainer(in filter: HashTable): List<Trainer> - метод, который выводит список тренеров, работающих в зале;

+ GetWorkouts(in filter:HashTable): List<Workout> - метод, который выводит список тренировок в зале;

+AddServices(in services: List<Service>): bool – метод, позволяющий добавлять услуги, которые могут проводится в зале;

+RemoveServices(in id: List<int>): bool – метод, позволяющий удалять услуги, которые могут проводится в зале.

**IService**

Интерфейс содержащий методы связанные добавление, удаление и выводом услуг.

*Функции:*

+ AddService(in objectID: int, new: Service[1..\*]): bool – метод, позволяющий добавлять объекту выбранные услуги;

+ RemoveService(in objectID: int, serviceID: int[1..\*]): bool – метод, позволяющий удалять выбранные услуги у объекта;

+ GetServices(in objectID: int, sorting: bool): Service[1..\*] – метод, позволяющий выводить все услуги объекта.

**IWorkShift**

Интерфейс содержащий методы связанные добавление, удаление и выводом рабочих смен.

*Функции:*

+ AddWorkShift(in objectID: int, new: WorkShift[1..\*]): bool – метод, позволяющий добавлять объекту выбранные рабочие смены;

+ RemoveWorkShift (in objectID: int, workshiftID: int[1..\*]): bool – метод, позволяющий удалять выбранные рабочие смены у объекта;

+ GetWorkShifts (in objectID: int, sorting: bool): WorkShift[1..\*] – метод, позволяющий выводить все рабочие смены объекта.

**IWorkout**

Интерфейс содержащий методы связанные добавление, удаление и выводом тренировок.

*Функции:*

+ AddWorkout (in objectID: int, new: Workout[1..\*]): bool – метод, позволяющий добавлять объекту выбранные тренировки;

+ RemoveWorkout (in objectID: int, workoutID: int[1..\*]): bool – метод, позволяющий удалять выбранные тренировки у объекта;

+ GetWorkouts (in objectID: int, sorting: bool): Workout[1..\*] – метод, позволяющий выводить все тренировки объекта.

**IContract**

Интерфейс содержащий методы связанные добавление, удаление и выводом контрактов.

*Функции:*

+ AddContract(in objectID: int, new: Contract[1..\*]): bool – метод, позволяющий добавлять объекту выбранные контракты;

+ RemoveContract(in objectID: int, contractID: int[1..\*]): bool – метод, позволяющий удалять выбранные контракты у объекта;

+ GetContracts (in objectID: int, sorting: bool): Contract[1..\*] – метод, позволяющий выводить все контракты объекта.

**IClient**

Интерфейс содержащий методы связанные добавление, удаление и выводом клиентов.

*Функции:*

+ AddClient(in objectID: int, new: Client[1..\*]): bool – метод, позволяющий добавлять объекту выбранных клиенты;

+ RemoveClient(in objectID: int, clientID: int[1..\*]): bool – метод, позволяющий удалять выбранных клиентов у объекта;

+ GetClients (in objectID: int, sorting: bool): Client[1..\*] – метод, позволяющий выводить всех клиентов объекта.

**IHall**

Интерфейс содержащий методы связанные добавление, удаление и выводом залов.

*Функции:*

+ AddHall(in objectID: int, new: Hall[1..\*]): bool – метод, позволяющий добавлять объекту выбранные залы;

+ RemoveHall(in objectID: int, hallid: int[1..\*]): bool – метод, позволяющий удалять выбранные залы у объекта;

+ GetHalls(in objectID: int, sorting: bool): Hall[1..\*] – метод, позволяющий выводить все залы объекта.

**IPass**

Интерфейс содержащий методы связанные добавление, удаление и выводом абонементов.

*Функции:*

+ AddPass(in objectID: int, new: Pass[1..\*]): bool – метод, позволяющий добавлять объекту выбранные абонементы;

+ RemovePass(in objectID: int, passID: int[1..\*]): bool – метод, позволяющий удалять выбранные абонементы у объекта;

+ GetPass(in objectID: int, sorting: bool): Pass[1..\*] – метод, позволяющий выводить все абонементы объекта.

**IEmployee**

Интерфейс содержащий методы связанные добавление, удаление и выводом абонементов.

*Функции:*

+ AddEmployee(in objectID: int, new: Employee[1..\*]): bool – метод, позволяющий добавлять объекту выбранных работников;

+ RemoveEmployee(in objectID: int, employeeID: int[1..\*]): bool – метод, позволяющий удалять выбранных работников у объекта;

+ GetEmployees(in objectID: int, sorting: bool): Employee[1..\*] – метод, позволяющий выводить всех работников объекта.

**Club**

Статичный класс, который работает с базой данных

*Функции:*

+ GetClients(in filter:HashTable): List<Client> - метод, который выводит список клиентов;

+ GetBranchs(in filter:HashTable): List<Branch> - метод, который выводит список филиалов;

+ GetHalls(in filter:HashTable): List<Hall> - метод, который выводит список залов;

+ GetOwners(in filter:HashTable): List<Owner> - метод, который выводит список владельцев;

+ GetBranchDirectors(in filter:HashTable): List<BranchDirector> - метод, который выводит список директоров филиала;

+ GetManagers(in filter:HashTable): List<Manager > - метод, который выводит список менеджер;

+ GetTrainers(in filter:HashTable): List<Trainer> - метод, который выводит список тренеров;

+ GetContracts(in filter:HashTable): List<Contract > - метод, который выводит список контрактов;

+ GetService(in filter:HashTable): List<Service > - метод, который выводит список услуг;

+ GetPass(in filter:HashTable): List<Pass> - метод, который выводит список абонеменов;

+ GetVisits(in filter:HashTable): List<Visit> - метод, который выводит список посещений клиента;

+ GetTimeTableWorkouts(in filter:HashTable): List<TimeTableWorkout> - метод, который выводит список расписаний тренировок;

+ GetTimeTableWorkShifts(in filter:HashTable): List<TimeTableWorkShift > - метод, который выводит список расписаний рабочей смены;

+ GetWorkout(in filter:HashTable): List<Workout> - метод, который выводит список тренировок;

+ GetWorkShift(in filter:HashTable): List<WorkShift> - метод, который выводит список рабочих смен;

+ GetNotifications(in filter:HashTable): List<Notification > - метод, который выводит уведомлений;

+ AddObj(in new: T[1..\*]): bool – метод, позволяющий добавлять объекты в базу данных(T – любой тип данных, существующий в рамках системы)