Lab1

Współbieżność w Javie

Kacper Ćwiertnia
02.10.2023

Treść Zadania

1. Napisać program (szkielet), który uruchamia 2 wątki, z których jeden zwiększa wartość zmiennej całkowitej o 1, drugi wątek zmniejsza wartość o 1. Zakładając że na początku wartość zmiennej Counter była 0, chcielibyśmy wiedzieć jaka będzie wartość tej zmiennej po wykonaniu 10000 operacji zwiększania i zmniejszania przez obydwa wątki.
2. Na podstawie 100 wykonań programu z p.1, stworzyć histogram końcowych wartości zmiennej Counter.
3. Spróbować wprowadzić mechanizm do programu z p.1, który zagwarantowałby przewidywalną końcową wartość zmiennej Counter. Nie używać żadnych systemowych mechanizmów, tylko swój autorski.
4. Napisać sprawozdanie z realizacji pp. 1-3, z argumentacją i interpretacją wyników.

Rozwiązanie

1. Dopisuję kod kolejnych klas zdefiniowanych w szkielecie.

```
class IThread extends Thread {
    2 usages
    private final Counter counter;
    1 usage
    public IThread(Counter couter){
        this.counter = couter;
    }

    public void run(){
        for(int i = 0; i < 10000; i++){
            this.counter.inc();
        }
    }
}
```

```java
class DThread extends Thread {

    2 usages
    private final Counter counter;
    1 usage
    public DThread(Counter couter){
        this.counter = couter;
    }

    public void run(){
        for(int i = 0; i < 10000; i++){
            this.counter.dec();
        }
    }
}
```

```java
public class Race {
    public static void main(String[] args) {
        Counter cnt = new Counter( n: 0);
        IThread incThread = new IThread(cnt);
        DThread decThread = new DThread(cnt);

        incThread.start();
        decThread.start();

        try {
            incThread.join();
            decThread.join();

            System.out.println("stan=" + cnt.value());
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
}
```

2. Dopisuję kod odpowiedzialny za 100 iteracji i zapisywanie ich wartości do pliku.

```
public class Race {
    public static void main(String[] args) {
        try {
            File iterationsFile = new File( pathname: "data/100iterations.csv");
            FileWriter iterationsFileWriter = new FileWriter(iterationsFile);
            iterationsFileWriter.write( str: "Iteration, Value\n");

            for(int i = 0; i < 100; i++) {
                Counter cnt = new Counter( n: 0);
                IThread incThread = new IThread(cnt);
                DThread decThread = new DThread(cnt);

                incThread.start();
                decThread.start();

                incThread.join();
                decThread.join();

                iterationsFileWriter.write( str: i + "," + cnt.value()+"\n");
            }

            iterationsFileWriter.close();
        } catch (IOException | InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
}
```

3. Aby usystematyzować wartość licznika trzeba synchronizować dostęp jego wartości między wątkami. W tym celu, tworzę zmienną isOccupied, która identyfikuję czy licznik aktualnie jest zajęty czy nie. Dodatkowo dopisuję 3 funkcje obsługujące te zmienną (sprawdzenie czy licznik jest zajęty, zajęcie go i oddanie go)

```
    private boolean isOccupied = false;
    1 usage
```

```
public boolean getIsOccupied(){
    return isOccupied;
}
2 usages
public void occupy(){
    isOccupied = true;
}
2 usages
public void free(){
    isOccupied = false;
}
```

Następnie w wątkach używam nowych elementów tak aby nie nadpisywały jednocześnie wartości licznika

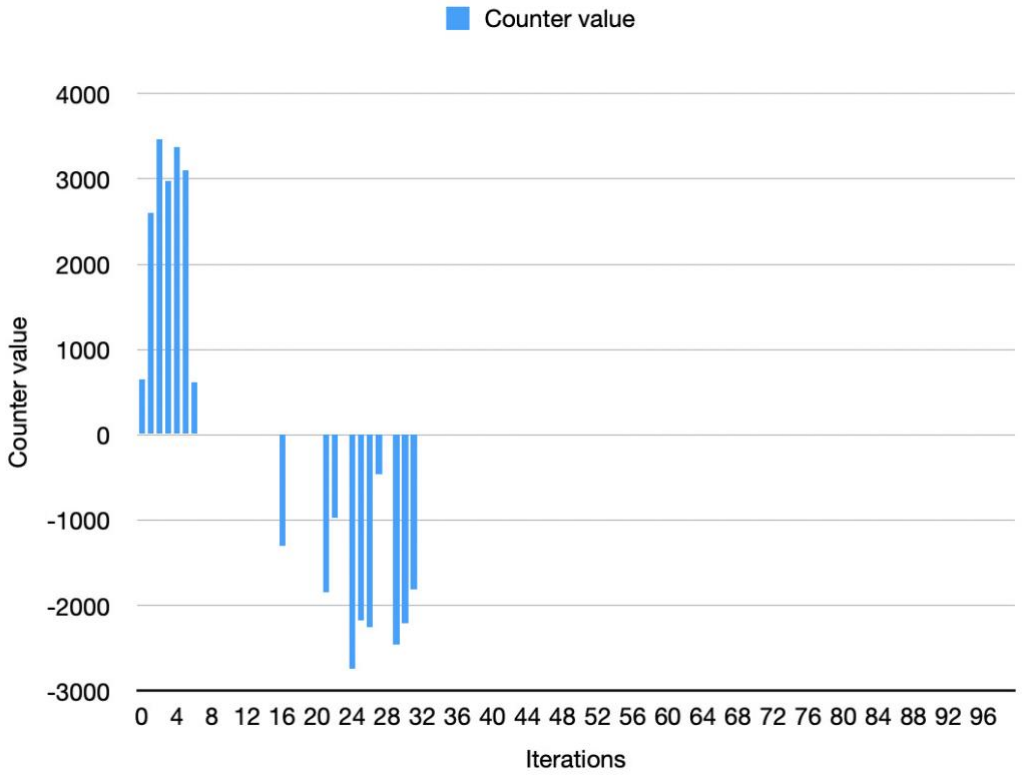
```
for(int i = 0; i < 10000; i++){
    while(counter.getIsOccupied()){
        try {
            Thread.sleep( millis: 100);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
    counter.occupy();
    counter.inc();
    counter.free();
}
```

```java
for(int i = 0; i < 10000; i++){
    while(counter.getIsOccupied()){
        try {
            Thread.sleep( millis: 100);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
    counter.occupy();
    counter.dec();
    counter.free();
}
```

Wyniki

2. Wyniki do podpunktu 2

Iteration	Value	19	0	40	0	60	0	80	0
0	652	20	0	41	0	61	0	81	0
1	2597	21	-1846	42	0	62	0	82	0
2	3462	22	-974	43	0	63	0	83	0
3	2982	23	0	44	0	64	0	84	0
4	3368	24	-2744	45	0	65	0	85	0
5	3097	25	-2178	46	0	66	0	86	0
6	619	26	-2255	47	0	67	0	87	0
7	0	27	-465	48	0	68	0	88	0
8	0	28	0	49	0	69	0	89	0
9	0	29	-2461	50	0	70	0	90	0
10	0	30	-2208	51	0	71	0	91	0
11	0	31	-1809	52	0	72	0	92	0
12	0	32	0	53	0	73	0	93	0
13	0	33	0	54	0	74	0	94	0
14	0	34	0	55	0	75	0	95	0
15	0	35	0	56	0	76	0	96	0
16	-1303	36	0	57	0	77	0	97	0
17	0	37	0	58	0	78	0	98	0
18	0	38	0	59	0	79	0	99	0
19	0	39	0						



3. Wyniki do podpunktu 3

Iteration	Value
0	0
1	0
2	0
3	-16
4	1
5	0
6	0
7	296
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0
33	0
34	0
35	0
36	0
37	0
38	0
39	0
40	0
41	0
42	0
43	0
44	0
45	0
46	0
47	0
48	0
49	0
50	0
51	0
52	0
53	0
54	0
55	0
56	0
57	0
58	0
59	0
60	0
61	0
62	0
63	0
64	0
65	0
66	0
67	0
68	0
69	0
70	0
71	0
72	0
73	0
74	0
75	0
76	0
77	0
78	0
79	0
80	0
81	0
82	0
83	0
84	0
85	0
86	0
87	0
88	0
89	0
90	0
91	0
92	0
93	0
94	0
95	0
96	0
97	0
98	0
99	0

Wnioski

Jak możemy zauważyć w podpunkcie 2 z powodu braku sychornizacji między wątkami dochodzi do niekontrolowanej edycji wartości countera co skutkuje jego niepoprawnym wynikem na koniec programu. W podpunkcie 3 dzięki zaimplementowaniu prostego mechanizmu sycnhronizacji wartość licznika zaczęła być bardziej przewidywalana.

Bibliografia

https://www.artima.com/insidejvm/ed2/threadsynch3.html

Zadanie dodatkowe

1. Teoretycznie najmniejszą wartością będzie 5, ponieważ, każdy z N wątków odczyta wartość 0, potem doda 1 i zapisze 1, następnie znów, każdy odczyta 1, doda 1 i zapisze 2. W taki sposób na końcu licznik będzie miał wartość 5.
2. Skoro dla 1 wątku najmniejszą (i jedyną) wartością jest 5, a powyżej przedstawiłem sposób działania dla większej ilości wątków to można wnioskować, że wartość licznika dla dowolnej ilości wątków na koniec programu zawsze będzie wynosiła 5.