**Курс:** Шаблоны проектирования приложений

**Тема:** Модуль 12 Диаграммы вариантов использования, Диаграмма состояний

Задание №1. Диаграмма состояний

Проектирование и реализация системы управления процессом бронирования номера в гостинице с использованием **диаграммы состояний** (State Diagram)

**Цель работы:**

Изучить принципы построения диаграммы состояний (State Diagram) и применить их для моделирования системы управления процессом бронирования номера в гостинице с последующей реализацией на языке C#.

**Описание задания:**

Необходимо разработать систему управления бронированием номеров в гостинице. В процессе бронирования система проходит через несколько состояний, начиная с выбора номера до завершения бронирования. В каждом состоянии возможны различные действия пользователя, такие как подтверждение бронирования, оплата и отмена бронирования.

**Основные состояния:**

1. **Idle** — начальное состояние, система ожидает действия пользователя.
2. **RoomSelected** — пользователь выбрал номер, но не подтвердил бронирование.
3. **BookingConfirmed** — бронирование подтверждено, но не оплачено.
4. **Paid** — бронирование оплачено, номер закреплен за пользователем.
5. **BookingCancelled** — бронирование отменено.

**Требования:**

1. Пользователь должен иметь возможность отменить бронирование на любом этапе до оплаты.
2. Если бронирование не оплачено, оно не может быть завершено.
3. После успешной оплаты бронирование завершается.

**Этапы выполнения:**

**1. Проектирование диаграммы состояний (UML):**

Создайте диаграмму состояний для описания процесса бронирования номера в гостинице. Диаграмма должна включать:

* Состояния: ожидание, выбор номера, подтверждение бронирования, оплата, завершение бронирования, отмена.
* Переходы между состояниями: выбор номера, подтверждение бронирования, оплата, отмена.

**Дополнительные задания (по желанию):**

* Добавить функцию изменения номера до подтверждения бронирования.
* Реализовать возможность расчета скидок при оплате.
* Добавить функционал для сохранения истории бронирований.

Задание №2. Диаграммы состояния

**Тема: Разработка и реализация диаграммы состояния для объекта в системе.**

**Цель работы**

1. Изучить принципы построения диаграмм состояния (State Diagram).
2. Научиться реализовывать переходы между состояниями объекта в приложении с использованием паттерна State.

**Условие задачи**

**Система: Управление заявками в системе бронирования билетов**

В системе бронирования билетов каждая заявка проходит через следующие состояния:

1. **Создана** – Заявка только что создана.
2. **Ожидает оплаты** – Заявка отправлена клиенту для оплаты.
3. **Оплачена** – Клиент завершил оплату.
4. **Подтверждена** – Система подтвердила успешное бронирование.
5. **Отменена** – Заявка была отменена.

**Правила переходов:**

* Заявка из состояния **Создана** переходит в **Ожидает оплаты** после отправки клиенту.
* Из **Ожидает оплаты**:
  + В **Оплачена** – если клиент произвел оплату.
  + В **Отменена** – если клиент не успел оплатить в установленный срок.
* Из **Оплачена** переходит в **Подтверждена** после проверки бронирования.
* **Отменена** – это финальное состояние.

**Часть 1: Построение диаграммы состояния**

1. Постройте диаграмму состояния для объекта **Заявка (BookingRequest)**.
2. Используйте следующие элементы:
   * Состояния: Создана, Ожидает оплаты, Оплачена, Подтверждена, Отменена.
   * Переходы: Опишите события или условия, вызывающие переходы (например, "Клиент оплатил").
   * Финальное состояние: Отмена.

Для построения диаграммы используйте UML-редактор, например, **StarUML**, **Lucidchart**, или **Draw.io**.

Задание №3. Диаграммы вариантов использования

**Тема: Система управления онлайн-курсов**

**Цель задания:**

Разработать систему для управления онлайн-курсами с использованием диаграммы вариантов использования. Реализовать её основные элементы, включая сложные взаимодействия между ролями пользователей и функциональными модулями.

**Описание системы**

Система предназначена для управления онлайн-курсами. Она позволяет пользователям взаимодействовать с курсами, преподавателями, а также отслеживать прогресс обучения.

**Роли пользователей:**

1. **Студент**:
   * Регистрация и авторизация.
   * Просмотр доступных курсов.
   * Запись на курс.
   * Прохождение тестов и отслеживание прогресса.
   * Оставление отзывов на курсы.
2. **Преподаватель**:
   * Создание и редактирование курсов.
   * Добавление материалов курса.
   * Создание тестов для курсов.
   * Просмотр статистики успеваемости студентов.
   * Модерация отзывов.
3. **Администратор**:
   * Управление учетными записями пользователей.
   * Управление категориями курсов.
   * Просмотр аналитики системы (число студентов, популярность курсов, успеваемость).

**Часть 1: Диаграмма вариантов использования**

Создайте диаграмму вариантов использования, включающую:

1. **Акторы**:
   * Студент
   * Преподаватель
   * Администратор
2. **Варианты использования**:
   * Регистрация и авторизация.
   * Просмотр курсов.
   * Запись на курс.
   * Прохождение тестов.
   * Создание и редактирование курсов.
   * Модерация отзывов.
   * Управление учетными записями.
   * Просмотр аналитики.
3. **Связи**:
   * Используйте связи **extends** и **includes**, где это применимо (например, регистрация может включать подтверждение почты).
4. **Иерархия ролей**:
   * Покажите, что преподаватель и администратор могут выполнять функции студента (например, проходить тесты для проверки их работы).