**Курс:** Шаблоны проектирования приложений

**Тема:** Модуль 11 Диаграммы вариантов использования. Диаграмма классов

**Цель:**

Изучить и применить концепцию **диаграмм классов (Class Diagram)** для проектирования объектно-ориентированных систем. В рамках задания вы создадите диаграмму классов для простой системы управления библиотекой и реализуете её на языке C#.

**Описание задания:**

Разработайте систему управления библиотекой, которая включает такие сущности, как **Книга**, **Читатель**, и **Библиотекарь**. Система должна позволять читателям брать книги в аренду, возвращать их и предоставлять библиотекарю возможность управлять книгами.

**Часть 1: Создание диаграммы классов**

1. **Классы и их атрибуты**:
   * **Book** (Книга):
     + Атрибуты: название, автор, ISBN, статус (в наличии/арендована).
   * **Reader** (Читатель):
     + Атрибуты: имя, список арендованных книг.
   * **Librarian** (Библиотекарь):
     + Атрибуты: имя.
   * **Library** (Библиотека):
     + Атрибуты: список книг.
2. **Отношения между классами**:
   * Класс **Library** имеет композицию с классом **Book** (библиотека содержит книги).
   * Класс **Reader** может арендовать несколько книг, следовательно, имеет связь с классом **Book** (агрегация).
   * Класс **Librarian** управляет книгами в библиотеке (связь с классом **Library**).

**Ожидаемый результат:**

Программа должна вывести информацию о доступных книгах в библиотеке, аренде книги читателем, а также об изменениях в списке доступных книг после аренды и возврата.

**Дополнительные задания:**

1. Добавьте возможность поиска книг по названию или автору.
2. Реализуйте функцию для отображения всех книг в библиотеке, включая те, которые находятся в аренде.
3. Добавьте ограничение на максимальное количество книг, которые читатель может взять в аренду одновременно.

**Цель:** Научиться проектировать и визуализировать архитектуру распределенной системы, **используя диаграмму компонентов в UML**, а также реализовать ее основные элементы на языке C#.

**Описание задачи**

Разработайте архитектуру и реализацию распределенной системы бронирования отелей. В системе будет несколько компонентов, каждый из которых представляет собой отдельный модуль с определенной логикой. Необходимо спроектировать систему, в которой взаимодействие между компонентами происходит через интерфейсы, что позволит системе быть модульной и легко расширяемой.

**Описание компонентов системы**

1. **UI-компонент** – интерфейс для пользователей (консольное приложение), где пользователи могут искать отели, бронировать номера и просматривать свои бронирования.
2. **HotelService** – компонент для работы с данными отелей:
   * Управляет информацией об отелях (название, местоположение, классы номеров, цены).
   * Предоставляет методы для поиска отелей по заданным критериям (местоположение, класс, цена и т.д.).
3. **BookingService** – компонент для бронирования номеров:
   * Осуществляет бронирование номера на определенные даты.
   * Проверяет доступность номера.
   * Хранит данные о бронированиях (пользователь, отель, даты).
4. **PaymentService** – компонент для обработки платежей:
   * Управляет информацией о платежах.
   * Проверяет оплату бронирований.
   * Поддерживает оплату различными методами (например, карта, PayPal и т.д.).
5. **NotificationService** – компонент для отправки уведомлений пользователям:
   * Отправляет уведомления о подтверждении бронирования.
   * Отправляет напоминания перед заездом и подтверждения об оплате.
6. **UserManagementService** – компонент для управления данными пользователей:
   * Регистрация и авторизация пользователей.
   * Хранение информации о пользователях.
   * Поддерживает методы для регистрации и входа.

**Результаты выполнения:**

1. **Диаграмма компонентов** с отображением всех компонентов, интерфейсов и связей.
2. **Исходный код компонентов** с реализацией интерфейсов и их взаимодействием.
3. **Консольный интерфейс**, позволяющий пользователю взаимодействовать с системой.
4. **Отчет**, содержащий описание архитектуры системы, диаграмму компонентов, код и описание работы каждого компонента, а также результаты тестирования.

**Итог:** В результате выполнения данной работы вы изучите, как проектировать архитектуру приложения с помощью диаграммы компонентов, реализовывать взаимодействие компонентов через интерфейсы и создавать независимые модули на языке C#.