



Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный электротехнический
университет “ЛЭТИ” им.В.И.Ульянова (Ленина)»

Кафедра ВТ

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

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

Студент гр. 3311

Локтионов Т.И.

Преподаватель

Павловский М.Г

Санкт-Петербург

2024

1. Техническое задание

1.1 Введение

Программный комплекс (ПК) «Система управления клиникой» предназначен для автоматизации процессов учёта и управления информацией о пациентах, врачах и расписании приёмов в клинике. Система разработана с использованием языка программирования **Java** и графической библиотеки **Swing** для создания настольного приложения.

1.2 Основание для разработки

Основанием для разработки ПК является учебный проект по дисциплине «**Объектно-ориентированное программирование**».

1.3 Назначение разработки

ПК «Система управления клиникой» предназначен для автоматизации следующих задач:

- Ведение базы данных о пациентах, включая их ФИО, диагноз, лечащего врача и статус лечения.
- Управление информацией о врачах, включая их специализацию и график приёма.
- Учет и управление расписанием приёмов.
- Поиск пациентов по имени, болезни или лечащему врачу.
- Добавление, редактирование и удаление данных о пациентах, врачах и расписании.
- Генерация отчётов в формате **PDF** и **HTML**.
- Сохранение и загрузка данных в формате **XML-файлов**.

Если в проекте отсутствует база данных, хранение данных осуществляется в текстовых или XML-файлах.

1.4 Требования к программе

1.4.1 Требования к функциональным характеристикам

1.4.1.1 Перечень функций

ПК обеспечивает выполнение следующих функций:

- Отображение данных о пациентах, врачах и приёмах в виде таблиц с возможностью фильтрации и поиска.
- Добавление новых записей о пациентах, врачах и приёмах.
- Удаление и редактирование существующих записей.
- Сохранение изменений в XML-файл.
- Загрузка данных из XML-файла при запуске программы.
- Экспорт данных о приёмах и пациентах в **PDF** и **HTML**.

1.4.1.2 Требования к составу выполняемых функций

1.4.1.2.1 Управление данными о пациентах

- Сохранение информации о пациентах, включая:
 - ФИО пациента
 - Возраст пациента

- Диагноз
- Статус лечения (напри. в ожидании, активный, завершён)

1.4.1.2.2 Управление данными о врачах

- Сохранение информации о врачах, включая:
 - ФИО врача
 - Специализация
 - Расписание приёма

1.4.1.2.3 Управление расписанием приёмов

- Учет данных о приёмах, включая:
 - Дата и время приёма
 - Лечащий врач
 - Пациент
 - Статус приёма (подтверждён, отменён, завершён)

1.4.1.3 Требования к организации и форме представления данных

Выходные данные:

- Информация о пациентах, врачах и приёмах представляется в табличной форме с возможностью фильтрации и поиска.
- Экспорт данных в **XML**, **PDF** и **HTML** форматы.

Входные данные:

- Ввод данных осуществляется оператором через диалоговые окна.
- Для каждой категории данных (пациенты, врачи, приёмы) предусмотрено отдельное окно ввода.

1.4.2 Требования к надежности

ПК должен устойчиво функционировать при соблюдении следующих условий:

- Стабильная работа операционной системы **Windows**.
- Работа с **XML-файлами** для хранения данных.
- Обработка исключений при вводе некорректных данных или ошибках доступа к файлам.

1.4.3 Условия эксплуатации

- Однопользовательский режим работы.

1.4.4 Требования к составу и параметрам технических средств

Программный комплекс должен функционировать на ПК со следующими минимальными характеристиками:

- **Процессор:** Intel Core i3 или аналогичный.
- **ОЗУ:** 4 ГБ.
- **HDD:** 10 ГБ свободного пространства.
- **Монитор:** с разрешением 1366x768 и выше.
- **Манипулятор:** мышь и клавиатура.

1.4.5 Требования к информационной и программной совместимости

1. Программа должна быть разработана на языке **Java**.
2. Совместимость с ОС **Windows 10/11**.
3. Поддержка русскоязычного и англоязычного интерфейсов.
4. Использование ключевых конструкций языка Java:
 - **Инкапсуляция и наследование**
 - **Конструкторы с параметрами**
 - **Абстрактные классы**
 - **Обработка исключений**

1.5 Требования к программной документации

Программная документация должна соответствовать стандартам **ЕСПД** и включать:

- Описание процесса проектирования ПК.
- Руководство пользователя.
- Исходные тексты программы.

1.6 Стадии и этапы разработки

1. Разработка объектной модели системы.
2. Создание структуры данных (XML-файлы).
3. Разработка функционала для работы с XML-файлами (загрузка, сохранение, обновление).
4. Создание GUI-интерфейса с использованием **Swing**.
5. Реализация **CRUD-операций** (добавление, редактирование, удаление данных).
6. Реализация экспорта данных в **PDF** и **HTML**.
7. Тестирование всех функций программы.
8. Подготовка документации и финальная отладка.

1.7 Порядок контроля и приемки

Контроль качества осуществляется на основании следующих критериев:

1. Корректная работа функций **добавления, изменения и удаления данных**.
2. Отображение данных в таблицах с поддержкой фильтрации и поиска.
3. Обработка ошибок при вводе данных или сохранении информации в XML-файлы.
4. Стабильная работа приложения при стандартных условиях эксплуатации.

2. Проектирование ПК

2.1 Описание вариантов использования ПК

Диаграмма прецедентов (*use case diagram*) наглядно демонстрирует, какие действия пользователь может выполнять в системе, как система на них реагирует, и как происходит взаимодействие с данными.

Основные акторы системы:

- **Пользователь (User)** – выполняет базовые действия: добавление, удаление, редактирование и поиск информации.
- **Администратор (Admin)** – управляет доступом, регистрирует пользователей и имеет расширенные права.
- **Система (System)** – обрабатывает данные, сохраняет и загружает их из XML-файлов.

Основные прецеденты (Use Cases):

1. **Добавить пациента:** Добавление нового пациента с необходимыми данными.
2. **Удалить пациента:** Удаление информации о пациенте.
3. **Редактировать пациента:** Внесение изменений в данные пациента.
4. **Сохранить данные:** Сохранение информации в XML-файл.
5. **Загрузить данные:** Загрузка информации из XML-файла.
6. **Поиск пациента:** Поиск пациента по имени, врачу или диагнозу.
7. **Генерация отчётов:** Создание отчётов в формате PDF или HTML.
8. **Регистрация пользователя:** Создание учетных записей для новых сотрудников.

Связи между прецедентами:

- **Использование (uses):** Например, прецедент «*Сохранить данные*» может использовать «*Загрузить данные*» для проверки целостности информации перед сохранением.
- **Расширение (extends):** Прецедент «*Поиск пациента*» может быть расширен дополнительными фильтрами (по дате, статусу и т.д.).

Взаимодействие с данными:

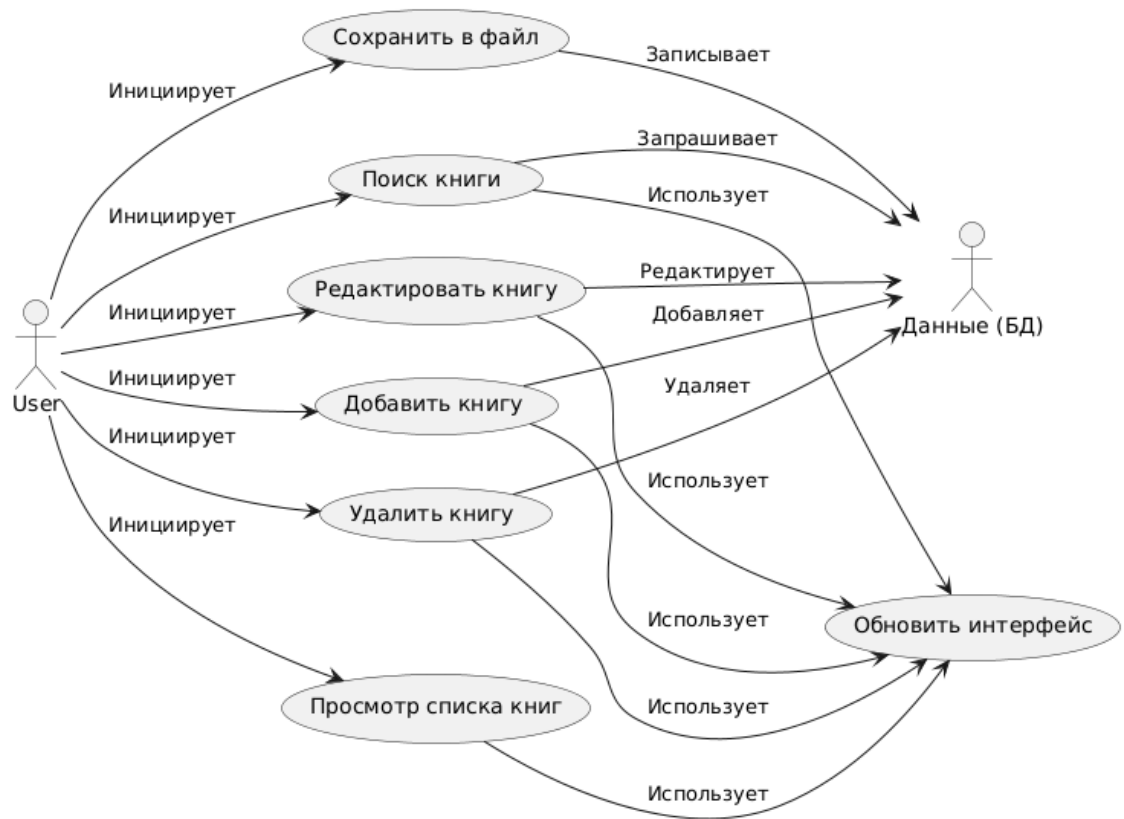
- Операции добавления, редактирования и удаления данных обновляют XML-файл.
- Действия, связанные с поиском и фильтрацией, обращаются к данным для получения актуальной информации.
- Прецедент «*Обновить интерфейс*» используется после каждой операции, чтобы отобразить изменения.

Описание диаграммы прецедентов:

- Актор «*Пользователь*» взаимодействует со всеми основными прецедентами, связанными с управлением пациентами, приёмами и

отчётами.

- Актор «Администратор» отвечает за регистрацию пользователей и контроль доступа.
- Прецеденты «Сохранить данные» и «Загрузить данные» логически связаны, так как оба работают с XML-файлами.



2.2 Создание прототипа интерфейса пользователя.

Описание прецедентов даёт общее представление о функциональных возможностях системы, однако для успешной реализации требуется детальная проработка пользовательского интерфейса (*User Interface, UI*).

Элементы пользовательского интерфейса:

- Поля ввода данных:
 - Текстовые поля для ввода имени пациента, болезни, даты приёма, статуса и других данных.
- Элементы управления:
 - Кнопки для выполнения основных операций:
 - *Добавить пациента*
 - *Удалить пациента*
 - *Поиск пациента*
 - *Сохранить данные*
 - *Загрузить данные*
 - Выпадающие списки для выбора врача, даты или статуса приёма.
- Таблицы (JTable):
 - Для отображения списка пациентов, врачей и приёмов.
 - Возможность фильтрации и сортировки данных.
- Действия пользователя:
 - Выбор записи для редактирования.
 - Ввод данных в текстовые поля.
 - Нажатие кнопок для выполнения операций.
- Реакции системы:
 - Отображение сообщений об успешном выполнении операций.
 - Вывод предупреждений в случае ошибок.
 - Обновление данных в таблице после каждой операции.

Примеры экранных форм:

1. Главное окно:
 - Отображение таблицы с пациентами.
 - Панель управления с кнопками (*Добавить пациента, Удалить пациента, Сохранить данные, Загрузить данные*).
2. Окно добавления пациента:
 - Поля для ввода данных (Имя, Болезнь, Врач, Дата приёма, Статус).
 - Кнопка *Сохранить*.
3. Окно поиска пациента:
 - Поле ввода ключевого слова.
 - Кнопка *Поиск*.

- Таблица с результатами поиска.
4. Окно генерации отчётов:
- Кнопка для экспорта отчётов в PDF или HTML.
 - Поле для выбора параметров отчёта.

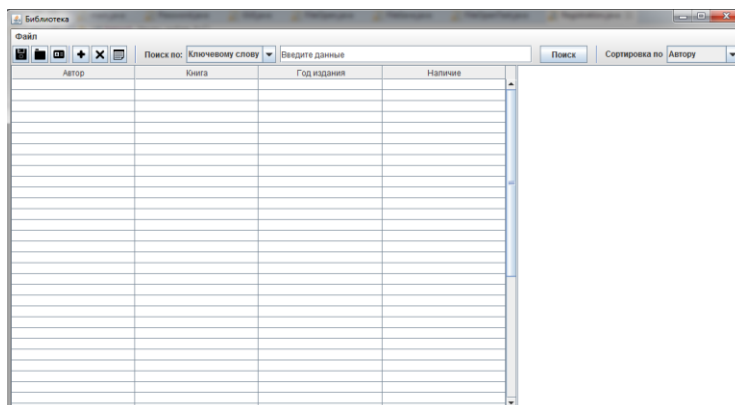
Связь интерфейса с логикой приложения:

- При нажатии кнопки «Добавить пациента» вызывается соответствующий метод для добавления данных.
- После успешного выполнения операции вызывается метод обновления интерфейса.
- При возникновении ошибки система отображает соответствующее уведомление.

Итоговая структура интерфейса пользователя:

- Вкладка 1: Управление пациентами.
- Вкладка 2: Управление расписанием.
- Вкладка 3: Генерация отчётов.
- Вкладка 4: Настройки и права доступа.

Макет:



Готовый результат:

Клиника - Список пациентов						
Файл						
Сохранить Добавить Удалить			Поиск по: Имени пациента Поиск...			
Имя пациента	Болезнь	Врач	Специализация врача	Дата приёма	Статус	
Ivan Ivanov	Acute Respiratory Viral Infe...	Doctor A	Therapist	01.10.2024	Accepted	
Tim	Depression	Doctor B	Psychiatrist	02.10.2024	Waiting	
Maria Petrova	Asthma	Doctor C	Therapist	03.10.2024	Cancelled	
Aleksei Smirnov	Gastritis	Doctor D	Gastroenterologist	04.10.2024	Accepted	
Olga Kuznetsova	Migraine	Doctor E	Neurologist	05.10.2024	Waiting	
Victor Tikhonov	Arthritis	Doctor F	Rheumatologist	08.10.2024	Accepted	
Anna Sidorova	Pneumonia	Doctor G	Pulmonologist	09.10.2024	Waiting	
Maksim Nikiforov	Hypertension	Doctor H	Cardiologist	10.10.2024	Accepted	
Lidia Krylova	Diabetes	Doctor I	Endocrinologist	11.10.2024	Cancelled	
Sergey Pavlov	Lung Cancer	Doctor J	Oncologist	12.10.2024	Accepted	
Ekaterina Morozova	Osteochondrosis	Doctor K	Orthopedist	13.10.2024	Waiting	
Nikolay Fomin	Pancreatitis	Doctor L	Gastroenterologist	14.10.2024	Accepted	
Vera Lebedeva	Insomnia	Doctor M	Neurologist	15.10.2024	Waiting	
Oleg Sokolov	Allergy	Doctor N	Allergist	16.10.2024	Accepted	
Nadezhda Saveleva	Tonsillitis	Doctor O	Otolaryngologist	17.10.2024	Cancelled	
Yuriy Grishin	Kidney Disease	Doctor P	Nephrologist	18.10.2024	Accepted	
Marina Sokolova	Cholecystitis	Doctor Q	Surgeon	19.10.2024	Waiting	
Ilya Borisov	Glaucoma	Doctor R	Ophthalmologist	20.10.2024	Accepted	
Aliona Gavrilova	Obesity	Doctor S	Dietitian	21.10.2024	Cancelled	
Stanislav Popov	Epilepsy	Doctor T	Neurologist	22.10.2024	Accepted	
Elena Yakovleva	Anemia	Doctor U	Hematologist	23.10.2024	Waiting	
Dmitry Kravtsov	Chronic Stress	Doctor V	Psychologist	24.10.2024	Accepted	
Oksana Chernyshova	Varicose Veins	Doctor W	Surgeon	25.10.2024	Waiting	
Andrey Zuev	Stomach Cancer	Doctor X	Oncologist	26.10.2024	Accepted	

2.3 Разработка объектной модели ПК

Объектная модель приложения не описывает внутреннюю структуру программного комплекса в коде, а отображает основные понятия предметной области клиники в виде совокупности типов объектов (сущностей). Эти сущности выделяются из анализа требований и прецедентов, связанных с учётом пациентов, проведением приёмов, работой врачей и назначением процедур.

Сущности

На диаграмме каждая сущность изображается прямоугольником, внутри которого указываются её **название**, **атрибуты** и **операции**. Ниже представлены основные сущности для рассматриваемой предметной области:

Пациент (Patient)

- **Атрибуты:**
 - *имя*: String — имя пациента.
 - *болезнь*: String — заболевание, с которым обратился пациент.
 - *статус*: String — текущее состояние приёма (например, «Waiting», «Accepted», «Canceled»).
- **Операции:**
 - *добавить()* — добавляет информацию о новом пациенте в систему.
 - *удалить()* — удаляет сведения о пациенте из списка.
 - *редактировать()* — изменяет данные о пациенте (например, обновляет болезнь или статус).

Врач (Doctor)

- **Атрибуты:**
 - *имя*: String — имя врача.
 - *специализация*: String — специализация (терапевт, хирург и т. д.).
- **Операции:**
 - *добавить()* — добавляет нового врача в систему.
 - *удалить()* — удаляет врача из списка.
 - *редактировать()* — редактирует сведения о враче.

Приём (Appointment)

- **Атрибуты:**
 - *дата*: Date — дата приёма пациента.
 - *время*: Time — время начала приёма.
 - *врач*: String (или ссылка на объект Doctor) — врач, ведущий приём.
 - *пациент*: String (или ссылка на объект Patient) — пациент, записанный на приём.
- **Операции:**
 - *добавить()* — создаёт новый приём (запись в расписании).

- удалить() — отменяет приём.
- редактировать() — корректирует время, дату или статус приёма.

Пользователь (User)

- **Атрибуты:**
 - *имя*: String — логин пользователя (например, имя сотрудника).
 - *пароль*: String — пароль для доступа к системе.
- **Операции:**
 - регистрация() — регистрирует нового пользователя (сотрудника) в системе.
 - вход() — авторизует пользователя и предоставляет доступ к функционалу.
 - выход() — завершает текущую пользовательскую сессию.

Ассоциации между сущностями

Ассоциации описывают отношения между разными сущностями. Они обозначаются линиями на диаграмме, сопровождаются названием (семантикой) и указывают *кратность* (количество возможных связей).

- **Пациент — Приём**
 - Ассоциация: «записан на»
 - Кратность: 1 .. * со стороны пациента (один пациент может иметь много приёмов) и 1 со стороны приёма (каждый приём связан конкретно с одним пациентом).
- **Врач — Приём**
 - Ассоциация: «ведёт»
 - Кратность: 1 .. * для врача (врач может вести несколько приёмов) и 1 со стороны приёма (каждый приём ведётся конкретным врачом).
- **Пользователь — (Пациент, Врач, Приём)**
 - Пользователь (например, администратор или регистратор клиники) может управлять (добавлять, удалять, редактировать) данными о пациентах, врачах и приёмах.

Элементы диаграммы сущностей

Сущности:

- **Пациент (Patient)**
 - *имя*: String
 - *болезнь*: String
 - *статус*: String
 - Операции: добавить(), удалить(), редактировать()
- **Врач (Doctor)**
 - *имя*: String
 - *специализация*: String
 - Операции: добавить(), удалить(), редактировать()
- **Приём (Appointment)**

- *дата*: Date
- *время*: Time
- *врач*: String (или Doctor)
- *пациент*: String (или Patient)
- Операции: добавить (), удалить (), редактировать ()
- **Пользователь (User)**
 - *имя*: String
 - *пароль*: String
 - Операции: регистрация (), вход (), выход ()

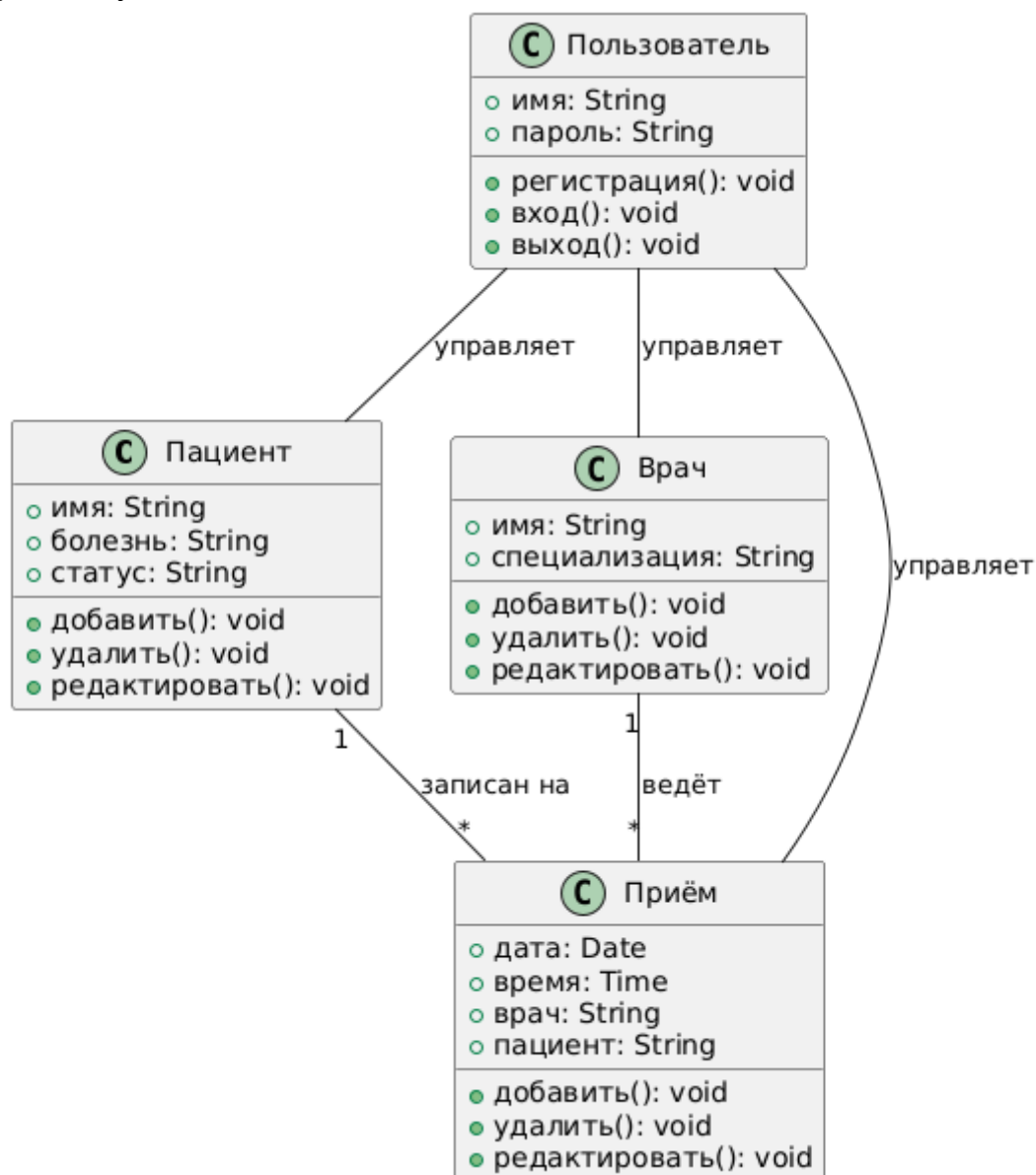
Ассоциации:

- **Пациент и Приём**: «записан на» (1 .. * / 1)
- **Врач и Приём**: «ведёт» (1 .. * / 1)
- **Пользователь и (Пациент, Врач, Приём)**: «управляет»
(означает, что пользователь создаёт, редактирует или удаляет записи)

Таким образом, объектная модель приложения отражает структуру предметной области клиники, позволяя наглядно увидеть, как основные сущности (пациент, врач, приём) и пользовательские операции (добавление, редактирование, удаление, регистрация) связаны между собой. Эта модель служит базисом для дальнейшей реализации логики в коде, создания пользовательского интерфейса и взаимодействия с хранилищами данных (например, XML-файлами).

Ниже приведён **пример** объектной модели для предметной области клиники (пациенты, врачи, приёмы, пользователи). Подобная диаграмма отражает сущности, их атрибуты и операции, а также показывает ассоциации («записан на», «ведёт» и т. д.). Обратите внимание, что это лишь одна из возможных реализаций в стиле ASCII; её можно дорабатывать или расширять при необходимости.

Диаграмма сущностей:



В данной диаграмме:

- **Пациент (Patient)** имеет атрибуты *имя*, *болезнь*, *статус* и операции для добавления, удаления, редактирования данных.
- **Приём (Appointment)** хранит *дату*, *время*, ссылки на *врача* и *пациента*.
- **Врач (Doctor)** характеризуется *именем* и *специализацией*.
- **Пользователь (User)** — сущность для управления доступом (авторизация, регистрация, завершение сеанса).

Ассоциации здесь показаны в виде пояснительных стрелок:

- Пациент «записан на» приём.
- Врач «ведёт» приём.
- Пользователь может «управлять» (или «создавать / редактировать / удалять») записями о пациентах, врачах и приёмах.

2.4 Построение диаграммы программных классов

Построение диаграммы классов для приложения управления клиникой

Диаграмма классов (*class diagram*) иллюстрирует будущие программные классы и интерфейсы на основе объектной модели клиники. Для каждого класса указываются три раздела: имя класса, состав его атрибутов и методы (операции). Графически класс отображается прямоугольником, в котором последовательно перечисляются название, поля и методы.

Структура классов

Пациент (Patient)

- Атрибуты:
 - + имя: String — имя пациента.
 - + болезнь: String — болезнь, с которой пациент обратился.
 - + статус: String — текущий статус приёма (например, «Waiting», «Accepted», «Canceled»).
- Методы:
 - + добавить(): boolean — добавляет нового пациента.
 - + удалить(): boolean — удаляет пациента из списка.
 - + редактировать(): boolean — редактирует информацию о пациенте (имя, болезнь, статус).

Врач (Doctor)

- Атрибуты:
 - + имя: String — имя врача.
 - + специализация: String — специализация врача (терапевт, хирург и т. д.).
- Методы:
 - + добавить(): boolean — добавляет нового врача.
 - + удалить(): boolean — удаляет врача из списка.
 - + редактировать(): boolean — редактирует информацию о враче (имя, специализация и т. д.).

Приём (Appointment)

- Атрибуты:
 - + дата: Date — дата приёма.
 - + время: Time — время начала приёма.
 - + врач: String — указание, какой врач ведёт приём.
 - + пациент: String — указание, какой пациент записан на приём.
- Методы:
 - + добавить(): boolean — добавляет новую запись о приёме.
 - + удалить(): boolean — удаляет приём из расписания.
 - + редактировать(): boolean — редактирует данные о приёме (дату, время, врача, пациента).

Пользователь (User)

- Атрибуты:
 - + имя: String — имя (логин) пользователя в системе.
 - + пароль: String — пароль для входа.
- Методы:
 - + регистрация(): boolean — регистрирует нового пользователя в системе.
 - + вход(): boolean — даёт возможность пользователю войти под своим логином и паролем.
 - + выход(): boolean — завершает сессию пользователя.

Связи между классами

На диаграмме классов обычно различают три основных типа отношений: ассоциация, агрегация и наследование.

1. Ассоциации

- Пациент ↔ Приём: «записан на»
 - Кратность: у пациента может быть несколько приёмов (1..*), тогда как каждый конкретный приём ссылается на одного пациента (1..1).
- Врач ↔ Приём: «ведёт»
 - Кратность: один врач ведёт несколько приёмов (1..*), но каждый приём закреплён за одним конкретным врачом (1..1).

2. Агрегация

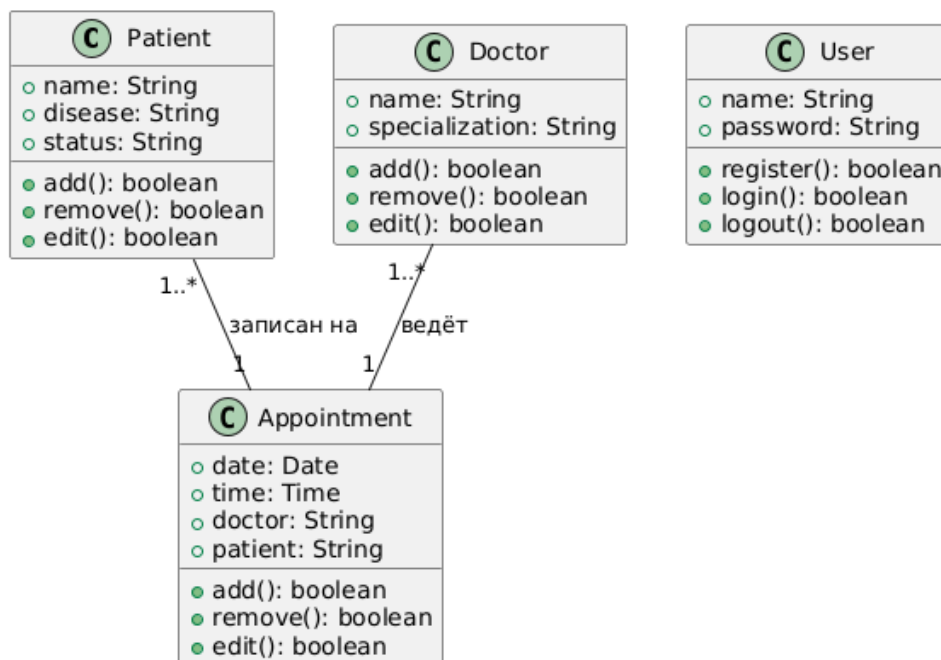
- Если, к примеру, класс «Врач» агрегирует класс «Приём» (или наоборот), то это может быть показано с помощью пустого ромба на линии связи. Это указывает, что, например, в рамках врача может вестись список нескольких приёмов.

3. Наследование

- Если в системе есть разные виды пользователей (например, «Администратор» и «Медсестра»), можно ввести классы Administrator и Nurse, которые будут наследовать базовый класс User. На диаграмме наследование обозначается стрелкой с белым треугольником, направленным от производного класса к базовому классу.

Таким образом, диаграмма классов даёт представление о том, как в приложении будут структурированы ключевые сущности (пациент, врач, приём, пользователь), какие у них есть данные (атрибуты) и какие операции они могут выполнять (методы), а также отражает взаимосвязи (ассоциации, агрегации, наследование) между классами.

Диаграмма классов:



2.5 Описание поведения ПК

Общие сведения

Поведение приложения описывает, какие действия выполняются в ходе работы системы, не углубляясь при этом в детали механизмов реализации. Одним из способов наглядно представить поведение является **диаграмма последовательностей** (*sequence diagram*). Она показывает, в каком порядке происходят взаимодействия (запросы) между пользователем и объектами системы в рамках определённого сценария.

Идентификация пользователей и объектов

1. Определите, **какие пользователи** (например, администратор, оператор регистратуры) и **какие объекты** (классы «Пациент», «Врач», «Приём», «Пользователь» и т. д.) участвуют в рассматриваемом сценарии.
2. Изобразите эти объекты в верхней части диаграммы последовательностей в виде прямоугольников, подписав внутри прямоугольника (или под ним) имя объекта с подчёркиванием и название класса, которому он принадлежит (например, `_p1 : Patient`).
3. Проведите от каждого объекта **вертикальную пунктирную линию**, обозначающую «линию жизни» (*lifeline*).

Выбор операций

- Из проектной документации (объектной модели, диаграммы классов) определите, какие **операции** выполняют данные объекты в данном сценарии (например, `добавитьПациента()`, `удалитьПациента()`, `редактироватьПриём()` и т. д.).
- Если каких-то операций, необходимых для сценария, раньше не описывали, добавьте их в модель.

Отображение запросов (сообщений)

- Каждый запрос на выполнение операции отображается **горизонтальной стрелкой**: она начинается на линии жизни (lifeline) того объекта или пользователя, который инициирует вызов, и заканчивается на линии жизни объекта-исполнителя.
- Над стрелкой указывается **номер** запроса, **название** операции и при необходимости **параметры** (например, 1.1 addPatient(name, disease)).
- Расположение стрелки по вертикали показывает момент времени вызова (чем ниже, тем позже).

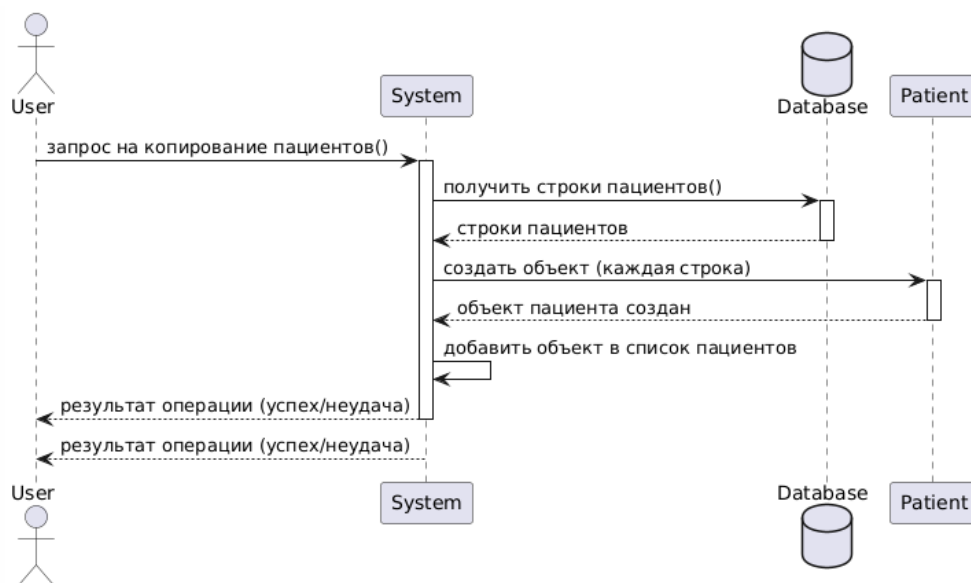
Порядок выполнения операций

- **Прямоугольник** на линии жизни объекта означает интервал выполнения конкретной операции (активность объекта).
- Для удобства используют **вложенную систему нумерации**: если операция 1 вызывает подоперации 1.1, 1.2 и т. д., то внутри 1.1 могут быть подшаги 1.1.1, 1.1.2 и так далее.
- Порядок стрелок сверху вниз отражает временную последовательность событий.

Условные ветвления и создание/уничтожение объектов

- При необходимости на диаграмме можно показать **логические условия** (if-else), когда одни операции выполняются только при выполнении определённого условия.
- Если объект **создаётся** в процессе сценария, его линия жизни начинается не сверху диаграммы, а с момента вызова конструктора (стрелка, указывающая на появление объекта).

Если объект **уничтожается** до завершения сценария, его линия жизни заканчивается символом «X» (крестик), указывающим момент уничтожения.



2.6 Построение диаграммы действий

Диаграммы действий (*activity diagrams*) применяются для наглядного описания сложных операций, таких как добавление нового пациента, редактирование информации о приёмах или копирование данных из внешнего источника. Основная цель создания таких диаграмм — **визуализировать логику** (алгоритм) выполнения операций, определяя порядок действий и варианты переходов между ними.

Графическое представление диаграммы действий напоминает блок-схему:

- Вершины (*nodes*) соответствуют **действиям** (операциям системы).
- Рёбра (*edges*) показывают **переходы** от одного действия к другому.

Каждая диаграмма действий должна иметь **единственное начальное** и **единственное конечное** состояние. Обычно её строят вертикально, чтобы поток действий шёл сверху вниз.

Основные элементы диаграммы действий

1. Начальное состояние

Обозначает, с чего начинается процесс (например, «Запрос на добавление пациента»).

2. Действия

Различные операции, такие как «Собрать данные о пациенте», «Проверить введённую информацию», «Внести пациента в базу данных» и т. д.

3. Переходы

Связывают действия между собой, показывая **последовательность** их выполнения.

4. Конечное состояние

Обозначает результат, например «Пациент успешно добавлен».

5. Параллельные процессы (если нужны)

Диаграмма действий позволяет выделять и сливать **параллельные потоки** при помощи «штанги синхронизации» (линии, разделяющей и вновь объединяющей несколько ветвей выполнения).

Пример диаграммы действий

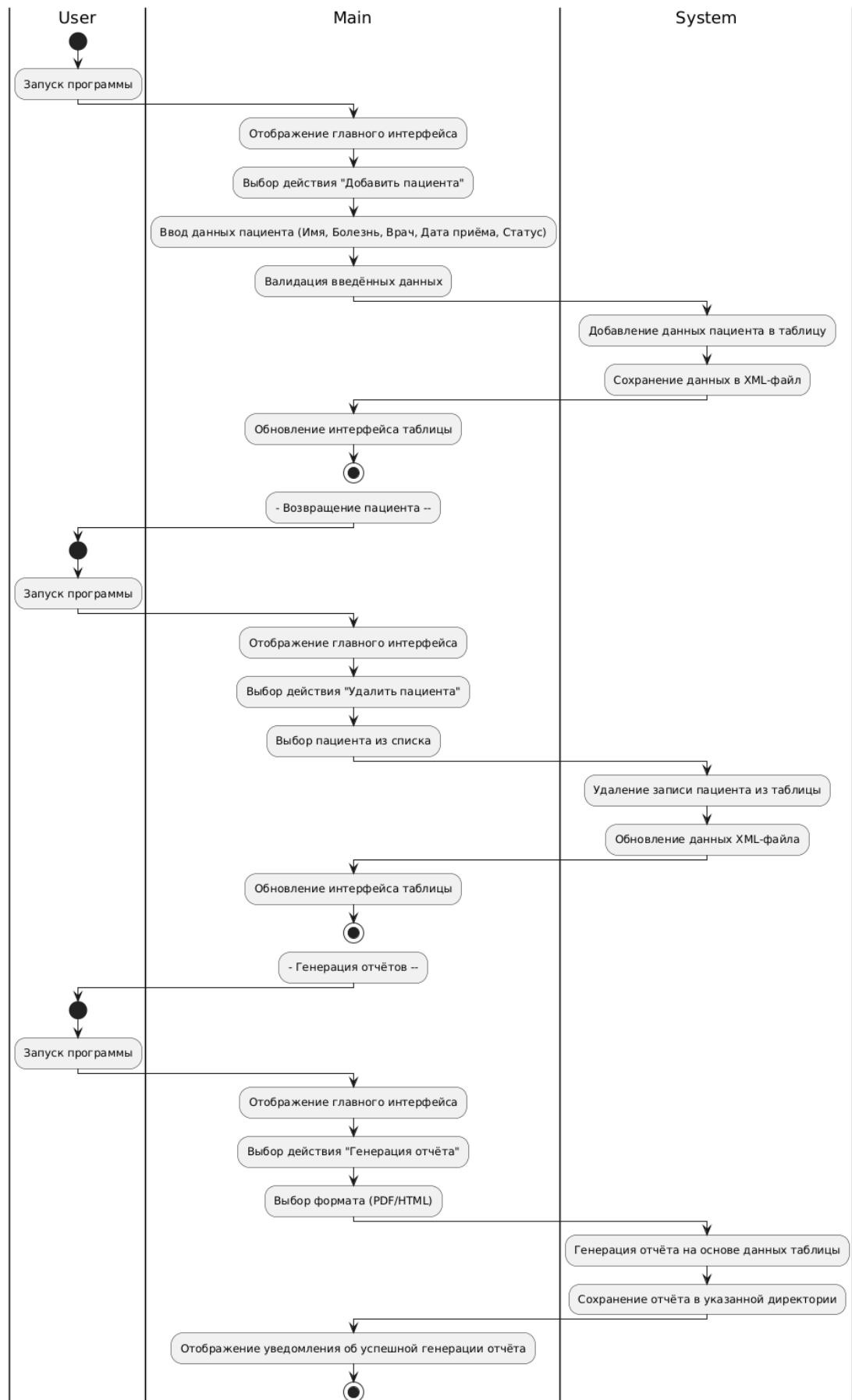
Допустим, нам нужно отобразить **процесс добавления нового пациента** в клинику:

1. **Начальное состояние:** «Запрос на добавление пациента».
2. **Действие 1:** «Собрать данные пациента» (имя, болезнь, дата приёма и т. д.).
3. **Действие 2:** «Проверить корректность данных»:
 - Если данные корректны, переходим к следующему действию.
 - Если данные некорректны, переходим к действию «Вывести сообщение об ошибке» и завершаем процесс.
4. **Действие 3:** «Добавить пациента в базу данных» (или сохранить запись в XML).

5. **Конечное состояние:** «Пациент успешно добавлен».

Таким образом, диаграмма действий в формате «начальное состояние → последовательность действий → конечное состояние» помогает **наглядно** показать, какие шаги выполняются при добавлении пациента, какова их последовательность и какие есть **разветвления** (условные переходы) в процессе.

Диаграмма:



2.7 Реализация отношений «многие ко многим»

В системе «Учёт пациентов» реализована связь «**многие ко многим**» между пациентами и врачами. Один пациент может наблюдаться у нескольких врачей, а один врач может вести несколько пациентов. Для реализации такой связи используется **связующая таблица**.

Описание сущностей и ключей

Пациенты (Patients)

- Первичный ключ (Primary Key): patient_id
- Атрибуты:
 - name: String — имя пациента
 - age: Integer — возраст пациента
 - disease: String — диагноз
 - status: String — статус пациента

Врачи (Doctors)

- Первичный ключ (Primary Key): doctor_id
- Атрибуты:
 - doc_name: String — имя врача
 - specialty: String — специализация врача

Связующая таблица (Patient_Doctor)

- Составной первичный ключ (Composite Primary Key): (patient_id, doctor_id)
- Внешний ключ (Foreign Key):
 - patient_id → Patients(patient_id)
 - doctor_id → Doctors(doctor_id)
- Атрибуты:
 - appointment_date: Date — дата приёма
 - notes: String — заметки врача

Описание схемы отношений по ключам

Пациенты (Patients)

- Каждый пациент имеет уникальный patient_id.
- Может иметь несколько записей в таблице Patient_Doctor, что означает наблюдение у нескольких врачей.

Врачи (Doctors)

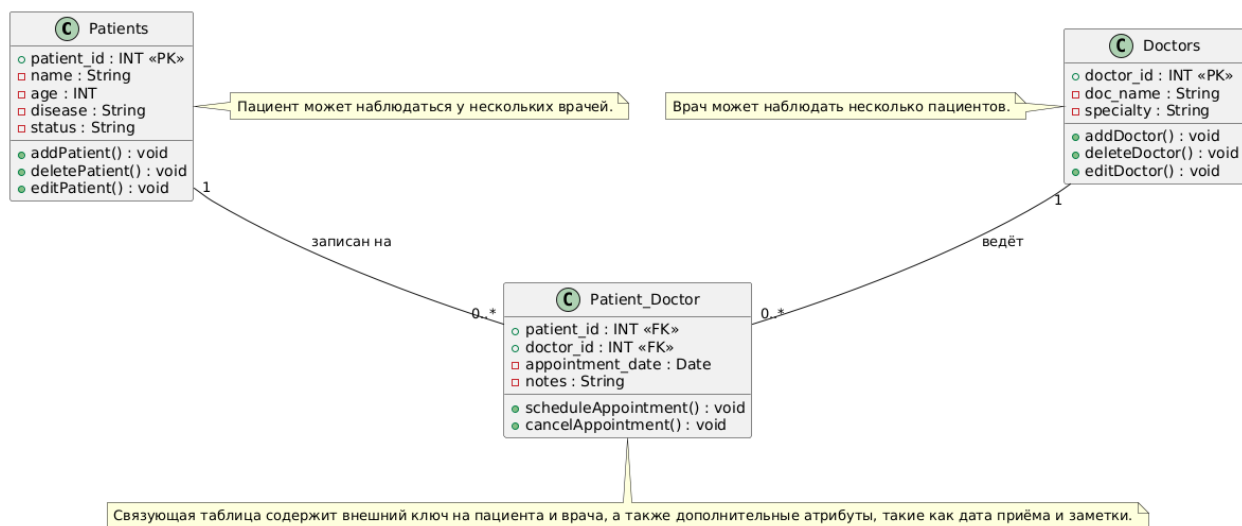
- Каждый врач имеет уникальный doctor_id.
- Может иметь несколько записей в таблице Patient_Doctor, что означает наличие нескольких пациентов.

Patient_Doctor

- Содержит составной ключ (patient_id, doctor_id).
- Содержит дополнительные атрибуты, такие как appointment_date и notes.
- Обеспечивает связь между пациентами и врачами через внешние ключи.

Преимущества явной связи по ключам

- Обеспечивает целостность данных за счёт использования внешних ключей.
- Позволяет однозначно идентифицировать каждую запись взаимодействия между пациентом и врачом.
- Легко масштабируется при добавлении новых атрибутов в таблицу Patient_Doctor.



3. Руководство оператора

3.1 Назначение программы

Программный комплекс «Учёт и администрирование данных поликлиники» предназначен для автоматизации процессов управления информацией о пациентах, врачах, расписании приёмов и статусах лечения. Программа разработана как настольное приложение с использованием **языка Java** и библиотеки **Swing** для создания графического интерфейса. Данные хранятся в **XML-файлах**, что позволяет удобно сохранять и восстанавливать записи между сеансами работы.

Основные функции программы:

- Просмотр информации о пациентах, врачах и их приёмах.
- Добавление, редактирование и удаление данных о пациентах.
- Добавление, редактирование и удаление данных о врачах.
- Назначение и редактирование приёмов с указанием даты, времени и статуса.
- Поиск информации по пациентам, врачам или параметрам приёма.
- Генерация отчётов в форматах **PDF** и **HTML**.
- Сохранение и загрузка данных в **XML-файлы**.

Программа предоставляет **интуитивно понятный интерфейс**, который позволяет администраторам и сотрудникам эффективно управлять данными и быстро получать доступ к необходимой информации.

3.2 Условия выполнения программы

Программа разработана для операционных систем Windows и требует следующих минимальных характеристик оборудования:

1. **Процессор:** Intel Core i3 или эквивалент.
2. **Оперативная память (RAM):** не менее 4 ГБ.
3. **Жесткий диск (HDD/SSD):** не менее 10 ГБ свободного пространства.
4. **Графический интерфейс:** монитор с разрешением **1366x768** и выше.
5. **Устройства ввода:** стандартная клавиатура и мышь.

Программные компоненты:

- Программа написана на **Java** с использованием библиотеки **Swing** для интерфейса.
- Данные хранятся в **XML-файлах** для обеспечения гибкости и возможности резервного копирования.
- Для экспорта отчётов используются **JasperReports**.

3.3 Описание задачи

Функциональные возможности программы:

1. Управление данными о пациентах:
 - Добавление новых записей о пациентах.

- Редактирование данных существующих пациентов.
 - Удаление записей о пациентах.
 - Поиск пациента по имени, диагнозу или дате приёма.
2. Управление данными о врачах:
 - Добавление и редактирование данных о врачах.
 - Указание специализации врача.
 - Назначение пациентов к конкретным врачам.
 3. Управление расписанием приёмов:
 - Назначение даты и времени приёма.
 - Указание статуса приёма («Ожидается», «Принят», «Отменён»).
 - Возможность редактирования данных о приёме.
 4. Генерация отчётов:
 - Экспорт данных о пациентах и врачах в **PDF** и **HTML** форматы.
 - Автоматическая генерация отчётов по приёмам.
 5. Сохранение и загрузка данных:
 - Сохранение текущих данных в **XML-файл**.
 - Загрузка данных из **XML-файла** при старте программы.

Используемые элементы языка Java:

- Инкапсуляция: для защиты данных и обеспечения доступа через публичные методы.
- Наследование: для создания логически связанных классов и упрощения кода.
- Конструкторы с параметрами: для удобного создания объектов с начальной инициализацией.
- Абстрактные классы: для выделения общих характеристик объектов.
- Полиморфизм: для унификации методов работы с объектами разных типов.
- Обработка исключений: для надёжной работы с вводом данных и файлами.
- Динамическое создание объектов: для гибкого управления данными в процессе выполнения.

Основные объекты данных:

1. Пациенты: включают имя, диагноз, дату приёма и статус.
2. Врачи: содержат информацию о специализации и назначенных пациентах.
3. Расписание приёмов: дата, время и статус приёма.

Данные хранятся в **XML-файлах**, что обеспечивает лёгкий доступ, редактирование и переносимость данных между устройствами.

3.4 Входные и выходные данные

Входные данные:

1. Пациенты:

- Имя пациента (*String*)
- Диагноз (*String*)
- Дата приёма (*Date*)
- Статус (*String*: «Ожидается», «Принят», «Отменён»)

Пример:

- Имя: Иван Иванов
- Диагноз: Простуда
- Дата приёма: 15.11.2024
- Статус: Ожидается

2. Врачи:

- Имя врача (*String*)
- Специализация (*String*)
- Назначенные пациенты (*List<Пациент>*)

Пример:

- Имя: Петров Пётр Сергеевич
- Специализация: Терапевт

3. Расписание приёмов:

- Дата (*Date*)
- Время (*Time*)
- Пациент (*String*)
- Врач (*String*)
- Статус (*String*)

Пример:

- Дата: 15.11.2024
- Время: 14:30
- Пациент: Иван Иванов
- Врач: Петров Пётр Сергеевич
- Статус: Ожидается

Выходные данные:

1. Список пациентов:

- Отображение данных о пациентах в таблице.
- Возможность фильтрации по дате, врачу или статусу.

2. Список врачей:

- Информация о врачах и их специализациях.
- Пациенты, назначенные к врачу.

3. Расписание приёмов:

- Детализированный список приёмов с указанием даты, времени и статуса.

4. Отчёты:

- Экспорт данных в **PDF** и **HTML**.
- Автоматическая генерация отчётов по пациентам и приёмам.

Пример отчёта:

- Пациент: Иван Иванов
- Врач: Петров Пётр Сергеевич
- Дата приёма: 15.11.2024
- Статус: Принят

5. Сохранение данных:

- Сохранение всех данных в **XML-файл**.
- Загрузка данных из **XML-файла** при запуске программы.

3.5 Выполнение программы

3.5.1 Первоначальный интерфейс

Клиника - Список пациентов

СохранитьДобавитьУдалить

Поиск по: Имени пациентаПоиск...

Поиск

Имя пациента	Болезнь	Врач	Специализация врача	Дата приёма	Статус
Ivan Ivanov	Acute Respiratory Viral Infection	Doctor Ambus	Therapist	01.10.2024	Accepted
Tim	Depression	Doctor B	Psychiatrist	02.10.2024	Waiting
Maria Petrova	Asthma	Doctor C	Therapist	03.10.2024	Cancelled
Aleksei Smirnov	Gastritis	Doctor D	Gastroenterologist	04.10.2024	Accepted
Olga Kuznetsova	Migraine	Doctor E	Neurologist	05.10.2024	Waiting
Victor Tikhonov	Arthritis	Doctor F	Rheumatologist	08.10.2024	Accepted
Anna Sidorova	Pneumonia	Doctor G	Pulmonologist	09.10.2024	Waiting
Maksim Nikiforov	Hypertension	Doctor H	Cardiologist	10.10.2024	Accepted
Lidia Krylova	Diabetes	Doctor I	Endocrinologist	11.10.2024	Cancelled
Sergey Pavlov	Lung Cancer	Doctor J	Oncologist	12.10.2024	Accepted
Ekaterina Morozova	Osteochondrosis	Doctor K	Orthopedist	13.10.2024	Waiting
Nikolay Fomin	Pancreatitis	Doctor L	Gastroenterologist	14.10.2024	Accepted
Vera Lebedeva	Insomnia	Doctor M	Neurologist	15.10.2024	Waiting
Oleg Sokolov	Allergy	Doctor N	Allergist	16.10.2024	Accepted
Nadezhda Saveleva	Tonsillitis	Doctor O	Otolaryngologist	17.10.2024	Cancelled
Yury Grishin	Kidney Disease	Doctor P	Nephrologist	18.10.2024	Accepted
Marina Sokolova	Cholecystitis	Doctor Q	Surgeon	19.10.2024	Waiting
Ilya Borisov	Glaucoma	Doctor R	Ophthalmologist	20.10.2024	Accepted
Alyona Gavrilova	Obesity	Doctor S	Dietitian	21.10.2024	Cancelled
Stanislav Popov	Epilepsy	Doctor T	Neurologist	22.10.2024	Accepted
Elena Yakovleva	Anemia	Doctor U	Hematologist	23.10.2024	Waiting
Dmitry Kravsov	Chronic Stress	Doctor V	Psychologist	24.10.2024	Accepted
Oksana Chernyshova	Varicose Veins	Doctor W	Surgeon	25.10.2024	Waiting
Andrey Zuev	Stomach Cancer	Doctor X	Oncologist	26.10.2024	Accepted

По имени

Поток

3.5.2 Добавление пациента

СохранитьДобавитьУдалить

Имя пациента	Болезнь	Врач
Ivan Ivanov		Ambus
Tim		B
Maria Petrova		C
Aleksei Smirnov		D
Olga Kuznetsova		E
Victor Tikhonov		F
Anna Sidorova		G
Maksim Nikiforov		H
Lidia Krylova	Diabetes	Doctor I
Sergey Pavlov	Lung Cancer	Doctor J

Input

Введите имя пациента:

OKCancel

3.5.3 Поиск

Поиск по: Имени пациентаПоиск...

Поиск

Имени пациентаИмени врачаНазванию болезни

Имя пациента	Болезнь	Врач	Специализация врача	Дата приёма	Статус
Ivan Ivanov	Acute Respiratory Viral Infection	Doctor Ambus	Therapist	01.10.2024	Accepted
Tim	Depression	Doctor B	Psychiatrist	02.10.2024	Waiting
Maria Petrova	Asthma	Doctor C	Therapist	03.10.2024	Cancelled

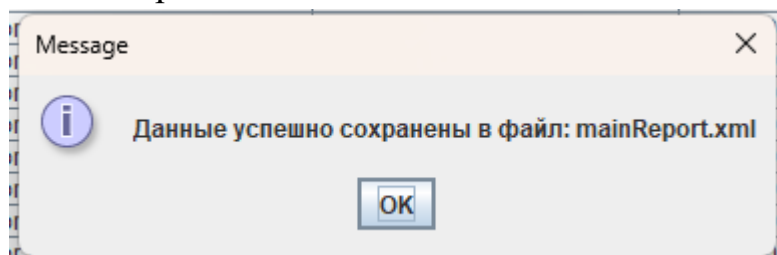
СохранитьДобавитьУдалить

Поиск по: Имени пациентаTim

3.5.4 Удаление (до/после)

Oksana Chernyshova	Varicose Veins	Doctor W	Surgeon	25.10.2024	Waiting
Andrey Zuev	Stomach Cancer	Doctor X	Oncologist	26.10.2024	Accepted
Dmitry Kravsov	Chronic Stress	Doctor V	Psychologist	24.10.2024	Accepted
Oksana Chernyshova	Varicose Veins	Doctor W	Surgeon	25.10.2024	Waiting

3.5.5 Сохранение



3.5.6 сортировка по имени (до/после)

Имя пациента ▲	
Aleksei Smirnov	G
Alyona Gavrilova	O
Anna Sidorova	Pi
Dmitry Kravtsov	C
Ekaterina Morozova	O
Elena Yakovleva	Ar
Ilya Borisov	G
Ivan Ivanov	Ac
Lidia Krylova	D
Maksim Nikiforov	H
Maria Petrova	As
Marina Sokolova	C
Nadezhda Saveleva	T
Nikolay Fomin	P
Oksana Chernyshova	V
Oleg Sokolov	Al
Olga Kuznetsova	M
Sergey Pavlov	L
Stanislav Popov	E
Tim	D
Vera Lebedeva	In
Victor Tikhonov	Ar
Yuriy Grishin	Ki

Имя пациента ▼	
Yuriy Grishin	P
Victor Tikhonov	A
Vera Lebedeva	I
Tim	C
Stanislav Popov	E
Sergey Pavlov	L
Olga Kuznetsova	M
Oleg Sokolov	A
Oksana Chernyshova	V
Nikolay Fomin	F
Nadezhda Saveleva	T
Marina Sokolova	C
Maria Petrova	A
Maksim Nikiforov	F
Lidia Krylova	C
Ivan Ivanov	A
Ilya Borisov	C
Elena Yakovleva	A
Ekaterina Morozova	C
Dmitry Kravtsov	C
Anna Sidorova	F
Alyona Gavrilova	C
Aleksei Smirnov	C

Заключение

Итоги разработки

В результате проделанной работы был разработан программный комплекс (ПК) для управления данными клиники, предназначенный для администрирования и учёта информации о пациентах, врачах и расписании приёмов. Также было составлено руководство оператора, детально описывающее функциональные возможности программы и взаимодействие пользователя с интерфейсом.

Основные этапы проектирования

1. Описание вариантов использования:
Были определены ключевые сценарии взаимодействия пользователей с системой, такие как добавление, редактирование, удаление данных, а также генерация отчётов и сохранение данных в XML-файлы.
2. Прототип интерфейса пользователя:
Разработан графический интерфейс с удобными элементами управления (кнопки, таблицы, диалоговые окна) для эффективного взаимодействия с системой.
3. Объектная модель и диаграммы:
Создана объектная модель приложения, включающая сущности пациентов, врачей и приёмов, а также разработаны диаграммы классов и диаграммы действий, описывающие структуру и логику работы приложения.
4. Описание поведения системы:
Были проработаны сценарии использования и их реализация в системе, что позволило эффективно распределить ответственность между компонентами приложения.

Функциональные возможности ПК

- Добавление, удаление и редактирование данных о пациентах, врачах и расписании приёмов.
- Поиск и фильтрация данных по ключевым словам и критериям.
- Генерация отчётов в форматах PDF и HTML.
- Сохранение и загрузка данных в XML-файлы для обеспечения целостности и резервного копирования информации.
- Логирование действий пользователя для упрощения диагностики и контроля работы системы.

Соответствие требованиям

Курсовой проект удовлетворяет заявленным требованиям, предоставляя:

- Интуитивно понятный интерфейс.
- Стабильную работу при обработке данных.
- Гибкость при добавлении и обновлении данных.

Перспективы развития

1. Интеграция с базами данных: Переход на полноценную СУБД (например, MySQL) для хранения и управления данными.
2. Статистические отчёты: Добавление аналитических инструментов для построения графиков и диаграмм.
3. Улучшение интерфейса: Переход на JavaFX для создания более современного и удобного GUI.
4. Облачная синхронизация: Возможность удалённого доступа к данным через облачные технологии.

Заключительные выводы

Разработанный программный комплекс для управления клиникой успешно решает поставленные задачи и демонстрирует:

- Эффективное управление данными о пациентах и врачах.
- Гибкость и надёжность в работе с информацией.
- Простоту в освоении благодаря интуитивно понятному интерфейсу.

Работа над проектом позволила развить навыки в области:

- Проектирования и реализации объектно-ориентированных систем.
- Применения многопоточности для оптимизации производительности.
- Обработки данных в формате XML.

Полученные знания и опыт могут быть использованы для разработки более сложных и масштабируемых приложений в будущем.

Ссылки:

>> репозиторий:

<https://github.com/iconLti/LTprojects/tree/master/OOP/Java%20projects/CourseWork>

