Лабораторная работа №4

Задание: Разработать диаграммы прецедентов и диаграмму классов UML по техническому заданию 1-й работы.

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.

Задание на дополнительные баллы:

* использование наследования: 1 балл;
* использование коллекций: 1 балл;
* использование абстрактных классов: 1 балл;
* использование статических методов и/или классов: 1 балл;

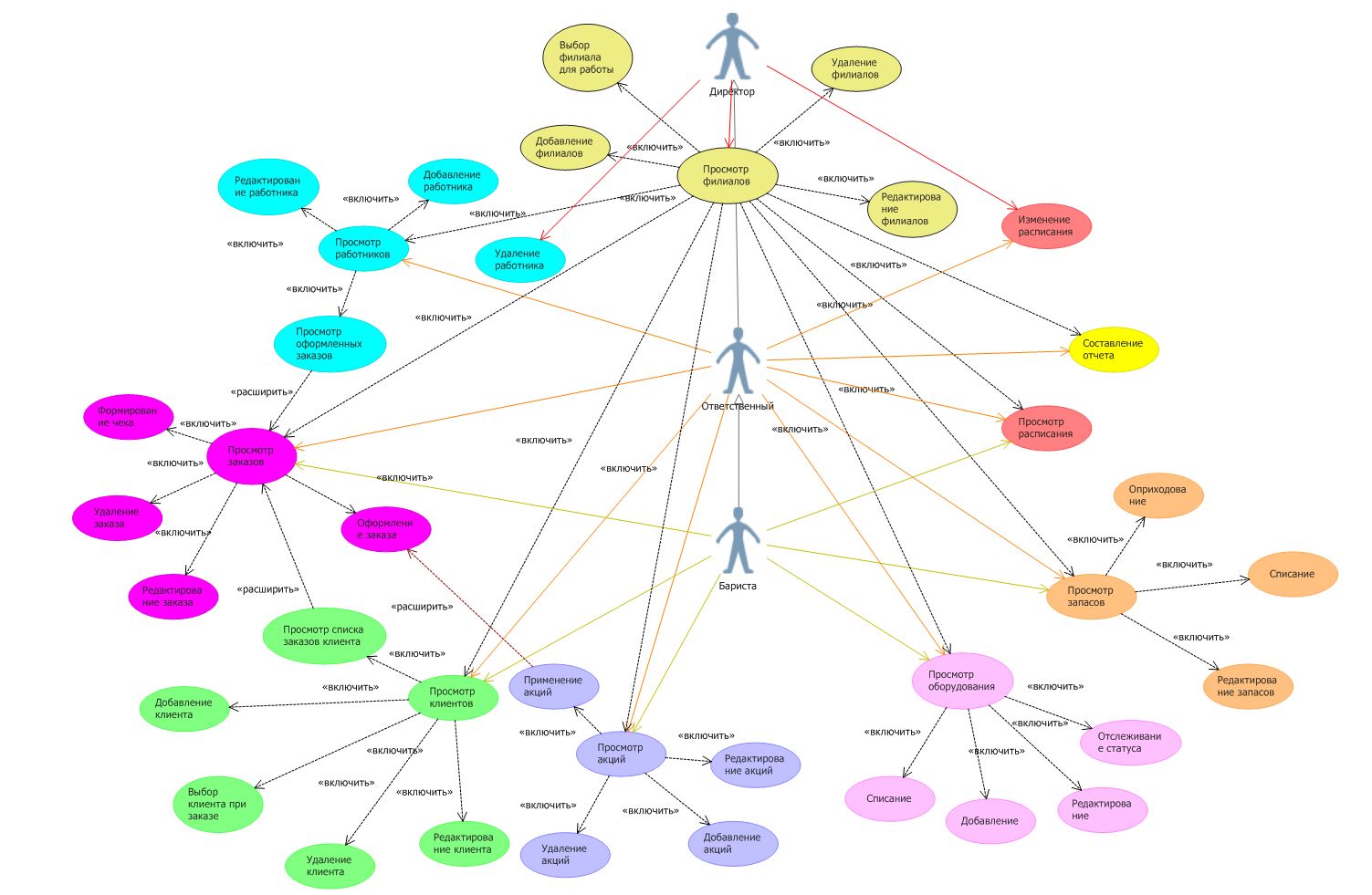


Рисунок 1 Диаграмма прецедентов

Описание прецедентов

Таблица 1 Описание прецедента "Просмотр филиалов"

|  |  |
| --- | --- |
| Имя прецедента | Просмотр филиалов |
| Краткое описание | Прецедент позволяет просматривать список всех филиалов |
| Акторы | Директор |
| Предусловия | Должен быть хотя бы один филиал, иначе предлагается создать новый |
| Основной поток | Выводятся названия филиалов в выпадающий список, по их выбору появляется адрес и описание. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 2 Описание прецедента "Выбор филиала для работы"

|  |  |
| --- | --- |
| Имя прецедента | Выбор филиала для работы |
| Краткое описание | Прецедент позволяет выбрать филиал для работы |
| Акторы | Директор |
| Предусловия | Должен быть выполнен прецедент “Просмотр филиалов» |
| Основной поток | Выбирается филиал из загруженного выпадающего списка. |
| Постусловия | Если прецедент был успешно завершен, то появляется возможность работы с филиалом. |
| Альтернативные потоки | — |

Таблица 3 Описание прецедента "Добавление филиала"

|  |  |
| --- | --- |
| Имя прецедента | Добавление филиала |
| Краткое описание | Прецедент позволяет добавлять новые филиалы |
| Акторы | Директор |
| Предусловия | Должен быть выполнен прецедент “Просмотр филиалов” |
| Основной поток | Открывается форма добавления нового филиала, куда заносятся все необходимые данные. |
| Постусловия | Если прецедент был успешно завершен, то форма добавления филиала закрывается, и в БД добавляется новый филиал. |
| Альтернативные потоки | Если имеются пустые поля, либо введены некорректные данные, то пользователю выдается сообщение об ошибке и дается возможность исправить данные. |

Таблица 4 Описание прецедента "Изменение данных о филиале"

|  |  |
| --- | --- |
| Имя прецедента | Изменение данных о филиале |
| Краткое описание | Прецедент позволяет редактировать данные о филиале |
| Акторы | Директор |
| Предусловия | Должен быть выполнен прецедент “Просмотр списка филиалов” |
| Основной поток | Открывается форма редактирования филиала, где изменяются необходимые данные. |
| Постусловия | Если прецедент был успешно завершен, то форма редактирования филиала закрывается, и измененные данные о филиале сохраняются |
| Альтернативные потоки | Если имеются пустые поля, либо введены некорректные данные, то пользователю выдается сообщение об ошибке и дается возможность исправить данные. |

Таблица 5 Описание прецедента "Удаление филиала"

|  |  |
| --- | --- |
| Имя прецедента | Удаление филиала |
| Краткое описание | Прецедент позволяет удалять филиал из БД |
| Акторы | Директор |
| Предусловия | Должен быть выполнен прецедент “Просмотр списка филиалов” |
| Основной поток | При нажатии на кнопку “Удалить”, текущая запись удаляется. |
| Постусловия | Если прецедент был успешно завершен, то филиал удаляется из базы данных. |
| Альтернативные потоки | — |

Таблица 6 Описание прецедента “Просмотр списка работников”

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка путевок |
| Краткое описание | Прецедент позволяет просматривать список работников |
| Акторы | Директор, Ответственный |
| Предусловия | Должен быть хотя бы один работник в филиале, должен быть выполнен прецедент «Выбор филиала для работы» (для Директора) |
| Основной поток | Выводится информация о работниках в виде таблица. Таблица содержится во вкладке «работники». |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 7 Описание прецедента "Просмотр оформленных заказов"

|  |  |
| --- | --- |
| Имя прецедента | Просмотр оформленных заказов. |
| Краткое описание | Прецедент позволяет просмотреть список оформленных работником заказов |
| Акторы | Директор, Ответственный. |
| Предусловия | Должны быть выполнены прецеденты: «Просмотр филиалов» и «Просмотр работников» |
| Основной поток | Открывается форма со списком заказов. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 8 Описание прецедента "Просмотр акций"

|  |  |
| --- | --- |
| Имя прецедента | Просмотр акций |
| Краткое описание | Прецедент позволяет просматривать информацию об акциях |
| Акторы | Директор, ответственный, бариста |
| Предусловия | Должна быть хотя бы одна акция |
| Основной поток | На вкладке «акции» находится информация о названиях, времени действия и значении акций. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 9 Описание прецедента "Применение акций"

|  |  |
| --- | --- |
| Имя прецедента | Применение акций |
| Краткое описание | Прецедент позволяет добавлять акции к заказу |
| Акторы | Директор, ответственный, бариста |
| Предусловия | Должна быть хотя бы одна акция. |
| Основной поток | На вкладке заказа работник может применить акции к заказу. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 10 Описание прецедента "Выбор клиента при заказе"

|  |  |
| --- | --- |
| Имя прецедента | Выбор клиента при заказе |
| Краткое описание | Прецедент позволяет выбирать клиента при заказе |
| Акторы | Директор, ответственный, бариста |
| Предусловия | Должен быть хотя бы один клиент, должен быть выполнен прецедент «Просмотр клиентов». |
| Основной поток | Открывается форма клиентов, где происходит поиск и добавление нужного клиента к заказу. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 11 Описание прецедента "Отслеживание статуса"

|  |  |
| --- | --- |
| Имя прецедента | Отслеживание статуса |
| Краткое описание | Прецедент позволяет менять и следить за статусом работы оборудования |
| Акторы | Директор, ответственный, бариста |
| Предусловия | Должен быть хотя бы один объект типа «оборудование», должен быть выполнен прецедент «Просмотр оборудования» |
| Основной поток | Во вкладке «Оборудование» происходит изменение и просмотр статуса. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 12 Описание прецедента "Составление отчета"

|  |  |
| --- | --- |
| Имя прецедента | Составление отчета |
| Краткое описание | Прецедент позволяет формировать отчет по результатам работы кофейни. |
| Акторы | Директор, ответственный |
| Предусловия | Филиал должен проработать некоторое время для накопления информации. |
| Основной поток | На вкладке «Отчет» можно сформировать отчет за определенный период. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 13 Описание прецедента "Формирование заказа"

|  |  |
| --- | --- |
| Имя прецедента | Формирование заказа |
| Краткое описание | Прецедент позволяет формировать заказ |
| Акторы | Директор, ответственный, бариста |
| Предусловия | Должна быть открыта вкладка «Заказ» |
| Основной поток | На форме заказа работник может сформировать заказ. |
| Постусловия | Если прецедент был успешно завершен, то в БД добавляется новый заказ. |
| Альтернативные потоки | — |

Остальные прецеденты аналогичны прецедентам, описанным в таблицах (1–4).

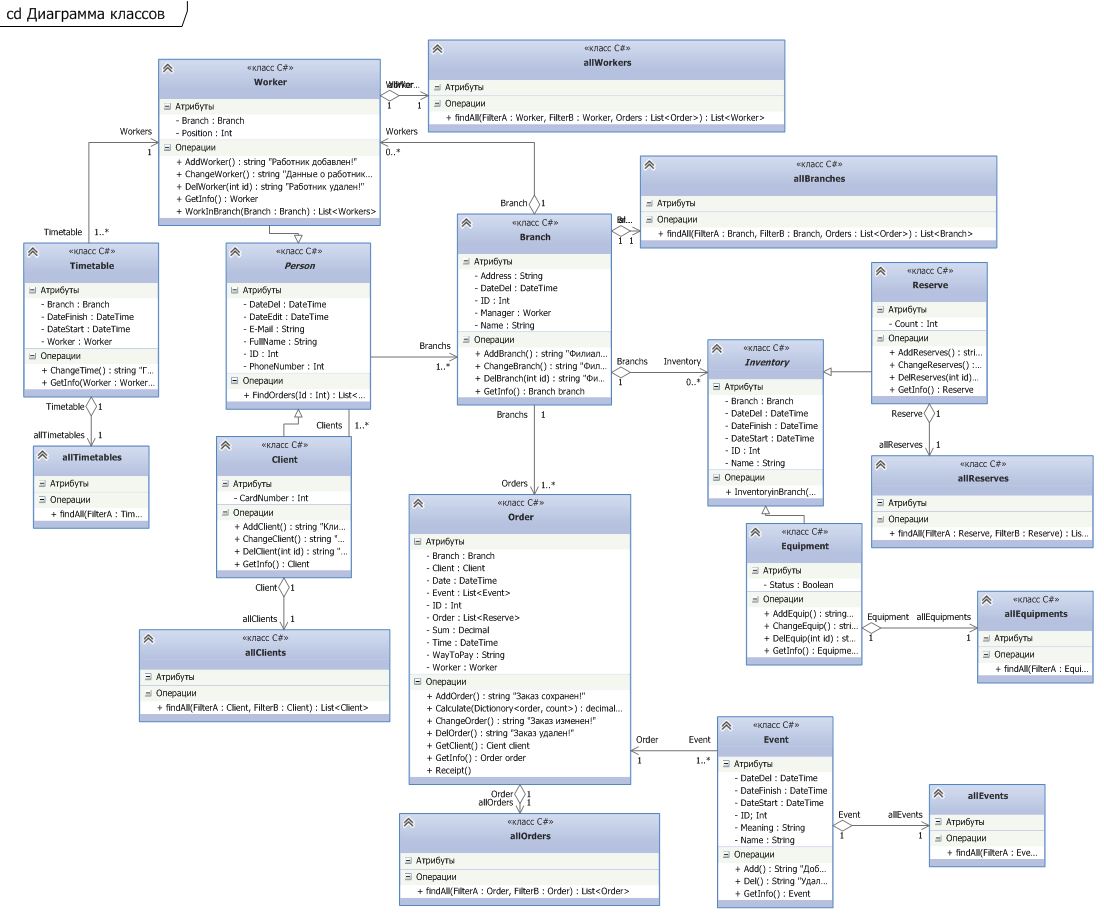
**Диаграммы классов**

Рисунок 2 Диаграммы классов

**Person**

Класс, являющийся родительским для классов Client и Worker. Содержит атрибуты: DateDel (дата удаления), DateEdit (дата создания), E-Mail (адрес почты), FullName (полное имя), ID (идентификатор), PhoneNumber (номер телефона). Является абстрактным.

+ findOrders(id: Int): List<Order> - функция, которая получает id и позволяет получать список заказов персоны.

**Worker**

Класс, являющийся наследником класса Person. Содержит атрибуты: Branch (филиал), Position (должность).

+ AddWorker() : string «Работник добавлен!» — функция, которая позволяет добавлять работника; **(1)**

+ ChangeWorker() : string «Данные о работнике изменены!» — функция, которая позволяет редактировать работника; **(2)**

+ DelWorker() : string «Работник удален!» — функция, которая позволяет удалять работника; **(3)**

+ GetInfo() : Worker — функция, которая возвращает информацию о работнике; **(4)**

+ WorkinBranch(Branch : Branch) : List <Workers> — функция, которая позволяет получить список работников филиала;

**allWorkers (5)**

Класс, связанный агрегацией «один-ко-многим» с классом Worker. Содержит открытую функцию «findAll».

+ + findAll(FilterA : Worker, FilterB : Worker, Orders : List<Order>) : List<Worker> — функция с входными параметрами «filterA», «filterB»(фильтр), «Orders»(заказы). Функция возвращает список работников.

**Client**

Класс, являющийся наследником класса Person. Содержит атрибут CardNumber (номер карты).

+ AddClient() : string «Клиент добавлен!» — функция, которая позволяет добавлять клиента;

+ ChangeClient() : string «Данные о клиенте изменены!» — функция, которая позволяет редактировать клиента;

+ DelClient() : string «Клиент удален!» — функция, которая позволяет удалять клиента;

+ GetInfo() : Client— функция, которая возвращает информацию о клиенте;

**allClients**

Класс, связанный агрегацией «один-ко-многим» с классом Client. Содержит открытую функцию «findAll».

+ + findAll(FilterA : Client, FilterB : Client) : List<Client>— функция с входными параметрами «filterA», «filterB»(фильтр). Функция возвращает список клиентов.

**Timetable**

Класс, связанный агрегацией «один-ко-многим» с классом Worker. Содержит атрибуты: Branch (филиал), DateFinish (время конца смены), DateStart (время начала смены), Worker (работник).

+ ChangeTime() – аналогична **(2)** функции;

+ GetInfo(Worker: Worker, DateStart) : Timetable timetable – Возвращает расписание работника

**allTimetables** аналогичен **(5)** классу.

Класс, связанный агрегацией «один-ко-многим» с классом Timetable. Содержит открытую функцию «findAll».

+ findAll(FilterA : Timetable, FilterB : Timetable) : List<Timetable>

**Branch**

Содержит атрибуты: Address (Адрес), DateDel (Дата удаления), ID (идентификатор), Manager (ответственный). Связан агрегацией «один-ко-многим» с классами Worker, Client, Inventory, Orders.

+ AddBranch() – аналогичен **(1)** функции;

+ ChangeBranch() – аналогичен **(2)** функции;

+ DelBranch() – аналогичен **(3)** функции;

+ GetInfo() – аналогичен **(4)** функции;

**allBranches** аналогичен **(5)** классу.

Класс, связанный агрегацией «один-ко-многим» с классом Branch. Содержит открытую функцию «findAll».

+ + findAll(FilterA : Branch, FilterB : Branch, Orders : List<Order>) : List<Branch>

**Inventory**

Класс, являющийся родительским для классов Reserve и Equipment. Содержит атрибуты: Branch (филиал), DateDel (дата удаления), DateFinish (дата окончания срока годности), DateStart (дата поступления), ID (идентификатор), Name (наименование). Является абстрактным.

+ InventoryinBranch(Branch: Branch) : List <Inventory> — возвращает список инвентаря в филиале;

**Equipment**

Наследник класса Inventory. Содержит атрибут Status (Статус).

+ AddEquip() – аналогичен **(1);**

+ ChangeEquip() – аналогичен **(2);**

+ DelEquip() – аналогичен **(3);**

+ GetInfo() – аналогичен **(4)**;

**allEquipments** аналогичен **(5)** классу.

Класс, связанный агрегацией «один-ко-многим» с классом Equipment. Содержит открытую функцию «findAll».

+ findAll(FilterA : Equipment, FilterB : Equipment): List<Equipment>

**Reserve**

Класс, являющийся наследником класса Inventory. Содержит атрибут Count (Количество).

+ AddReserves() – аналогичен **(1)**

+ ChangeReserves() – аналогичен **(2)**

+ DelReserves() – аналогичен **(3)**

+ GetInfo() – аналогичен **(4)**

**allReserves** аналогичен **(5)** классу.

Класс, связанный агрегацией «один-ко-многим» с классом Reserve. Содержит открытую функцию «findAll».

+ findAll(FilterA : Reserve, FilterB : Reserve): List<Reserve>

**Order**

Содержит атрибуты: Branch (Филиал), Client (Клиент), Date (Дата), Event (Список акций), ID (Идентификатор), Order (Список товаров), Sum (Сумма), Time (Время), WayToPay (Способ оплаты), Worker (Работник). Связан композицией «один-ко-многим» с классом Event.

+ AddOrder() – аналогичен **(1);**

+ Calculate(Dictionary<order, count>, List<event>) : sum – статический метод, позволяет рассчитать итоговую сумму заказа.

+ ChangeOrder() – аналогичен **(2);**

+ DelOrder() – аналогичен **(3);**

+ GetClient() : Client client – метод, позволяющий получить клиента, сделавшего данный заказ.

+ GetInfo() – аналогичен **(4);**

+ Receipt() – формирует чек.

+ City()— конструктор класса City;

**allOrders** аналогичен **(5)** классу.

Класс, связанный агрегацией «один-ко-многим» с классом Orders. Содержит открытую функцию «findAll».

+ findAll(FilterA : Order, FilterB : Order): List<Order>

**Event**

Содержит атрибуты: DateDel (Дата удаления), DateFinish (Дата окончания), DateStart (Дата начала), Meaning (Значение), Name (Наименование), ID (Идентификатор).

+ Add() – аналогичен **(1);**

+ Del() – аналогичен **(3);**

+ GetInfo() – аналогичен **(4).**

**allEvents** аналогичен **(5)** классу.

Класс, связанный агрегацией «один-ко-многим» с классом Event. Содержит открытую функцию «findAll».

+ findAll(FilterA : Event, FilterB : Event): List<Event>