Лабораторная работа №2

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 перегрузки операций (в одном классе): 1 балл;
* использование переопределения (между классами): 1 балл.

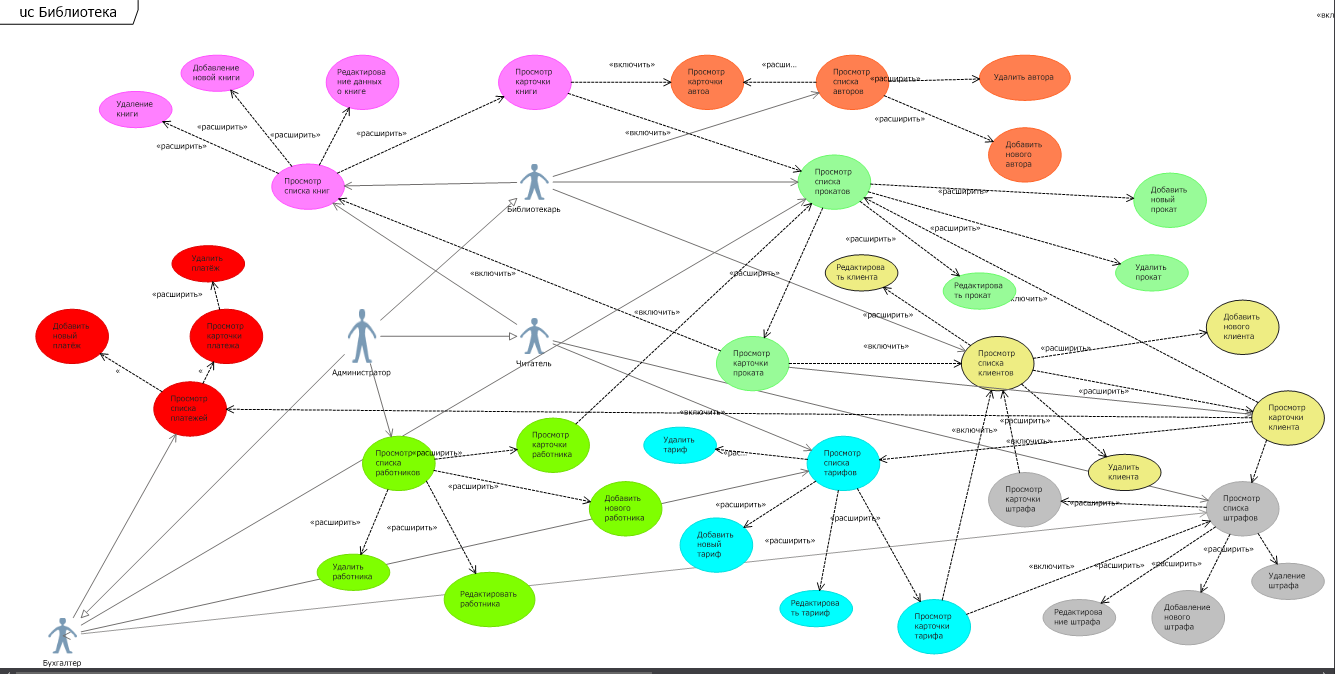


Рисунок Диаграмма прецедентов

Описание прецедентов

Таблица 1 Описание прецедента "Просмотр списка работников"

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка работников |
| Краткое описание | Прецедент позволяет просматривать список всех работников |
| Актёры | Библиотекарь, бухгалтер, администратор |
| Предусловия | Должен быть хотя бы один работник. Если таковых нет, то пользователю предлагается создать нового работника. |
| Основной поток | Выводится информация обо всех работниках в виде таблицы |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 2 Описание прецедента "Добавление записи нового работника"

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

Таблица 3 Описание прецедента "Изменение данных о работнике"

|  |  |
| --- | --- |
| Имя прецедента | Изменение данных о работнике |
| Краткое описание | Прецедент позволяет редактировать данные о работнике |
| Актёры | Администратор |
| Предусловия | Должен быть выполнен прецедент “Просмотр списка работников” |
| Основной поток | Открывается форма редактирования работника, где изменяются необходимые данные. |
| Постусловия | Если прецедент был успешно завершен, то форма редактирования работника закрывается, и измененные данные о работника сохраняются |
| 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 " Удаление работника"

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

Таблица 5 Описание прецедента “Просмотр списка книг”

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка книг |
| Краткое описание | Прецедент позволяет просматривать список всех книг |
| Актёры | Библиотекарь |
| Предусловия | Должна быть хотя бы одна книга |
| Основной поток | Выводится информация обо всех книгах в виде таблицы. Таблица содержится как в карточке клиента (в ней содержатся все книги конкретного клиента), так и отдельно (в ней содержится список всех книг) |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 6 Описание прецедента "Добавление книги"

|  |  |
| --- | --- |
| Имя прецедента | Добавление книги |
| Краткое описание | Прецедент позволяет добавлять новые книги |
| Актёры | Библиотекарь |
| Предусловия | Должен быть выполнен прецедент Просмотр списка книг. |
| Основной поток | Открывается форма добавления книги, куда вносятся необходимые данные. |
| Постусловия | Если прецедент был успешно завершен, то в БД добавляется новая книга. |
| Альтернативные потоки | Если имеются пустые поля, либо введены некорректные данные, то пользователю выдается сообщение об ошибке и дается возможность исправить данные. |

Таблица 7 Описание прецедента "Изменение данных о книге"

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

Таблица 8 Описание прецедента "Удаление книги"

|  |  |
| --- | --- |
| Имя прецедента | Удаление книги |
| Краткое описание | Прецедент позволяет удалять книги |
| Актёры | Библиотекарь |
| Предусловия | Должен быть выполнен прецедент “Просмотр списка книг” |
| Основной поток | При нажатии на кнопку “Удалить”, текущая запись удаляется. |
| Постусловия | Если прецедент был успешно завершен, то книга удаляется из базы данных. |
| Альтернативные потоки | — |

Таблица 9 Описание прецедента “Просмотр списка прокатов”

|  |  |
| --- | --- |
| Имя прецедента | Просмотр списка прокатов |
| Краткое описание | Прецедент позволяет просматривать список всех прокатов |
| Актёры | Администратор, библиотекарь |
| Предусловия | Должен быть хотя бы один прокат |
| Основной поток | Выводится информация обо всех прокатах в виде таблицы. |
| Постусловия | Если прецедент был успешно завершен, то пользователь может выполнять другие действия. |
| Альтернативные потоки | — |

Таблица 10 Описание прецедента "Добавление проката"

|  |  |
| --- | --- |
| Имя прецедента | Добавление проката |
| Краткое описание | Прецедент позволяет добавлять новые книги |
| Актёры | Библиотекарь |
| Предусловия | Должны быть выполнены прецеденты “Просмотр списка книг”, “Просмотр списка читателей” |
| Основной поток | Открывается форма добавления проката, куда вносятся необходимые данные. |
| Постусловия | Если прецедент был успешно завершен, то в БД добавляется новый прокат. |
| Альтернативные потоки | Если имеются пустые поля, либо введены некорректные данные, то пользователю выдается сообщение об ошибке и дается возможность исправить данные. |

Таблица 11 Описание прецедента "Изменение данных о прокате"

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

Таблица 12 Описание прецедента "Удаление проката"

|  |  |
| --- | --- |
| Имя прецедента | Удаление проката |
| Краткое описание | Прецедент позволяет удалять прокат |
| Актёры | Библиотекарь |
| Предусловия | Должен быть выполнен прецедент “Просмотр списка прокатов” |
| Основной поток | При нажатии на кнопку “Удалить”, текущая запись удаляется. |
| Постусловия | Если прецедент был успешно завершен, то пользователь удаляется из базы данных. |
| Альтернативные потоки | — |

Остальные прецеденты аналогичны прецедентам, описанным выше

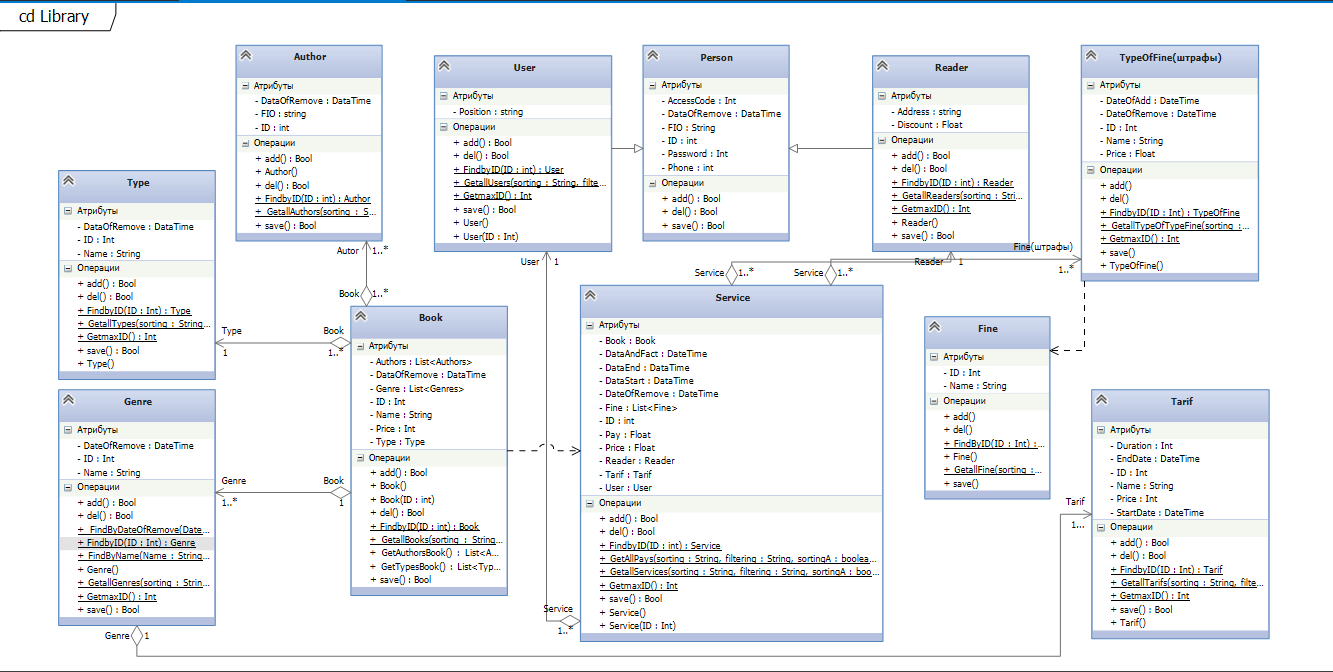


Рисунок Диаграмма классов

**Person**

Класс Person является предком классов Reader и User. Этот класс содержит атрибуты: FIO, ID, Phone, DataOfRemove, Password, AccessCode.

+add() – функция добавления человека в БД;

+del() – функция удаления человека из БД;

+save() – функция сохранения изменений;

**Reader**

Данный класс наследует все операции класса Person. Этот класс содержит атрибуты Person и атрибуты: Address (домашний адрес), Discount(скидка).

+Reader() – конструктор класса Reader без параметров

+ GetMaxID() : Int- получение максимального ID

+ FindByID(ID : Int) : Reader- поиск читателя по ID

+ GetallReaders(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Reader>) - выводит список всех читателей, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

**User**

Данный класс наследует все операции класса Person. Этот класс содержит атрибуты, описывающие отдельного работника и атрибуты: Position(должность).

+User() – конструктор класса User без параметров

+User(ID : Int) – конструктор класса User принимающий в качестве параметра ID работника

+ FindByID(ID : Int) : User - поиск ползователя по ID

+ GetMaxID() : Int- получение максимального ID

+ GetallUsers(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<User>) - выводит список всех пользователей, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

**Service**

Класс отображающий оказанные услуги. Атрибуты: ID, DataOfRemove, Price, Pay, Tarif, Fine(штраф), Book(Книга), User(Работник), Reader(читатель), DataStart(дата начала), DataEnd(дата возврата), DataEndFact(дата возврата по факту).

+add() – функция добавления услуги в БД;

+del() – функция удаления услуги из БД;

+save() – функция сохранения изменений;

+Service () – конструктор класса Service без параметров

+ Service (ID : Int) – конструктор класса Service принимающий в качестве параметра ID услуги

+ FindByID(ID : Int) : Service - поиск услуги по ID

+ GetMaxID() : Int- получение максимального ID

+ GetAllPays(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Pay>) - выводит список всех платежей, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

+ GetallServices(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Service>) - выводит список всех услуг, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

**Book**

Класс отображающий все книги.

Атрибуты: Autor(автор), Genre(жанр), ID, Name, Price(цена), DataOfRemove (дата удаления), Type(тип книги: журнал, аудио-кига и т.д). Связан с классами Type, Autor, Genre при создании карточки книги нужно указать автора и тип книги.

+add() – функция добавления книги в БД;

+del() – функция удаления книги из БД;

+save() – функция сохранения изменений;

+Book() – конструктор класса Book без параметров

+ Book (ID : Int) – конструктор класса Book принимающий в качестве параметра ID книги

+ FindByID(ID : Int) : Book - поиск книги по ID

+ GetMaxID() : Int- получение максимального ID

+ GetallBooks(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Book>) - выводит список всех книг, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

+ GetAuthorsBook() : List<Authors> - выводит авторов этой книги

+ GetTypesBook() : List<Types> - выводит типы данной книги

**Autor**

Класс отображающий автора.

Атрибуты: FIO, ID, DataOfRemove

+add() – функция добавления автора в БД;

+del() – функция удаления автора из БД;

+save() – функция сохранения изменений;

+Autor() – конструктор класса Autor без параметров

+ FindByID(ID : Int) : Autor - поиск автора по ID

**+** GetallAuthors(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Author>) - выводит список всех авторов, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

**Type**

Класс отображающий тип книги.

Атрибуты: Name, ID, DataOfRemove

+add() – функция добавления типа в БД;

+del() – функция удаления типа из БД;

+save() – функция сохранения изменений;

+ Type() – конструктор класса Type без параметров

+ FindByID(ID : Int) : Type - поиск типа по ID

+ GetMaxID() : Int- получение максимального ID

**+** GetallTypes(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Type>) - выводит список всех типов, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

**Genre**

Класс отображающий жанр книги.

Атрибуты: Name, ID, DataOfRemove

+add() – функция добавления типа в БД;

+del() – функция удаления типа из БД;

+save() – функция сохранения изменений;

+ Genre() – конструктор класса Genre без параметров

+ Genre (ID : Int) – конструктор класса Genre принимающий в качестве параметра ID книги

+ FindByID(ID : Int) : Book - поиск книги по ID

+ GetMaxID() : Int- получение максимального ID

+ GetallGenres(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Genre>) - выводит список всех жанров, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

**Fine**

Класс, отображающий штрафы. Атрибуты: ID, Name. Связан с классом TypeOfFine, при оформлении штрафа нужно указать его тип.

+add() – функция добавления услуги в БД;

+del() – функция удаления услуги из БД;

+save() – функция сохранения изменений;

+Fine () – конструктор класса Fine без параметров

+ FindByID(ID : Int) : Fine - поиск штрафа по ID

+ GetallFines(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Book>) - выводит список всех штрафов, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

**TypeOfFine**

Класс отображающий типы штрафов.

Атрибуты: ID, Name, Price, DataOfRemove, DateOfAdd

+add() – функция добавления книги в БД;

+del() – функция удаления книги из БД;

+save() – функция сохранения изменений;

+ FindByID(ID : Int) : TypeOfFine - поиск книги по ID

+ GetMaxID() : Int- получение максимального ID

+ GetallTypeOfFine (sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Book>) - выводит список всех типов штрафов, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)

**Tarif**

Класс отображающий все тарифы.

Атрибуты: Duration(продолжительность), ID, Name, Price(цена), StartDate, EndDate

+add() – функция добавления книги в БД;

+del() – функция удаления книги из БД;

+save() – функция сохранения изменений;

+Tarif () – конструктор класса Tarif без параметров

+ FindByID(ID : Int) : Tarif - поиск книги по ID

+ GetMaxID() : Int- получение максимального ID

+ GetallTarifs(sorting : String, filtering : String, sortingA : boolean, count : int, page : int) : List<Tarif>) - выводит список всех тарифов, sorting-сортировка по первому параметру, filtering- фильтрация, sorting-сортировка по второму параметру, count-количество строк на странице, page-текущая страница)