**Department of Software Engineering**

**Course Code: CS332**

**Class: BESE7AB-BSCS 6AB**

**Lab 01: Multithreaded Programming**

**Date: September 16, 2019**

**Time: 09:00 – 11:50 am**

**Name:** Ahmad Amjad Mughal

**Reg No:** 121672

**Class:** BSCS-6C

**Lab 01: Multithreaded Programming**

**Introduction**

This lab is design to revise the concepts of multithreaded programming.

**Objectives**

Analyze the execution time of single thread and multi-thread program.

**Tools/Software Requirement**

MS Visual Studio 2013 Or any other programming language

**Description**

**Lab Tasks**

**Q1:** Below is the C code to calculate pi, it took 1000 microseconds to calculate and display the calculated value. Your task is as follows:

* Convert the following C code into C#/ Java and calculate the execution time.

**Task 1:**

**import** java.util.Scanner;

**public** **class** PICalculation {

**public** **static** **void** main(String[] args) {

// Create a Scanner object

Scanner input = **new** Scanner (System.*in*);

**long** startTime = System.*currentTimeMillis*();

**double** pi;

**double** sequenceFormula = 0;

**for**(**int** counter = 1; counter < 1000000; counter += 2)

{

sequenceFormula = sequenceFormula + ((1.0 / (2.0 \* counter - 1)) - (1.0 / (2.0 \* counter + 1)));

}

pi = 4 \* sequenceFormula;

**long** stopTime = System.*currentTimeMillis*();

System.*out*.println("Value of PI for 1000000 terms = " + pi);

System.*out*.println("Start Time is = " +(startTime)+ " Milliseconds");

System.*out*.println("Stop Time is = " +(stopTime)+ " Milliseconds");

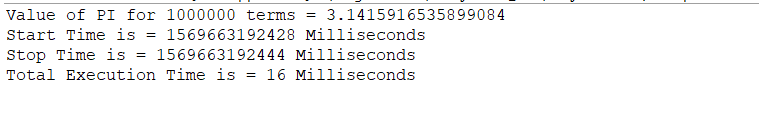
System.*out*.println("Total Execution Time is = " +(stopTime - startTime) + " Milliseconds");

input.close();

}

}

**Output**

****

* Convert your C# /Java code into multithreaded version and analyze the execution time.

**Task 2:**

**class** MultiThreading **extends** Thread

{

**public** **void** run()

{

**try**

{

**double** pi;

**double** sequenceFormula = 0;

**long** startTime = System.*currentTimeMillis*();

**for**(**int** counter = 1; counter < 1000000; counter += 2)

{

sequenceFormula = sequenceFormula + ((1.0 / (2.0 \* counter - 1)) - (1.0 / (2.0 \* counter + 1)));

}

pi = 4 \* sequenceFormula;

System.*out*.println(" Value of PI for 1000000 terms = " + pi);

**long** stopTime = System.*currentTimeMillis*();

System.*out*.println("Total Execution Time is = " +(stopTime - startTime) + " Milliseconds");

}

**catch** (Exception e)

{

// Throwing an exception

System.*out*.println ("Exception is caught");

}

}

}

**public** **class** MainClass

{

**public** **static** **void** main(String[] args)

{

**int** n = 2; // Number of threads

System.*out*.println("Total Number of threads running "+n);

**for** (**int** i=0; i < n; i++)

{

MultiThreading threads = **new** MultiThreading();

threads.start();

}

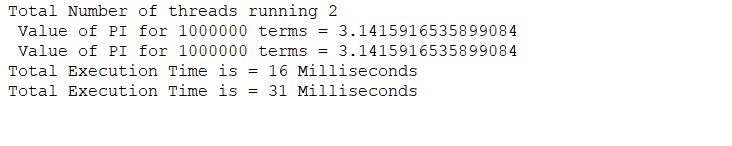
}

}

}

}

**Output**

****

* **Is it better to use multithreaded version compared to single threaded in PI calculation?**

**Ans.** In my case, multithreaded takes more time as the duplicate code is executing by two threads and they output the same PI value. Loops take more time in multithread program as compared to single threaded program. I would prefer single thread PI calculation as it takes slightly less time.

**Deliverables**

Upload well formatted code files on LMS.