|  |  |
| --- | --- |
| Name: | Edward Eisenberger |
| ID# | 1066164 |
| Assignment 2 | |
| Date of Submission | May 31, 2019 |

Table of Contents

[Overview 3](#_Toc10195490)

[Part A: ThreadPooling 3](#_Toc10195491)

[Summary 3](#_Toc10195492)

[Results 3](#_Toc10195493)

[Thread Pool Start 3](#_Toc10195494)

[Thread Pool Array 4](#_Toc10195495)

[Part B: ProducerConsumer 5](#_Toc10195496)

[Summary 5](#_Toc10195497)

[Results 5](#_Toc10195498)

[Part C: SemaphoreTest 8](#_Toc10195499)

[Summary 8](#_Toc10195500)

[Results 8](#_Toc10195501)

[Part D: MutexTest 9](#_Toc10195502)

[Summary 9](#_Toc10195503)

[Results 9](#_Toc10195504)

# Overview

Assignment 2 consisted of implementing the examples in the Handout on “Multithreading in Csharp” from page 12 to page 23 (excluding process and thread monitoring).

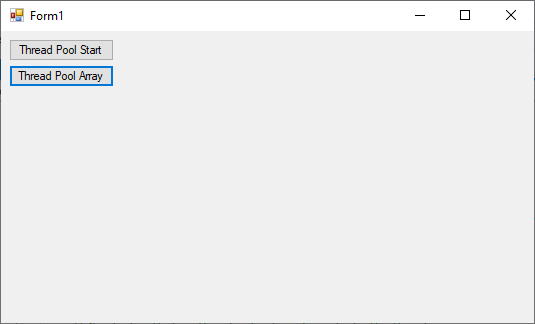
# Part A: ThreadPooling

## Summary

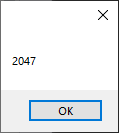
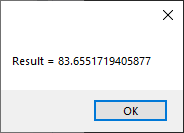
The ThreadPooling project introduced using the ThreadPool to launch worker threads. Worker objects were launched using the ThreadPool and utilize the ManualResetEvent to indicate completion of its task.

## Results

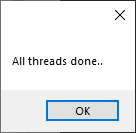
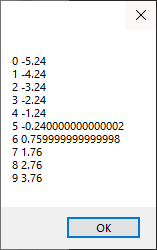
The output of the project is shown below:



### Thread Pool Start

### Thread Pool Array

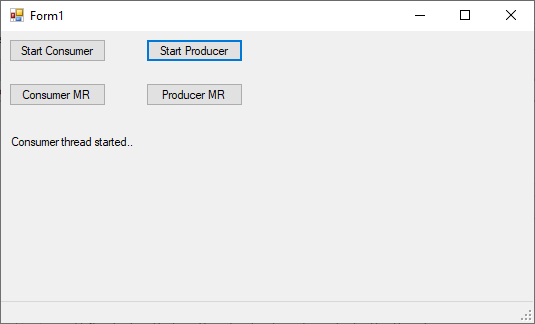
# Part B: ProducerConsumer

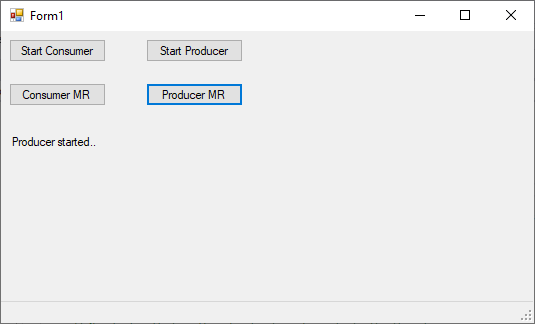
## Summary

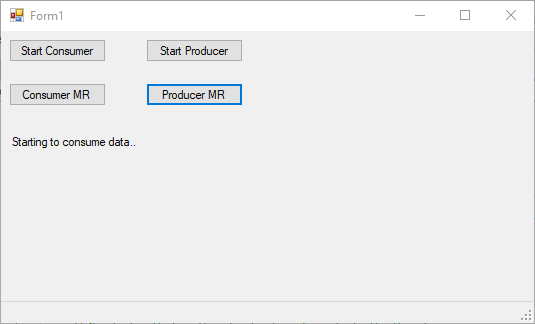
The ProducerConsumer project introduced the concept of synchronizing threads where one thread produces data and another thread consumes the data. One method utilizes the Monitor.Wait in the Consumer thread and Monitor.Pulse in the producer thread. The second method utilizes the ManualResetEvent for synchronization, ManResetEvent.WaitOne and ManualResetEvent.Reset in the consumer thread and ManualResetEvent.Set in the producer thread.

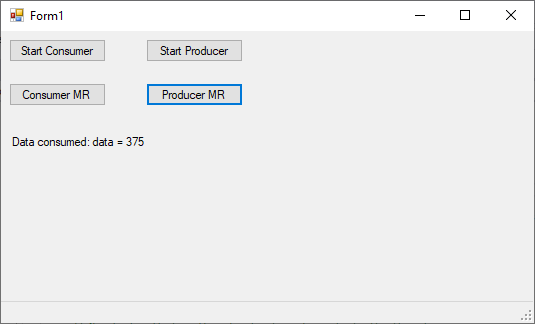
## Results

The following images are results of the project:









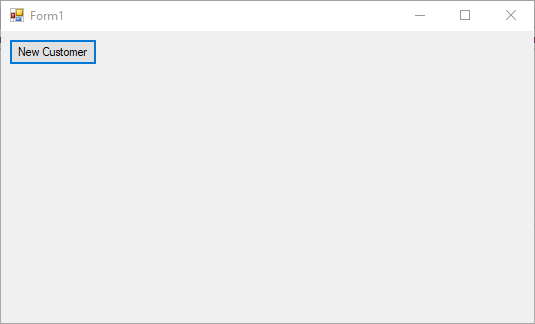
# Part C: SemaphoreTest

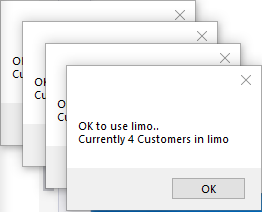
## Summary

The SemaphoreTest application introduced the concept of semaphores. A LimoCar class is implemented which uses a semaphore to limit access to use of the Limo. 4 users are allowed to the access to the Limo at a time, additional users must wait until the semaphore is released before they are allowed to use the Limo.

## Results

The following images are results of the project:





# Part D: MutexTest

## Summary

The MutexTest application introduced the concept of utilizing a Mutex to control access to a shared resource.

## Results

The following images are the results of the application:

