UNIVERSITY OF WOLLONGONG IN DUBAI

UNIVERSITY OF WOLLONGONG IN DUBAI

ECTE331 (DB224) Real-time Embedded Systems

Name: Ammar Nasreldin Aly

ID: 8536272

Submitted content: Project [Part B] Report

UNIVERSITY OF WOLLONGONG IN DUBAI

Introduction:

The project part B code is designed to demonstrate the use of multithreading and synchronization in Java. Specifically, it involves the creation of two threads, ThreadA and ThreadB, which communicate with each other using a shared Data object. The Data object contains various variables (A1, A2, A3, B1, B2, B3) and Boolean flags (goFunA1, goFunA2, goFunA3, goFunB1, goFunB2, goFunB3) that play a crucial role in the synchronization and communication between the threads.

The main goal of this code is to demonstrate how threads can be used to perform complex calculations and data processing tasks concurrently, while ensuring that the shared resources are accessed and updated safely. The code uses various synchronization mechanisms such as wait() and notify() to coordinate the execution of the threads and ensure that the calculations are performed correctly.

Implementation:

The code is divided into three main components: the Data class, the TaskB class, and the two thread classes, ThreadA and ThreadB.

The Data class contains the variables and flags that are shared between the threads. Lines (4-14).

```
4  class Data{
5
6      int A1,A2,A3,B1,B2,B3;
7      boolean goFunA1= false;
8      boolean goFunA2= false;
9      boolean goFunA3= false;
10     boolean goFunB1= false;
11     boolean goFunB2= false;
12     boolean goFunB3= false;
13
14 }
```

The TaskB class is the main class that creates and starts the threads. It creates a Data object and then starts two threads, ThreadA and ThreadB, which run concurrently. The threads access and update the shared Data object using synchronization mechanisms. Lines (16-40).

```
16 public class TaskB {
17
18     public static void  main(String[] args) throws InterruptedException {
19             int testSize = 10;
20             int i;
21             Data mySample = new Data();
22
23
24             for (i=0; i< testSize ; i++) {
25                 System.out.println("iteration "+ i );
26                 mySample.goFunA2 = false;
27                 mySample.goFunA3 = false;
28                 mySample.goFunB2 = false;
29                 mySample.goFunB3 = false;
30
31                 ThreadA a = new ThreadA(mySample);
32                 ThreadB b = new ThreadB(mySample);
33                 a.start();
34                 b.start();
35
36                 a.join();
37                 b.join();
38             }
39     }
40 }
41
```

UNIVERSITY OF WOLLONGONG IN DUBAI

ThreadA and ThreadB each have their own run method, which contains a series of calculations that involve updating the shared variables. The calculations are performed in a specific order, with each thread waiting for specific conditions to be met before proceeding to the next step.

ThreadA lines (43-90).

```
43 class ThreadA extends Thread{
44     private Data sample;
45
46     public ThreadA(Data sample) {
47         this.sample = sample ;
48     }
49     public void run() {
50         synchronized(sample) {
51             int n = 500 ;
52             sample.A1 = n* (n+1)/2 ;
53             System.out.println("A1 = "+ sample.A1);
54             sample.goFunB2 = true;
55             sample.notify();
56         }
57         synchronized(sample) {
58
59             try {
60                 while (sample.goFunA2 == false) {
61                     sample.wait();
62                     System.out.println("Thread A2 waiting ");
63                 }
64
65                 int n = 300;
66                 sample.A2 = sample.B2 + n*(n+1)/2;
67                 System.out.println("A2 = " + sample.A2);
68                 sample.goFunB3 = true;
69                 sample.notify();
70
71             } catch (InterruptedException e) {
72                 e.printStackTrace();
73             }
74         }
75
76         synchronized(sample) {
77             try {
78                 while (sample.goFunA3 == false) {
79                     sample.wait();
80                     System.out.println("Thread A3 waiting ");
81                 }
82                 int n = 400;
83                 sample.A3 = sample.B3 + n*(n+1)/2;
84                 System.out.println("A3 = " + sample.A3);
85             } catch (InterruptedException e) {
86                 e.printStackTrace();
```

ThreadB lines (96-147).

```
96 class ThreadB extends Thread {
97     private Data sample;
98
99     public ThreadB(Data sample) {
100         super();
101         this.sample = sample;
102     }
103
104     public void run(){
105
106         synchronized(sample) {
107             int n = 250;
108             sample.B1 = n*(n+1)/2;
109             System.out.println("B1 = " + sample.B1);
110             sample.notify();
111         }
112
113         synchronized(sample) {
114             try {
115                 while (sample.goFunB2 == false) {
116                     sample.wait();
117                     System.out.println("Thread B2 waiting ");
118                 }
119
120                 int n = 200;
121                 sample.B2 = sample.A1 + n*(n+1)/2;
122                 System.out.println("B2 = " + sample.B2);
123                 sample.goFunA2 = true;
124                 sample.notify();
125             } catch (InterruptedException e) {
126                 e.printStackTrace();
127             }
128         }
129
130         synchronized(sample) {
131             try {
132                 while (sample.goFunB3 == false) {
133                     sample.wait();
134                     System.out.println("Thread B3 waiting ");
135                 }
136
137                 int n = 400;
138                 sample.B3 = sample.A2 + n*(n+1)/2;
139                 System.out.println("B3 = " + sample.B3);
```

In this code, ThreadA waits for goFunB2 to be true before performing its second calculation, and ThreadB waits for goFunA2 to be true before performing its second calculation. This ensures that the calculations are performed in a specific order and that the shared variables are updated correctly.

UNIVERSITY OF WOLLONGONG IN DUBAI

Calculations:

A1 = 500 * (500 + 1)/2 = 125250

B1 = 250 * (250 + 1)/2 = 31375

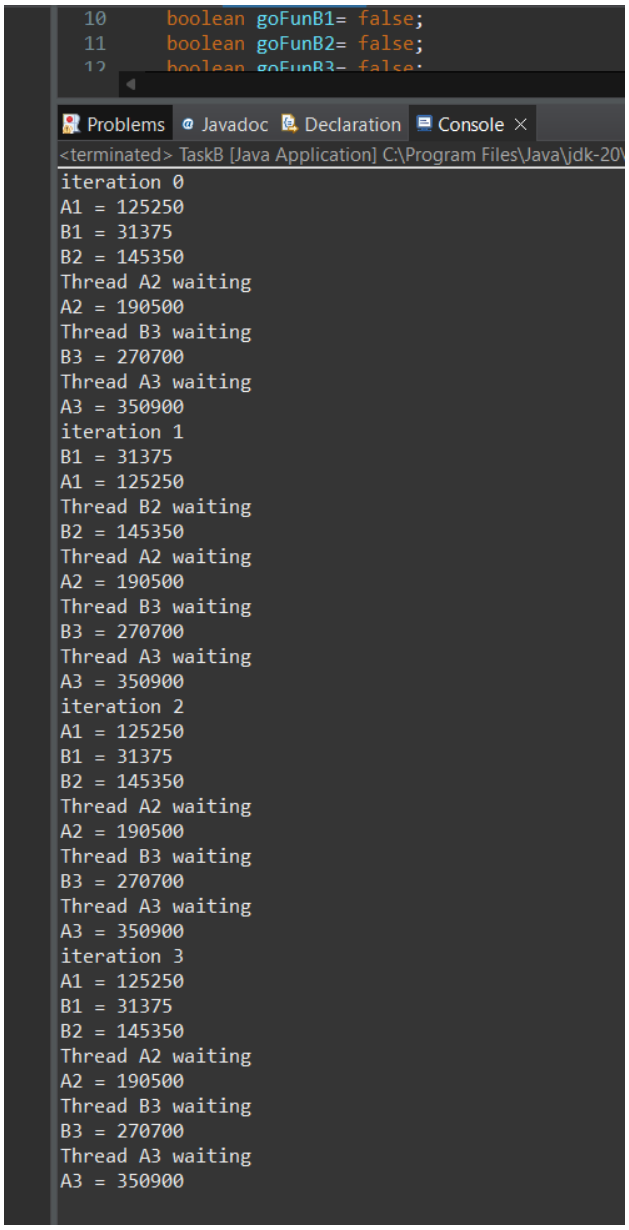
B2 = A1 + (200 * (200 + 1)/2) = 145350

A2 = B2 + (300 * (300 + 1)/2) = 190500

B3 = A2 + (400 * (400 + 1)/2) = 270700

A3 = B3 + (400 * (400 + 1)/2) = 350900

Results:



The code produces the expected results for four iterations from 0 to 3. Notably, the output shows identical results for each iteration, which confirms that the code is functioning correctly. Moreover, the waiting messages printed during execution demonstrate the proper usage of synchronization mechanisms between threads.

Specifically, the output shows that:

- B2, A2, B3 and A3 are correctly calculated based on the values of A1, B2, A2 and B3, respectively.
- The waiting messages printed by ThreadA and ThreadB indicate that they are correctly waiting for each other to complete their calculations before proceeding.

This outcome validates the effectiveness of the synchronization mechanisms implemented in the code, ensuring that the threads communicate correctly and avoid race conditions.