МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра Вычислительной техники

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ ПО ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

Тема: «Создание программного комплекса средствами объектно-ориентированного программирования»

Студент гр. 3312 Васильев. Т. В.

Преподаватель Павловский М.Г.

Санкт-Петербург

1. Техническое задание

Введение

Курсовая работа посвящена разработке программного комплекса для управления школьными операциями. Основной задачей программы является автоматизация работы администратора школы, включая управление данными о студентах, преподавателях, предметах и классах. Программа предоставляет удобный интерфейс для внесения, изменения и удаления информации, а также для получения аналитической отчетности по преподавателям, студентам и учебным предметам. Целью работы является закрепление и углубление теоретических знаний, приобретение практических навыков по проектированию и разработке программного обеспечения на объектноориентированном языке.

Основания для разработки

Разработка выполняется в рамках учебного задания по курсовой работе. Тема: "Разработка информационной системы для завуча школы".

Назначение разработки

Программный комплекс предназначен для хранения, управления и анализа данных об учениках, учителях, классах и предметах. Эксплуатационное назначение – упрощение работы завуча за счет автоматизации хранения и обработки данных.

При разработке программного кода на языке Java необходимо учитывать следующие обязательные требования

- Использование закрытых и открытых членов классов.
- Применение наследования.
- Создание конструкторов с параметрами.
- Определение абстрактных базовых классов.
- Реализация виртуальных функций.
- Обработка исключительных ситуаций.
- Динамическое создание объектов.

Техническое задание

Задание 2. Разработать ПК для завуча школы. В ПК должны храниться сведения об учениках школы, учителях, предметах, которые они преподают. Завуч может добавлять, изменять и удалять информацию о учителе или ученике. Завучу могут потребоваться следующие сведения:

- какие преподаватели ведут тот или иной предмет;
- в каких классах преподает учитель;
- количество учеников в школе и в указанном классе;
- отчет об успеваемости в школе (количество отличников, двоечников с указанием их фамилий).

Описание Программного Комплекса

Программный комплекс "Информационная система для завуча школы" предназначен для автоматизации учета данных об учениках, учителях, классах и предметах, а также для анализа успеваемости учеников. Он разработан с использованием языка программирования Java и предоставляет графический интерфейс для взаимодействия пользователя с системой.

Функциональные возможности:

- *Работа с данными об учениках*:

Добавление, редактирование и удаление информации (ФИО, класс, успеваемость).

Просмотр списка всех учеников с возможностью фильтрации по классам.

- *Работа с данными об учителях:*

Управление сведениями о преподавателях (ФИО, предметы, классы, в которых они работают)

- *Работа с данными о классах:*

Создание, изменение и удаление классов.

- *Работа с данными о предметах:*

Создание, редактирование и удаление предметов.

- *Генерация отчета:*

Статистика по успеваемости учеников: Количество отличников, двоечников с указанием их фамилий.

Программа имеет интуитивно понятный интерфейс. Основное окно включает:

1. Выпадающий список для выбора таблицы данных (ученики, учителя, классы, предметы).
2. Панель управления с кнопками для добавления, редактирования, удаления записей и генерации отчета.
3. Таблицу для отображения данных.
4. Дополнительные диалоговые окна используются для ввода и редактирования данных.

Требования к функциональным характеристикам

1. Возможность назначать преподавателей для различных классов, а также редактировать и управлять этими связями.
2. Программа позволяет назначать преподавателей к определённым предметам, управлять связями.
3. Редактирование данных о предметах, классах, учениках и преподавателях
4. Добавление и удаление данных о предметах, классах, учениках и преподавателях
5. Фильтрация учеников по классам.
6. Сохранение данных в XML файл.
7. Загрузка данных из XML файла.
8. Создание текстового отчета об успеваемости учеников школы с возможностью задавать параметры успеваемости.

Требования к надежности

- Проверка корректности ввода данных (например, числовые значения успеваемости).
- Устойчивость к ошибкам пользователя (обработка неверных данных с уведомлением).
- Сохранение данных в файл XML для обеспечения их сохранности.

Условия эксплуатации

Программа устанавливается на персональный компьютер с операционной системой Windows (версии 7 и выше). Интернет-соединение не требуется, так как все данные хранятся локально. Обслуживание программы заключается в регулярном резервном копировании данных и, при необходимости, обновлении программы для исправления ошибок или добавления нового функционала. Для работы с программой требуется один пользователь (завуч) с базовыми навыками работы на компьютере.

Требования к составу и параметрам технических средств

Для корректной работы программы необходим персональный компьютер с процессором не менее 1.5 ГГц, оперативной памятью объемом 4 ГБ и минимум 200 МБ свободного дискового пространства. Программа требует установленной Java Runtime Environment (JRE) версии 8 или выше.

Требования к информационной и программной совместимости

Программа использует XML-файлы для хранения данных, что обеспечивает совместимость с различными системами обработки информации. Входные данные включают информацию об учениках (ФИО, класс, успеваемость), учителях (ФИО, предметы, классы), классах (название, идентификатор) и предметах (название, идентификатор). Выходные данные представлены в виде текстовых отчетов о преподавателях, классах, учениках и их успеваемости. Для реализации программы использован язык Java с использованием стандартных библиотек для работы с графическим

интерфейсом (Swing) и файлами.

Требования к информационной и программной совместимости

Стадии и этапы разработки

1. Анализ требований к программному комплексу.
2. Проектирование архитектуры приложения и интерфейса.
3. Реализация функций добавления, редактирования, удаления данных и отчетов.
4. Тестирование работы программы.
5. Подготовка пояснительной записки и программной документации.

2. Проектирование ПК

Описание вариантов использования ПК

Разработка программного комплекса для завуча школы предусматривает автоматизацию ряда ключевых процессов. Эти процессы детализированы с помощью диаграммы вариантов использования (Use Case Diagram), которая отражает взаимодействие пользователя (завуча) с системой.

Актор и основные прецеденты

Актором в данной системе выступает завуч, который взаимодействует с программой для управления школьной информацией. Основные функции программного комплекса представлены следующими прецедентами:

- Добавление данных: завуч может добавлять информацию об учениках, учителях, классах и предметах.
- Редактирование данных: изменение ранее внесенной информации.
- Удаление данных: удаление записей об учениках, учителях, классах и предметах.

- Просмотр данных: возможность просмотра сведений, например, о том, какие учителя преподают определенные предметы или в каких классах они работают.
- Генерация отчета: создание отчетов об успеваемости учеников (количество отличников и двоечников).

Типы связей между прецедентами

1. Связь коммуникации: завуч взаимодействует с каждым прецедентом напрямую. Например, инициирует добавление данных через графический интерфейс.
2. Связь использования (uses): функции добавления, редактирования и удаления данных используют базовые операции, такие как валидация введенной информации или сохранение изменений в базе данных.
3. Связь расширения (extends): например, генерация отчета может быть расширена функционалом фильтрации по классам или учителям.

Список акторов и прецедентов

Акторы:

- Завуч: Основной пользователь системы, который управляет данными и генерирует отчеты.

Прецеденты:

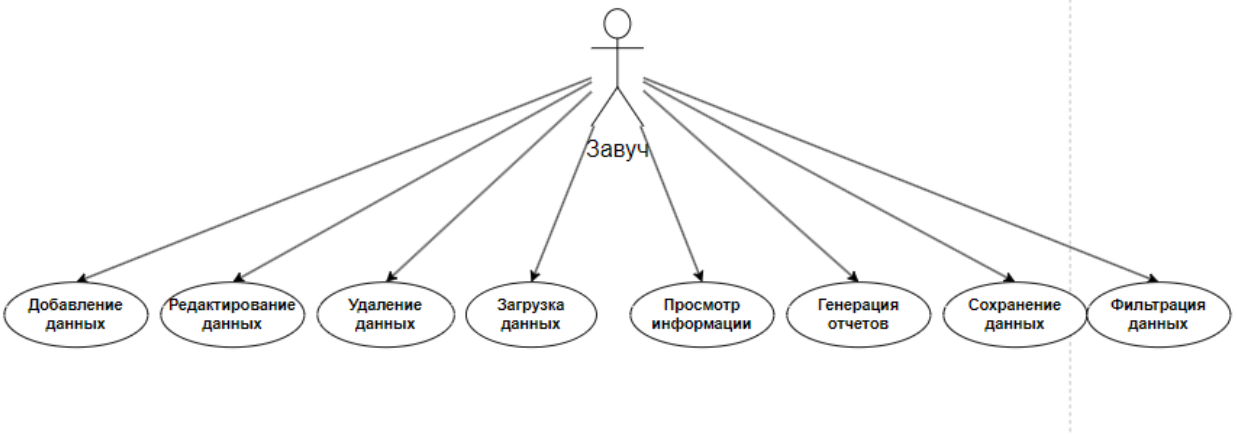
- *Добавление данных*: Завуч добавляет новую информацию об учениках, учителях, классах или предметах.
- *Редактирование данных*: Завуч вносит изменения в существующую информацию (ученики, учителя, классы, предметы).
- *Удаление данных*: Завуч удаляет ненужные или устаревшие записи из системы.

- *Просмотр информации*: Завуч просматривает данные о том, какие учителя ведут определенные предметы, в каких классах они преподают, и общую статистику учеников.
- *Фильтрация данных*: Завуч отбирает данные по определенным критериям, например, список учеников конкретного класса или преподавателей по предметам.
- *Генерация отчетов*: Завуч создает отчет об успеваемости учеников (количество отличников и двоечников с указанием фамилий).
- *Сохранение данных*: Программа сохраняет текущие данные в файл для их последующего использования.
- *Загрузка данных*: Завуч загружает ранее сохраненные данные из файла для работы с ними в программе.

Связи

Пользователь(Завуч) связан со всеми прецедентами, так как он выполняет все действия.

Диаграмма прецедентов



Создание прототипа пользовательского интерфейса

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).

Создание прототипа интерфейса пользователя

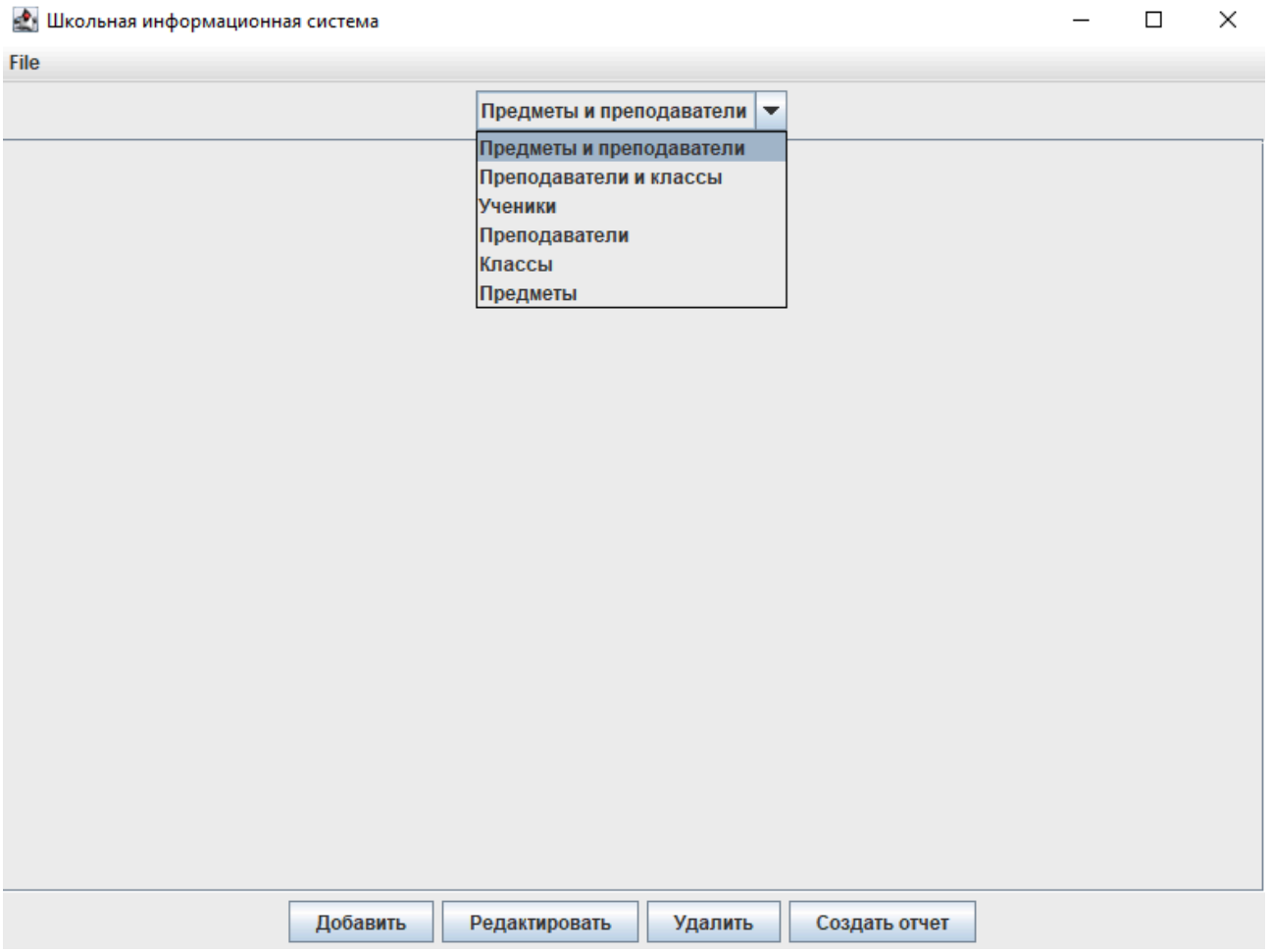


Рис.2.1. Выбор таблицы.

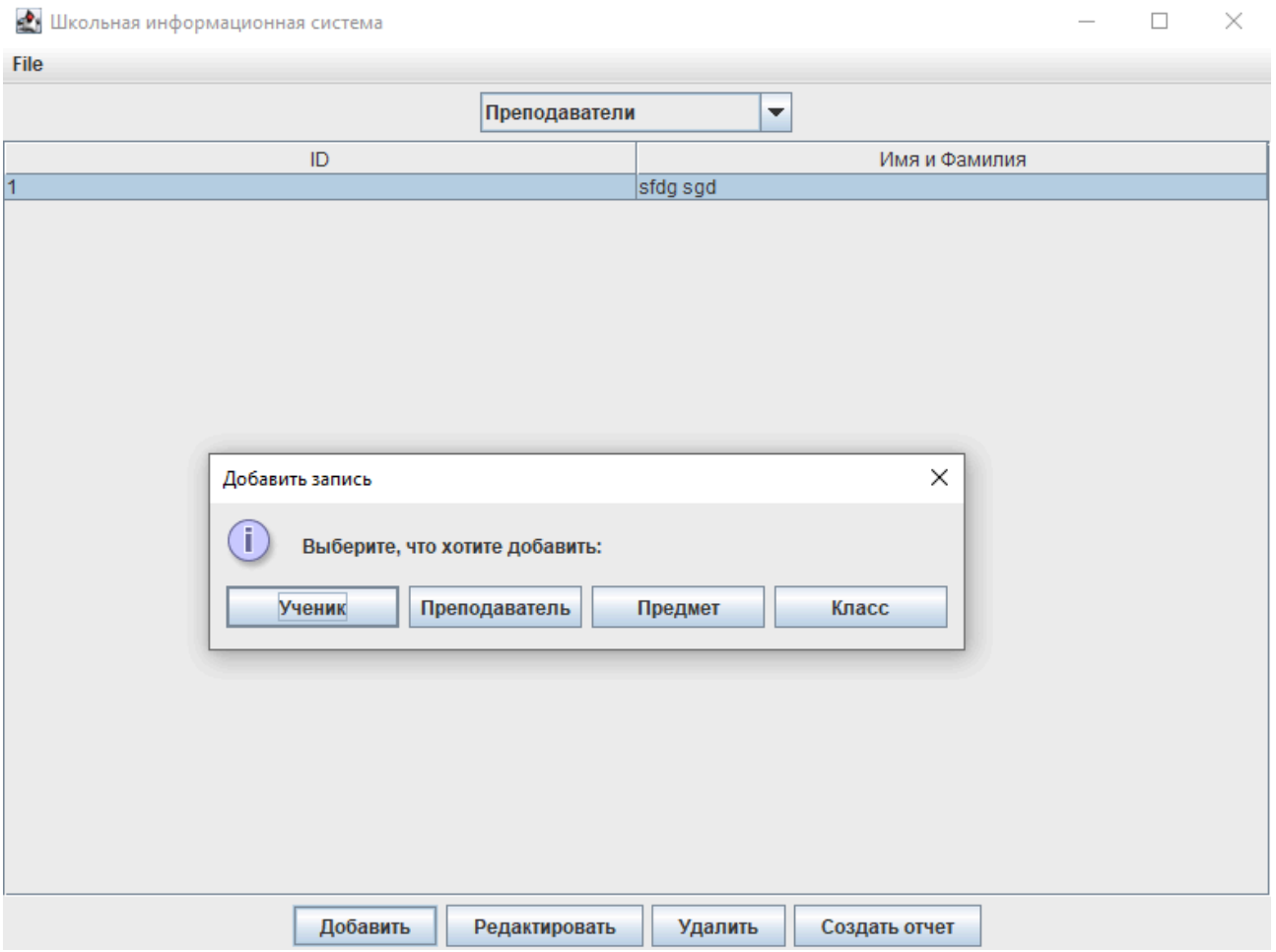


Рис.2.2. Окно добавления.

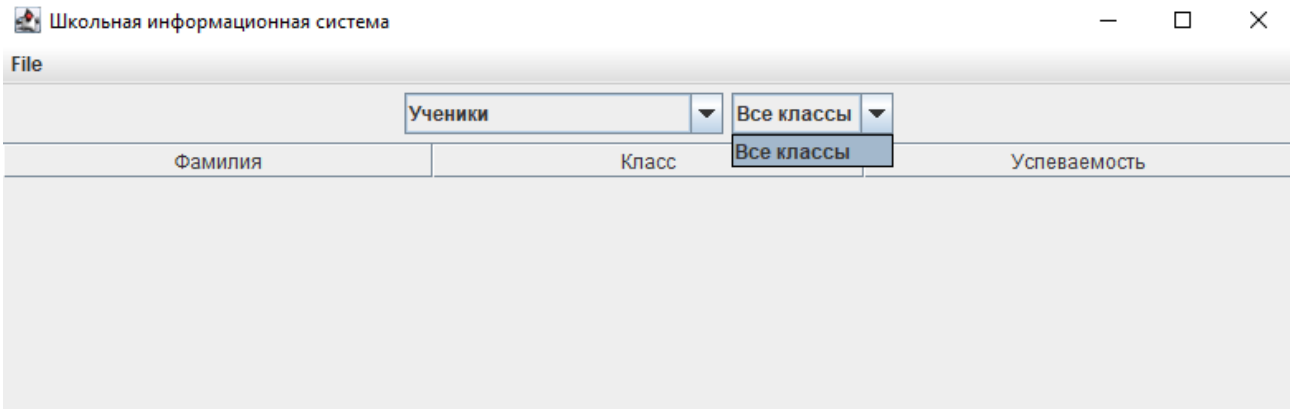


Рис.2.3.Фильтрация по классам.

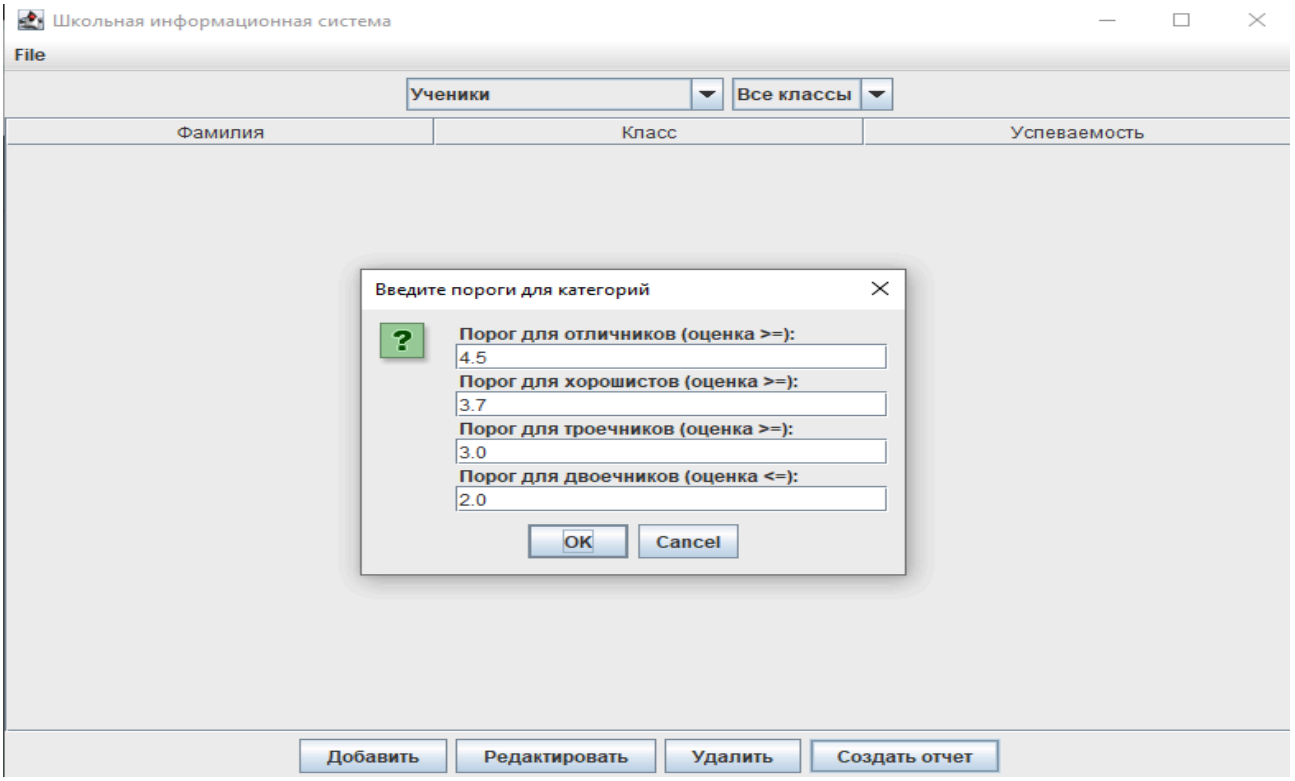


Рис.2.4.Создание отчета.

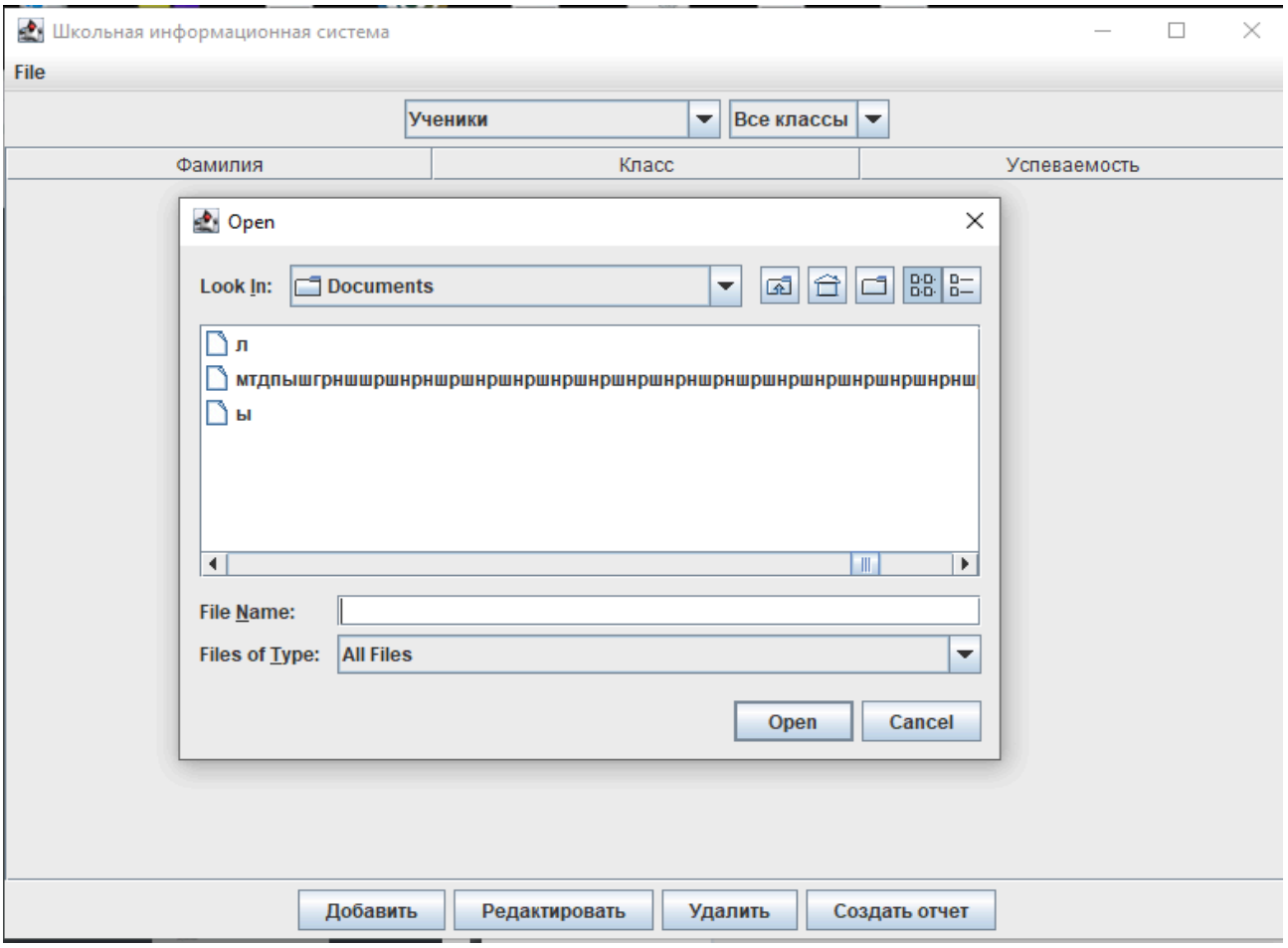


Рис.2.5.Загрузка данных.

Экранная форма	Элементы управления	Действия пользователя	Отклик системы
Список предметов и преподавателей	Таблица с полями: предмет, преподаватель. Кнопки: «Добавить», «Редактировать»,	Нажать кнопку «Добавить» — открыть форму для ввода данных о преподавателе и предмете. Нажать кнопку «Редактировать» — изменить данные. Нажать кнопку	Открыть форму для добавления/редактирования данных о предмете и преподавателе.
Список преподавателей и классов	Таблица с полями: преподаватель, класс. Кнопки: «Добавить», «Редактировать», «Удалить».	Нажать кнопку «Добавить» — открыть форму для ввода данных о преподавателе и классе. Нажать кнопку «Редактировать» — изменить данные. Нажать кнопку	Открыть форму для добавления/редактирования данных о преподавателе и классе.
Список учеников	Таблица с полями: фамилия, имя, класс. Кнопки: «Добавить», «Редактировать», «Удалить».	Нажать кнопку «Добавить» — открыть форму для ввода данных ученика. Нажать кнопку «Редактировать» — изменить данные. Нажать кнопку «Удалить» — удалить выбранного ученика.	Открыть форму для добавления/редактирования данных ученика. Удалить ученика из списка.
Список преподавателей	Таблица с полями: фамилия, имя, предметы. Кнопки: «Добавить», «Редактировать», «Удалить».	Нажать кнопку «Добавить» — открыть форму для ввода данных о преподавателе. Нажать кнопку «Редактировать» — изменить данные. Нажать кнопку «Удалить» — удалить выбранного преподавателя.	Открыть форму для добавления/редактирования данных преподавателя. Удалить преподавателя из списка.

Список классов	Таблица с полями: класс. Кнопки: «Добавить», «Редактировать», «Удалить».	Нажать кнопку «Добавить» — открыть форму для ввода данных о классе. Нажать кнопку «Редактировать» — изменить данные. Нажать кнопку «Удалить» — удалить выбранный класс.	Открыть форму для добавления/редактирования данных класса. Удалить класс из списка.
Список предметов	Таблица с полями: название предмета. Кнопки: «Добавить», «Редактировать», «Удалить».	Нажать кнопку «Добавить» — открыть форму для ввода данных о предмете. Нажать кнопку «Редактировать» — изменить данные. Нажать кнопку «Удалить» — удалить выбранный предмет.	Открыть форму для добавления/редактирования данных предмета. Удалить предмет из списка.
Отчет по успеваемости	Кнопка: «Создать отчет».	Нажать кнопку «Создать отчет» — сгенерировать отчет по успеваемости.	Сохранить файл, содержащий отчет с количеством отличников и двоечников.
Выбор класса (для учеников)	Выпадающий список (ComboBox): «Все классы».	Выбрать класс для отображения учеников.	Обновить отображение списка учеников в выбранном классе.
Сохранение данных	Пункт меню: «Файл» → «Сохранить данные»	Нажать пункт меню «Сохранить данные».	Сохранить текущие данные в файл XML. Отобразить сообщение о результате сохранения (успех/ошибка).
Загрузка данных	Пункт меню: «Файл» → «Загрузить данные».	Нажать пункт меню «Загрузить данные».	Загрузить данные из XML-файла. Обновить таблицы данными из файла или показать сообщение об ошибке.

Разработка объектной модели ПК

Объектная модель отражает основные понятия предметной области в виде классов с их атрибутами, операциями и связями. Она описывает структуру и поведение объектов системы, помогает определить ключевые сущности и их взаимодействие. Модель используется для визуализации, анализа и проектирования системы, облегчая переход от абстрактных требований к конкретной реализации.

Сущности

На диаграмме сущность представляется в виде прямоугольника, внутри которого указано имя сущности, ее атрибуты и операции. В нашем приложении можно выделить следующие сущности:

1. *Ученик*

   Атрибуты:

   - *id: int — уникальный идентификатор.*
   - *фамилия: String*
   - *имя: String*
   - *класс: String*

   Операции:

   - *добавитьУченик()*
   - *изменитьУченик()*
   - удалитьУченик()
2. *Преподаватель*

   Атрибуты:

   - *id: int — уникальный идентификатор.*
   - *фамилия: String*
   - *имя: String*
   - *предметы: List<String>*

Операции:

- *добавитьПреподаватель()*
- *изменитьПреподаватель()*
- удалитьПреподаватель()

3. *Предмет*

Атрибуты:

- *id: int — уникальный идентификатор.*
- *название: String*

Операции:

- *добавитьПредмет()*
- *изменитьПредмет()*
- удалитьПредмет()

4. *Класс*

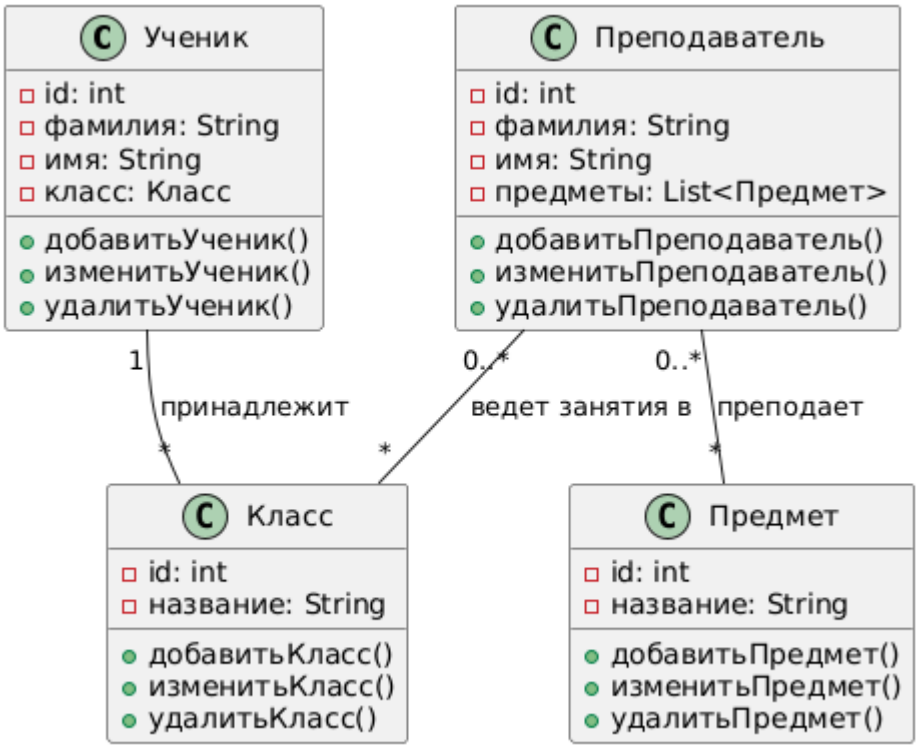
Атрибуты:

- *id: int — уникальный идентификатор.*
- *название: String*

Операции:

- *добавитьКласс()*
- *изменитьКласс()*
- удалитьКласс()

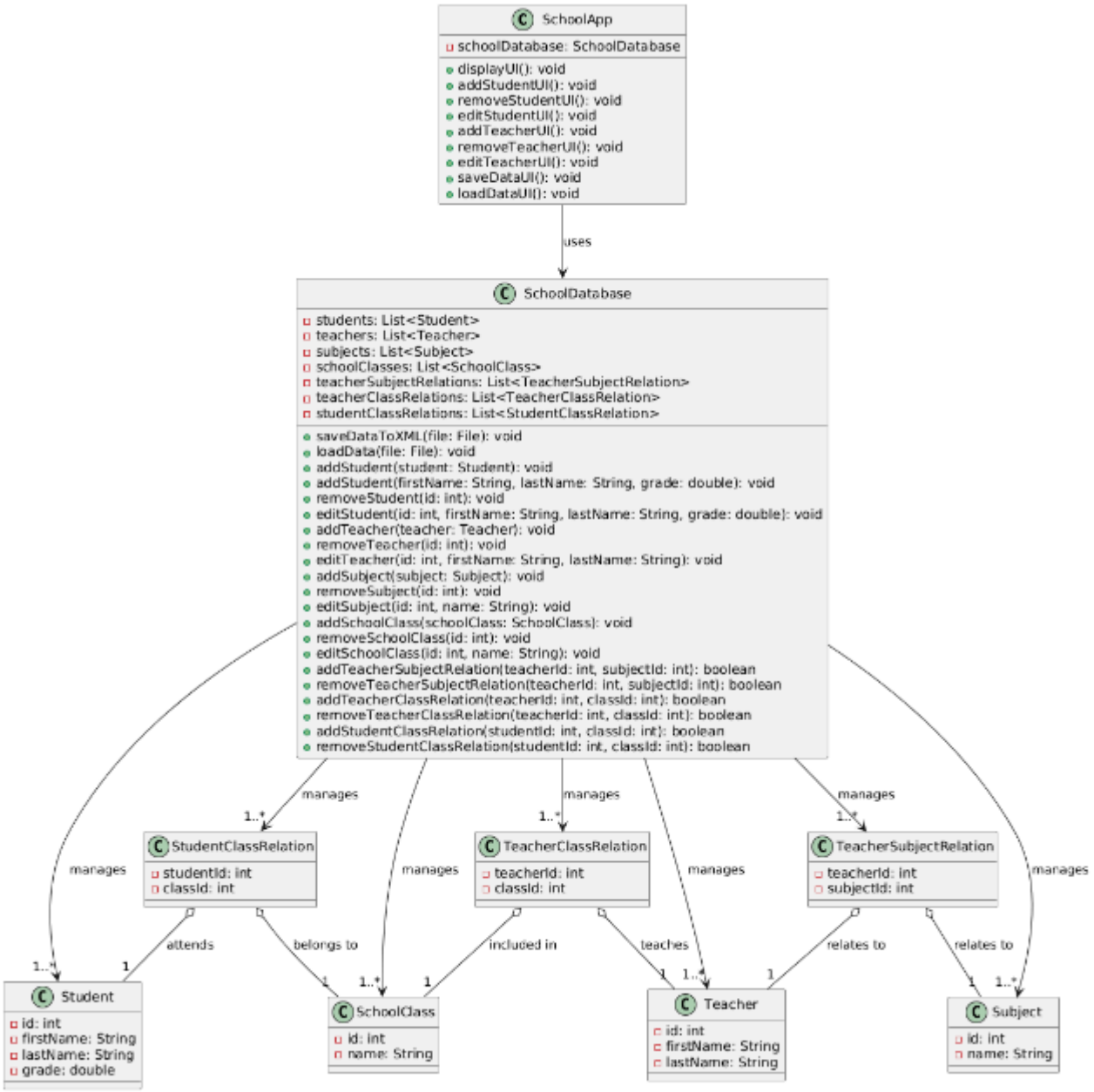
Диаграмма сущностей



Построение диаграммы программных классов

Построение диаграммы программных классов представляет собой процесс визуализации структуры системы. На диаграмме отображаются основные программные классы, их атрибуты, методы и взаимосвязи. Это позволяет формализовать архитектуру системы, определить взаимодействия между компонентами и задать механизм доступа к данным.

Диаграмма классов



Описание поведения ПК

Описание поведения программного комплекса (ПК) заключается в представлении последовательности действий, выполняемых системой, без детализированного механизма их реализации. Одной из важнейших составляющих такого описания является **диаграмма последовательностей** (sequence diagram). Эта диаграмма визуализирует порядок выполнения операций в определенном сценарии, указывая, какие события (запросы) генерируются пользователями и объектами системы, а также как они взаимодействуют.

Диаграмма последовательности имеет две оси:

- **Вертикальная ось** представляет собой временную шкалу, которая отображает последовательность выполнения операций.
- **Горизонтальная ось** отображает объекты, взаимодействующие в процессе выполнения сценария.

Основные этапы построения диаграммы последовательности:

1. **Идентификация пользователей и объектов**:
   На первом шаге необходимо определить, какие пользователи и объекты программного комплекса участвуют в реализации сценария. Затем объекты изображаются в виде прямоугольников, расположенных горизонтально, а для каждого объекта рисуется линия жизни (пунктирная вертикальная линия).
2. **Определение операций**:
   Следующим шагом из объектной модели выбираются операции, которые участвуют в сценарии. Если они ещё не определены, их необходимо добавить в модель. Эти операции будут отображены на диаграмме последовательности.
3. **Запросы и взаимодействие объектов**:
   Каждой операции на диаграмме последовательности соответствует горизонтальная линия со стрелкой, начинающаяся от объекта, который вызывает операцию, и заканчивающаяся на объекте, который её выполняет. На стрелке указывается порядковый номер операции, её имя и параметры. После операции можно добавить комментарий, поясняющий её смысл.
4. **Время выполнения операций**:
   Операция, выполняемая в ответ на запрос, будет отображаться на линии жизни объекта в виде прямоугольника. Порядок выполнения операций определяется их номером: чем ниже горизонтальная линия, тем позже операция выполняется.

5. **Вложенность операций**:
   Вложенные операции получают свои номера, начиная с 1. Для отображения повторяющихся операций используется нумерация с квадратными скобками и звёздочками для указания, что операция выполняется несколько раз. Также можно указать диапазон повторений.
6. **Условия и создание/удаление объектов**:
   На диаграмме можно отобразить операции с условием (if-else), а также моменты создания и уничтожения объектов. Если объект создаётся или уничтожается, его линия жизни начинается или заканчивается в соответствующие моменты времени. Символ объекта рисуется в начале его линии жизни, а при уничтожении объекта его линия жизни помечается крестиком "X".

Связь многие ко многим

В рамках разработки системы управления данными о школе была реализована связь "многие ко многим" между следующими сущностями:

1. **Учителя и предметы**
2. **Учителя и классы**
3. **Студенты и классы**

Эти связи были реализованы через использование вспомогательных классов, которые хранят информацию о том, какие учителя преподают какие предметы, в каких классах учат учителя, а также в каких классах учатся студенты.

1. Связь "Учитель - Предмет" (Many-to-Many)

Связь между учителями и предметами реализована через класс, который представляет собой таблицу связей между учителем и предметом. В этой таблице хранятся пары идентификаторов учителя и предмета. Это позволяет одному учителю преподавать несколько предметов, а одному предмету — преподавать несколько учителей. Для добавления или удаления связей проверяется наличие

соответствующих элементов в базе данных, что обеспечивает целостность данных.

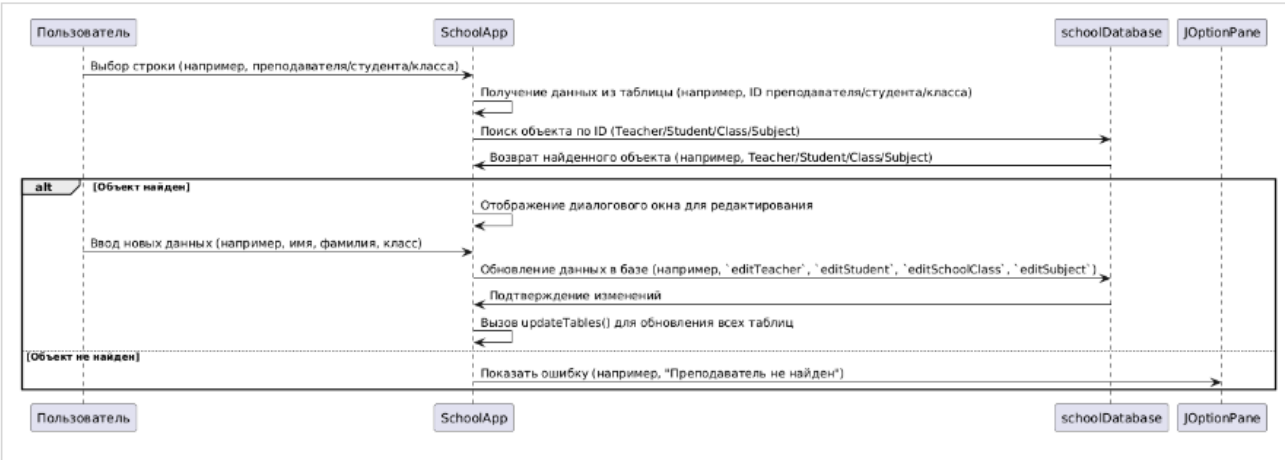
2. Связь "Учитель - Класс" (Many-to-Many)

Связь между учителями и классами реализована через класс, который хранит идентификаторы учителей и классов. Это позволяет одному учителю преподавать в нескольких классах, а одному классу — иметь несколько учителей. Подобно предыдущей связи, добавление и удаление связей осуществляется с проверкой существования учителей и классов в базе данных для предотвращения ошибок.

3. Связь "Студент - Класс" (Many-to-Many)

Связь между студентами и классами реализована через класс, который хранит идентификаторы студентов и классов. Это позволяет одному студенту учиться в нескольких классах, а одному классу — иметь несколько студентов. Проверка целостности данных также проводится при добавлении или удалении связей между студентами и классами.

Диаграмма последовательностей



Построение диаграммы действий

Диаграмма действий помогает наглядно представить алгоритм работы с данными и управления их редактированием в системе, обеспечивая понимание

всех шагов, которые выполняются от начала до конца операции

Описание шагов на диаграмме

1. *Начало: Начальный узел, откуда начинается процесс.*

2. *Выбор строки: Пользователь выбирает строку в таблице (например, из таблицы преподавателей, студентов или классов).*

3. *Извлечение данных из таблицы: Извлекаются данные о выбранном объекте из таблицы (например, ID преподавателя или имя студента).*

4. *Поиск объекта в базе данных: Программа ищет объект в базе данных по полученному ID.*

5. *Объект найден?: Проверка наличия объекта в базе данных. Если объект найден, продолжаем, если нет — выводим ошибку.*

6. *Показать форму для редактирования: Отображаем форму редактирования для введения новых данных.*

7. *Ввод данных: Пользователь вводит данные в форму.*

8. *Проверка данных: Проверка на корректность введенных данных. Если данные корректны, продолжаем.*

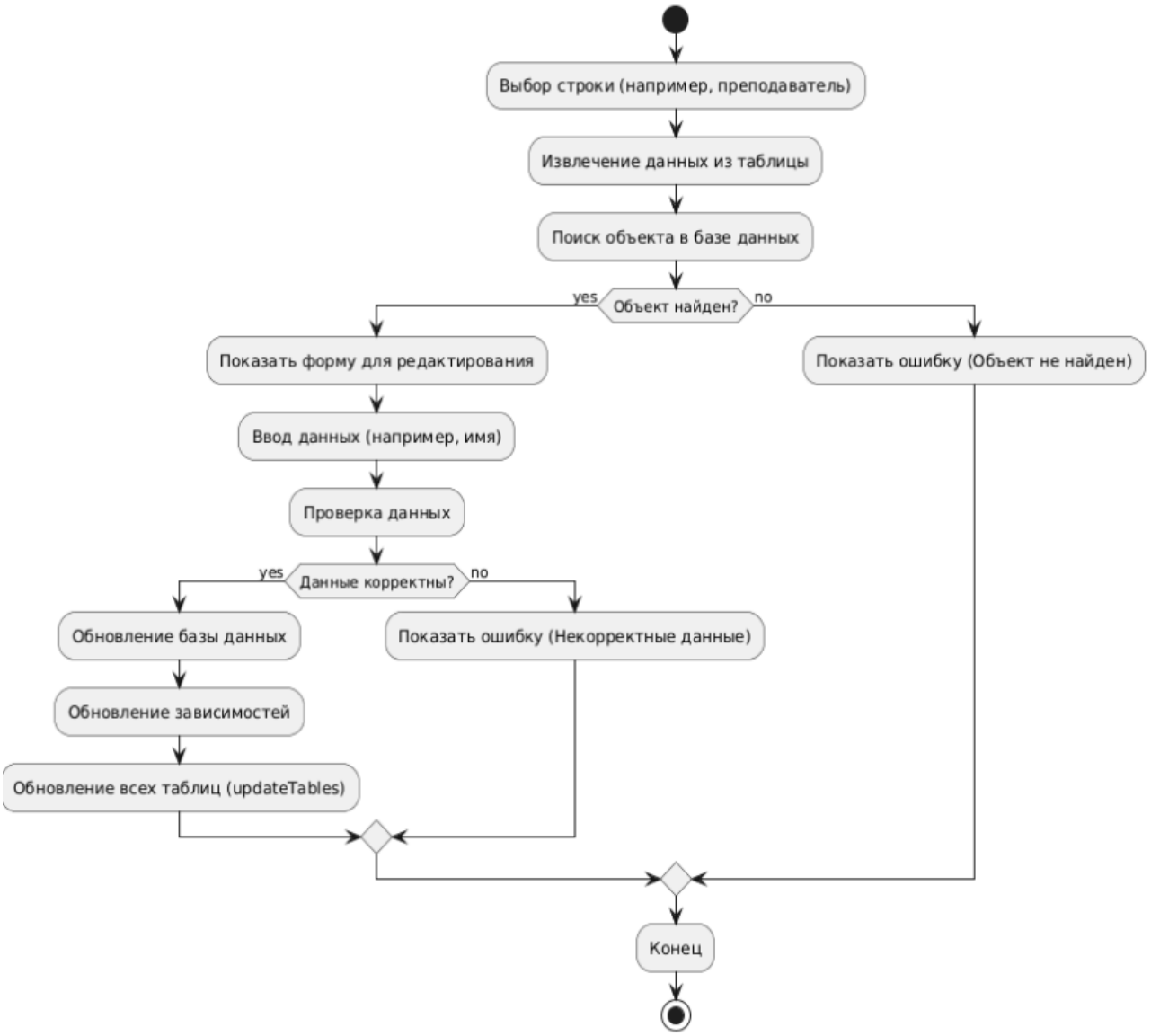
9. *Обновление базы данных: Обновляем записи в базе данных с новыми данными.*

10. *Обновление зависимостей: Если изменяются связи (например, класс преподавателя), обновляем их в базе.*

11. *Обновление всех таблиц: Вызываем метод updateTables(), чтобы обновить все таблицы и отобразить изменения.*

12. *Конец: Процесс завершен, и пользователь может продолжить работу*

Диаграмма действий



Руководство оператора

Программа: Школьная информационная система

1. Назначение программы

Программа предназначена для управления данными о преподавателях, студентах, классах и предметах в образовательном учреждении. Она позволяет:

- Добавлять, редактировать и удалять данные о преподавателях, студентах, классах и предметах.
- Управлять зависимостями между преподавателями и классами, а также

между студентами и классами.
- Генерировать отчеты о числе студентов в классах, преподавании предметов и успеваемости студентов.
- Обновлять информацию и отображать её в таблицах для удобного просмотра и редактирования.

2. Условия выполнения программы

Программа должна выполняться на ПК с операционной системой **Windows** или **Linux**. Для ее работы необходимы следующие требования:

- Минимальные требования:
    - Процессор: 1 ГГц или выше.
    - Оперативная память: 2 ГБ.
    - Свободное место на диске: 100 МБ.
    - Java.
- Рекомендуемые требования:
    - Процессор: 2.5 ГГц или выше.
    - Оперативная память: 4 ГБ или выше.
    - Свободное место на диске: 200 МБ.
    - Java.

3. Описание задачи

Задача программы состоит в управлении данными о преподавателях, студентах, классах и предметах. Оператор должен использовать программу для:

- Редактирования информации о преподавателях, студентах и классах.
- Обновления зависимостей между классами и преподавателями, а также между студентами и классами.
- Генерации отчета об успеваемости.

4. Входные и выходные данные

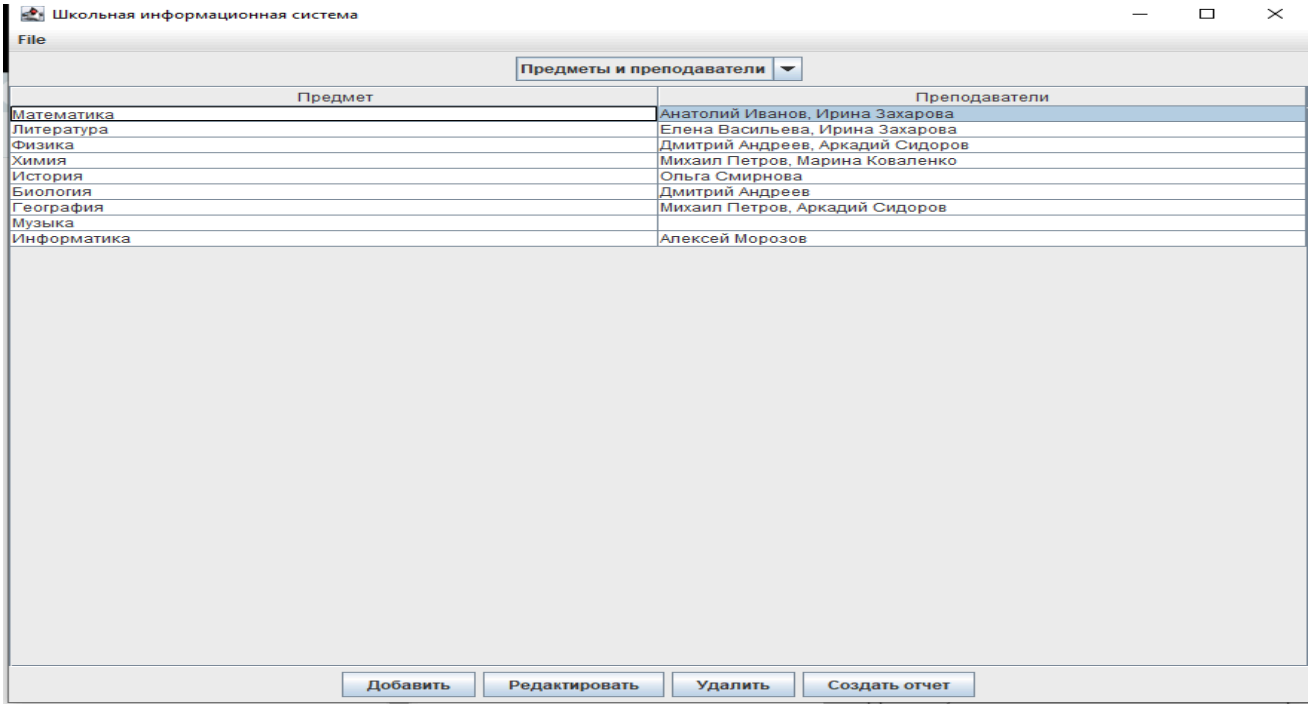
Входные данные:

- Для редактирования преподавателей, студентов и классов требуется ввод данных через текстовые поля, такие как имя, фамилия, класс, успеваемость и т. д.
- Для изменения зависимостей между преподавателями и классами или студентами и классами необходимо выбрать соответствующие элементы в интерфейсе (например, выбрать класс из списка).

Выходные данные:

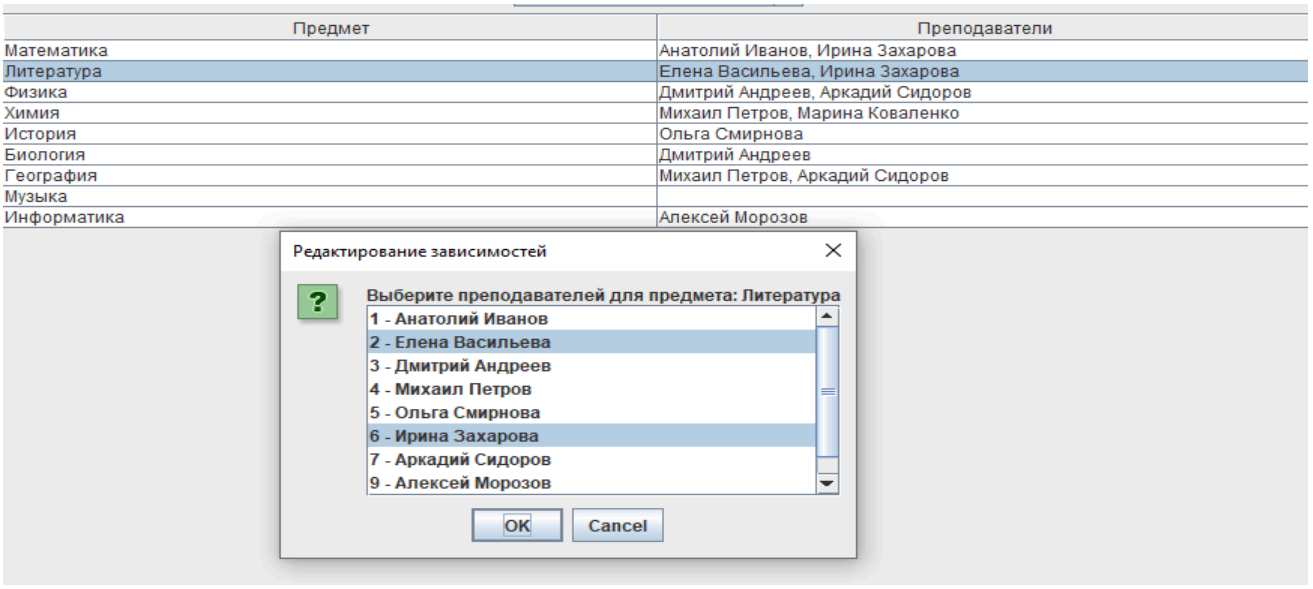
- После выполнения операций редактирования или добавления данных, программа отображает обновленные данные в таблицах.
- Программа также генерирует отчеты по количеству студентов, успеваемости студентов.

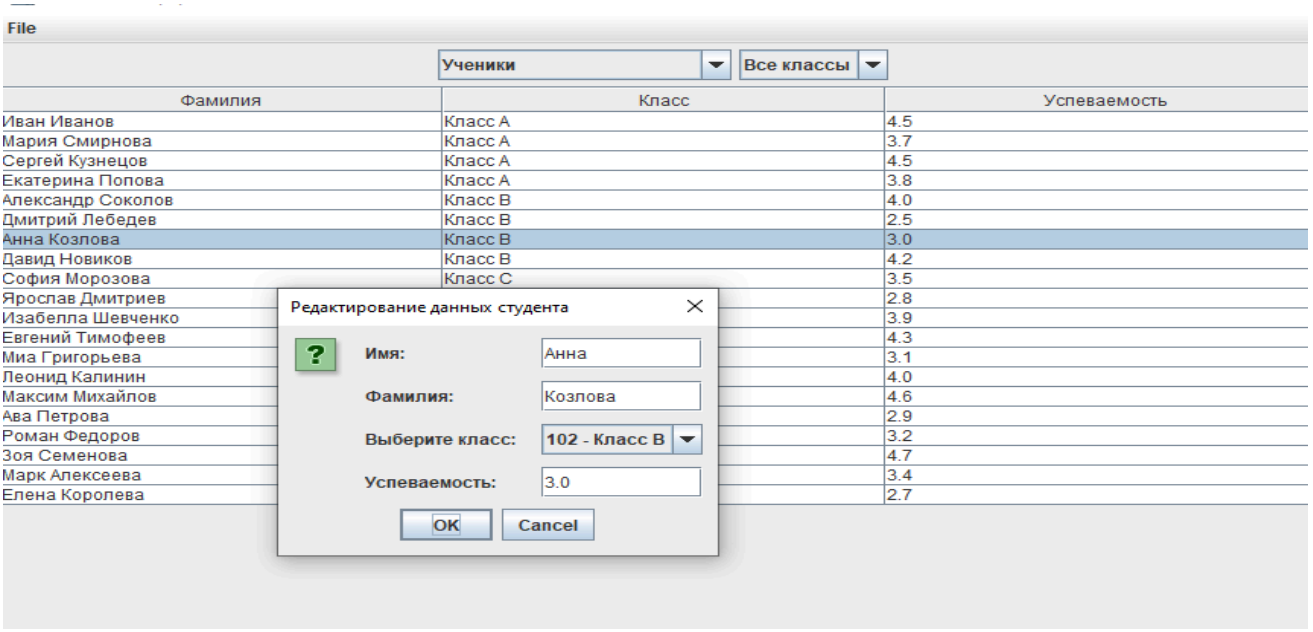
5. Выполнение программы



1. **Выбор операции**:

- Оператор может выбрать одну из доступных операций,например, таких как:
    - Редактирование данных преподавателя.
    - Редактирование данных студента.
    - Редактирование данных класса.
    - Редактирование данных предмета.
- Для этого необходимо выбрать соответствующую строку в таблице, после чего программа предложит форму для ввода данных.





2. **Редактирование данных**:
    - После выбора строки и открытия формы, оператор вводит данные и нажимает **OK** для сохранения или **Cancel** для отмены.
    - Если введенные данные корректны, они сохраняются в базе данных, и таблицы обновляются.
3. **Обновление таблиц**:
    - После выполнения операции программа автоматически обновляет все таблицы, отображающие актуальные данные.

6. Проверка программы

Для проверки корректности работы программы:

- Проверьте, что после редактирования данных преподавателей, студентов и классов информация обновляется в таблицах.
- Убедитесь, что после выполнения операций отображаются актуальные данные (например, количество студентов в классе, информация о преподавателях).

Контрольные примеры:

- Добавление нового преподавателя.
- Изменение информации о студенте (например, смена класса).
- Редактирование названия класса.

Заключение

В рамках курсового проектирования был разработан программный комплекс для управления учебным процессом, что позволило закрепить теоретические знания и развить практические навыки в объектно-ориентированном программировании и проектировании. Были выполнены следующие задачи: изучение предметной области, объектно-ориентированное проектирование с использованием UML, реализация системы на объектно-ориентированном языке и подготовка программной документации.

Ссылка на репозиторий: https://github.com/lovushker/OOP-Coursework