**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ВТ**

**Пояснительная записка**

**к курсовой работе**

**по дисциплине «Объектно-ориентированное программирование на языке Java»**

**Создание программного комплекса средствами объектно-ориентированного программирования**

|  |  |
| --- | --- |
| Студент гр. 7307 | Торопов В. А. |
| Преподаватель | Павловский М.Г. |

Санкт-Петербург

2019

1.Техническое задание

## 1.1 Введение

Программный комплекс (ПК) для работников библиотеки предназначен для использования в составе системы программно-информационного обеспечения для хранения и контроля сведений об имеющихся в библиотеке книгах и о читателях библиотеки

## 1.2 Основание для разработки

Основанием для разработки ПК «Библиотека» является курсовой проект по дисциплине «Объектно-ориентированное программирование».

## 1.3 Назначение разработки

ПК «Библиотека» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности лица (ОЛ), ответственного за контроль библиотеки

ПК «Библиотека» предназначен для автоматизации следующих процессов:

* учет и администрирование информации о книгах;
* учет и администрирование информации о читателях;
* получение справочной информации о работе библиотеки;

## 1.4 Требования к программе

### 1.4.1 Требования к функциональным характеристикам

*1.4.1.1 Перечень функций*

ПК «Библиотека» должен обеспечивать выполнение следующих функций:

* просмотр, добавление, удаление и изменение в базе данных (БД);
* выдача справочной информации, хранимой в БД, по запросам ОЛ.

*1.4.1.2 Требования к составу выполняемых функций*

*1.4.1.2.1 Функция «*просмотр, добавление, удаление и изменение в базе данных (БД)*»*

Ввод, просмотр, добавление, удаление и изменение в БД должны обеспечивать ведение и хранение следующих данных:

* данных о чистателях;
* данных о книгах.

*1.4.1.2 Требования к организации и форме представления выходных данных*

Выходные данные должны быть представлены в виде таблицы содержащий описание необходимых информационных объектов, выполненного посредствам представления его характеристик.

*1.4.1.3 Требования к организации и форме представления входных данных*

Входная информация для задачи «Библиотека» содержится в приходно-расходной документации. Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов.

### 1.4.2 Требования к надежности

ПК «Библиотека» должен устойчиво функционировать при соблюдении гарантии устойчивого функционирования операционной системы и системы управления базой данных. Под устойчивой работой ПК понимается непрерывное функционирование программы в отсутствии критических сбоев, приводящих к аварийному завершению. Кроме того, должен быть обеспечен контроль входных данных на предмет соответствия предполагаемому типу.

### 1.4.3 Условия эксплуатации

Выполнение ПК «Библиотка» своих функций должно быть обеспечено для однопользовательского режима работы с монопольным доступом к базе данных.

### 1.4.4 Требование к составу и параметрам технических средств

Задача должна решаться на ПЭВМ типа IBM РС или совместимой с ней с процессором Pentium III 500 и выше, ОЗУ не менее 128Мб, HDD не менее 4 Гб, монитор SVGA (цветной)15", видеокарта 64 Мб, клавиатура 102 кл., манипулятор типа "мышь".

### 1.4.5 Требование к информационной и программной совместимости

Выходная и входная информация ПК «Библиотека» должна быть удобна для визуального восприятия. ПК должен быть выполнен на языке программирования высокого уровня Python и должен быть совместим с операционной системой Windows.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Python:

* закрытые и открытые члены классов;
* наследование;
* конструкторы с параметрами;
* абстрактные базовые классы;
* виртуальные функции;
* обработка исключительных ситуаций;
* динамическое создание объектов.

## 1.5 Требования к программной документации

Программная документация (ПД) должна удовлетворять требованиям стандартов ЕСПД.

Документация должна быть представлена в следующем составе:

1. описание процесса проектирования ПК;
2. руководство оператора;
3. исходные тексты ПК.

## 1.6 Стадии и этапы разработки

1. Разработка технического задания;
2. Описание вариантов использования ПК;
3. Создание прототипа интерфейса пользователя;
4. Разработка объектной модели ПК;
5. Построение диаграмм программных классов;
6. Описание поведения ПК;
7. Построение диаграмм действий;

## 1.7 Порядок контроля и приемки

В процессе приема работы устанавливается соответствие ПК и прилагаемой документации требованиям, обозначенным в техническом задании.

# 2 Проектирование ПК

## 2.1 Описание вариантов использования ПК

Развернутое описание функциональных требований осуществляется на этапе проектирования комплекса. Для того чтобы детализировать требования, необходимо выделить процессы, происходящие в заданной предметной области. Описание таких процессов на UML выполняется в виде прецедентов (use case). Прецеденты являются сценарием или вариантом использования ПК при взаимодействии с внешней средой. Они являются продолжением описаний требований и функциональных спецификаций, указанных в техническом задании. Прецедент изображается в виде эллипса, в котором содержится имя прецедента. Название прецедента обязательно включает в себя глагол, выражающий суть выполняемой функции. С помощью прецедентов описывается функционирование ПК с точки зрения внешнего пользователя, который называется в UML актором (actor). Актор представляет собой любую внешнюю по отношению к моделируемой системе сущность (человек, программная система, устройство), которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Актор на диаграмме изображается пиктограммой в виде человечка, под которым указано его имя. Совокупность функций, реализуемых ПК, изображается в виде диаграммы (use case diagram). Для построения диаграммы необходимо определить акторы, прецеденты (функции) и взаимоотношение между акторами и прецедентами, и между прецедентами, если один прецедент расширяет или использует другой. В языке UML для вариантов использования и действующих лиц поддерживается несколько типов связей. Это связи коммуникации (communication), использования (uses) и расширения (extends).

Связь коммуникации — это связь между прецедентом и актором. На языке UML связь коммуникации изображают в виде стрелки. Направление стрелки показывает, кто инициирует коммуникацию. При задании коммуникации необходимо указать данные, которые вводит или получает пользователь. Кроме данных на концах стрелки можно указать кратности отношения, которые характеризуют количество взаимодействующих между собой акторов и прецедентов. На диаграммах прецедентов наиболее распространенными являются две формы записи кратности 1 и 1 .. \*. Первая форма записи означает, что один актор (прецедент) участвует во взаимодействии, а вторая форма записи, что один или несколько акторов (прецедентов) участвуют во взаимодействии.

Связь использования предполагает, что один прецедент всегда применяет функцио­нальные возможности другого. С помощью таких связей структурируют прецеденты, показывая тем самым, какой прецедент является составной частью другого прецедента. Такой включаемый прецедент является абстрактным прецедентом в том смысле, что он не может исполняться независимо от других прецедентов, а лишь в их составе. Связь использования изображается с помощью стрелок и слова «uses» (использование). Направление стрелки указывает, какой прецедент используется для реализации функциональности другого прецедента.

Связь расширения задается в том случае, если необходимо показать родственные отношения между двумя прецедентами. Один из них является базовым, а другой его расширением. Базовый прецедент не зависит от расширяющих прецедентов и способен функционировать без них. С другой стороны, расширяющие прецеденты без базового прецедента функционировать не могут. Связи расширения изображают в виде стрелки со словом «extends» (расширение), которая имеет направление от базового прецедента к расширяемому.

Прецеденты необходимо ранжировать, чтобы в начальных циклах разработки реализовать наиболее приоритетные из них. Разбиение функциональности системы на отдельные прецеденты служит примерно той же цели, что и разбиение сложного алгоритма на подпрограммы. Основная стратегия должна заключаться в том, чтобы сначала сконцентрировать внимание на тех прецедентах, которые в значительной мере определяют базовую архитектуру ПК.

Диаграмма прецедентов представлена на рис. 2.1.

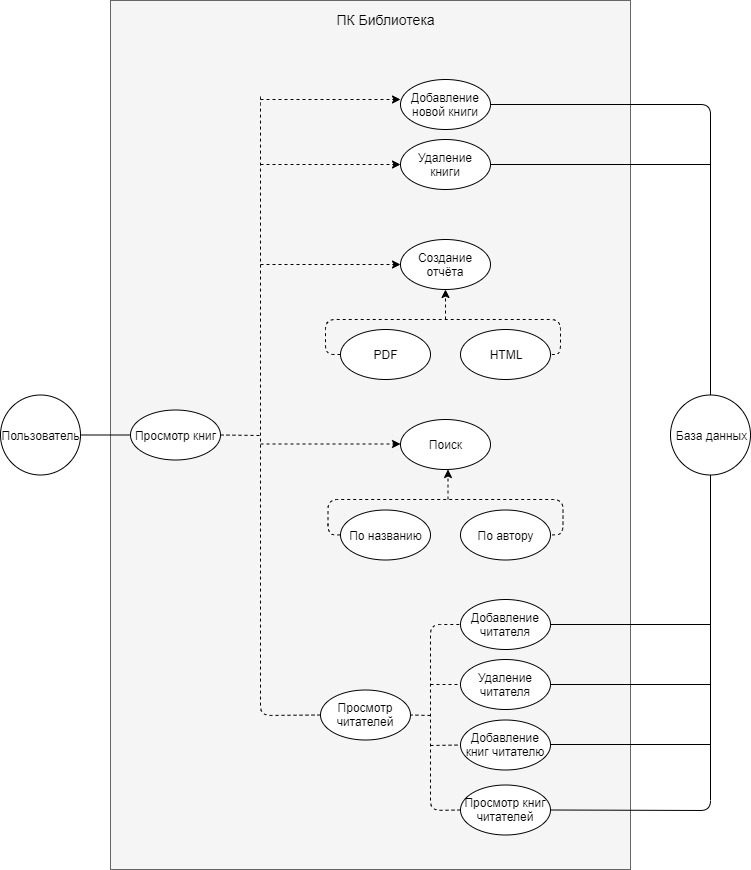


Рис.2.1 Диаграмма прецедентов

## 2.3 Создание прототипа интерфейса пользователя

Описание прецедента выражает общую сущность процесса без детализации его реализации. Проектные решения, связанные с интерфейсом пользователя, при этом опускаются. Для разработки пользовательского интерфейса необходимо описать процесс в терминах реальных проектных решений, на основе конкретных технологий ввода-вывода информации. Когда речь идет об интерфейсе пользователя, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с конкретными устройствами. Для каждой экранной формы указываются поля ввода и перечень элементов управления, действия пользователя (нажать кнопку, выбрать пункт меню, ввести данные, нажать правую/левую кнопку мыши) и отклики системы (отобразить данные, вывести подсказку, переместить курсор). Такое описание интерфейса представляется в виде таблицы экранных форм.

В табл. 2.1 представлено описание экранных форм.

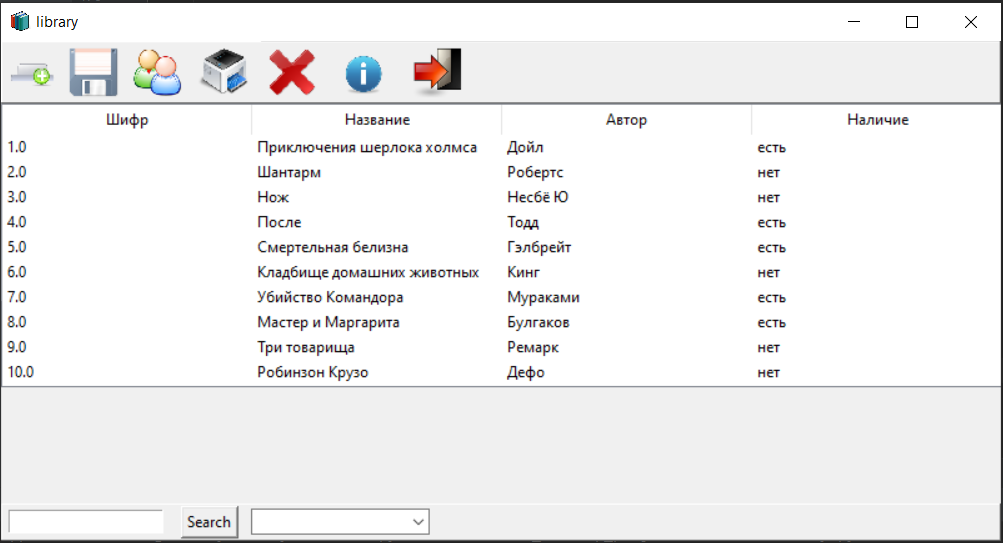


Рис.2.2. Главная экранная форма

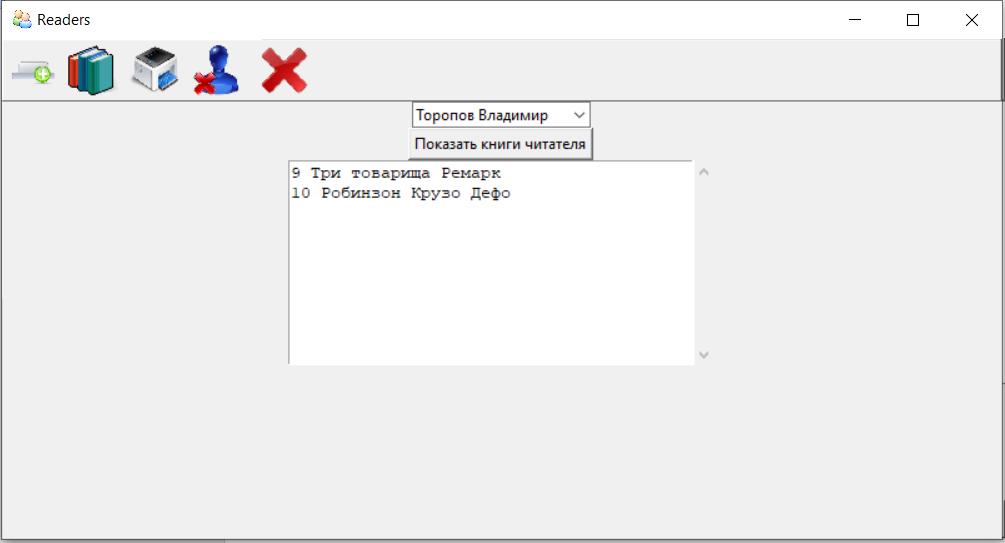
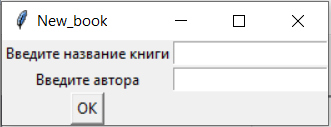


Рис. 2.3. Экранная форма просмотра читателей



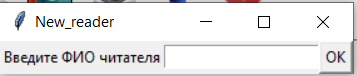
Рис. 2.4. Экранная форма добавления книги

Рис. 2.5. Экранная форма добавления читателя

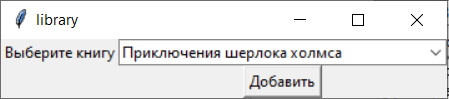


Рис. 2.6. Экранная форма для добавления книги читателю

Таблица 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| Экранная форма | Элементы управления | Действия пользователя | Отклик системы |
| Главная экранная форма | Кнопки:  «Добавить новую книгу»,  «Просмотр списка читателей»  «Сохранить»  «Создать отчет»  «Удалить книгу»  «Выход» | нажать кнопку «Добавить новую книгу» | Открыть окно «New\_book» на добавление |
| нажать кнопку «Просмотр списка читателей» | Открыть окно «Readers» для просмотра |
| нажать кнопку «Сохранить» | Сохранить изменения |
| нажать кнопку «Создать отчет» | Создать HTML отчет, из XML файла |
| Двойной клик на строку таблицы | Выделить строку таблицы |
| Выделить одну строку таблицы и нажать кнопку «Удалить книгу» | Удаление книги из таблицы |
| Нажать кнопку «Выход» | Завершить работу приложения |
| Readers | Кнопки:  «Добавить читателя»  «Добавить книгу читателю»  «Удалить читателя»  «Списать книгу у читателя» «Показать книги читателя» | Нажать кнопку «Добавить читателя» | Открыть окно «New\_reader» на добавление |
| Нажать кнопку «Добавить книгу» | Открыть окно «New\_book» на добавление |
| Нажатие кнопки  «Удалить читателя» | Удаление читателя из БД |
| Нажать кнопку «Списать книгу у читателя» | Открыть окно «Delete\_book» для изменения |
| Нажать кнопку «Показать книги читателя» | Показ книг читателя |
| New\_book | Кнопки:  «Ок» | Нажатие кнопки  «Ok» | Закрытие формы. Добавление\изменение данных в базе данных. |
| New\_reaader | Кнопки:  «Ok» | Нажатие кнопки  «Ok» | Закрытие формы. Добавление\изменение данных в базе данных. |

## 2.3. Разработка объектной модели ПК

Объектная модель не описывает структуру ПК, она отображает основные понятия предметной области в виде совокупности типов объектов (сущностей). Сущности строятся путем выделения их из предметной области и анализа прецедентов. На диаграмме сущность обозначается прямоугольником, внутри которого записывается имя сущности, ее атрибуты и операции.

Атрибуты описывают свойства сущности. В объектную модель включаются те атрибуты, для которых определены соответствующие требования или для которых предполагается хранить определенную информацию. Атрибут характеризуется именем и типом. Для атрибута рекомендуется использовать простые типы данных (число, строка, дата, время и другие).

Описание операций помогает определить поведение объектов сущности. На этом этапе, прежде всего, определяется внутреннее поведение каждого объекта сущности, без учета взаимодействия с другими объектами предметной области. На диаграмме обычно указывается только имя операции, а ее подробное описание приводится в отдельной таблице. В таблице должно содержаться краткое описание назначения операции, ее имя и список входных и выходных параметров.

Ассоциация между сущностями отражает некоторое бинарное отношение между ними. Ассоциация обозначается проведенной между сущностями линией, с которой связывается определенное имя. Имя записывается в глагольной форме, и оно должно отражать семантический смысл отношения. Стрелка на линии указывает, в каком направлении нужно читать имя. На концах линии могут содержаться выражения, определяющие количественную связь между экземплярами сущности (кратность). Кратность определяет, сколько экземпляров одной сущности может быть ассоциировано с одним экземпляром другой сущности. Примеры кратностей:

0 ..\* - нуль или больше,

1 .. \* - один или больше,

1 – ровно один.

Необходимо устанавливать отношения ассоциации между двумя сущностями в том случае, если объект одной сущности должен знать об объекте другой. Прежде всего, следует включать в модель те ассоциации, которые отражают структурные отношения («содержит», «включает», «хранит» и т.д.), или те, которые должны сохраняться в течение некоторого времени.

Диаграмма сущностей представлена на рис. 2.8. Детальное описание операций представлено в табл. 2.2.

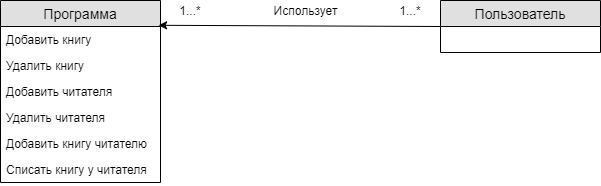


Рис. 2.8. Диаграмма сущностей

Таблица 2.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сущность | Имя операции | Параметры операции | | | Тип возвращаемого значения | Назначение операции |
| Вид | Название | Тип |
| Программа | Добавить книгу | Вх. | Название книги | Строка | БД | Добавляет в БД новую книгу |
| Автор | Строка |
| Удалить книгу | Вх. | Шифр книги | Целое | БД | Удаляет выбранную книгу |
| Добавить читателя | Вх. | ФИО | Строка | БД | Добавляет в БД нового читателя |
| Удалить читателя | Вх. | ФИО читателя | Строка | БД | Удаляет выбранного читателя |
| Добавить книгу читателю | Вх. | Шифр книги | Целое | БД | Добавляет книгу данному читателю |
| ФИО читетеля | Строка |
| Списать книгу у читателя | Вх. | Шифр книги | Целое | БД | Списывает книгу у читателя |
| ФИО читателя | Строка |

## 2.4 Построение диаграммы программных классов

Диаграмма классов (class diagram) иллюстрирует спецификации будущих программных классов и интерфейсов. Она строится на основе объектной модели. В описание класса указываются три раздела: имя класса, состав компонентов класса и методы класса. Графически класс изображается в виде прямоугольника. Имя программного класса может совпадать с именем сущности или быть другим. Но поскольку для записи идентификаторов переменных в языках программирования используют латинские буквы, то и имена программных классов и имена их атрибутов, как правило, записываются латинскими буквами. Атрибуты и операции класса перечисляются в горизонтальных отделениях этого прямоугольника. Атрибутам и методам классов должны быть присвоены права доступа. Права доступа помечаются специальными знаками:

+ - означает открытый (public) доступ;

— - означает скрытый (private) доступ;

# - означает наследуемый (protected) доступ.

При описании атрибутов после двоеточия указывается их тип, а при описании методов класса возвращаемое значение (для конструкторов возвращаемое значение не указывается).

В диаграмме классов могут вводиться дополнительно новые атрибуты, операции и связи или осуществляться конкретизация ассоциаций, указанных в объектной модели. На диаграмме классов могут быть три вида отношений: ассоциация, агрегация и наследование.

На диаграмме классов ассоциация имеет такое же обозначение, как и в объектной модели. На линиях ассоциации может присутствовать стрелка. Это стрелка видимости, которая показывает направление посылки запросов в ассоциации. Стрелка видимости также показывает, какой из классов содержит компоненты для реализации отношения ассоциации, иными словами, кто является инициатором посылки запроса к другому объекту. Ассоциация без стрелки является двунаправленной.

Агрегирование - это отношение между классами типа целое/часть. Агрегируемый класс в той или иной форме является частью агрегата. На практике это может быть реализовано по-разному. Например, объект класса-агрегата может хранить объект агрегируемого класса, или хранить ссылку на него. Агрегирование изображается на диаграмме полым ромбом на конце линии со стороны агрегирующего класса (агрегата). Если агрегируемый объект может быть создан только тогда, когда создан агрегат, а с уничтожением агрегата уничтожаются и все агрегируемые объекты, то такое агрегирование называется сильным и отображается в виде закрашенного ромба.

Наследование - это отношение типа общее-частное между классами. Его следует вводить в том случае, когда поведение и состояние различных классов имеют общие черты. Наследование связывает конкретные классы с общими или в терминологии языков программирования производные классы (подклассы) с базовыми классами (суперклассами). На диаграммах наследование изображается в виде стрелки с полым треугольником, идущей от производного класса к базовому. Если один производный класс наследует несколько базовых, то такое наследование называется множественным.

Диаграмма классов представлена на рис. 2.9.

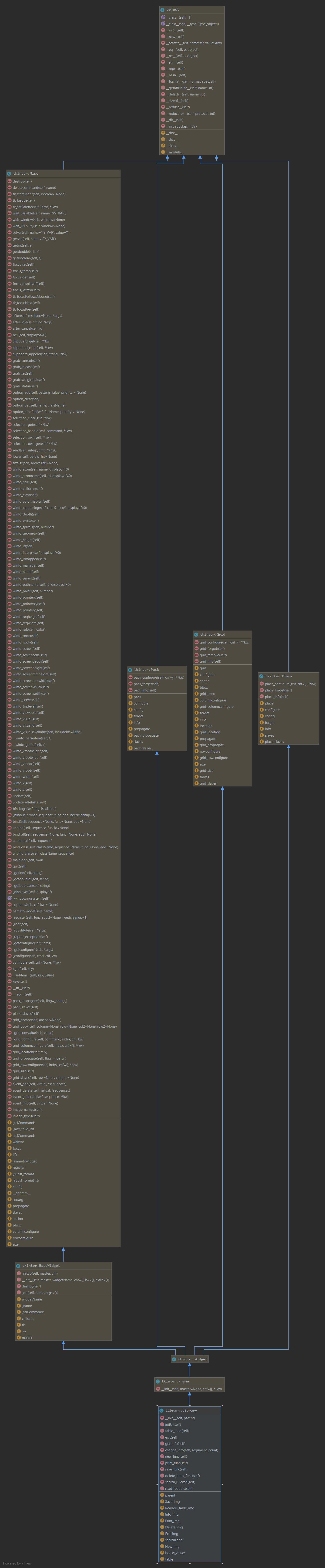


Рис.2.9. Диаграмма классов

## 2.5 Описание поведения ПК

Поведение ПК представляет собой описание того, какие действия выполняет ПК, без определения механизма их реализации. Одной из составляющей такого описания является диаграмма последовательностей (seguence diagram). Диаграмма последовательностей является схемой, которая для определенного сценария прецедента показывает генерируемые пользователями и объектами события (запросы) на выполнение некоторой операции и их порядок. Диаграммы последовательности имеют две размерности: вертикальная представляет время, горизонтальная - различные объекты. Чтобы построить диаграмму последовательностей необходимо выполнить следующие действия:

1. Идентифицировать пользователей и объекты программных классов, участвующие в начальной стадии реализации сценария прецедента, и их изображения в виде прямоугольников расположить наверху в одну линию. Для каждого пользователя и объекта нарисовать вертикальную пунктирную линию, которая является линией их жизни. Внутри прямоугольника указываются подчеркнутое имя объекта и имя класса, к которому принадлежит объект.
2. Из объектной модели выбрать те операции, которые участвуют в реализации сценария. Если такие операции не были определены при построении диаграммы программных классов, то необходимо их описать и внести в модель.
3. На диаграмме последовательностей каждому запросу на выполнение операции должна соответствовать горизонтальная линия со стрелкой, начинающаяся от вертикальной линии того пользователя или объекта, который вызывает операцию, и заканчивающаяся на линии жизни того пользователя или объекта, который будет ее выполнять. Над стрелкой указывается номер операции, число итераций, имя операции и в скобках ее параметры. После описания операции может следовать комментарий, поясняющий смысл операции и начинающийся со знака "//".

Операция, которая реализует запрос, на линии жизни объекта обозначается прямоугольником. Порядок выполнения операций определяется ее номером, который указывается перед именем, и положением горизонтальной линии на диаграмме. Чем ниже горизонтальная линия, тем позже выполняется операция. В диаграммах последовательности принято применять вложенную систему нумерации, так как это позволяет отобразить их вложенность. Нумерация операций каждого уровня вложенности должна начинаться с 1.

На диаграмме последовательностей можно описать вызов операции по условию (конструкция if-else) и показать моменты создания и уничтожения объектов. Если объект создается или уничтожается на отрезке времени, представленном на диаграмме, то его линия жизни начинается и заканчивается в соответствующих точках, в противном случае линия жизни объекта проводится от начала до конца диаграммы. Символ объекта рисуется в начале его линии жизни; если объект создается не в начале диаграммы, то сообщение о создании объекта рисуется со стрелкой, проведенной к символу объекта. Если объект уничтожается не в конце диаграммы, то момент его уничтожения помечается большим крестиком "Х".

Диаграмма последовательностей для операции добавления читателя представлена на рис. 2.10.

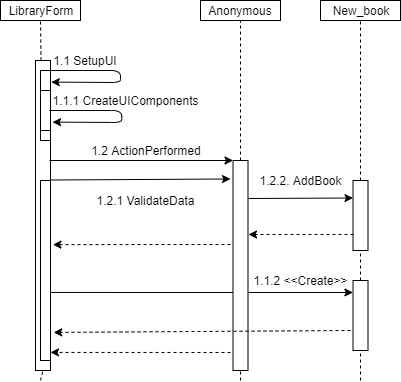


Рис.2.10 Диаграмма последовательностей

2.6 Построение диаграммы действий

Диаграмма действий (activity diagram) строится для сложных операций. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются действия, а дугами — переходы от одного действия к другому. Она очень похожа на блок-схемы алгоритмов. Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояние. Диаграмму деятельности принято строить таким образом, чтобы действия следовали сверху вниз. Отличительной чертой диаграммы действий является то, что в ней можно отобразить параллельные процессы. Для этой цели используется специальный символ (линия синхронизации), который позволяет задать разделение и слияние потоков управления. При этом разделение имеет один входящий переход и несколько выходящих, а слияние, наоборот, имеет несколько входящих переходов и один выходящий.

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Чтобы связать объекты с действиями, необходимо явно указать их на диаграмме деятельности. Для графического представления объектов, используются прямоугольник, в котором указывается подчеркнутое имя класса. Подчеркнутое имя означает, что на диаграмме задается объект, а не его класс. Далее после имени можно указать в прямых скобках значения атрибутов объекта после выполнения предшествующего действия. Такие прямоугольники объектов присоединяются к переходам отношением зависимости с помощью пунктирной линией со стрелкой.

Диаграмма действий представлена на рис. 2.11.



Рис. 2.11 Диаграмма действий

# 3. Руководство оператора

## 3.1 Назначение программы

ПК «Библиотека» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности ОЛ, ответственного за контроль библиотеки

В рамках ПК «Библиотека» ОЛ может:

* добавлять, править и удалять информацию о читателях;
* добавлять, править и удалять информацию о книгах;
* получать справочную информацию по работе библиотеки;

## 3.2 Условия выполнения программы

ПК предназначен для функционирования под операционной средой Windows (XP, 2x) при поддержке Sqile

Персональная электронно-вычислительная машина (ПЭВМ) должна обладать следующими характеристиками:

1. тип процессора Pentium III 500 и выше;
2. объем ОЗУ – не менее 128Мб;
3. объем жесткого диска – не менее 4Гб;
4. видеокарта – 64Мб;
5. стандартная клавиатура;
6. манипулятор типа "мышь".

## 3.3 Описание задачи

Разработать ПК для работников библиотеки. В ПК должны храниться сведения об имеющихся в библиотеке книгах и о читателях библиотеки. Библиотекарю могут потребоваться следующие сведения:

* какие книги закреплены за читателем;
* кто автор и как называется книга с заданным шифром.

Библиотекарь может вносить следующие изменения:

* запись нового читателя в библиотеку;
* пополнение библиотеки;
* списывание старой книги;
* изменение шифра книги.

Необходимо предусмотреть возможность выдачи справки о колличестве читателей библиотеки и работе библиотеки за месяц (количество выданных книг, число записавшихся читателей).

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Python:

* закрытые и открытые члены классов;
* наследование;
* конструкторы с параметрами;
* абстрактные базовые классы;
* виртуальные функции;
* обработка исключительных ситуаций;
* динамическое создание объектов.

С целью выполнения поставленной задачи в процессе проектирования разработана общая модель ПК с выявлением основных объектов и связей между ними. На основании полученной модели разработаны программные классы.

Требования к коду ПК учтены при создании программных классов и непосредственном написании программы.

## 3.4 Входные и выходные данные

Выходные данные должны быть представлены в виде таблице содержащий описание необходимых информационных объектов, выполненного посредствам представления его характеристик.

Входная информация для задачи «Библиотека» содержится в приходно-расходной документации. Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов. Вводимая информация может выбираться или набираться из списка предлагаемых значений.

## 3.5 Выполнение программы

### 3.5.1. Запуск программы

При запуске программы на экране появится главное диалоговое окно представленное на рис. 3.1. Туда сразу загружаются все данные.

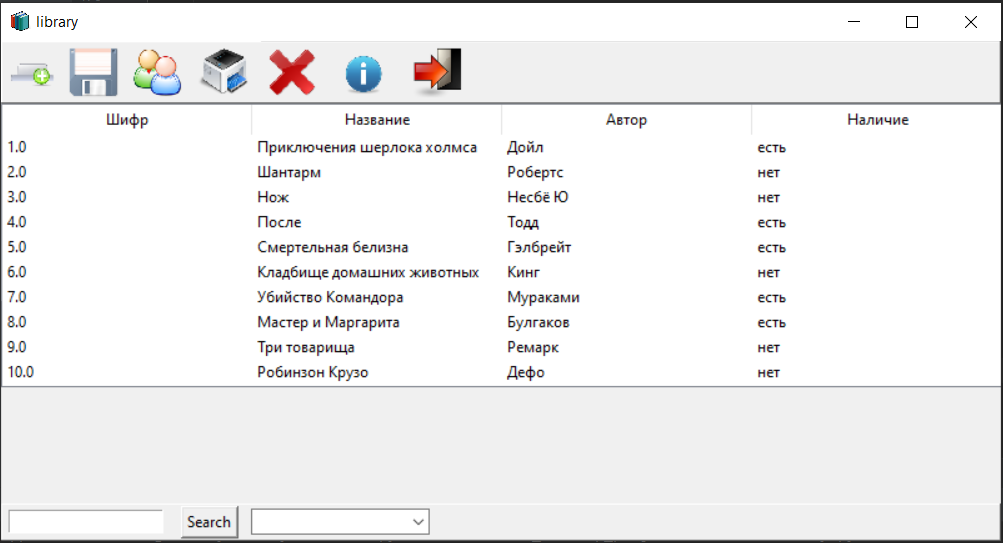


Рис. 3.1 Главное диалоговое окно

### 3.5.2. Выполнение основных функций

*3.5.2.1 Добавление книги*

При нажатии на кнопку Добавить новую книгу, откроется новое окно (рис. 3.2.), куда необходимо ввести данные, которые необходимы для добавления книги

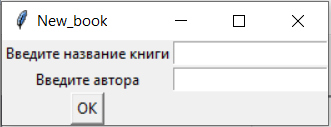


Рис. 3.2 Окно добавления книги

После ввода данных необходимо нажать кнопку «ОК» для добавления книни»

*3.5.2.2 Удаление книги*

При нажатии кнопки удалить книгу появится диалоговое окно, которое уточнит выбранное действие и при согласии выбранная книга удалится из базы данных(Рис. 3.3)

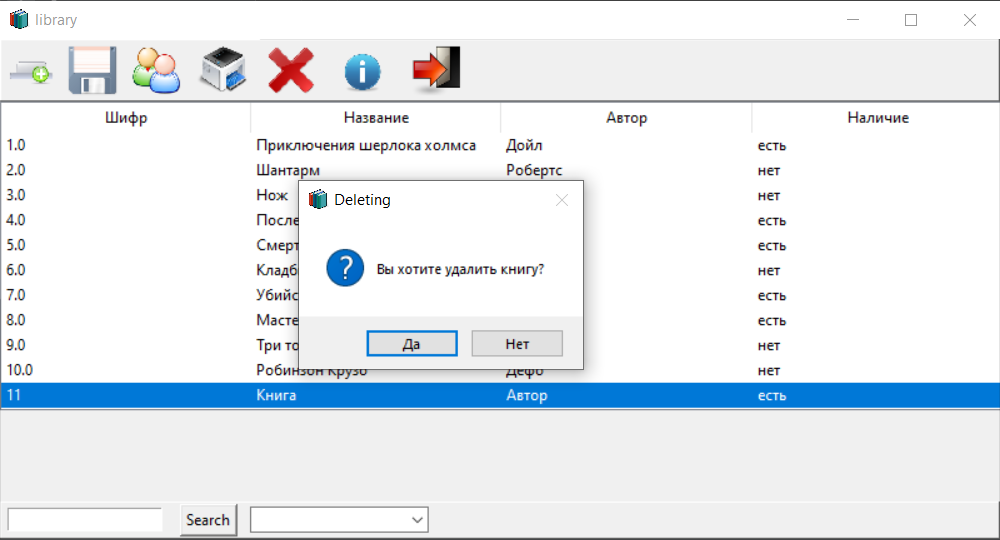


Рис. 3.3 Окно удаления книги

*3.5.2.3 Создание отчёта*

При нажатии кнопки Показать отчёт будет открыто новое окно с необходимой для нас информацией (рис 3.4.)

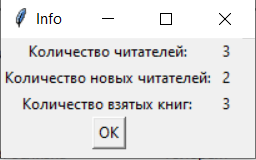


Рис. 3.4 Окно отчёта

*3.5.2.4 Просмотр всех читателей*

При нажатии на кнопку “Просмотр читателей”, откроется окно, где отображены все читатели и информация об их книгах (рис 3.5.)

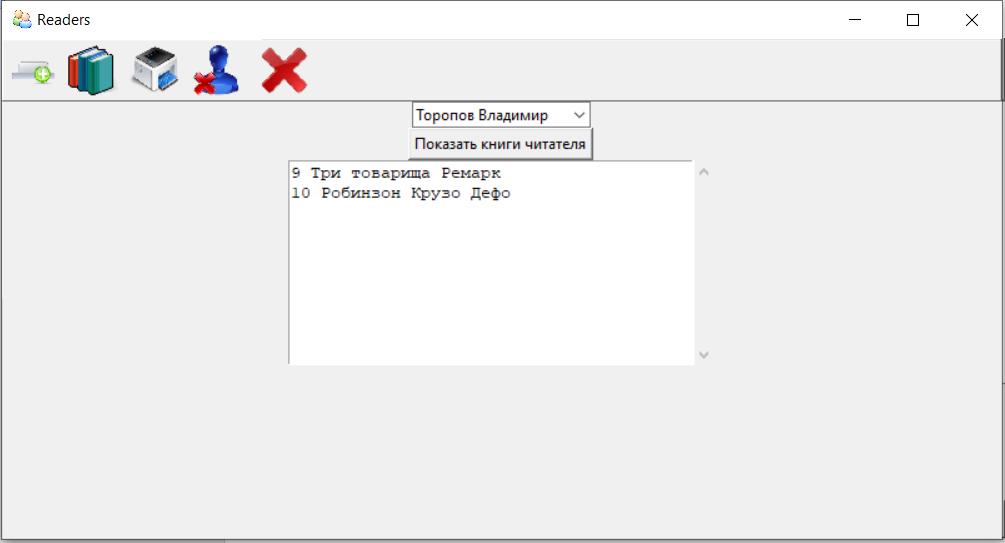


Рис. 3.5 Просмотр всех существующих водителей

*3.5.2.5 Добавление читателя*

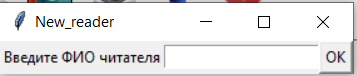
При нажатии кнопки Добавить читателя откроется диалоговое окно New\_reader, которое потребует ввести нужную информацию (Рис 3.6)

Рис. 3.6 Добавление читателя

При нажатии кнопки «ОК» читатель будет добавлен в БД

*3.5.2.6 Добавление книги читателю*

Для того чтобы добавить книгу читателю необходимо нажать на кнопку Добавить книгу читателю, после чего откроется окно Добавления книги читателю(рис 3.7.)

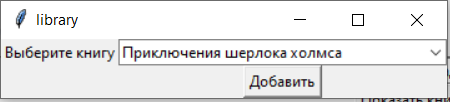


Рис. 3.7 Добавление книги читателю

При выборе книги и нажатии кнопки «Добавить», книга будет добавлена читателю

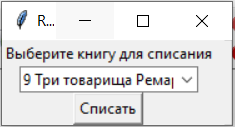
Аналогично выполняется списание книги у читателя (рис 3.8)

Рис. 3.8 Списание книги у читателя

# Заключение

В результате проделанной работы разработан ПК «Библиотека», предназначенный для администрирования информации и автоматизации деятельности ОЛ, ответственного за контроль Библиотеки

В процессе проектирования созданы описание вариантов использования ПК, прототип интерфейса пользователя, объектная модель ПК, диаграмма классов, описание поведения ПК, диаграмма действия ПК.

Курсовой проект удовлетворяет поставленным требованиям.

# Исходный текст:

Класс Library:

class Library(Frame):  
 *"""Library class"""* def \_\_init\_\_(self, parent):  
 Frame.\_\_init\_\_(self, parent)  
 self.parent = parent  
 self.initUI()  
  
 def initUI(self):  
  
 toolbar = Frame(self.parent, bd=1, relief=RAISED)  
 toolbar\_bot = Frame(self.parent, bd=1, relief=RAISED)  
  
 self.New\_img = ImageTk.PhotoImage(Image.open("../resources/new\_ico.png").resize((40, 40)))  
 newButton = Button(toolbar, image=self.New\_img, relief=FLAT, command=self.new\_func)  
 newButton.pack(side=LEFT, padx=1, pady=1)  
  
 self.Save\_img = ImageTk.PhotoImage(Image.open("../resources/save\_ico.png").resize((40, 40)))  
 saveButton = Button(toolbar, image=self.Save\_img, relief=FLAT, command=self.save\_func)  
 saveButton.pack(side=LEFT, padx=2, pady=1)  
  
 self.Readers\_table\_img = ImageTk.PhotoImage(Image.open("../resources/readers\_table\_ico.ico").resize((40, 40)))  
 readertableButton = Button(toolbar, image=self.Readers\_table\_img, relief=FLAT, command=self.read\_readers)  
 readertableButton.pack(side=LEFT, padx=3, pady=1)  
  
 self.Print\_img = ImageTk.PhotoImage(Image.open("../resources/print\_ico.png").resize((40, 40)))  
 printButton = Button(toolbar, image=self.Print\_img, relief=FLAT, command=self.print\_func)  
 printButton.pack(side=LEFT, padx=4, pady=1)  
  
 self.Delete\_img = ImageTk.PhotoImage(Image.open("../resources/delete\_book\_ico.ico").resize((40, 40)))  
 deleteButton = Button(toolbar, image=self.Delete\_img, relief=FLAT, command=self.delete\_book\_func)  
 deleteButton.pack(side=LEFT, padx=5, pady=1)  
  
 self.Info\_img = ImageTk.PhotoImage(Image.open("../resources/info.ico").resize((40, 40)))  
 infoButton = Button(toolbar, image=self.Info\_img, relief=FLAT, command=self.get\_info)  
 infoButton.pack(side=LEFT, padx=6, pady=1)  
  
 self.Exit\_img = ImageTk.PhotoImage(Image.open("../resources/exit\_ico.png").resize((40, 40)))  
 exitButton = Button(toolbar, image=self.Exit\_img, relief=FLAT, command=self.exit)  
 exitButton.pack(side=LEFT, padx=7, pady=1)  
  
 self.searchLabel = Entry(toolbar\_bot, width=20)  
 self.searchLabel.pack(side=LEFT, padx=5, pady=1)  
  
 searchButton = Button(toolbar\_bot, text="Search", command=self.search\_Clicked)  
 searchButton.pack(side=LEFT, padx=9, pady=1)  
  
 toolbar.pack(side=TOP, fill=X)  
 toolbar\_bot.pack(side=BOTTOM, fill=X)  
  
 self.books\_values = []  
 self.table\_read()  
  
 autorCombo = Combobox(toolbar\_bot)  
 autorCombo['values'] = tuple(self.books\_values)  
 autorCombo.pack(side=LEFT, padx=1, pady=1)  
  
 self.pack()  
  
 def table\_read(self):  
 *"""books info reading function"""* self.table = Treeview(self)  
  
 rb = xlrd.open\_workbook('../resources/books.xls', formatting\_info=False)  
 sheet = rb.sheet\_by\_index(0)  
  
 self.table.grid(columnspan=sheet.ncols)  
  
 self.table["columns"] = sheet.row\_values(0)  
 self.table["show"] = "headings"  
 for cols in range(sheet.ncols):  
 self.table.heading(sheet.row\_values(0)[cols], text=sheet.row\_values(0)[cols])  
  
 index = iid = 0  
 for i in range(1, sheet.nrows):  
 self.table.insert('', index, iid, values=sheet.row\_values(i))  
 self.books\_values.append(sheet.row\_values(i)[2])  
 index = iid = index + 1  
  
 def exit(self):  
 *"""exit function"""* self.quit()  
  
 def get\_info(self):  
 def close\_window():  
 info\_window.destroy()  
  
 info\_file = open("../resources/report.txt", 'r')  
 txt = info\_file.read()  
 count\_readers = txt[0]  
 count\_new\_readers = txt[2]  
 count\_books = txt[4]  
 info\_file.close()  
  
 info\_window = Toplevel(self)  
 info\_window.title("Info")  
 count\_readers\_lbl = Label(info\_window, text="Количество читателей: ")  
 count\_readers\_lbl.grid(column=0, row=0)  
 count\_readers\_lbl\_c = Label(info\_window, text=count\_readers)  
 count\_readers\_lbl\_c.grid(column=2, row=0)  
 count\_new\_readers\_lbl = Label(info\_window, text="Количество новых читателей: ")  
 count\_new\_readers\_lbl.grid(column=0, row=1)  
 count\_new\_readers\_lbl\_c = Label(info\_window, text=count\_new\_readers)  
 count\_new\_readers\_lbl\_c.grid(column=2, row=1)  
 count\_books\_lbl = Label(info\_window, text="Количество взятых книг: ")  
 count\_books\_lbl.grid(column=0, row=2)  
 count\_books\_lbl\_c = Label(info\_window, text=count\_books)  
 count\_books\_lbl\_c.grid(column=2, row=2)  
 button = Button(info\_window, text="OK", command=close\_window)  
 button.grid(column=0, row=4)  
  
 def change\_info(self, argument, count):  
 info\_file = open("../resources/report.txt", 'r')  
 txt = info\_file.read()  
 info\_mas = []  
 info\_mas.append(txt[0]) #count\_readers  
 info\_mas.append(txt[2]) #count\_new\_readers  
 info\_mas.append(txt[4]) #count\_new\_books  
 info\_mas[argument] = int(info\_mas[argument]) + count  
 info\_file.close()  
 info\_file = open("../resources/report.txt", 'w')  
 info\_file.write(info\_mas[0] + '\n' + info\_mas[1] + '\n' + info\_mas[2] + '\0')  
 info\_file.close()  
  
  
 def new\_func(self):  
 *"""Adding of reader function"""* def add\_reader():  
 n = self.table["height"]  
 new\_book = []  
 new\_book.append(n + 1)  
 new\_book.append(name\_txt.get())  
 new\_book.append(autor\_book\_field.get())  
 new\_book.append('есть')  
 self.table["height"] = n + 1  
 self.table.insert('', n + 1, n + 1, values=new\_book)  
 res = messagebox.showinfo('Добавление книги', 'Успешно')  
 new\_book\_window.destroy()  
  
 new\_book\_window = Toplevel(self)  
 new\_book\_window.title("New\_book")  
 name\_book\_lbl = Label(new\_book\_window, text="Введите название книги")  
 name\_book\_lbl.grid(column=0, row=0)  
 name\_txt = Entry(new\_book\_window, width=20)  
 name\_txt.grid(column=1, row=0)  
 autor\_book\_lbl = Label(new\_book\_window, text="Введите автора")  
 autor\_book\_lbl.grid(column=0, row=1)  
 autor\_book\_field = Entry(new\_book\_window, width=20)  
 autor\_book\_field.grid(column=1, row=1)  
 button = Button(new\_book\_window, text="OK", command=add\_reader)  
 button.grid(column=0, row=2)  
  
 def print\_func(self):  
 *"""printing function"""* unittest.main()  
  
 def save\_func(self):  
 *"""saving data function"""* txt = ""  
 rw = xlwt.Workbook('../resources/books\_write.xls')  
 sheet = rw.add\_sheet('Sheet1', cell\_overwrite\_ok=True)  
 for i in range(4):  
 sheet.write(0, i, self.table.heading(i)["text"])  
 for i in range(self.table["height"]):  
 value = ""  
 for j in range(4):  
 #print(self.table.item(i)["values"][j])  
 value = value + self.table.item(i)["values"][j] + ' '  
 sheet.write(i + 1, j, self.table.item(i)["values"][j])  
 txt = txt + value + '\n'  
 rw.save('../resources/books\_write.xls')  
  
 txt = "some string for pdf"  
 pdf = FPDF()  
 pdf.add\_page()  
 pdf.set\_font('Arial', size=12)  
 pdf.cell(200, 10, txt=txt)  
 pdf.output('../resources/report.pdf')  
  
 html\_template = """  
 <html>  
 <head>  
 <title>Отчет номер: {current\_date}</title>  
 </head>  
 <body>  
 <table border="1">  
 <tr>  
 <td>Имя</td>  
 <td>Результат</td>  
 </tr>  
 <tr>  
 <td>{arg\_w\_name}</td>  
 <td>{arg\_w\_res}</td>  
 </tr>  
 </table>  
 </body>  
 </html>  
 """  
 html\_file = open("../resources/report\_html.html", 'w')  
 html\_file.write(html\_template.format(current\_date="2015-03-31", arg\_w\_name="Vladimir Toropov", arg\_w\_res="report"))  
 mb = messagebox.showinfo('Success', 'Успешно сохранено')  
  
  
 def delete\_book\_func(self):  
 *"""removal book function"""* a = self.table.selection()  
 res = messagebox.askquestion('Deleting', 'Вы хотите удалить книгу?')  
 if res:  
 self.table.delete(a)  
 self.table['height'] = self.table['height'] - 1  
  
 def search\_Clicked(self):  
 *"""search function"""* for i in range(self.table["height"]):  
 self.table.selection\_remove(i)  
 search = self.searchLabel.get()  
 for i in range(self.table["height"]):  
 if search in self.table.item(i)["values"][1]:  
 self.table.selection\_add(i)  
  
 def read\_readers(self):  
 readers\_window = Toplevel(self)  
 readers\_window.title("Readers")  
 readers\_window.iconbitmap("../resources/readers\_table\_ico.ico")  
 readers\_window.geometry("800x400")  
  
 readers\_app = Readers(readers\_window, self.table)  
  
 self.update()

Класс Unittestclass:

class Unittestclass(unittest.TestCase):  
 def test\_def(self):  
 self.assertEqual(5 + 5, 10)

Класс Readers:

class Readers(Frame):  
 *"""Readers class"""* def \_\_init\_\_(self, parent, table):  
 Frame.\_\_init\_\_(self, parent)  
 self.parent = parent  
 self.table = table  
 self.initUI()  
  
 def initUI(self):  
 toolbar = Frame(self.parent, bd=1, relief=RAISED)  
 toolbar\_bot\_readers = Frame(self.parent, bd=1, relief=RAISED)  
  
 self.New\_img = ImageTk.PhotoImage(Image.open('../resources/new\_ico.png').resize((40, 40)))  
 newButton = Button(toolbar, image=self.New\_img, relief=FLAT, command=self.new\_reader\_func)  
 newButton.pack(side=LEFT, padx=1, pady=1)  
  
 self.New\_book\_img = ImageTk.PhotoImage(Image.open('../resources/library\_ico.png').resize((40, 40)))  
 newbookButton = Button(toolbar, image=self.New\_book\_img, relief=FLAT, command=self.add\_book\_reader)  
 newbookButton.pack(side=LEFT, padx=1, pady=1)  
  
 self.Print\_img = ImageTk.PhotoImage(Image.open("../resources/print\_ico.png").resize((40, 40)))  
 printButton = Button(toolbar, image=self.Print\_img, relief=FLAT, command=self.print\_reader\_func)  
 printButton.pack(side=LEFT, padx=2, pady=1)  
  
 self.Delete\_img = ImageTk.PhotoImage(Image.open("../resources/delete\_reader\_ico.ico").resize((40, 40)))  
 deleteButton = Button(toolbar, image=self.Delete\_img, relief=FLAT, command=self.delete\_reader\_func)  
 deleteButton.pack(side=LEFT, padx=3, pady=1)  
  
 self.Delete\_book\_img = ImageTk.PhotoImage(Image.open("../resources/delete\_book\_ico.ico").resize((40, 40)))  
 deletebookButton = Button(toolbar, image=self.Delete\_book\_img, relief=FLAT, command=self.delete\_book\_reader\_func)  
 deletebookButton.pack(side=LEFT, padx=4, pady=1)  
  
 toolbar.pack(side=TOP, fill=X)  
  
 t1 = threading.Event()  
 t2 = threading.Event()  
  
 self.first\_thread = threading.Thread(target=self.threads\_word, args=(t1, t2))  
 self.second\_thread = threading.Thread(target=self.threads\_word, args=(t2, t1))  
  
 self.first\_thread.start()  
 self.second\_thread.start()  
  
 t1.set()  
  
 self.first\_thread.join()  
 self.second\_thread.join()  
  
 self.list\_readers()  
  
 self.pack()  
  
 def change\_info(self, argument, count):  
 info\_file = open("../resources/report.txt", 'r')  
 txt = info\_file.read()  
 info\_mas = []  
 info\_mas.append(txt[0]) #count\_readers  
 info\_mas.append(txt[2]) #count\_new\_readers  
 info\_mas.append(txt[4]) #count\_new\_books  
 info\_mas[argument] = str(int(info\_mas[argument]) + count)  
 info\_file.close()  
 info\_file = open("../resources/report.txt", 'w')  
 info\_file.write(info\_mas[0] + '\n' + info\_mas[1] + '\n' + info\_mas[2] + '\0')  
 info\_file.close()  
  
 def threads\_word(self, wait\_thread, set\_thread):  
 wait\_thread.wait()  
 wait\_thread.clear()  
 txt\_file = open('../resources/threads.txt', 'a')  
 txt\_file.write("Поток")  
 txt\_file.close()  
 set\_thread.set()  
  
 def add\_book\_reader(self):  
 def write\_book\_reader():  
 current\_name = self.readers.get()  
 current\_reader = self.root[0]  
 for i in self.root:  
 if i.attrib['name'] == current\_name:  
 current\_reader = i  
 break  
  
 current\_book = books.get()  
 current\_book\_table = self.table.item(0)  
 for i in range(self.table['height']):  
 if current\_book == self.table.item(i)['values'][1]:  
 self.table.set(i, column=3, value='нет')  
 current\_book\_table = self.table.item(i)  
  
 # for i in range(self.table['height']):  
 # print(self.table.item(i))  
  
 book = ET.SubElement(current\_reader, 'book')  
 book.set('code', current\_book\_table['values'][0])  
 book\_title = ET.SubElement(book, 'title')  
 book\_title.text = current\_book\_table['values'][1]  
 book\_autor = ET.SubElement(book, 'autor')  
 book\_autor.text = current\_book\_table['values'][2]  
  
 rw = xlwt.Workbook('../resources/books.xls')  
 sheet = rw.add\_sheet('Sheet1', cell\_overwrite\_ok=True)  
 for i in range(4):  
 sheet.write(0, i, self.table.heading(i)["text"])  
 for i in range(self.table["height"]):  
 for j in range(4):  
 sheet.write(i + 1, j, self.table.item(i)["values"][j])  
 rw.save('../resources/books.xls')  
  
 new\_xml = ET.tostring(self.root, 'UTF-8')  
 xml\_file = open("../resources/readers.xml", 'wb')  
 xml\_file.write(new\_xml)  
 xml\_file.close()  
  
 success\_window = Toplevel(nb)  
 success\_window.title("Success")  
 lbl\_i = Label(success\_window, text="Книга успешно добавлена")  
 lbl\_i.grid(column=0, row=0)  
 button\_i = Button(success\_window, text="OK", command=close\_nb\_window)  
 button\_i.grid(column=0, row=3)  
  
 self.change\_info(2, 1)  
 self.list\_readers()  
  
 def close\_nb\_window():  
 nb.destroy()  
  
 nb = Toplevel(self)  
 books = Combobox(nb, state='readonly')  
 elements = []  
 for i in range(self.table["height"]):  
 if self.table.item(i)['values'][3] == "есть":  
 current = self.table.item(i)['values']  
 elements.append(current[1])  
 books["values"] = tuple(elements)  
 books.current(0)  
 books['width'] = 40  
 lbl = Label(nb, text="Выберите книгу")  
 lbl.grid(column=0, row=0)  
 books.grid(column=1, row=0)  
 button = Button(nb, text="Добавить", command=write\_book\_reader)  
 button.grid(column=1, row=1)  
  
 def list\_readers(self):  
 try:  
 logging.basicConfig(filename='test\_debug.log', level=logging.DEBUG)  
 """reading readers function"""  
 tree = ET.parse("../resources/readers.xml")  
 self.root = tree.getroot()  
 self.readers = Combobox(self, state='readonly')  
 elements = []  
 logging.debug("Файл открыт, combobox создан")  
  
 for reader in self.root:  
 elements.append(reader.attrib['name'])  
  
 logging.debug("Прочитаны элементы списка читатели")  
  
 self.readers['values'] = tuple(elements)  
 self.readers.current(0)  
 self.readers.grid(column=0, row=0)  
  
 button = Button(self, text="Показать книги читателя", command=self.books\_of\_reader)  
 button.grid(column=0, row=1)  
  
 self.books = scrolledtext.ScrolledText(self, width=40, height=10)  
 self.books.grid(column=0, row=5)  
  
 logging.debug("Читатели заполнены")  
  
 self.books\_of\_reader()  
 except:  
 mb = messagebox.showerror("ERROR", "ошибка при удалении")  
  
 def books\_of\_reader(self):  
 logging.basicConfig(filename='../resources/test\_warning.log', level=logging.WARNING)  
 self.books.delete(1.0, END)  
 text = ""  
 current = self.root[0]  
 current\_name = self.readers.get()  
  
 logging.warning("начало списка мб пустым")  
  
 for i in self.root:  
 if i.attrib['name'] == current\_name:  
 current = i  
 break  
 for book in current:  
 text = text + book.attrib['code'] + ' ' + book[0].text + ' ' + book[1].text + '\n'  
 self.books.insert(INSERT, text)  
  
 logging.warning("список может быть пустым")  
  
 def new\_reader\_func(self):  
 *"""Adding a reader function"""* def clicked():  
 name = fio\_txt.get()  
 reader = ET.SubElement(self.root, 'reader')  
 reader.set('name', name)  
  
 new\_xml = ET.tostring(self.root, 'UTF-8')  
 xml\_file = open("../resources/readers.xml", 'wb')  
 xml\_file.write(new\_xml)  
 xml\_file.close()  
  
 success\_window = Toplevel(new\_reader\_window)  
 success\_window.title("Success")  
 lbl = Label(success\_window, text="Читатель успешно добавлен")  
 lbl.grid(column=0, row=0)  
 button = Button(success\_window, text="OK", command=close\_reader\_window)  
 button.grid(column=0, row=3)  
  
 self.change\_info(0, 1)  
 self.change\_info(1, 1)  
 self.list\_readers()  
  
 def close\_reader\_window():  
 new\_reader\_window.destroy()  
  
 new\_reader\_window = Toplevel(self)  
 new\_reader\_window.title("New\_reader")  
 name\_reader\_lbl = Label(new\_reader\_window, text="Введите ФИО читателя")  
 name\_reader\_lbl.grid(column=0, row=0)  
 fio\_txt = Entry(new\_reader\_window, width=20)  
 fio\_txt.grid(column=1, row=0)  
 ok\_button = Button(new\_reader\_window, text='OK', command=clicked)  
 ok\_button.grid(column=2, row=0)  
  
  
 def print\_reader\_func(self):  
 print('print readers')  
  
 def delete\_reader\_func(self):  
 *"""Removal reader function"""* try:  
 current\_name = self.readers.get()  
 current = self.root[0]  
 for i in self.root:  
 if i.attrib['name'] == current\_name:  
 current = i  
 break  
 self.root.remove(current)  
 self.change\_info(0, -1)  
 new\_xml = ET.tostring(self.root, 'UTF-8')  
 xml\_file = open("../resources/readers.xml", 'wb')  
 xml\_file.write(new\_xml)  
 xml\_file.close()  
 except:  
 mb = messagebox.showerror("ERROR", "ошибка при удалении")  
  
 self.list\_readers()  
  
 def delete\_book\_reader\_func(self):  
 *"""Removal book of reader function"""* try:  
 def delete\_book():  
 current\_book\_name = books.get()  
 current\_book = current[0]  
 for i in current:  
 if i[0].text in current\_book\_name:  
 current\_book = i  
 break  
 current.remove(current\_book)  
  
 for i in range(self.table['height']):  
 if self.table.item(i)['values'][1] in current\_book\_name:  
 self.table.set(i, column=3, value='есть')  
  
 rw = xlwt.Workbook('../resources/books.xls')  
 sheet = rw.add\_sheet('Sheet1', cell\_overwrite\_ok=True)  
 for i in range(4):  
 sheet.write(0, i, self.table.heading(i)["text"])  
 for i in range(self.table["height"]):  
 for j in range(4):  
 sheet.write(i + 1, j, self.table.item(i)["values"][j])  
 rw.save('../resources/books.xls')  
  
 new\_xml = ET.tostring(self.root, 'UTF-8')  
 xml\_file = open("../resources/readers.xml", 'wb')  
 xml\_file.write(new\_xml)  
 xml\_file.close()  
  
 success\_window = Toplevel(books\_reader\_window)  
 success\_window.title("Success")  
 lbl = Label(success\_window, text="Книга успешно списана")  
 lbl.grid(column=0, row=0)  
 button = Button(success\_window, text="OK", command=close\_reader\_window)  
 button.grid(column=0, row=3)  
  
 self.list\_readers()  
  
 def close\_reader\_window():  
 books\_reader\_window.destroy()  
  
 books\_reader\_window = Toplevel(self)  
 books\_reader\_window.title('Reader\'s books')  
  
 books = Combobox(books\_reader\_window, state='readonly')  
 current = self.root[0]  
 current\_name = self.readers.get()  
 for i in self.root:  
 if i.attrib['name'] == current\_name:  
 current = i  
 break  
 elements = []  
 for book in current:  
 elements.append(book.attrib['code'] + ' ' + book[0].text + ' ' + book[1].text + '\n')  
  
 lbl = Label(books\_reader\_window, text="Выберите книгу для списания")  
 lbl.grid(column=0, row=0)  
  
 books['values'] = tuple(elements)  
 books.current(0)  
 books.grid(column=0, row=2)  
  
 DeleteButton = Button(books\_reader\_window, text="Списать", command=delete\_book)  
 DeleteButton.grid(column=0, row=5)  
 except:  
 mb = messagebox.showerror("ERROR", "Ошибка при удалении")

Main:

def main():  
 *"""Main function"""* window = Tk()  
 window.title("library")  
 window.iconbitmap('../resources/library\_ico.ico')  
 window.geometry("800x400")  
 app = Library(window)  
 window.mainloop()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()