Ильяхова Алиса, Б9122-02.03.01сцт

Лабораторная работа №8. Создание и запуск задач

Вариант 9

1. Цель и задачи работы

Цель:

Изучить механизм работы с классами задач.

Задачи:

- Изучить возможности класса Task;
- Научиться создавать и запускать задачи;
- Научиться работать с массивами задач.

2. Реализация индивидуального задания

2.1. Условие варианта 9

Согласно таблице на стр. 31:

- **Тип делегата**: лямбда-выражение
- **Решаемая задача**: Метод возвращает результат шифрования строки: каждый исходный символ строки заменяется шифрованным символом, код которого на n больше кода исходного символа.
- **Входные параметры**: Два параметра — исходная строка, число сдвига n.

2.2. Метод шифрования

Реализован метод EncryptString:

- Принимает строку и сдвиг.
- Выполняет побайтовое шифрование.
- Имитирует долгую операцию через Thread.Sleep(2000).
- Возвращает зашифрованную строку.

2.3. Создание и запуск задач

- Создан массив тестовых данных: пары (строка, сдвиг).
- Для каждой пары создана задача через Task.Run().

- Использовано **лямбда-выражение** для передачи параметров.
- Переменная index захвачена для корректной передачи в замыкание.

2.4. Ожидание завершения

- Выполнено ожидание завершения всех задач через Task.WaitAll(tasks).
- После завершения выведены результаты через свойство Result.

3. Ответы на контрольные вопросы

1. **Что такое класс Task? Чем класс Task отличается от класса Thread?**
    a. Task — это высокоуровневая абстракция над потоком, представляющая **единицу работы**.
    b. Thread — низкоуровневый объект, управляющий реальным потоком ОС.
    c. Task использует пул потоков, автоматически управляет ресурсами, поддерживает возврат значений и исключения.
2. **Как получить идентификатор текущей задачи? Как получить идентификатор текущего потока?**
    a. Идентификатор задачи: Task.CurrentId (может быть null вне задачи).
    b. Идентификатор потока: Thread.CurrentThread.ManagedThreadId.
3. **Опишите возможные методы создания и запуска задач.**
    a. Task.Run(action) — запускает задачу немедленно.
    b. new Task(action); task.Start() — создаёт задачу в состоянии Created, запускает вручную.
    c. Task.Factory.StartNew(action) — гибкий способ с настройками.
4. **Какие параметры задачи может установить программист при реализации задачи? Опишите возможные варианты применения параметра типа TaskCreationOptions.**
    a. LongRunning — указывает, что задача займёт много времени (создаётся отдельный поток).
    b. PreferFairness — обеспечивает честное планирование.
    c. AttachedToParent — привязывает задачу к родительской.

4. Экранные формы и листинг программы

4.1. Консольный вывод программы

```
=== Лабораторная работа №8. Вариант 9 ===
Создание и запуск задач

[Задача ID: 17] Начало шифрования...
[Задача ID: 18] Начало шифрования...
[Задача ID: 19] Начало шифрования...
[Задача ID: 19] Шифрование завершено.
[Задача ID: 17] Шифрование завершено.
[Задача ID: 18] Шифрование завершено.

=== Результаты шифрования ===
Задача 1: "Khoor"
Задача 2: "\twqi"
Задача 3: "Zljyl{"

 Все задачи завершены.
```

4.2. Полный листинг программы с комментариями

```
using System;
using System.Threading;
using System.Threading.Tasks;

// Лабораторная работа №8. Создание и запуск задач
// Вариант 9

class Program
{
    // Метод шифрования строки
    static string EncryptString(string input, int shift)
    {
        if (input == null) return null;
        Console.WriteLine($"[Задача ID: {Task.CurrentId}] Начало шифрования...");
        Thread.Sleep(2000); // Имитация долгой операции

        char[] buffer = new char[input.Length];
        for (int i = 0; i < input.Length; i++)
        {
            buffer[i] = (char)(input[i] + shift);
        }

        string result = new string(buffer);
```

```
        Console.WriteLine($"[Задача ID: {Task.CurrentId}] Шифрование завершено.");
        return result;
    }

    static void Main()
    {
        Console.WriteLine("=== Лабораторная работа №8. Вариант 9 ===");
        Console.WriteLine("Создание и запуск задач\n");

        // Тестовые данные
        var testData = new (string text, int shift)[]
        {
            ("Hello", 3),
            ("World", 5),
            ("Secret", 7)
        };

        // Массив задач
        Task<string>[] tasks = new Task<string>[testData.Length];

        // Создание и запуск задач
        for (int i = 0; i < testData.Length; i++)
        {
            int index = i; // захват переменной
            tasks[i] = Task.Run(() =>
            {
                return EncryptString(testData[index].text, testData[index].shift);
            });
        }

        // Ожидание завершения всех задач
        Task.WaitAll(tasks);

        // Вывод результатов
        Console.WriteLine("\n=== Результаты шифрования ===");
        for (int i = 0; i < tasks.Length; i++)
        {
            Console.WriteLine($"Задача {i + 1}: \"{tasks[i].Result}\"");
        }

        Console.WriteLine("\n Все задачи завершены.");
    }
}
```

5. Вывод

В ходе выполнения лабораторной работы №8 были:

- Реализован метод шифрования строки согласно варианту 9;

- Созданы и запущены несколько задач с использованием Task.Run;
- Продемонстрировано ожидание завершения всех задач через Task.WaitAll;
- Показано получение результатов через свойство Result;
- Использовано лямбда-выражение для передачи параметров.