**Курс:** Шаблоны проектирования приложений

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

Задание №1. Диаграммы вариантов использования

**Тема задания: Система управления сетью библиотек**

Система предназначена для управления несколькими библиотеками, позволяя пользователям взаимодействовать с книгами, а администраторам – управлять филиалами и каталогами.

**Описание системы:**

**Роли пользователей:**

1. **Читатель**:
   * Регистрация в системе.
   * Просмотр доступных книг.
   * Поиск книг по автору, названию, жанру.
   * Бронирование книги.
   * Отмена бронирования.
   * Просмотр истории бронирований.
2. **Библиотекарь**:
   * Добавление и удаление книг.
   * Учет выданных и возвращенных книг.
   * Просмотр списка активных бронирований.
   * Управление каталогом.
3. **Администратор**:
   * Управление филиалами библиотеки (добавление/удаление филиалов).
   * Управление учетными записями пользователей и библиотекарей.
   * Просмотр аналитики (количество выданных книг, самые популярные книги и т. д.).

**Часть 1: Разработка диаграммы вариантов использования**

Создайте диаграмму вариантов использования, которая включает:

1. **Акторов**:
   * Читатель
   * Библиотекарь
   * Администратор
2. **Варианты использования**:
   * Регистрация пользователя
   * Просмотр и поиск книг
   * Бронирование книги
   * Управление книгами (добавление/удаление)
   * Выдача/возврат книг
   * Управление филиалами
   * Просмотр аналитики
3. **Связи**:
   * Используйте связи **extends** и **includes**, чтобы отразить зависимость между вариантами использования.
4. **Иерархию ролей**:
   * Покажите, что библиотекарь и администратор наследуют часть функциональности читателя.

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

Задание №2. Диаграмма состояния

**Проектирование и реализация автомата по продаже билетов с использованием диаграммы состояний (State Machine) на языке C#**

**Цель работы:**

Изучить принципы построения **диаграмм состояний** (State Diagram) и применить их для разработки системы управления автоматом по продаже билетов с последующей реализацией на языке C#.

**Описание задания:**

Необходимо спроектировать и реализовать систему управления автоматом по продаже билетов, которая будет имитировать процесс покупки билетов. Автомат должен проходить через несколько состояний в зависимости от действий пользователя (внесение денег, выбор билета, отмена покупки и т.д.).

Основные состояния:

1. **Idle** (ожидание) — начальное состояние, автомат ожидает действия от пользователя.
2. **WaitingForMoney** (ожидание внесения денег) — состояние, в котором автомат ждет от пользователя внесения средств.
3. **MoneyReceived** (деньги получены) — состояние, когда внесены необходимые средства.
4. **TicketDispensed** (билет выдан) — состояние, когда билет был выдан пользователю.
5. **TransactionCanceled** (транзакция отменена) — состояние, когда пользователь отменил покупку до завершения процесса.

**Этапы выполнения работы:**

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

Создайте диаграмму состояний для описания процесса работы автомата по продаже билетов. Она должна включать следующие состояния и переходы между ними:

* Переход из состояния **Idle** в **WaitingForMoney** при выборе билета.
* Переход из состояния **WaitingForMoney** в **MoneyReceived** при внесении необходимой суммы.
* Переход из состояния **MoneyReceived** в **TicketDispensed** после выдачи билета.
* Возможность отмены транзакции из состояния **WaitingForMoney** или **MoneyReceived** в состояние **TransactionCanceled**.

**2. Реализация на языке C#:**

Реализуйте автомат по продаже билетов на языке C# с использованием паттерна "состояние" (State Pattern). Для каждого состояния создайте соответствующий класс и обеспечьте управление переходами между состояниями.

**Дополнительные задания (по желанию):**

* Добавить новые состояния, например, возможность возврата сдачи.
* Реализовать разные типы билетов с разной стоимостью.
* Добавить функцию проверки остатка билетов в автомате.