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 МОЭВМ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ по дисциплине «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»
«Создание программного комплекса средствами объектно-ориентированного программирования»

Выполнил: Сапронов К.Д.

Факультет КТИ

Группа №3311

Подпись преподавателя: _____

Санкт-Петербург 2024

Введение

Данная курсовая работа посвящена разработке программного комплекса (ПК) для администратора гостиницы. Программа предоставляет удобный интерфейс для управления информацией о номерах, клиентах, работниках и ценах. Основной целью является упрощение работы администратора, автоматизация процессов внесения данных и получения аналитической информации.

Основание для разработки

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

Техническое задание

Задание 10. Разработать ПК для администратора гостиницы. В ПК должна храниться информация о проживающих клиентах, номерах гостиницы и служащих гостиницы. Администратор гостиницы может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

- список работников гостиницы;
- список свободных номеров с указанием его вместимости;
- прейскурант цен.
- справка о жильцах гостиницы (ФИО, срок проживания, номер);
- отчет о работе гостиницы за месяц (число клиентов, сколько дней был занят и свободен номер).

Технические возможности ПК:

- Управление данными о номерах, включая возможность добавления, редактирования и удаления информации.
- Управление данными о сотрудниках.
- Управление данными о ценах за разные типы номеров.
- Управление данными клиентов.
- Получение отчёта о работе гостиницы за месяц.
- Экспорт данных в формат PDF и HTML.
- Поддержка работы с файлами XML для сохранения и загрузки данных.

Требования к программе

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

- закрытые и открытые члены классов;

- наследование;
- конструкторы с параметрами;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

Описание Программного Комплекса:

Программный комплекс реализован с использованием языка программирования Java и представляет собой приложение с графическим интерфейсом. Для работы с таблицами используются компоненты `JTable` и `DefaultTableModel`, что позволяет эффективно управлять данными. Основные функции включают работу с таблицами, многопоточность для обработки больших объемов данных, а также экспорт данных в различные форматы. Интерфейс пользователя включает кнопки для добавления, редактирования, удаления данных, а также инструменты для фильтрации и сортировки записей.

Проектирование ПК

На этапе проектирования приложения для управления гостиницей необходимо детализировать функциональные требования, выделив ключевые процессы, происходящие в данной предметной области. Эти процессы можно описать с помощью прецедентов, которые представляют собой сценарии взаимодействия пользователя с приложением.

Прецеденты

- Добавить номер: Пользователь может вручную добавить новые номера в список
- Удалить номер: Пользователь может удалить выбранный номер из списка.
- Сохранить данные: Пользователь может сохранить изменения в списке номеров в XML-файл.
- Загрузить данные: Пользователь может загрузить список номеров из XML-файла.
- Поиск: Пользователь может выполнить поиск в списке.
- Регистрация пользователя: Новый пользователь может зарегистрироваться в
- системе для доступа к функциональности приложения.
- Генерация отчетов: Пользователь может сгенерировать отчеты в формате
- PDF или HTML.

Акторы

Акторы, взаимодействующие с приложением:

- Пользователь: Основной актор, который работает с данными (добавляет, удаляет или редактирует их)
- Администратор: Актор, который может управлять пользователями и их правами доступа.

Связи между акторами и прецедентами

На диаграмме прецедентов пользователь инициирует все основные операции с данными, администратор же управляет правами пользователя и процессом регистрации

- Связь использования: Если один прецедент всегда использует функциональность другого, это можно отразить с помощью связи использования. Например, прецедент "Сохранить данные" может использовать прецедент "Загрузить данные", если сохранение требует предварительной загрузки данных.
- Связь расширения: Если один прецедент может быть расширен другим, это также следует отразить. Например, прецедент "Поиск" может быть расширен прецедентом "Фильтрация по критериям", если пользователь хочет уточнить поиск.

Ранжирование прецедентов

Прецеденты необходимо ранжировать по приоритету, чтобы в начальных циклах разработки реализовать наиболее важные функции. Например, прецеденты "Добавить номер", "Удалить номер", “Сохранить в файл” и “Загрузить из файла” могут быть приоритетными, так как они составляют основную функциональность приложения.

Элементы диаграммы прецедентов

Акторы: Пользователь (взаимодействует с приложением), администратор (управляет пользователями и их правами доступа)

Прецеденты: добавить номер, удалить номер, сохранить данные, загрузить данные, поиск, регистрация пользователя, генерация отчетов

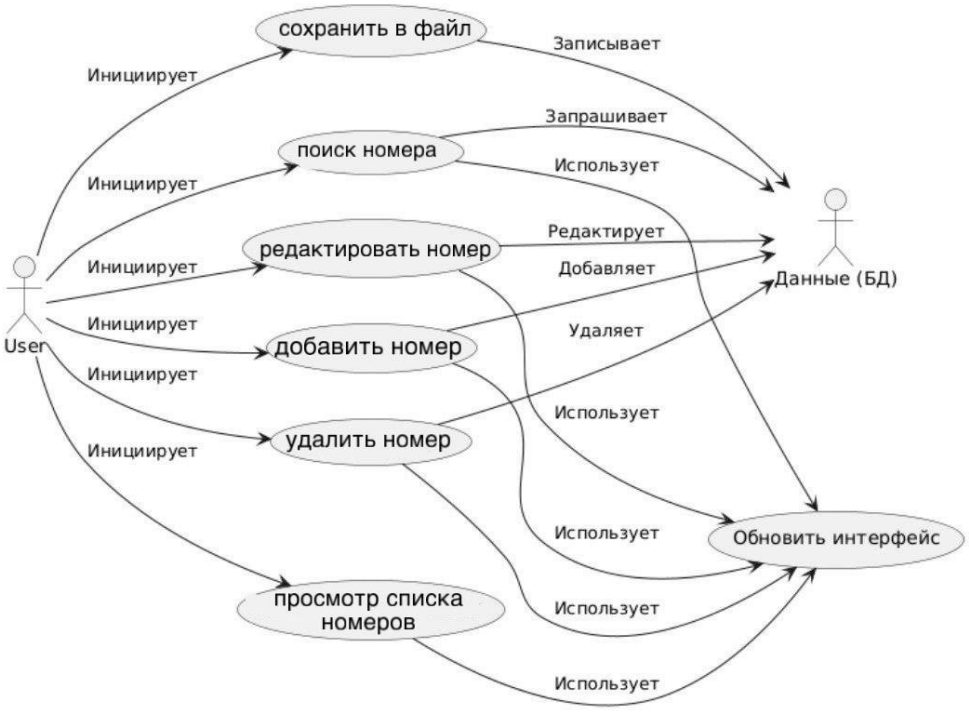
Связи:

Связи между акторами и прецедентами изображаются стрелками,

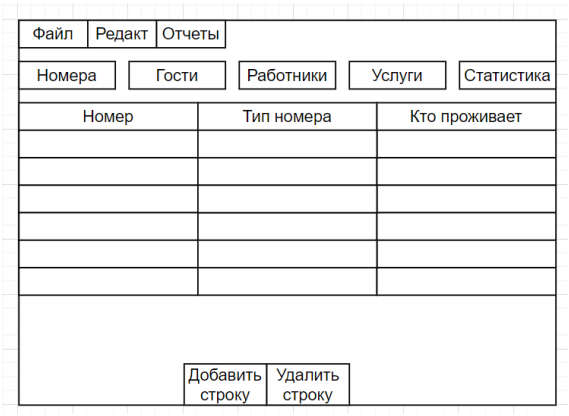
указывающими, кто инициирует взаимодействие, а также могут использоваться связи расширения или использования, если один прецедент опирается на функционал другого.

Описание диаграммы

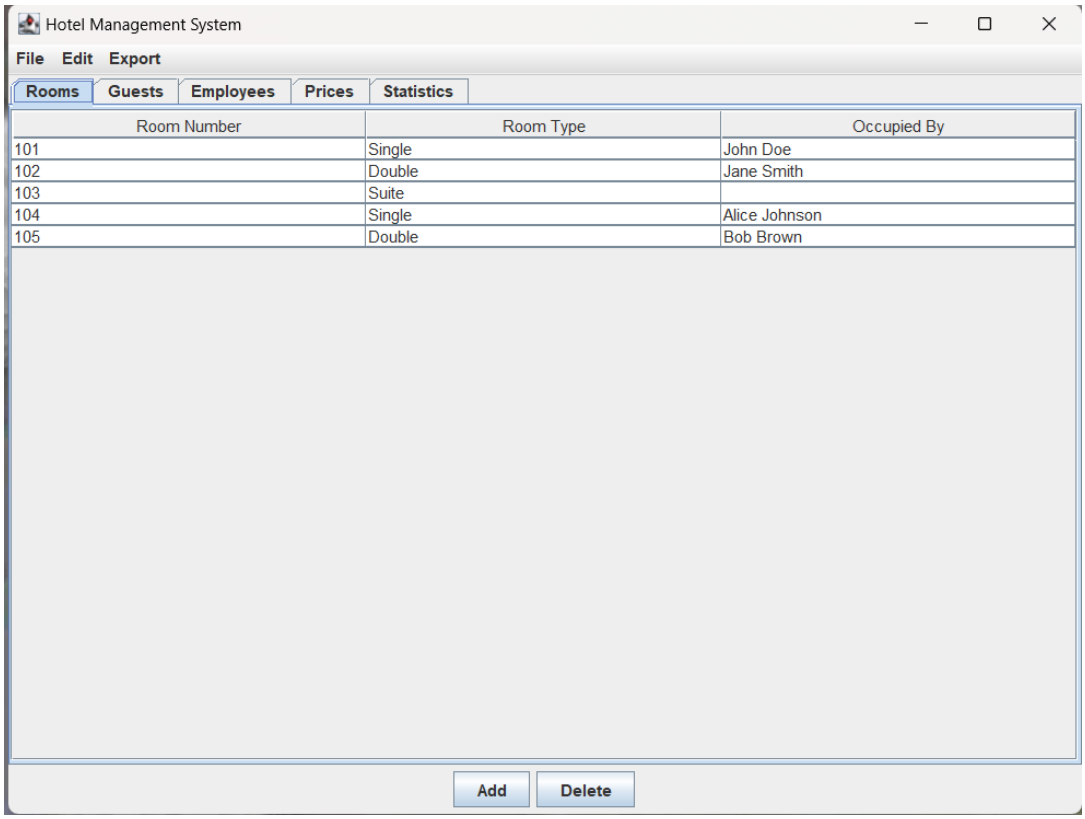
Пользователь связан со всеми прецедентами, так как он инициирует все действия.
Администратор может быть связан с прецедентами, связанными с управлением пользователями (например, регистрация пользователя).



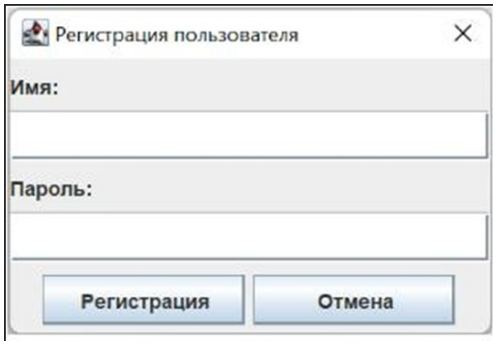
Создание прототипа интерфейса пользователя
Описание прецедента выражает общую сущность процесса без детализации его реализации. Проектные решения, связанные с интерфейсом пользователя, при этом опускаются. Для разработки пользовательского интерфейса необходимо описать процесс в терминах реальных проектных 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.
Создание прототипа интерфейса пользователя.
Главное окно:



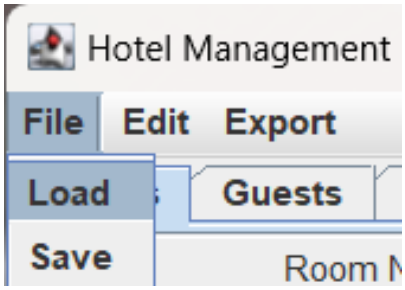
Готовый результат:

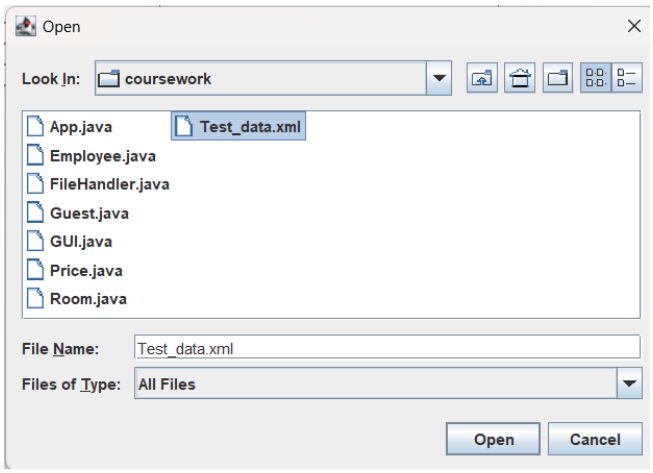


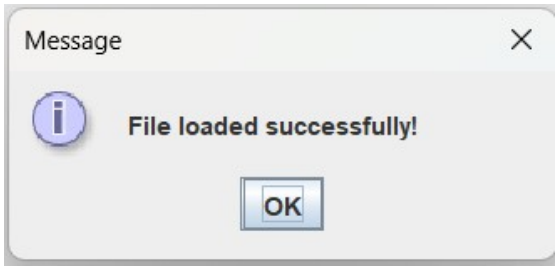
Регистрация пользователя

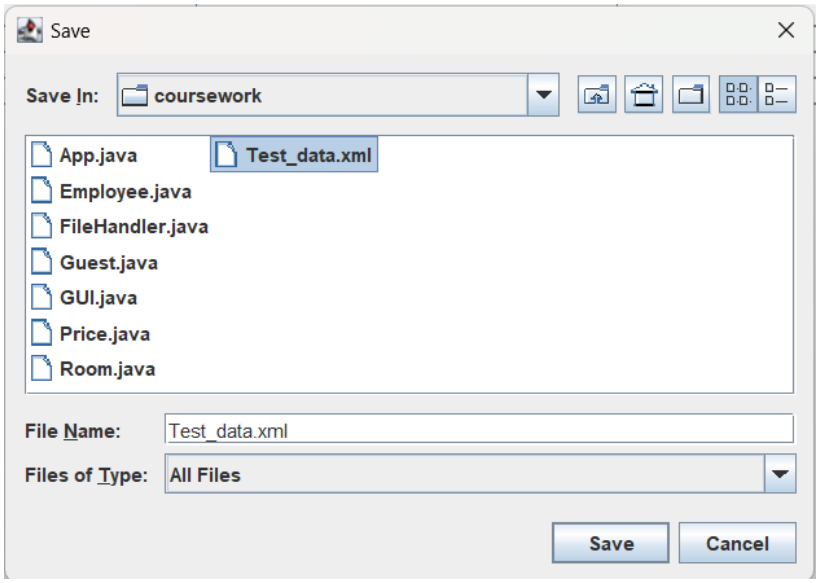


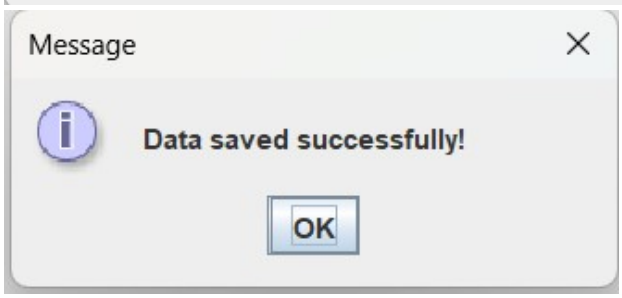
Сохранение и загрузка данных



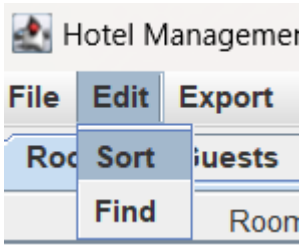


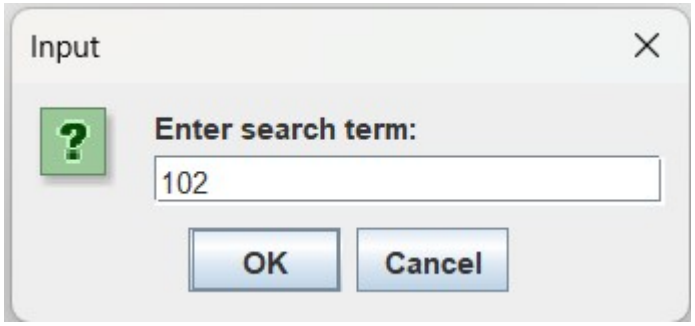


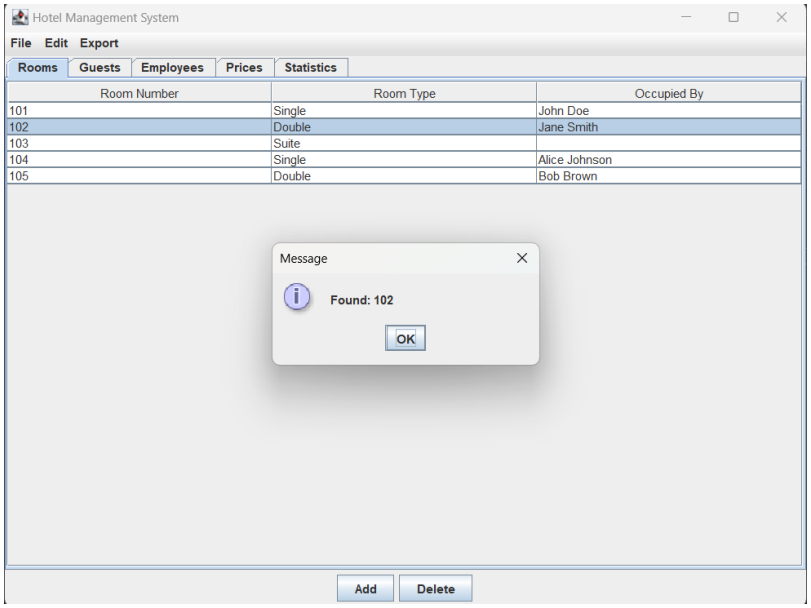




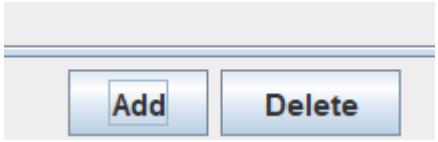
Сортировка и поиск

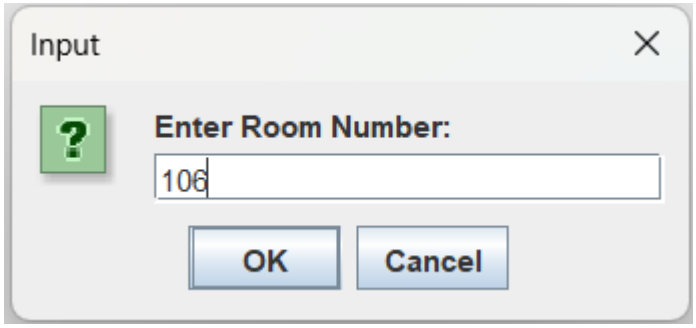


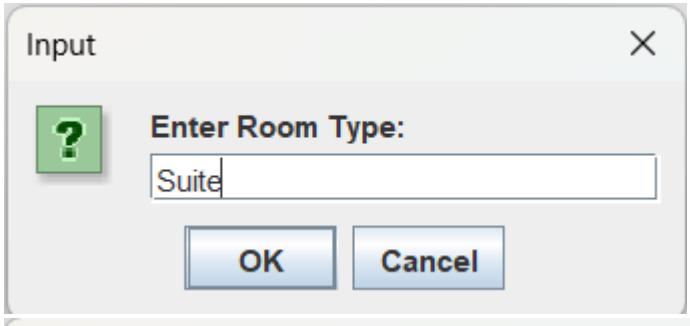


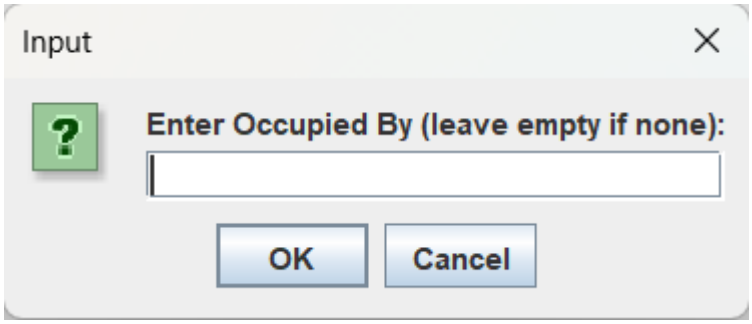


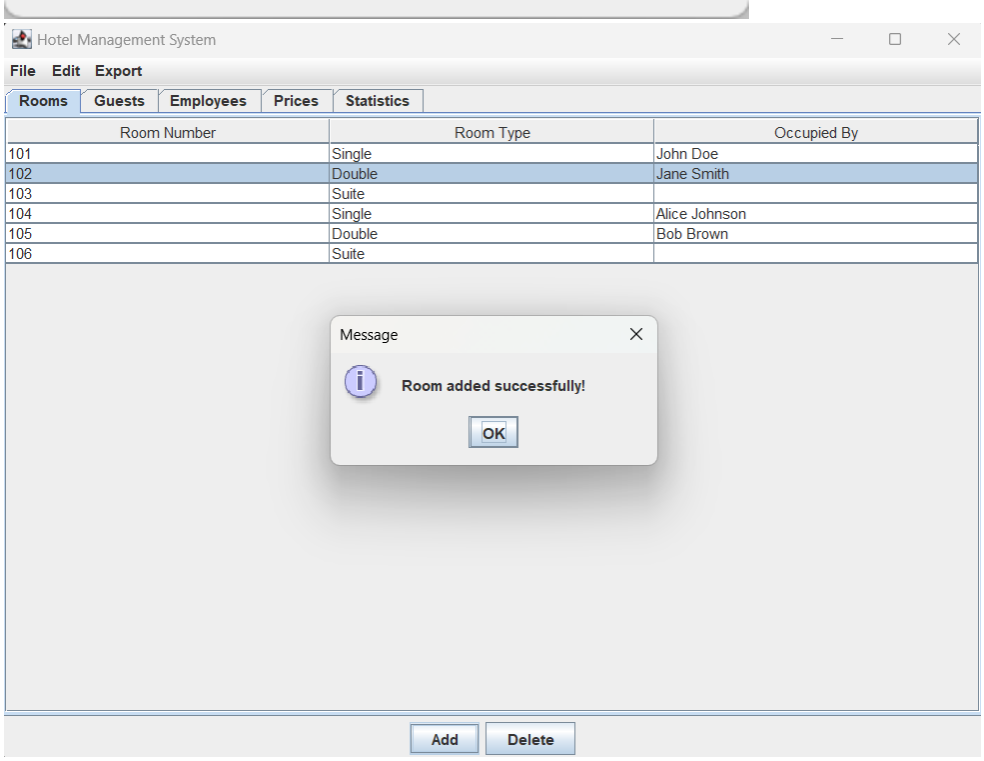
Добавление и удаление строки



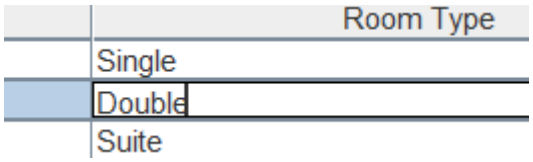




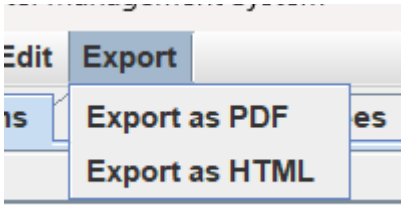




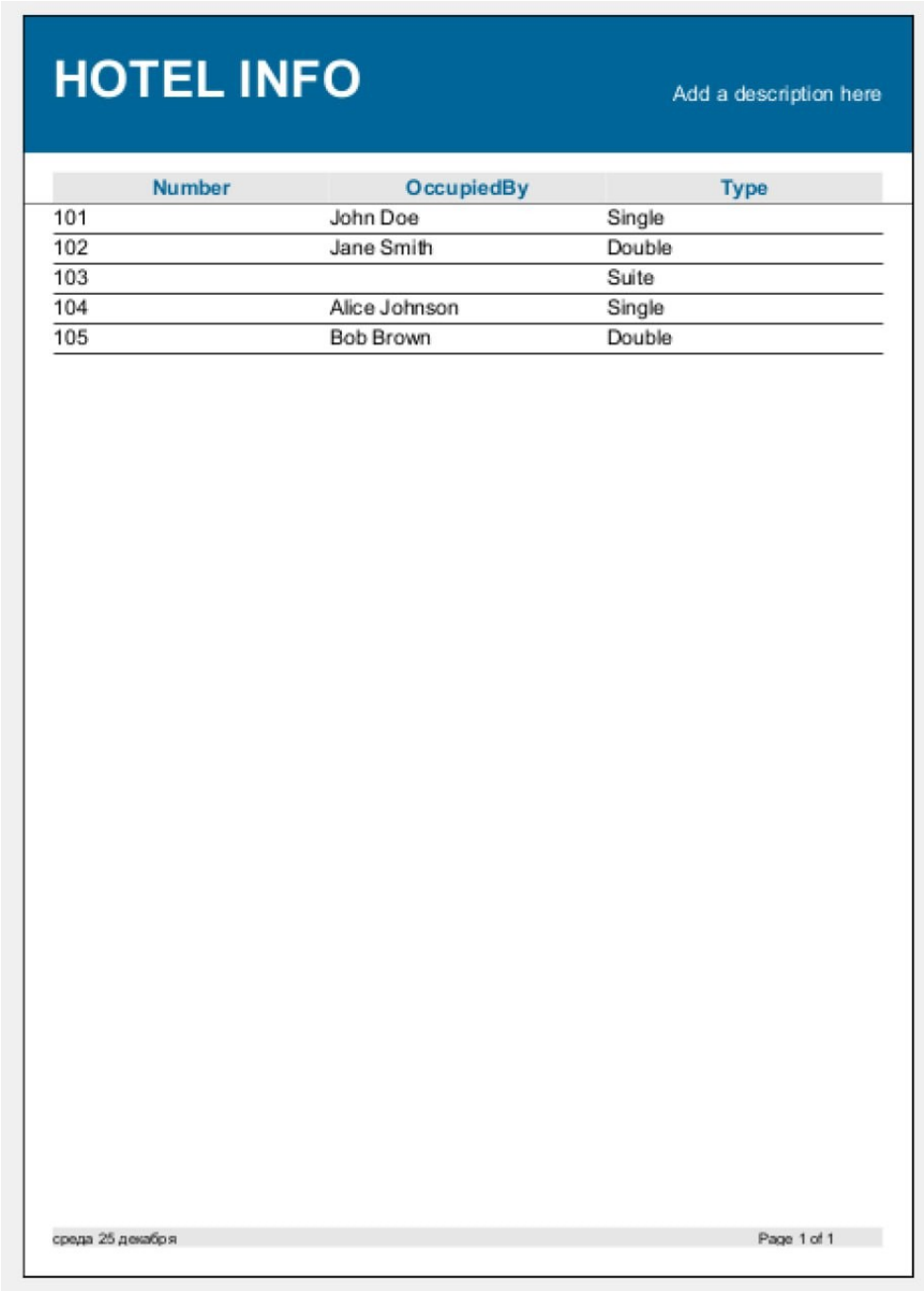
Редактирование данных



Создание отчета



Пример отчёта:



HOTEL INFO

Add a description here

Number	OccupiedBy	Type
101	John Doe	Single
102	Jane Smith	Double
103		Suite
104	Alice Johnson	Single
105	Bob Brown	Double

среда 25 декабря Page 1 of 1

Разработка объектной модели приложения для управления списком номеров

Объектная модель приложения не описывает структуру программного комплекса, а отображает основные понятия предметной области в виде совокупности типов объектов (сущностей). Сущности строятся путем выделения их из предметной области и анализа прецедентов, связанных с управлением данными о номерах, гостях, сотрудниках и услугах.

Сущности

На диаграмме сущность обозначается прямоугольником, внутри которого

записывается имя сущности, ее атрибуты и операции. В приложении могут быть выделены следующие сущности:

1. Номер (Room)

Атрибуты:

• номер: int — номер комнаты.

• гость: String — кем занят номер.

• тип: String – тип номера.

Операции:

• добавить(): добавляет новый номер в систему.

• удалить(): удаляет номер из списка.

• редактировать(): редактирует информацию о номере.

2. Гость (Guest)

Атрибуты:

• имя: String — ФИО гостя.

• номер: int — номер комнаты.

• время нахождения: int – сколько ночей гость проведет в отеле.

Операции:

• добавить(): добавляет нового гостя.

• удалить(): удаляет гостя из списка.

• редактировать(): редактирует информацию о госте.

3. Услуга (Service)

Атрибуты:

• название: String — название услуги (тип номера).

• цена: int — цена услуги.

Операции:

• добавить(): добавляет новую услугу.

• удалить(): удаляет услугу из списка.

• редактировать(): редактирует информацию об услуге.

4. Работник (Staff)

Атрибуты:

• имя: String — имя работника.

• должность: position — должность работника.

Операции:

• добавить(): добавляет нового работника.

• удалить(): удаляет работника из списка.

• редактировать(): редактирует информацию о работнике.

5. Пользователь (User)

Атрибуты:

• имя: String — имя пользователя.

• пароль: String — пароль для доступа к системе.

Операции:

• регистрация(): регистрирует нового пользователя.

• вход(): позволяет пользователю войти в систему.

• выход(): завершает сессию пользователя.

Ассоциации между сущностями

Ассоциация между сущностями отражает бинарные отношения между ними.

Ассоциации обозначаются линиями, соединяющими сущности, с указанием их семантического смысла. В приложении могут быть следующие ассоциации:

Гость и номер

• Ассоциация: "проживает в".

• Кратность: 1 – 0..* (каждый гость может жить только в одном номере одновременно, но в одном номере может жить несколько человек или не жить никто).

Номер и тип услуги:

• Ассоциация: "предлагает".

• Кратность: 1 – 1..* (каждый номер предлагает только один вид услуги, но услуга может быть предложена разными номерами).

Тип услуги и работник:

• Ассоциация: "выполняется".

• Кратность: 1..* - 1..*(в предоставлении услуги могут участвовать несколько работников, и каждый работник участвует в предоставлении нескольких услуг).

Элементы диаграммы сущностей

Сущности:

Номер (Room)

• Атрибуты:

• номер: int — номер комнаты.

• гость: String — кем занят номер.

• тип: String – тип номера.

• Операции:

• добавить(): добавляет новый номер в систему.

• удалить(): удаляет номер из списка.

• редактировать(): редактирует информацию о номере.

2. Гость (Guest)

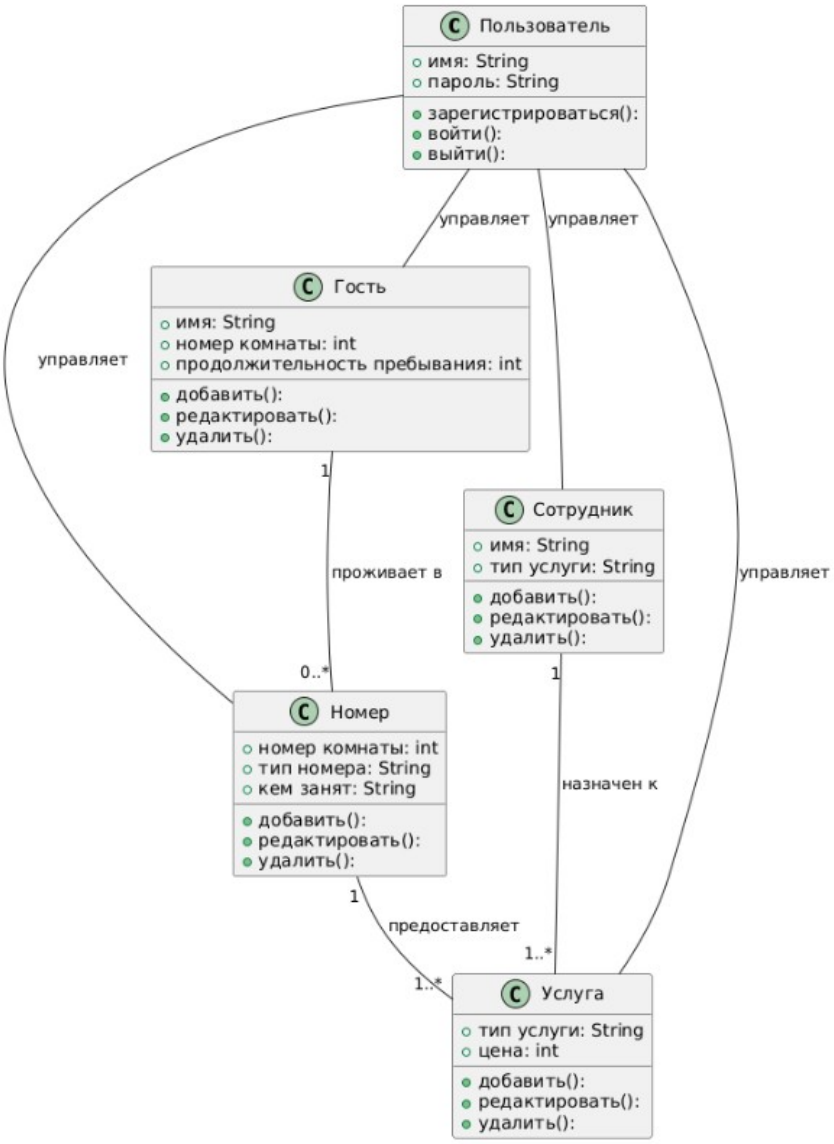
• Атрибуты:

• имя: String — ФИО гостя.

• номер: int — номер комнаты.

• время нахождения: int – сколько ночей гость проведет в отеле.

Операции:
• добавить(): добавляет нового гостя.
• удалить(): удаляет гостя из списка.
• редактировать(): редактирует информацию о госте.
3. Услуга (Service)
Атрибуты:
• название: String — название услуги (тип номера).
• цена: int — цена услуги.
Операции:
• добавить(): добавляет новую услугу.
• удалить(): удаляет услугу из списка.
• редактировать(): редактирует информацию об услуге.
4. Работник (Staff)
Атрибуты:
• имя: String — имя работника.
• должность: position — должность работника.
Операции:
• добавить(): добавляет нового работника.
• удалить(): удаляет работника из списка.
• редактировать(): редактирует информацию о работнике.
5. Пользователь (User)
Атрибуты:
• имя: String — имя пользователя.
• пароль: String — пароль для доступа к системе.
Операции:
• регистрация(): регистрирует нового пользователя.
• вход(): позволяет пользователю войти в систему.
• выход(): завершает сессию пользователя.
Ассоциации:
• Гость и Номер: "проживает в" (1 .. 1)
• Номер и тип услуги: "предлагает" (1 .. 1)
• Работник и тип услуги: "выполняет" (* .. 0)



Построение диаграммы классов

Диаграмма классов иллюстрирует будущие программные классы и интерфейсы на основе объектной модели гостиницы. Для каждого класса указываются три раздела: имя класса, состав его атрибутов и методы. Графически класс отображается прямоугольником, в котором последовательно перечисляются названия, поля и методы

1. Номер (Room)

Атрибуты:

• номер: int — номер комнаты.

• гость: String — кем занят номер.

• тип: String – тип номера.

Операции:

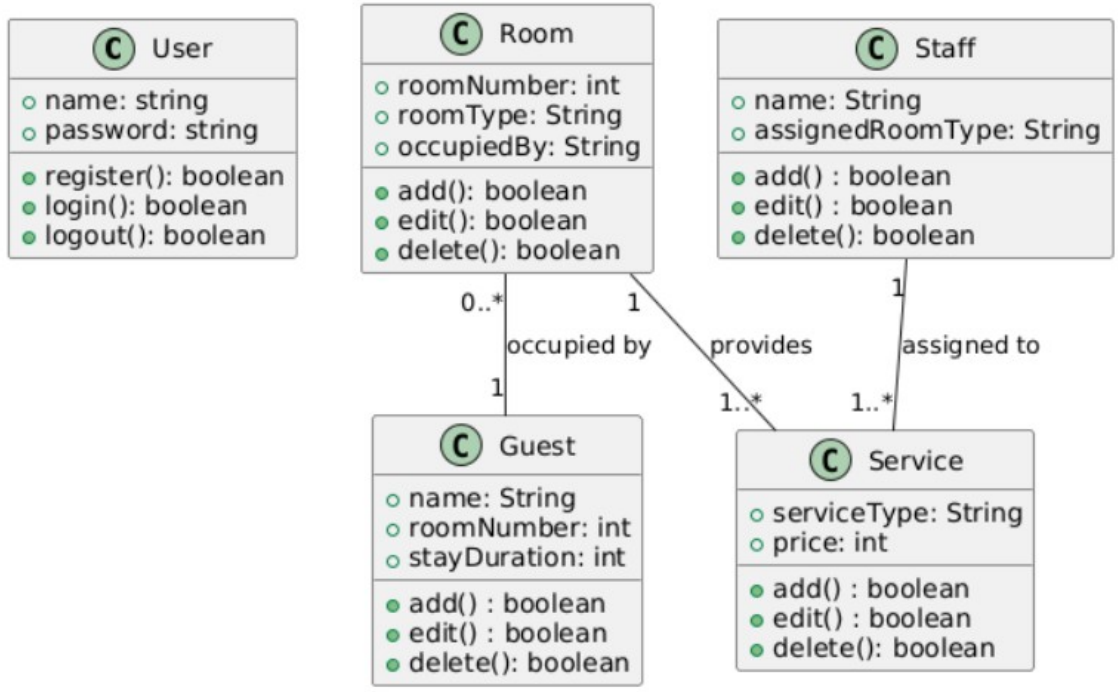
• добавить(): добавляет новый номер в систему.

• удалить(): удаляет номер из списка.

• редактировать(): редактирует информацию о номере.
2. Гость (Guest)
Атрибуты:
• имя: String — ФИО гостя.
• номер: int — номер комнаты.
• время нахождения: int – сколько ночей гость проведет в отеле.
Операции:
• добавить(): добавляет нового гостя.
• удалить(): удаляет гостя из списка.
• редактировать(): редактирует информацию о госте.
3. Услуга (Service)
Атрибуты:
• название: String — название услуги (тип номера).
• цена: int — цена услуги.
Операции:
• добавить(): добавляет новую услугу.
• удалить(): удаляет услугу из списка.
• редактировать(): редактирует информацию об услуге.
4. Работник (Staff)
Атрибуты:
• имя: String — имя работника.
• должность: position — должность работника.
Операции:
• добавить(): добавляет нового работника.
• удалить(): удаляет работника из списка.
• редактировать(): редактирует информацию о работнике.
5. Пользователь (User)
Атрибуты:
• имя: String — имя пользователя.
• пароль: String — пароль для доступа к системе.
Операции:
• регистрация(): регистрирует нового пользователя.
• вход(): позволяет пользователю войти в систему.
• выход(): завершает сессию пользователя.
Агрегация:
Если, например, класс Гость агрегирует класс Номер, это
может быть изображено полым ромбом на линии ассоциации,
указывая, что гость может жить сразу в нескольких номерах.
Наследование:

Если, например, класс Пользователь может быть базовым классом для классов Администратор и Обычный пользователь, это будет изображено стрелкой с полым треугольником, указывающей от производного класса к базовому.

Диаграмма классов



Описание поведения приложения.

Поведение приложения представляет собой описание того, какие действия выполняет система, без определения механизма их реализации. Одной из составляющих такого описания является диаграмма последовательностей. Диаграмма последовательностей является схемой, которая для определенного сценария прецедента показывает генерируемые пользователями и объектами события (запросы) на выполнение некоторой операции и их порядок.

Диаграммы последовательностей имеют две размерности: вертикальная представляет время, горизонтальная — различные объекты. Чтобы построить диаграмму последовательностей, необходимо
выполнить следующие действия:

1. **Идентификация пользователей и объектов:**

Определить пользователей (например, администратор, обычный

пользователь) и объекты программных классов (например, Номер, Гость, Сотрудник, Услуга, Пользователь), участвующие в начальной стадии реализации сценария прецедента. Изображения этих объектов в виде прямоугольников расположить наверху в одну линию. Для каждого пользователя и объекта нарисовать вертикальную пунктирную линию, которая будет представлять их линию жизни. Внутри прямоугольника указать подчеркнутое имя объекта и имя класса, к которому принадлежит объект.

2. **Выбор операций:**

Из объектной модели выбрать те операции, которые участвуют в реализации сценария. Например, операции добавления, удаления и редактирования объектов. Если такие операции не были определены при построении диаграммы классов, то необходимо их описать и внести в модель.

3. **Отображение запросов на выполнение операций:**

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 укажите номер операции, число итераций, имя операции и в скобках ее параметры.

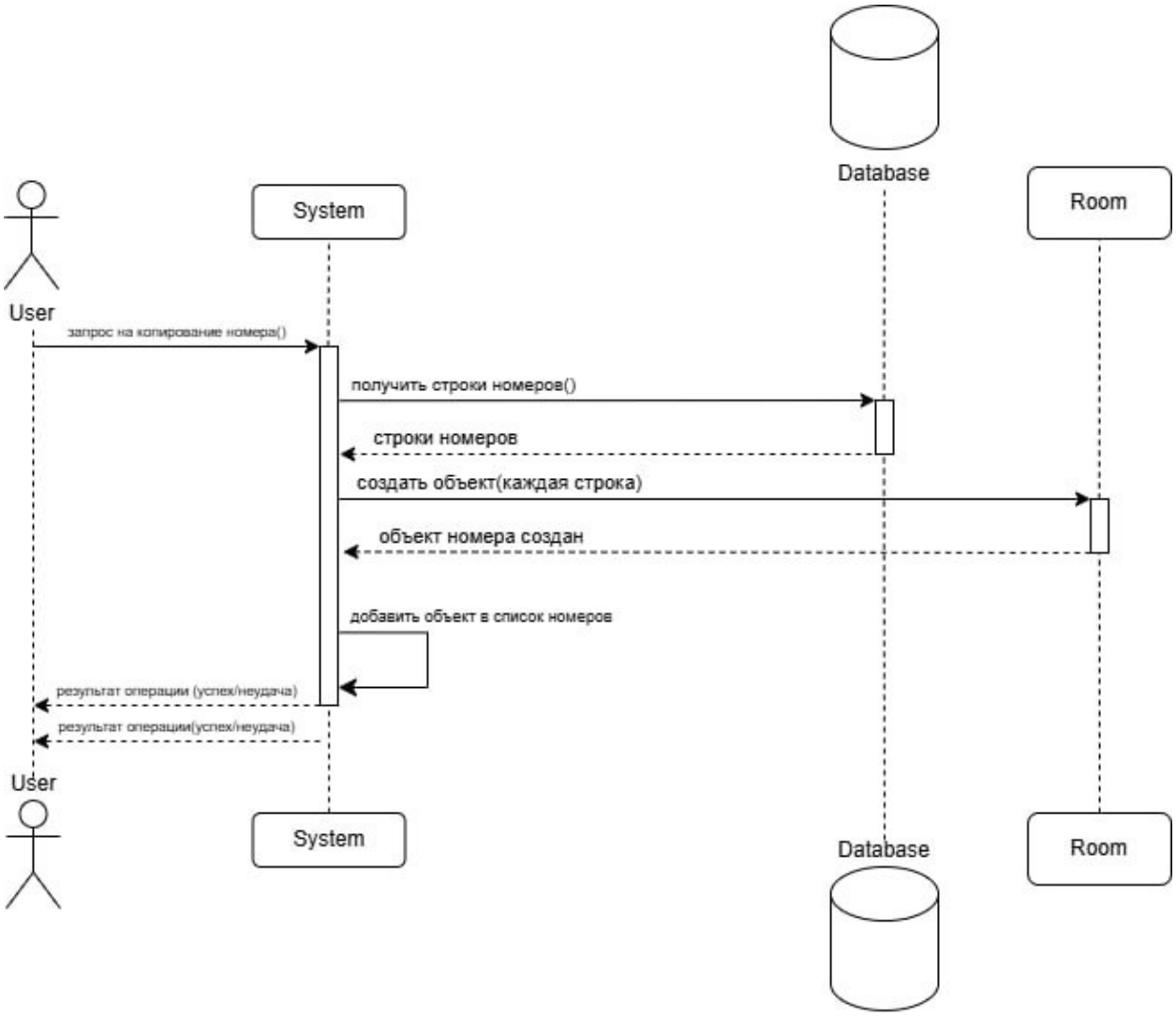
4. **Порядок выполнения операций:**

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.

5. **Условия и создание объектов:**

На диаграмме последовательностей можно описать вызов операции по условию (например, 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 "X".



Построение диаграммы действий.

Диаграмма действий строится для сложных операций. Основным направлением использования диаграмм действий является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения.

Графически диаграмма действий 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 действий должна иметь единственное начальное и единственное

конечное состояние. Диаграмму действий принято строить таким образом, чтобы действия следовали сверху вниз.

Основные элементы диаграммы действий

1. Начальное состояние: обозначает начало процесса, например, "Запрос на добавление номера".

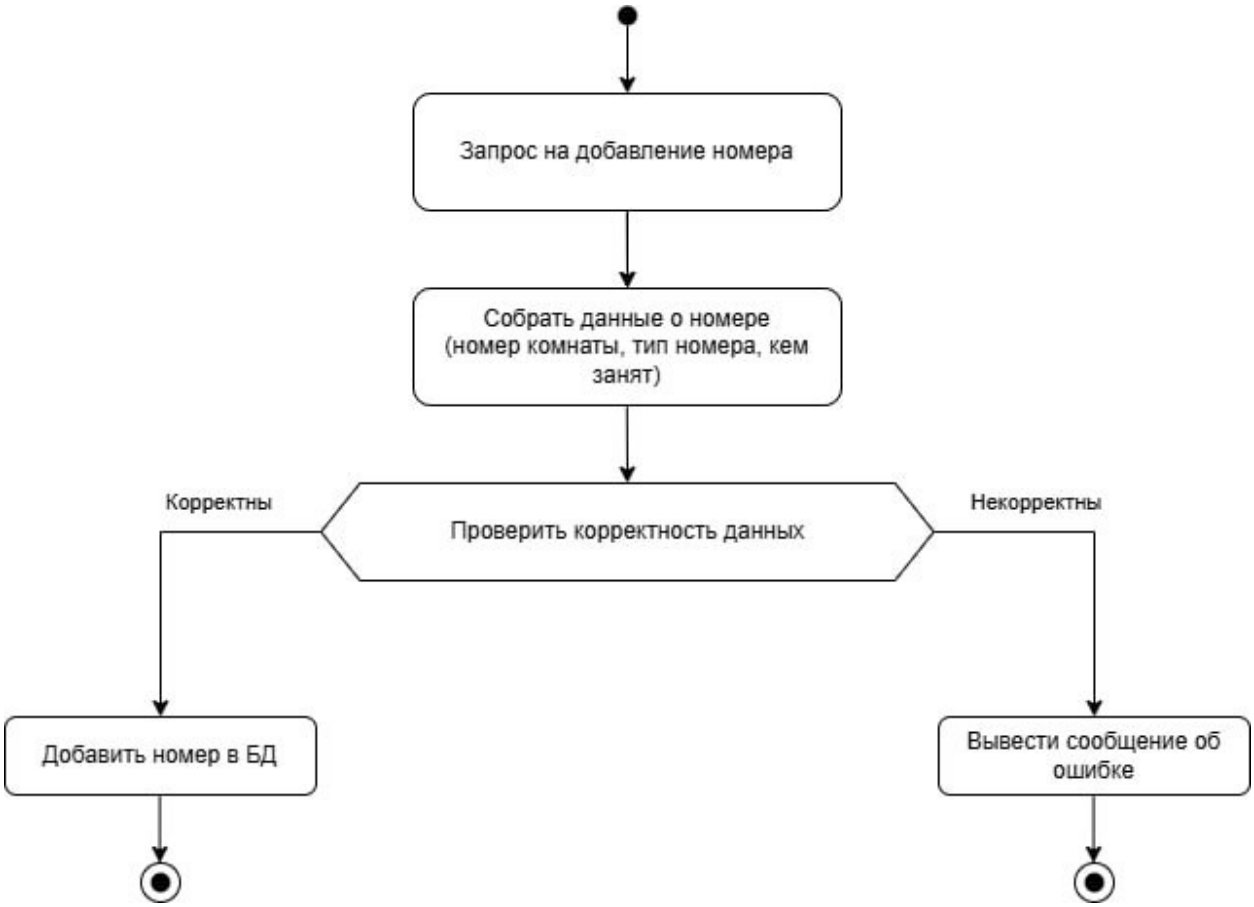
2 . Действия: представляют собой операции, такие как “Собрать данные о номере”.

3. Переходы: показывают последовательность выполнения действий.

4. Конечное состояние: обозначает завершение процесса, например, "Добавление номера в БД".

5. Параллельные процессы: если необходимо, можно отобразить параллельные действия, используя линию синхронизации для разделения и слияния потоков управления.

Пример диаграммы действий



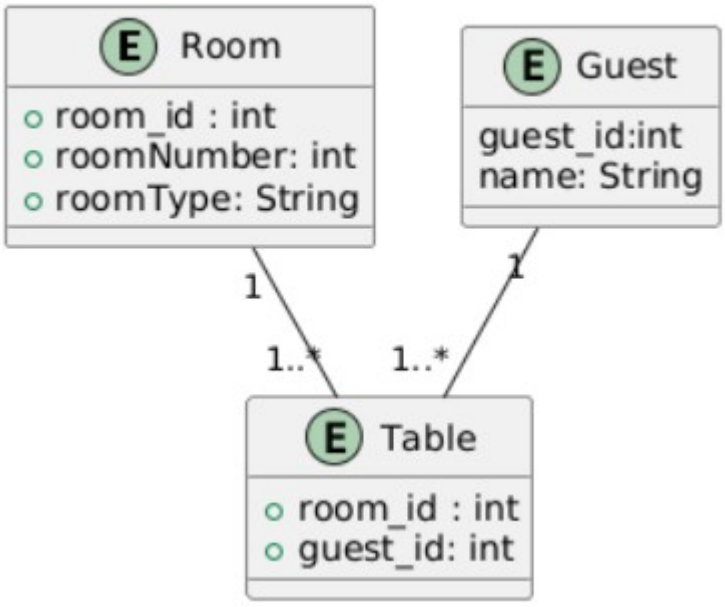
Реализация отношений "многие ко многим"

В системе реализовано отношение "многие ко многим" между номерами и гостями. Это позволяет нескольким гостям жить в одном номере, и одному гостю жить в нескольких номерах. Данная реализация обеспечивает гибкость в учете и администрировании данных.

Описание сущностей

1. Номер (Room):
   Атрибуты:
- Room_id: int - первичный ключ
- roomNumber: int – номер комнаты
- roomType: string – тип номера
2. Гость (Guest):
   Атрибуты:
   - Guest_id: int – первичный ключ
   - Name: string – имя гостя
3. Связующая таблица (Table):
   Атрибуты:
   - room_id: int – внешний ключ
   - guest_id: int – внешний ключ

Схема отношений «многие ко многим»



Описание схемы:

С помощью явной связи по ключам обеспечивается целостность данных, однозначная идентификация и легкость масштабирования программы.

Руководство оператора

Назначение программы

Программа предназначена для автоматизации деятельности, связанной с управлением информацией о номерах, гостях, работниках и услугах в рамках системы гостиницы. Она должна входить в состав автоматизированной системы учета и администрирования информации и предназначена для упрощения работы администраторов гостиниц.

В рамках программы администратор может:

- Управлять данными о номерах, включая возможность добавления, редактирования и удаления информации.
- Управлять данными о сотрудниках.
- Управлять данными о ценах за разные типы номеров.
- Управлять данными клиентов.
- Получать отчёт о работе гостиницы за месяц.

Условия выполнения программы

Программа предназначена для функционирования под операционной средой Windows (XP, 7 и выше) при поддержке базы данных (например, MS Access или MySQL). Персональная электронно-вычислительная машина (ПЭВМ) должна обладать следующими характеристиками:

1. тип процессора: Pentium IV 1.5 ГГц и выше;
2. объем ОЗУ – не менее 2 Гб;
3. объем жесткого диска – не менее 10 Гб;
4. видеокарта – 128 Мб;
5. стандартная клавиатура;
6. манипулятор типа "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.

Описание задачи

Функциональные возможности программы:

Управление данными о гостях:

- Добавление новых записей о гостях.
- Редактирование данных существующих гостей.
- Удаление записей о гостях.
- Поиск гостя по имени, номеру комнаты или дате заезда.

Управление данными о номерах:

- Добавление и редактирование данных о комнатах.
- Указание типа комнаты (например, стандарт, люкс, семейный).
- Назначение комнаты конкретному гостю.

Генерация отчётов:

- Экспорт данных в форматы PDF и HTML.
- Автоматическая генерация отчётов по занятости комнат и статистике за месяц.

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:

Гости: включают имя, номер комнаты, дату заезда и статус.

Номера: содержат информацию о типе комнаты и текущем состоянии (свободна, занята).

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

Заключение

В результате проделанной работы разработан программный комплекс для управления данными гостиницы, предназначенный для администрирования и учета информации о гостях, номерах и персонале. Также разработано руководство оператора, которое описывает работу с программой и основные функции системы.

В процессе проектирования были созданы: описание вариантов использования программы, прототип интерфейса пользователя, объектная модель, диаграмма классов и описание поведения системы.

В ходе реализации был учтён функционал для добавления, удаления и редактирования данных о гостях, персонале и бронированиях, а также для поиска информации. Программа позволяет сохранять данные в текстовый файл для дальнейшего анализа и резервного копирования. Курсовой проект удовлетворяет поставленным требованиям и обеспечивает эффективное управление информацией, позволяя гибко управлять данными в гостиничной системе.