Практическая работа по паттерну Singleton в C#

Цель работы

Целью этой практической работы является изучение и применение паттерна проектирования Singleton на языке программирования C#.

Задачи

1. Изучить основные концепции паттерна Singleton.
2. Реализовать простой Singleton класс.
3. Создать несколько сценариев использования Singleton в разных контекстах.

Теоретическая часть

Паттерн Singleton гарантирует, что у класса есть только один экземпляр, и предоставляет глобальную точку доступа к этому экземпляру. Это полезно, например, когда вам нужен общий объект для координации действий между разными частями приложения.

Реализация Singleton

Самый простой способ реализации Singleton в C# - это использовать ленивую инициализацию и статическое поле. Вот пример:

```
public class Singleton
{
    private static Singleton instance;
    private Singleton() { }

    public static Singleton Instance
    {
        get
        {
            if (instance == null)
            {
                instance = new Singleton();
            }
            return instance;
        }
    }
}
```

Практическая часть

Задача 1: Создание Singleton

1. Создайте новый проект C# в вашей среде разработки (например, Visual Studio).

2. Создайте класс Singleton и реализуйте его в соответствии с примером выше.

Задача 2: Варианты использования Singleton

1. Использование в логгере: Создайте класс Logger, который будет использовать Singleton для записи логов в единый журнал. Добавьте методы для записи логов разных уровней (например, Info, Error).

2. Использование в приложении для работы с базой данных: Создайте класс DatabaseConnection, который будет использовать Singleton для установки и поддержания одного соединения с базой данных.

3. Использование в приложении для хранения настроек: Создайте класс AppSettings, который будет использовать Singleton для хранения и предоставления настроек приложения.

4. Использование в кеше данных: Создайте класс CacheManager, который будет использовать Singleton для кэширования данных, чтобы избежать повторных запросов к источнику данных.

5. Использование в системе управления ресурсами: Создайте класс ResourceManager, который будет использовать Singleton для управления доступом к ресурсам, например, пулу потоков.

Задача 3: Демонстрация использования Singleton

В каждом из вариантов использования Singleton создайте простые примеры, которые демонстрируют, как можно получить доступ к Singleton экземпляру и использовать его функциональность.