ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки бакалавров по направлению   
38.03.05 Бизнес-информатика**

*Кононова Екатерина Сергеевна, 18БИ-1*

Отчет по практической части дисциплины «Моделирование процессов и систем» на тему: «Объектно-ориентированный анализ и проектирование программного обеспечения»

|  |  |
| --- | --- |
|  | Проверила:  преподаватель кафедры ИСиТ  Я.П. Куранова |

Нижний Новгород, 2020

Оглавление

[Введение 3](#_Toc39873329)

[Глоссарий проекта. 4](#_Toc39873330)

[Раздел 1. Варианты использования. 6](#_Toc39873331)

[1.1. Диаграмма вариантов использования системы. 6](#_Toc39873332)

[Раздел 2. Проектирование системы. 7](#_Toc39873333)

[2.1. Иерархия классов. 7](#_Toc39873334)

[2.2. Диаграмма классов системы. 7](#_Toc39873335)

[2.1.1. Описание классов 8](#_Toc39873336)

[2.1.2. Диаграмма пакетов 12](#_Toc39873337)

[2.2. Диаграммы взаимодействия 13](#_Toc39873338)

[2.2.1. Диаграммы последовательности. 13](#_Toc39873339)

[2.2.2. Диаграммы кооперации. 16](#_Toc39873340)

[2.3. Диаграммы состояний 19](#_Toc39873341)

[2.4. Диаграммы активностей 21](#_Toc39873342)

[Заключение 24](#_Toc39873343)

# Введение

**Вариант 13. Генеалогическое дерево.**

Тема:Объектно-ориентированный анализ и проектирование программного обеспечения. Программная система поддержки генеалогических деревьев.

**Программная система поддержки генеалогических деревьев.**

Система хранит сведения о персонах (ф. и. о., пол, дата рождения, дата смерти, биография) и о родственных связях между ними. Связи бывают только трех видов: "мужья-жены", "дети-родители" и "братья-сестры". Система обеспечивает возможность добавления данных о новых персонах и родственных связях, изменение введенных данных и удаление ненужных данных. Система следит за непротиворечивостью вводимых данных. Недопустимо, чтобы человек был собственным предком или потомком.

Пользователи системы могут осуществлять поиск полезной информации по дереву:

· находить для указанного члена семьи его детей;

· находить для указанного члена семьи его родителей;

· находить для указанной персоны братьев и сестер, если таковые есть;

· получать список всех предков персоны;

· получать список всех потомков персоны;

· получать список всех родственников персоны;

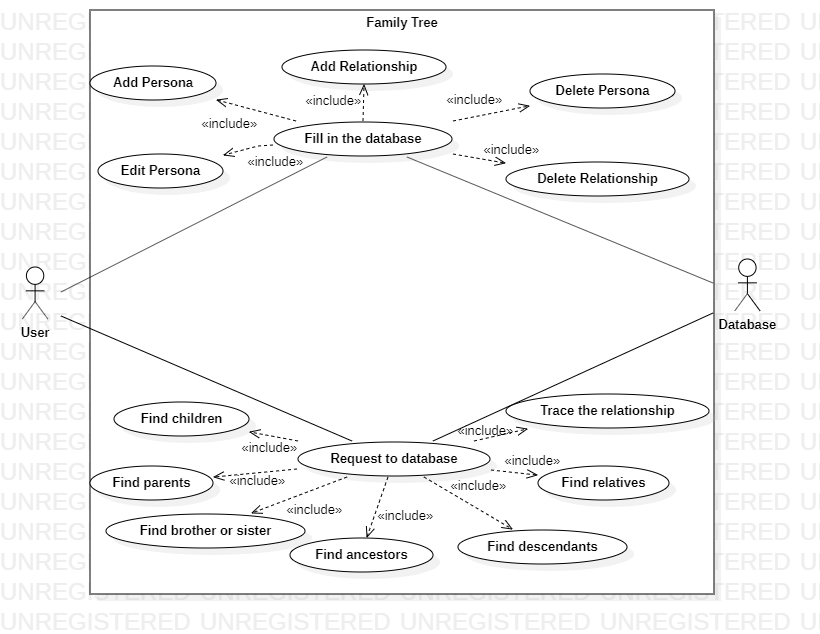
· прослеживать цепочку родственных связей от одной персоны до другой, если таковые есть (например, если Петр является шурином Ивана, то на запрос о родственных связях между Петром и Иваном выдается такой результат: «Петр – брат Ольги, Ольга – жена Ивана»).

# Глоссарий проекта.

|  |  |
| --- | --- |
| ***Наименование actor/use-case*** | ***Описание*** |
| User (actor) | Пользователь, работающий с системой. Этот актер инициирует все прецеденты (Рисунок 1). |
| Database (actor) | файл в памяти компьютера, хранящий таблицы данных. Взаимодействие с этим актером происходит при чтении и сохранении базы данных (Рисунок 1). |
| Fill in the database (use case) | **ВИ:** Включает в себя процедуры добавления, редактирования и удаления записей из базы (Рисунок ***1***).  **Предусловия:**  Система находится в состоянии «Ожидание действий от пользователя».  **Основные потоки событий:**  Пользователь выбирает, какое действие выполнить.  1. Работа с персонами.   * Добавить персону.   Система запрашивает данные о персоне. Пользователь вводит данные о персоне. Система проверяет, введено ли имя. Если имя введено, новая запись добавляется в базу.   * Изменить персону.   Пользователь выбирает персону из списка. Система запрашивает новые данные о персоне. Пользователь вводит данные о персоне. Система проверяет, введено ли имя. Если имя введено, изменения вносятся в базу.   * Удалить персону.   Пользователь выбирает персону из списка. Система проверяет, состоит ли персона в связях. Если не состоит, запись о персоне удаляется из базы.  2. Работа со связями.   * Добавить связь.   Система проверяет, имеется ли хотя бы 2 записи в таблице персон. Если имеется, пользователь выбирает участников и тип связи. Система проверяет, возможна ли связь между выбранными персонами. Если возможна, в базу добавляется новая запись.   * Удалить связь.   Пользователь выбирает нужную связь. Из базы удаляется запись о связи.  **Альтернативные потоки событий:**  При удалении персоны выясняется, что удаляемая персона состоит в связях.   1. Выводится сообщение об ошибке, и прецедент завершается.   При добавлении связи выясняется, что таблица персон содержит менее двух записей.   1. Выводится сообщение об ошибке и прецедент завершается.   **Постусловия:**  Система возвращается в состояние «Ожидание действий от пользователя». |
| Request to the database | **ВИ:** Включает в себя процедуры обработки пользовательских запросов и вывода найденной информации (Рисунок1).  **Предусловия:**  Система находится в состоянии «Ожидание действий от пользователя».  **Основные потоки событий:**  Пользователь выбирает, какой запрос выполнить.   * Исходные данные – это одна персона (найти детей, родителей и т.д.).   Пользователь выбирает нужный запрос в контекстном меню для нужной персоны.   * Исходные данные - 2 персоны (отследить цепочку связей).   Пользователь выбирает нужных персон из списка.  Система отображает результат запроса.  **Постусловия:**  Система возвращается в состояние «Ожидание действий от пользователя». |

# Раздел 1. Варианты использования.

## 1.1. Диаграмма вариантов использования системы.

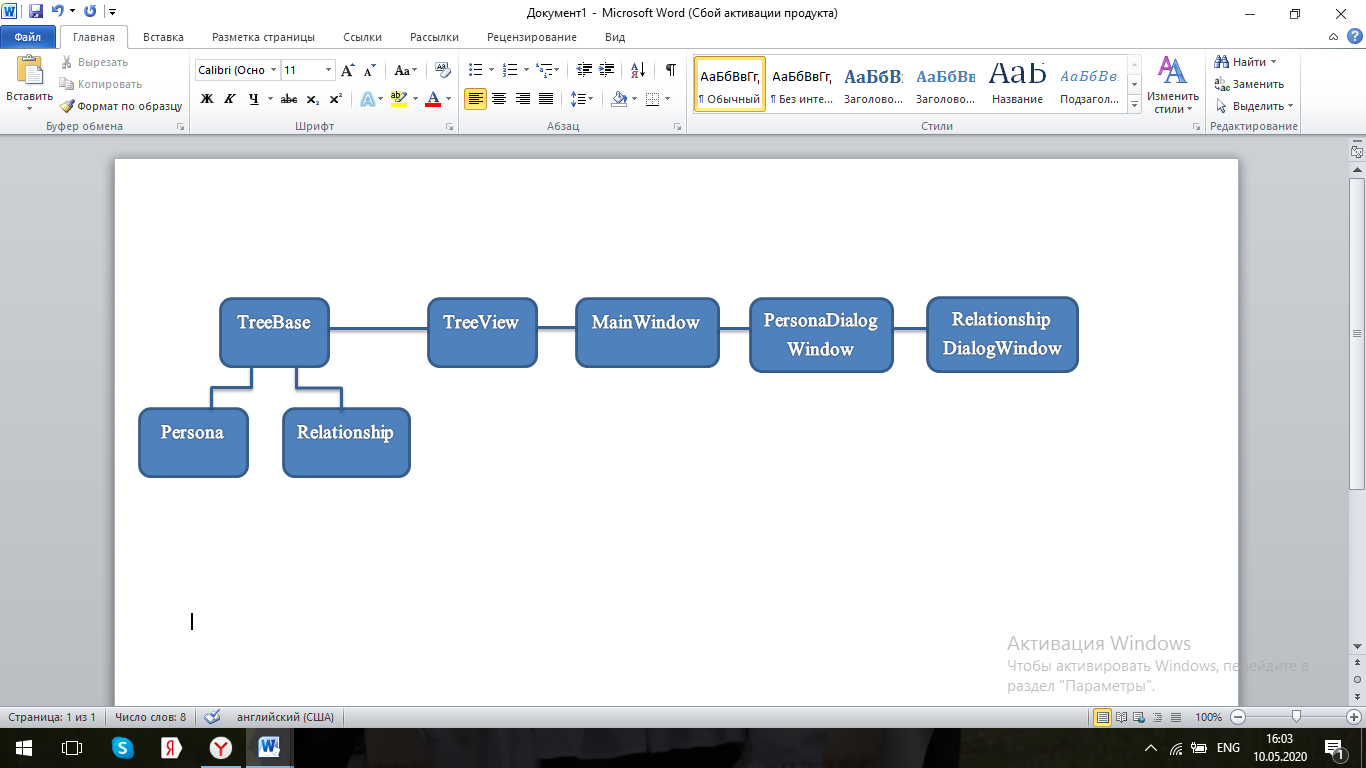


*Рисунок 1. Use case диаграмма системы генеалогических деревьев*

Имеется 2 эктора (Рисунок1) – пользователь и база данных (sql, например). Также 2 основных варианта использования – один отвечает за запросы к базе, обработку информации и вывода необходимых данных, второй вариант использования отвечает за заполнение базы. В «запрос» входят: поиск родственников, поиск родителей, поиск детей, братьев и сестёр, также поиск потомков и предков и прослеживание цепочки связей между конкретными 2 персонами. В «заполнение» включено добавление новой персоны и новой связи, их удаление, а также редактирование персоны (Рисунок 1). Оба вариант использования инициирует пользователь системы.

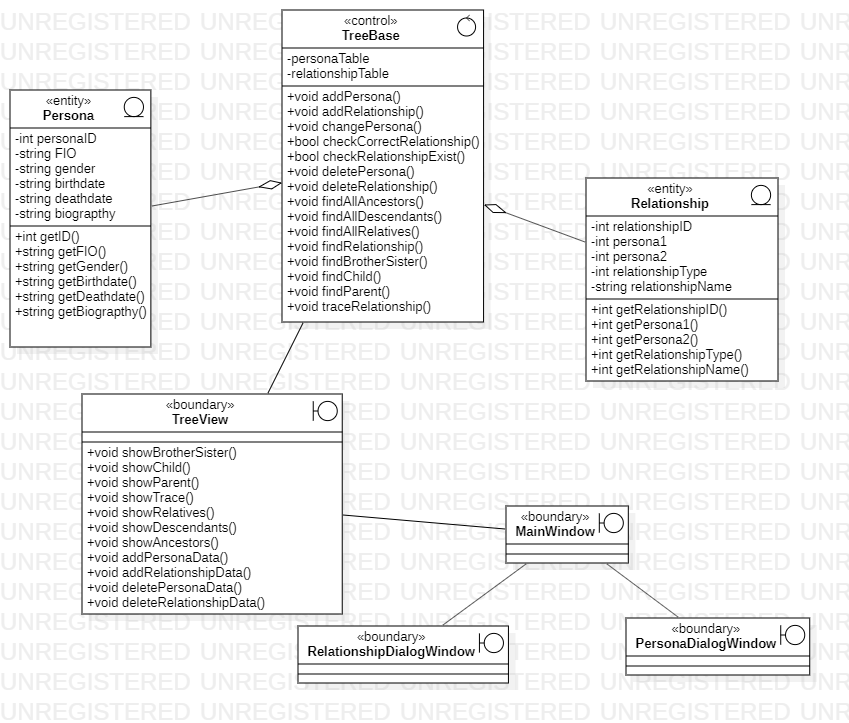
# Раздел 2. Проектирование системы.

## 2.1. Иерархия классов.



*Рисунок 2. Иерархия классов системы*

## 2.2. Диаграмма классов системы.



*Рисунок 3. Диаграмма классов*

На диаграмме представлены классы системы генеалогических деревьев (Рисунок 3). В системе есть два класса сущности(<<entity>>) – это классы Persona и Relationship, в которых хранятся данные о персонах и связях соответственно. Четыре граничных класса(<<boundary>>). Класс TreeView обрабатывает вводы пользователя, а также выводит информацию по запросам. Остальные три класса являются окнами приложения – главное, а также одно диалоговое окно для персоны и второе для связи. Класс TreeBase являетяс контролирующим классом(<<control>>). Именно в нем содержатся функции поиска, с помощью которых можно найти необходимую информацию, а также добавить связь или персону в базу данных.

### 2.1.1. Описание классов

**Persona** – класс, описывающий строку в таблице «Персона» (Рисунок 3). Это класс сущность(<<entity>>), так как он просто хранит данные и даже с выключением системы данные не удаляются. Класс является таблицей базы данных с полями, содержащими данные о человеке в генеалогическом дереве.

***Атрибуты:***

|  |  |  |
| --- | --- | --- |
| ***Имя*** | ***Описание*** | ***Тип*** |
| personaID | ID персоны | int |
| FIO | ФИО | string |
| Gender | пол | string |
| Birthdate | дата рождения | string |
| Deathdate | дата смерти | string |
| Biograpthy | биография | string |

***Функции:***

|  |  |  |
| --- | --- | --- |
| ***Имя*** | ***Описание*** | ***Сигнатура*** |
| getId | Возвращает ID персоны | int |
| getFIO | Возвращает ФИО персоны | string |
| getGender | Возвращает пол персоны | string |
| getBirthdate | Возвращает дату рождения персоны | string |
| getDeathdate | Возвращает дату смерти персоны | string |
| getBio | Возвращает биографию персоны | string |

**Relationship** – класс, описывающий строку в таблице «Связь» (Рисунок 3). Также вляется классом сущностью(<<entity>>), так как тоже хранит данные, но уже не о человеке, а о связи между двумя персонами. Класс является таблицей базы данных с полями, содержащими данные о связи в генеалогическом дереве.

***Атрибуты:***

|  |  |  |
| --- | --- | --- |
| ***Имя*** | ***Описание*** | ***Тип*** |
| relationshipID | ID связи | int |
| persona1 | ID первой персоны | int |
| persona2 | ID второй персоны | int |
| relationshipType | тип связи | int |
| relationshipName | название типа связи | string |

***Функции:***

|  |  |  |
| --- | --- | --- |
| ***Имя*** | ***Описание*** | ***Сигнатура*** |
| getRelationshipID | Возвращает ID связи | int |
| getPersona1 | Возвращает ID 1 персоны | int |
| getPersona2 | Возвращает ID 2 персоны | int |
| getRelationshipType | Возвращает тип связи | int |
| getRelationshipName | Возвращает название связи | string |

**TreeBase** – класс документа (Рисунок 3). Этот класс является управляющим, так как является координатором внутри системы. Именно с помощью этого класса можно получить информацию о необходимом человеке или связи или получить списки детей, родителей, родственников.

***Атрибуты:***

|  |  |  |
| --- | --- | --- |
| ***Имя*** | ***Описание*** | ***Тип*** |
| personaTable | таблица персон | array |
| relationshipTable | таблица связей | array |

***Функции:***

|  |  |  |
| --- | --- | --- |
| ***Имя*** | ***Описание*** | ***Сигнатура*** |
| addPersona | добавить персону в базу | void |
| addRelationship | добавить связь в базу | void |
| changePersona | отредактировать персону | void |
| checkCorrectRelationship | проверить возможность создания связи между указанными персонами | void |
| checkRelationshipExist | проверить, есть ли связи между указанными персонами | void |
| deletePersona | удалить персону из базы | void |
| deleteRelationship | Удалить связь из базы | void |
| findAllAncestors | рекурсивная функция поиска родителей | void |
| findAllDescendants | рекурсивная функция поиска детей | void |
| findAllRelatives | рекурсивная функция поиска родственников | void |
| findRelationship | рекурсивная функция поиска связей | void |
| findBrotherSister | функция поиска братьев и сестер | void |
| findChild | функция поиска детей | void |
| findParent | функция поиска родителей | void |
| traceRelationship | функция прослеживания связи между персонами | void |

**TreeView** – класс, который обеспечивает ввод-вывод данных пользователем (Рисунок 3). С его помощью происходит отображение данных пользователю и ввод данных для персон и связей.

***Функции:***

|  |  |  |
| --- | --- | --- |
| ***Имя*** | ***Описание*** | ***Сигнатура*** |
| addPersonaData | добавить информацию для персоны | void |
| addRelationshipData | добавить информацию для связи | void |
| deletePersonaData | удалить персону | void |
| deleteRelationshipData | Удалить связь | void |
| showAncestors | Показать список предков | void |
| showDescendants | Показать список потомков | void |
| showlRelatives | Показать список родственников | void |
| showBrotherSister | Показать братьев/сестёр | void |
| showChild | Показать информацию о детях | void |
| showParent | Показать информацию о родителях | void |
| showTrace | Отобразить цепочку связей | void |

**RelationshipDialogWindow** – класс диалогового окна для пользователя при вводе данных о новых связях (Рисунок 3).

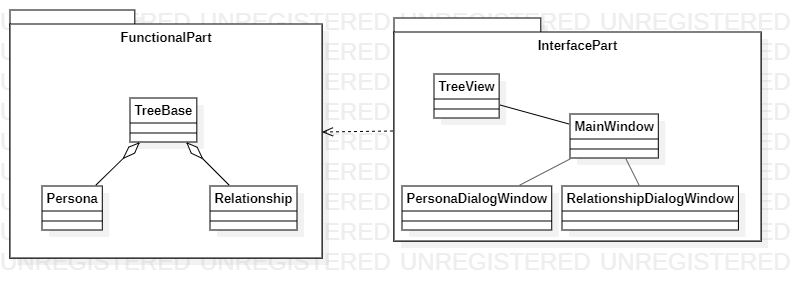
**PersonaDialogWindow -** класс диалогового окна для пользователя при вводе данных о новых персонах и редактировании уже существующих (Рисунок 3).

**MainWindow -** класс главного окна (Рисунок 3).

Эти классы являются граничными(<<boundary>>), так как они обеспечивают взаимодействие с внешними элементами, непосредственно с пользователем.

### 2.1.2. Диаграмма пакетов

Ниже (Рисунок 4) отображены пакеты. Классы были разделены по принципу выполнения определённых функций в системе: функциональной части и интерфейсной части. В пакете FunctionalPart содержатся классы Persona, Relationship, TreeBase. В этом пакете классы реализуют логику работы системы и хранение данных. В пакете InterfacePart содержатся TreeView, PersonaDialogWindow, RelationDialogWindow и MainWindow. Классы того пакета обеспечивают взаимообмен информацией между пользователем и системой, а также ввод и вывод необходимых данных.

****

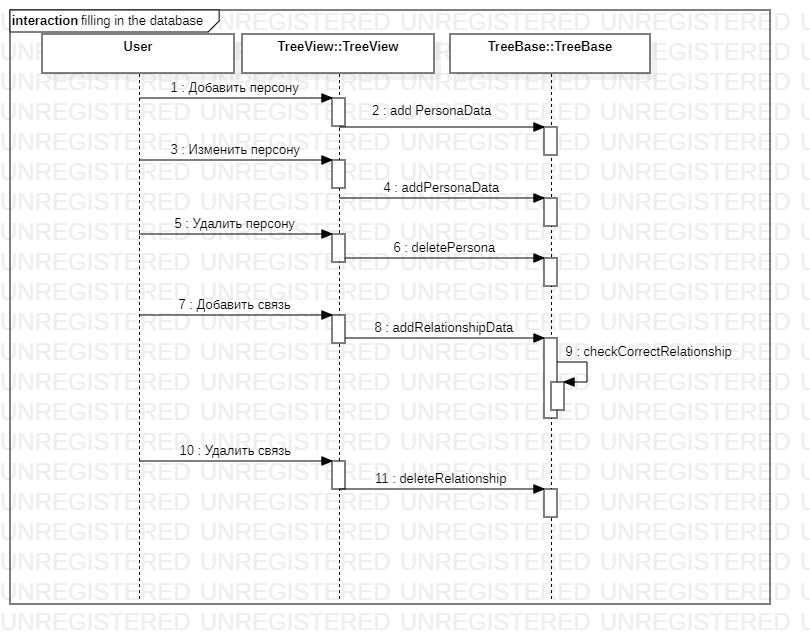
*Рисунок 4. Диаграмма пакетов классов системы*

## 2.2. Диаграммы взаимодействия

### 2.2.1. Диаграммы последовательности.

*Диаграмма последовательности для варианта использования «Fill in the database» на Рисунок 5.*

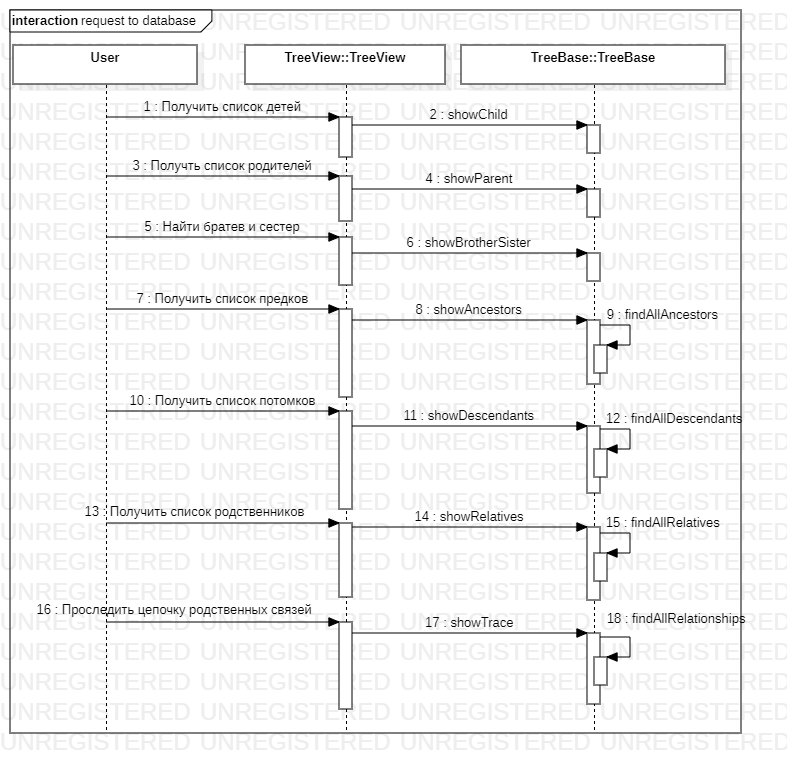
Участвуют пользователь, объект класса ввода-вывода пользовательских действий TreeView и объект класса TreeBase (Рисунок 5). Пользователь отправляет сообщение «добавить персону», объект класса обработчика отправляет addPersonaData. Пользователь отправляет сообщение «изменить персону», объект класса обработчика отправляет addPersonaData. Пользователь отправляет сообщение «удалить персону», объект класса обработчика отправляет deletePersona. Пользователь отправляет сообщение «добавить связь», объект класса обработчика отправляет addRelationshipData, а также отправляется сообщение на проверку корректности связи checkCorrectRelationsip. Пользователь отправляет сообщение «удалить связь», объект класса обработчика отправляет deleteRelationship.



*Рисунок 5. Диаграмма последовательности (fill in the database)*

*Диаграмма последовательности для варианта использования «Request to database» на Рисунок 6.*

Участвуют пользователь, объект класса ввода-вывода пользовательских действий TreeView и объект класса TreeBase (Рисунок 6). Пользователь отправляет сообщение «получить список детей», объект класса обработчика отправляет showChild. Пользователь отправляет сообщение «получить список родителей», объект класса обработчика отправляет showParent. Если пользователь отправляет сообщение «найти братьев и сестёр», тогда отправляется showBrotherSister. После сообщения от пользователя «получить список предков» передается сообщение объектом класса обработчика showAncestors, далее отправляется сообщение showAllAncestors. Если пользователь отправляет сообщение «получить список всех потомков», то сначала отправляется showDecedants, а потом showAllDescendans. Таким же образом отправляются сообщения о поиске родственников (showRelatives) после сообщения пользователя «получить список родственников». Далее отправляется showAllRelatives. Если пользователь отправляет сообщение «проследить цепочку связей», то сначала объектом класса обработчика отправляется showTrace, а затем объектом класс TreeBase findAllRelationship.



*Рисунок 6. Диаграмма последовательности (request to the database)*

### 2.2.2. Диаграммы кооперации.

*Диаграмма кооперации для варианта использования «Fill in the database»* *на Рисунок 7.*

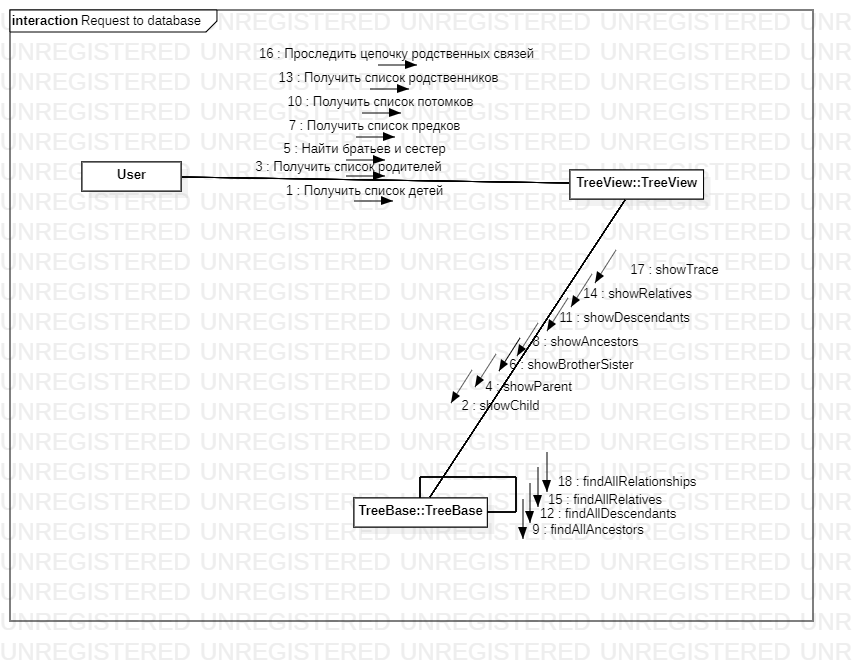
Участвуют пользователь, объект класса ввода-вывода пользовательских действий TreeView и объект класса TreeBase (Рисунок 7). Пользователь отправляет сообщение «добавить персону», объект класса обработчика отправляет addPersonaData. Пользователь отправляет сообщение «изменить персону», объект класса обработчика отправляет addPersonaData. Пользователь отправляет сообщение «удалить персону», объект класса обработчика отправляет deletePersona. Пользователь отправляет сообщение «добавить связь», объект класса обработчика отправляет addRelationshipData, а также отправляется сообщение на проверку корректности связи checkCorrectRelationsip. Пользователь отправляет сообщение «удалить связь», объект класса обработчика отправляет deleteRelationship.

**

*Рисунок 7.Диаграмма кооперации (fill in the database)*

*Диаграмма кооперации для варианта использования «Request to database» на Рисунок 8.*

Участвуют пользователь, объект класса ввода-вывода пользовательских действий TreeView и объект класса TreeBase (Рисунок 8). Пользователь отправляет сообщение «получить список детей», объект класса обработчика отправляет showChild. Пользователь отправляет сообщение «получить список родителей», объект класса обработчика отправляет showParent. Если пользователь отправляет сообщение «найти братьев и сестёр», тогда отправляется showBrotherSister. После сообщения от пользователя «получить список предков» передается сообщение объектом класса обработчика showAncestors, далее отправляется сообщение showAllAncestors. Если пользователь отправляет сообщение «получить список всех потомков», то сначала отправляется showDecedants, а потом showAllDescendans. Таким же образом отправляются сообщения о поиске родственников (showRelatives) после сообщения пользователя «получить список родственников». Далее отправляется showAllRelatives. Если пользователь отправляет сообщение «проследить цепочку связей», то сначала объектом класса обработчика отправляется showTrace, а затем объектом класс TreeBase findAllRelationship.



*Рисунок 8. Диаграмма кооперации (request to database)*

## 2.3. Диаграммы состояний

*Диаграмма состояний для экземпляра класса Persona (Рисунок 9).*

Может находиться в 4 состояниях:

1. Создана
2. Сохранена
3. Удалена
4. Изменена

Когда человек добавляет новую персону, то она создается и находится в состоянии «Persona создана» (Рисунок 9). Данные проверяются и, если противоречий нет, то персона сохраняется и переходит в состояние «Persona сохранена» и на этом все может завершиться. Но если поступает запрос на изменение, то вносятся изменения и она переходит в состояние «Persona изменена», если противоречий нет, то она снова сохраняется и всё завершается (Рисунок 9). В противном случае если противоречия найдены, тогда она удаляется и всё завершается.



*Рисунок 9. Диаграмма состояний экземпляра класса Persona*

*Диаграмма состояний для экземпляра класса Relationship (Рисунок 10).*

Может находиться в 3 состояниях:

1. Добавлена
2. Сохранена
3. Удалена

Когда человек добавляет новую связь, то она создается и находится в состоянии «Relationship добавлена» (Рисунок 10). Данные проверяются и, если противоречий нет, то связь сохраняется и переходит в состояние «Relationship сохранена» и на этом все может завершиться (Рисунок 10). В противном случае, если противоречия найдены, тогда она удаляется и всё завершается.

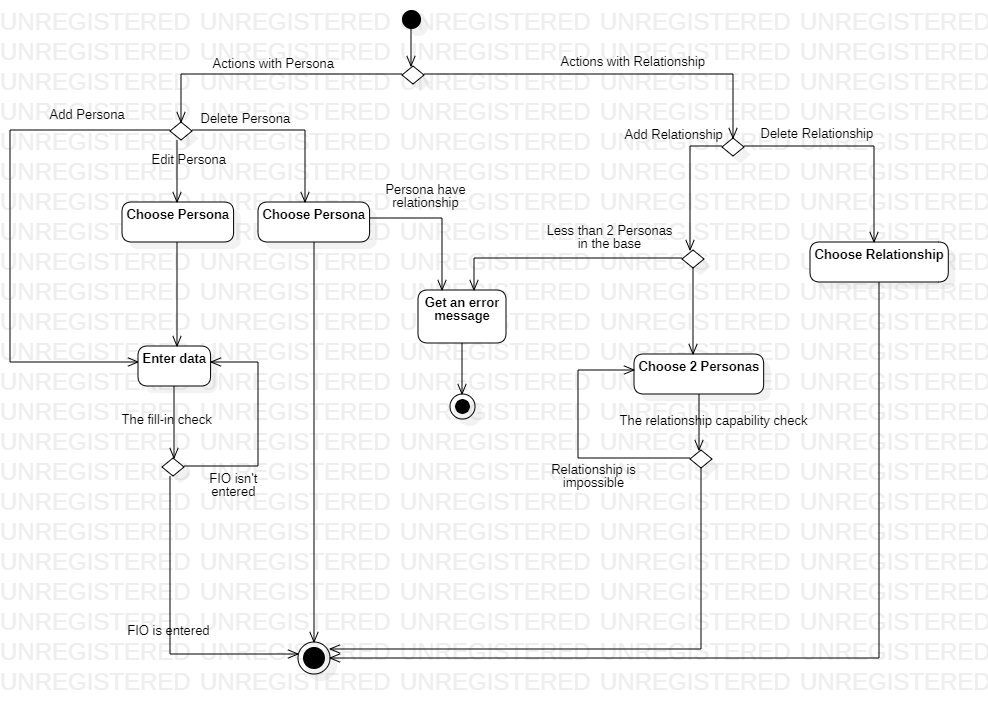


*Рисунок 10. Диаграмма состояний для экземпляра класса Relationship*

## 2.4. Диаграммы активностей

*Диаграмма активности для варианта использования «Fill in the database» (Рисунок 11).*

Пользователь выбирает действия с персоной или действия со связями (Рисунок 11). Если он выбирает действие со связями, то после этого ему нужно выбрать либо удалить связь и тогда связь удаляется, либо добавить связь и происходит проверка на наличие хотя бы 2 персон в базе (Рисунок 11). Если в базе меньше 2 персон, то выводится сообщение об ошибке. Если персон достаточно, тогда предлагается выбрать двух персон, если после выбора есть противоречия, возвращаемся к выбору персон, в противном случае данные изменяются в базе (Рисунок 11). Если выбрано «действия с персоной», тогда пользователь выбирает удалить персону, изменить или добавить. Если выбрано удалить персону, тогда пользователь выбирает нужную и если у нее есть связи, то выводится сообщение об ошибке, если связей нет, то она удаляется, и вносятся изменения в базу. Если выбрано «добавить персону», тогда пользователь вводит данные о персоне. Если ФИО введено, то производятся изменения в базе, если нет, то повторяется введение данных о персоне. Если выбрано «изменить персону», то пользователь выбирает, какую именно персону он хочет изменить (Рисунок 11). Дальше пользователь вводит данные, проверяется, введено ли ФИО, если да, то вносятся изменения. Если нет, то повторно вводятся данные.



*Рисунок 11. Диаграмма активности (fill in the database)*

*Диаграмма активности для варианта использования «Request to data base»(Рисунок 12).*

Пользователь выбирает (Рисунок 12) – исходные данные две персоны (отследить связь) или одна (найти детей, родителей и т.д.). После выбора нескольких, пользователь выбирает персон из списка, происходит поиск в базе и выводится результат. Если пользователь выбрал 1 персону, то нужно выбрать какую именно и какой запрос, происходит поиск в базе и выводится результат (Рисунок 12).



*Рисунок 12. Диаграмма активности (request to database)*

# Заключение

В ходе работы была спроектирована программная система поддержки генеалогических деревьев. Были выявлены основные классы, которые были разделены на функциональную и интерфейсную части. Для детального представления работы системы был сконструирован ряд диаграмм согласно требованиям к системе. Спроектированная система отвечает всем заложенным требованиям. К сожалению, пробная версия не была сделана, но такую систему можно было бы написать на языках программирования java или С++. Возможно, если бы писалась программа, то был бы заложен ряд ограничений, например, на количество записей в базе или ограничение в датах, которые можно было бы добавлять в дерево. Однако по модели можно сказать, что функционал понятен и, если делать выводы именно из модели, а не из возможной реализации, то все требования к системе были учтены, и данная модель соответствует техническому заданию системы.