Министерство образования и науки Российской Федерации ФГАОУ ВПО “УрФУ имени Президента России Б.Н. Ельцина”

Институт радиоэлектроники и информационных технологий Кафедра Информационных технологий

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отчет по лабораторной работе №2**

Студенты: Носков Д.В., Кондратьев А.С.

Гриппа: РИМ-160204

Преподаватель: Трофимов С.П.

Екатеринбург

2017 г.

**Цель работы**

В рамках лабораторной работы, необходимо решить следующий перечень задач:

1. **Блокирование потоков с помощью критических секций**

В основном потоке создаются два потока, каждый из которых выполняет один метод, реализующий интерфейс с пользователем (диалог с вводом информации о пользователе). Организуйте критический блок, в который включите все функции ввода. Убедитесь, что интерфейсы перемешиваются но не конфликтуют друг с другом.

1. **Синхронизация потоков с помощью объектов-событий**

Приложение запускает несколько потоков. Каждый поток ожидает нажатия клавиши Enter в основном потоке, после чего выводит полную информацию о пользователе. Информации о разных пользователях не должна перемешиваться.

Используйте объекты события AutoResetEvent или ManualResetEvent.

**Задание 1.**

Для выполнения условий задания был написан следующий код:

using System;  
using System.Threading.Tasks;  
  
namespace ThreadLocking  
{  
 internal interface IDialogWithUser  
 {  
 string AskUserName(int threadNumber);  
 bool AskIsMan(int threadNumber);  
 int AskAge(int threadNumber);  
 }  
  
 internal class DialogWithUser : IDialogWithUser  
 {  
 private static readonly object sync = new object();  
  
 public string AskUserName(int threadNumber)  
 {  
 lock (sync)  
 {  
 Console.Write($"I'm thread #{threadNumber} ");  
 Console.WriteLine("What is you name?");  
 return Console.ReadLine();  
 }  
 }  
  
 public bool AskIsMan(int threadNumber)  
 {  
 lock (sync)  
 {  
 Console.Write($"I'm thread #{threadNumber} ");  
 Console.WriteLine("Write 'true' if you are a man.");  
 return bool.Parse(Console.ReadLine());  
 }  
 }  
  
 public int AskAge(int threadNumber)  
 {  
 lock (sync)  
 {  
 Console.Write($"I'm thread #{threadNumber} ");  
 Console.WriteLine("What is you age?");  
 return int.Parse(Console.ReadLine());  
 }  
 }  
 }  
  
 internal class EntryPoint  
 {  
 private static readonly IDialogWithUser dialog = new DialogWithUser();  
  
 public static void Main(string[] args)  
 {  
 var tasks = new[]  
 {  
 Task.Run(() =>  
 {  
 dialog.AskUserName(1);  
 }),  
 Task.Run(() =>  
 {  
 dialog.AskIsMan(2);  
 }),  
  
 Task.Run(() =>  
 {  
 dialog.AskAge(3);  
 })  
 };  
  
 Task.WaitAll(tasks);  
 }  
 }  
}

В данном случае, был организован критический блок, в который были включены все функции ввода. По результатам выполнения кода, видно, что интерфейсы не конфликтуют друг с другом (Рис. 1).

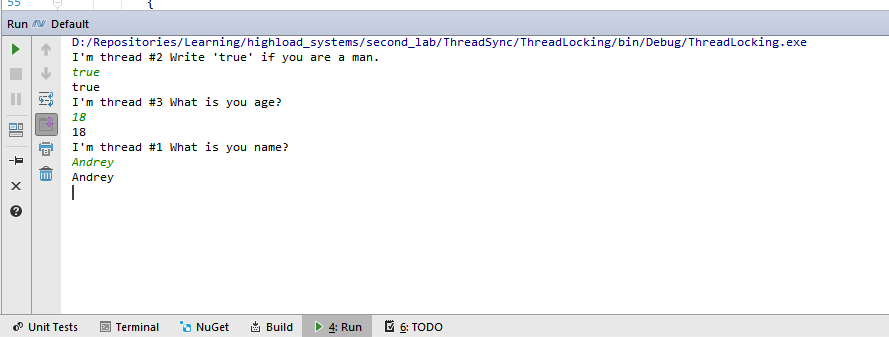


Рис. 1. Результат работы кода для задания 1

**Задание 2.**

Данная задача была реализована следующим образом:

using System;  
using System.Linq;  
using System.Threading;  
using System.Threading.Tasks;  
  
namespace ResetEvents  
{  
 internal class UserInfo  
 {  
 public string Name { get; set; }  
 public int Age { get; set; }  
 public bool IsMan { get; set; }  
  
 public override string ToString() => $"{Name} - {(IsMan ? "man" : "woman")} - {Age}";  
  
 }  
  
 internal class EntryPoint  
 {  
 private static UserInfo[] infos =  
 {  
 new UserInfo {Name = "Andrey", Age = 22, IsMan = true},  
 new UserInfo {Name = "Dima", Age = 22, IsMan = true},  
 new UserInfo {Name = "Sveta", Age = 21, IsMan = false},  
 new UserInfo {Name = "Vadim", Age = 22, IsMan = true},  
 new UserInfo {Name = "Zhana", Age = 24, IsMan = false}  
 };  
  
 public static void Main(string[] args)  
 {  
 using (var resetEvent = new ManualResetEvent(false))  
 {  
 var tasks = infos.Select(x => Task.Run(() =>  
 {  
 resetEvent.WaitOne();  
 Console.WriteLine(x.ToString());  
 })).ToArray();  
  
 Console.WriteLine("I'm a main thread. After 2 seconds I set reset event.");  
 Task.Delay(2000).Wait();  
 resetEvent.Set();  
 Task.WaitAll(tasks);  
 }  
 }  
 }  
}

Результат выполнения программы представлен на рисунке 2.

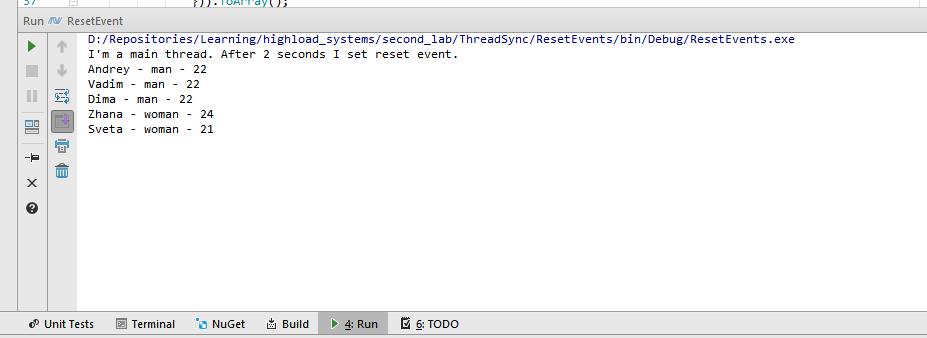


Рис. 2. Результат выполнения кода для задания 2

**Заключение**

В результате выполнения данной лабораторной работы было решено две задачи.

Для первой задачи была написана программа, включающая в себя один критический блок, который содержит в себе все функции ввода, а также получен результат подтверждающий, что интерфейсы взаимодействия с пользователем не вступают в конфликт друг с другом.

Для решения второй задачи был использован объект события ManualResetEvent, а также был получен результат, подтверждающий, что потоки не перемешались друг с другом (информация о пользователях осталась исходной).