Department of Software Engineering Mehran University of Engineering and Technology, Jamshoro			
Course: SW422–Distributed Computing			
Instructor	Rabeea Jaffari	**Practical/Lab No.**	07
Date		**CLOs**	CLO-3: P5 & CLO-3: P3
Signature		**Assessment Score**	

Topic	**To implement multithreading**
Objectives	- Learn multithreading, mutual exclusion and synchronization mechanisms

Lab Discussion: Theoretical concepts and Procedural steps

Lab Tasks

Submission Date:

1. **Multithreading Example Program.**

ThreadDemo.java

```
1  class NewThread implements Runnable {
2      Thread t;
3      NewThread(){
4          t=new Thread(this, "Demo Thread");
5          System.out.println("Child Thread: "+t);
6          t.start();
7
8      }
9              //Jawaria Sattar 16SW04
10     public void run(){
11         try{
12
13 for(int i=5; i>0; i--){
14     System.out.println("Child Thread: "+i);
15     Thread.sleep(500);
16 }
17
18         }
19
20         catch(InterruptedException e){
21             System.out.println("Child Interrupted");
22
23         }
24         System.out.println("Exiting Child thread ");
25
26     }
27 }
28
```

ThreadDemo.java

```
28
29 class ThreadDemo{
30     public static void main(String []args){
31
32         new NewThread();
33         try{
34
35             for(int i=5; i>0; i--){
36     System.out.println("Main Thread: "+i);
37     Thread.sleep(1000);
38 }
39         }
40 catch(InterruptedException e){
41             System.out.println("Main thread Interrupted");
42
43         }
44 System.out.println("Exiting Main thread ");
45
46     }
47
48
49 }
```

Command Prompt

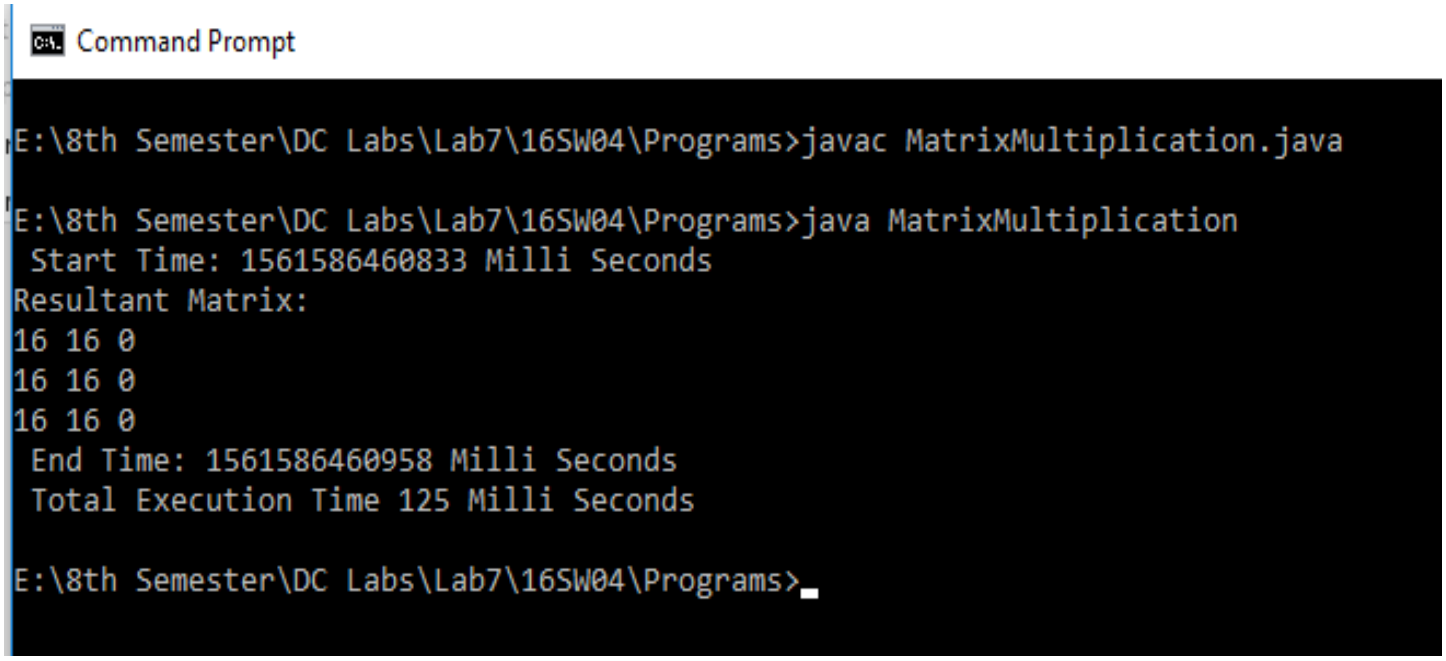
```
E:\8th Semester\DC Labs\Lab7\16SW04>javac *.java

E:\8th Semester\DC Labs\Lab7\16SW04>java ThreadDemo
Child Thread: Thread[Demo Thread,5,main]
Child Thread: 5
Main Thread: 5
Child Thread: 4
Main Thread: 4
Child Thread: 3
Child Thread: 2
Main Thread: 3
Child Thread: 1
Exiting Child thread
Main Thread: 2
Main Thread: 1
Exiting Main thread

E:\8th Semester\DC Labs\Lab7\16SW04>
```

2. **Write the same code for matrix multiplication and divide the code into ten threads. Also note the timestamp at the start and end of the program. Give your conclusion.**

Traditional Method:

MatrixMultiplication.java

```
1  import java.lang.*;
2  public class MatrixMultiplication{
3      public static void main (String []args){
4          long startTime=System.currentTimeMillis();
5          System.out.print(" Start Time: ");
6          System.out.print(startTime);
7          System.out.println(" Milli Seconds");
8
9          int x[][]={{0,4,0}, {0,4,0}, {0,4,0}};
10         int y[][]={{4,0,0}, {4,4,0}, {0,4,0}};
11
12           //Jawaria Sattar 16sw04
13
14         int result[][]=new int[3][3];
15          System.out.println("Resultant Matrix: ");
16         for(int i=0; i<3; i++){
17             for (int j=0; j<3; j++){
18                 result[i][j]=0;
19                 for (int k=0; k<3; k++){
20
21                     result[i][j]+=x[i][k]*y[k][j];
22                 }
23                 System.out.print(result[i][j]+" ");
24             }
25                 System.out.println();
26         }
27
28
29        long endTime=System.currentTimeMillis();
30        System.out.print(" End Time: ");
31         System.out.print(endTime);
32         System.out.println(" Milli Seconds");
33         long totalTime= endTime-startTime;
34          System.out.print(" Total Execution Time "+totalTime);
35          System.out.println(" Milli Seconds");
36     }
37
38 }
```


Command Prompt
E:\8th Semester\DC Labs\Lab7\16SW04\Programs>javac MatrixMultiplication.java
E:\8th Semester\DC Labs\Lab7\16SW04\Programs>java MatrixMultiplication
Start Time: 1561586460833 Milli Seconds
Resultant Matrix:
16 16 0
16 16 0
16 16 0
End Time: 1561586460958 Milli Seconds
Total Execution Time 125 Milli Seconds
E:\8th Semester\DC Labs\Lab7\16SW04\Programs>

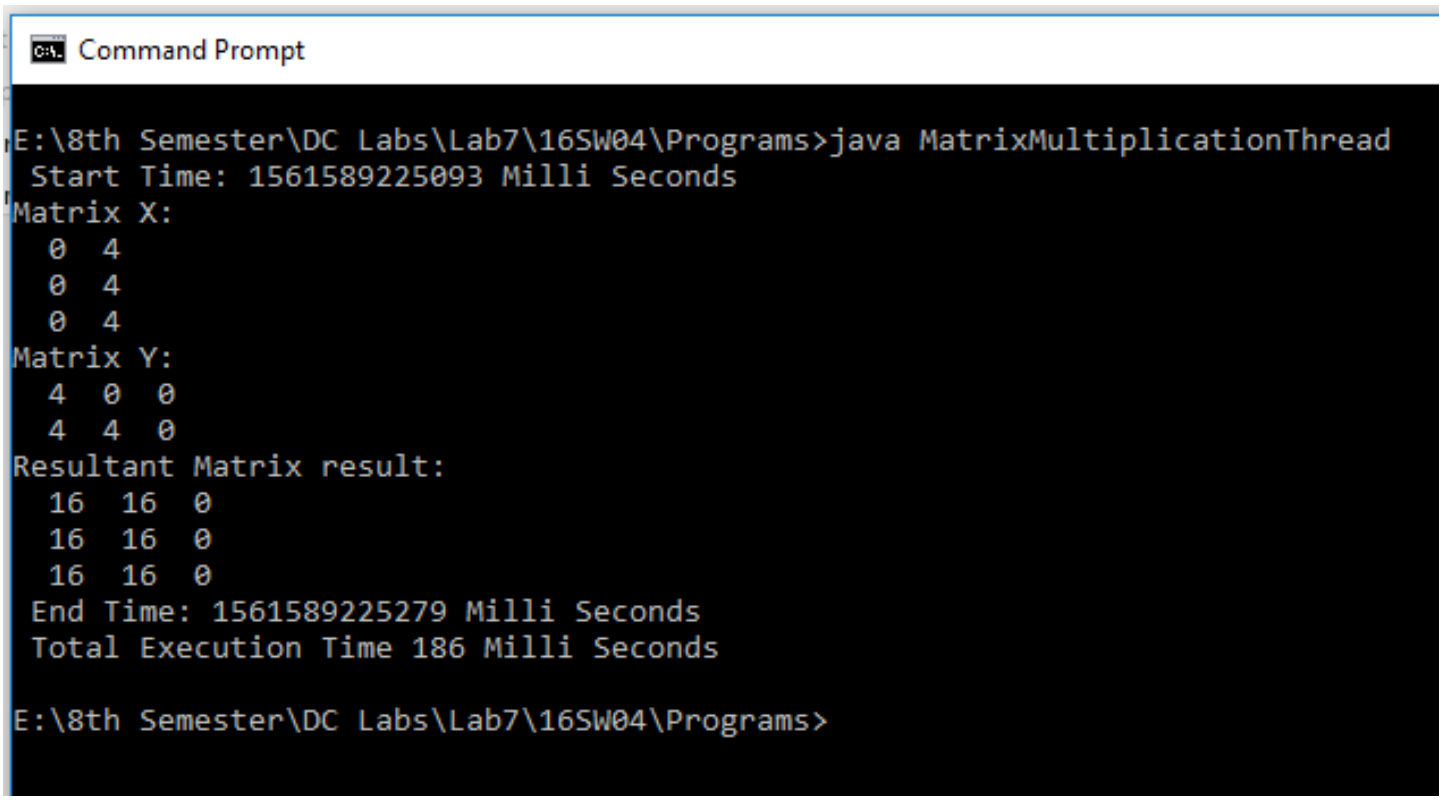
Multithreading Method:

MatrixMultiplicationThread.java*

```
public class  MatrixMultiplicationThread{
    public static final int No_of_Threads=9;
    public static void main (String [] args){
        int row;
        int column;
// 16SSW04   Jawaria Sattar
        int X[][]={{0,4,0}, {0,4,0}, {0,4,0}};
        int Y[][]={{4,0,0}, {4,4,0}, {0,4,0}};
        int result[][]=new int[3][3];
        int threadCount=0;
        long startTime=System.currentTimeMillis();
        System.out.print(" Start Time: ");
        System.out.print(startTime);
        System.out.println(" Milli Seconds");
        Thread [] thread= new Thread[No_of_Threads];

        try{

            for(row=0; row<3; row++){
                for(column=0; column<3; column++){

                    thread[threadCount]=new Thread(new WorkerThread(row, column, X,Y, result));
                    thread[threadCount].start();
                    thread[threadCount].join();
                    threadCount++;
            }

            }
        }
        catch(InterruptedException e){}
            System.out.println("Matrix X: ");
            for(row=0; row<3; row++){
                for(column=0; column<2; column++){
                    System.out.print("  "+X[row][column]);

                }
                   System.out.println();
            }
```

MatrixMultiplicationThread.java*

```
39            System.out.println("Matrix Y: ");
40
41 for(row=0; row<2; row++){
42                   for(column=0; column<3; column++){
43                        System.out.print("  "+Y[row][column]);
44
45                   }
46                      System.out.println();
47              }
48
49              System.out.println("Resultant Matrix result: ");
50              for(row=0; row<3; row++){
51                   for(column=0; column<3; column++){
52                        System.out.print("  "+result[row][column]);
53
54                   }
55                      System.out.println();
56              }
57 long endTime=System.currentTimeMillis();
58         System.out.print(" End Time: ");
59          System.out.print(endTime);
60          System.out.println(" Milli Seconds");
61          long totalTime= endTime-startTime;
62           System.out.print(" Total Execution Time "+totalTime);
63           System.out.println(" Milli Seconds");
64
65
66          }
67
68      }
69
70 class WorkerThread implements Runnable {
71
72 private int row;
73 private int  column;
74 private int X[][];
75 private int Y[][];
76 private int result[][];
77 public WorkerThread(int row, int column, int X[][], int Y[][], int result[][])
78 {
79     this.row=row;
80     this.column=column;
81     this.X=X;
82     this.Y=Y;
83     this.result=result;
84
85 }
86    //16SW04
87 @Override
88 public void run(){
89     for(int j=0; j<Y.length; j++){
90         result[row][column]+=X[row][j]*Y[j][column];
91     }
92
93 }
94
95
96
97 }
98
```



Conclusion:
Execution time of multithreading method is more than the execution time of traditional method.