Шакиров Р.Л. гр 4204 дата 05.12.2025
Лабораторная работа №.6
Делегаты и события
C#

Задание 3: Создать приложение, в котором генератор события после генерации первого события генерирует последующие события только в том случае, если приемник события уведомляет, что сообщение принято (квитирование). Для квитирования использовать первый параметр обработчика события.

```
using System;
using System.Collections.Generic;
using System.Threading;

namespace sobit
{
    // Аргументы события с поддержкой квитирования
    class ChangeEventArgs : EventArgs
    {
        string str;
        public string Str
        {
            get { return str; }
        }

        int change;
        public int Change
        {
            get { return change; }
        }

        // Делегат для квитирования
        public Action<bool> AcknowledgeCallback { get; }

        public ChangeEventArgs(string str, int change, Action<bool> acknowledgeCallback)
        {
            this.str = str;
            this.change = change;
            this.AcknowledgeCallback = acknowledgeCallback;
        }
    }

    class GenEvent // Генератор событий - издатель
    {
        public delegate void ChangeEventHandler(object source, ChangeEventArgs e);
        public event ChangeEventHandler OnChangeHandler;

        private bool _waitingForAck = false;
        private Queue<(string, int)> _pendingEvents = new Queue<(string, int)>();

        public void UpdateEvent(string str, int change)
        {
            if (change == 0)
                return;
```

```
            // Если ждем квитирования, ставим в очередь
            if (_waitingForAck)
            {
                Console.WriteLine($"Генератор: Событие '{str}'
поставлено в очередь (ожидание квитирования)");
                _pendingEvents.Enqueue((str, change));
                return;
            }

            _waitingForAck = true;

            // Создаем callback для квитирования
            Action<bool> ackCallback = (success) =>
            {
                _waitingForAck = false;
                Console.WriteLine($"Генератор: Получено
квитирование - {(success ? "УСПЕХ" : "ОШИБКА")}");

                // Обрабатываем следующее событие из
очереди
                ProcessNextEvent();
            };

            ChangeEventArgs e = new ChangeEventArgs(str, change, ackCallback);
            Console.WriteLine($"Генератор: Отправляю событие '{str}'");

            if (OnChangeHandler != null)
                OnChangeHandler(this, e);
        }

        private void ProcessNextEvent()
        {
            if (_pendingEvents.Count > 0)
            {
                var (str, change) = _pendingEvents.Dequeue();
                Console.WriteLine($"Генератор: Обрабатываю
следующее событие из очереди: '{str}'");
                UpdateEvent(str, change);
            }
        }
    }

    // Подписчик
    class RecEvent
    {
        private Random _random = new Random();

        // Обработчик события
        void OnRecChange(object source, ChangeEventArgs e)
        {
            int change = e.Change;
            Console.WriteLine($"Приемник: Получено '{e.Str}' -
{Math.Abs(e.Change)} тонны");

            // Имитация обработки с задержкой
```

```
            Thread.Sleep(1500);

            // Случайным образом определяем успешность
обработки
            bool success = _random.Next(0, 2) == 1;

            if (success)
            {
                Console.WriteLine($"Приемник: Обработка успешна для
'{e.Str}', отправляю квитирование...");
            }
            else
            {
                Console.WriteLine($"Приемник: Ошибка обработки для
'{e.Str}', уведомляю генератор...");
            }

            // Отправляем квитирование через callback
            e.AcknowledgeCallback?.Invoke(success);
        }

        // В конструкторе класса осуществляется
подписка
        public RecEvent(GenEvent gnEvent)
        {
            gnEvent.OnChangeHandler += new GenEvent.ChangeEventHandler(OnRecChange);
        }
    }

    // Второй приемник для демонстрации
    class SecondRecEvent
    {
        // Обработчик события
        void OnRecChange(object source, ChangeEventArgs e)
        {
            Console.WriteLine($"Второй приемник: Логирую изменение
'{e.Str}' на {e.Change} тонн");

            // Всегда успешная обработка для второго
приемника
            Thread.Sleep(500);
            Console.WriteLine($"Второй приемник: Логирование
завершено для '{e.Str}'");

            // Отправляем квитирование
            e.AcknowledgeCallback?.Invoke(true);
        }

        public SecondRecEvent(GenEvent gnEvent)
        {
            gnEvent.OnChangeHandler += new GenEvent.ChangeEventHandler(OnRecChange);
        }
    }

    class Class1
    {
```

```
        [STAThread]
        static void Main(string[] args)
        {
            Console.WriteLine("=== Система с квитированием событий
===¥n");

            GenEvent gnEvent = new GenEvent();
            RecEvent inventoryWatch = new RecEvent(gnEvent);
            SecondRecEvent secondReceiver = new SecondRecEvent(gnEvent);

            // Посылаем несколько событий подряд
            Console.WriteLine("Посылаю первое событие:");
            gnEvent.UpdateEvent("грузовика", -2);

            Console.WriteLine("¥nПосылаю второе событие:");
            gnEvent.UpdateEvent("автопоезда", 4);

            Console.WriteLine("¥nПосылаю третье событие:");
            gnEvent.UpdateEvent("фуры", 3);

            Console.WriteLine("¥nПосылаю четвертое событие:");
            gnEvent.UpdateEvent("контейнера", -1);

            // Даем время на обработку всех событий
            Thread.Sleep(10000);

            Console.WriteLine("¥n=== Все события обработаны ===");
            Console.ReadKey();
        }
    }
}
```

Вывод: рассмотрены делегаты и события