Table of Content

|  |  |  |
| --- | --- | --- |
| **S. No** | **Content** | **Page No.** |
| 1. | Multi-Threading | 1 |
| 2. | Multi-threading without synchronization | 1 |
| 3. | Multi-threading with synchronization | 1 |
| 4. | Observations and outcomes made without Synchronization | 2 |
| 5. | Observations and outcomes made with Synchronization | 4 |
| 6. | Reference Link for this project | 6 |
| 7. | GitHub reference | 6 |

**Project represented by:**

Sairam Konda - 02936213

Sai Deepak Kailasapu - 029362110

|  |  |
| --- | --- |
| **Tasks** | **Done By** |
| 1.Coding for multi-threading with and without synchronization | Sairam Konda, Sai Deepak Kailasapu |
| 2.Project Report | Sai Deepak Kailasapu |
| 3.Video (YouTube) | Sairam Konda |
| 4.Github | Sairam, Sai Deepak |

**Multi-Threading:**

Multithreading is a specialized form of multitasking, and a multitasking is the feature that allows your computer to run two or more programs concurrently. In general, there are two types of multitasking: process-based and thread-based.

**Multi-Threading without Synchronization:**

Non-Synchronized means that two or more threads can access the methods of that class at any given time. StringBuilder is an example of a non-synchronized class. Generally, a non-synchronized class is not thread safe.

**Multi-Threading with Synchronization:**

Synchronization is a process of handling resource accessibility by multiple thread requests. At times when more than one thread tries to access a shared resource, we need to ensure that resource will be used by only one thread at a time. The process by which this is achieved is called synchronization.

With this project we are going to show the handling and outcomes of multi-threading with synchronization by inserting the given snippet of code and building an executable file around the given snippet.

Text, letter

Description automatically generated

**Observations and outcomes made without Synchronization**

Let us consider an example of ‘Movie ticket booking’ to understand the process better. When multiple users try to book the same seat, multi-threading with synchronization sees that the seat can be accessible only to one user, again depending on the first come first server basis. Whereas, without Synchronization all the users will be able to access the same seat which is not the right process of booking.

Given below are the snapshots of the outputs of multi-Threading without synchronization passing 1, 2, 3(number of threads) values.

**Using one thread:**

Text

Description automatically generated

**Using two threads:**

A screenshot of a computer

Description automatically generated with medium confidence

**Using three threads:**

A screenshot of a computer

Description automatically generated with medium confidence

The above snapshots shows that the final values (shared variable) are different for different threads.

**Observations and outcomes made with Synchronization**

Given below are the snapshots of the outputs of multi-Threading with synchronization passing 1, 2 (number of threads) values.

Here it shows the final values (shared variable) are same for all the threads.

**Using one threads:**

A screenshot of a computer

Description automatically generated with medium confidence

**Using two threads:**

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Reference Link for this project:

https://www.youtube.com/watch?v=XzF7s5ho5hM

GitHub reference:

https://github.com/kondasairam/MiniProject-AdvanceOS-