



| | | | |
|---|---|---|---|
|  | Politechnika Bydgoska im. J.J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki | |  |
| Przedmiot: | Programowanie współbieżne | IST Studia stacjonarne Semestr 5, 2021/2022 | |
| Temat: | Wątki i pule wątków | | |
| Numer lab.: | 2 | Data wykonania: | 2021.10.27 |
| Prowadzący | mgr inż. Karol Hartwig | Data oddania: | 2022.01.07 |
| Autor: | Mariusz Jackowski | Indeks: | 113031 |

Instrukcja:

- Utworzyć projekt maven lub gradle.
- Przekleić zawartość załączników do pliku *Item.java*, uzupełnić brakujące importy.

Wątki

- Utworzyć listę 100 obiektów klasy *Item*
- Utworzyć 4 wątki produkujące (wywołujące metodę *produceMe()*), oraz 3 wątki konsumujące (wywołujące metodę *consumeMe()*).
- Zaimplementować aplikację w taki sposób aby wszystkie obiekty z listy były “wyprodukowane” przez wątki produkujące i “skonsumowane” przez wątki konsumujące. Zrealizować to w taki sposób aby czas zminimalizować czas produkcji i konsumpcji wszystkich obiektów.

Pule wątków

- Utworzyć pulę wątków.
- Wykonać zadanie z poprzedniego podpunktu tak, aby wykorzystać w tym celu pulę wątków, z tym że bez podziału na wątki produkujące i konsumujące (pojedynczy wątek, w zależności od potrzeb może wykonywać obydwa zadania).
- Wykonać testy dla różnych typów puli wątków.

Stream API

- Wykonać zadanie z poprzedniego punktu wykorzystując Stream API (*parallelStream*)

Materiały referencyjne:

- [Item.java](#)

Repozytorium:



Main.java

```
public class Main {
    private static TimeOperations T0 = new TimeOperations();
    private static ThreadPoolOperations TPO = new ThreadPoolOperations();
    private static volatile List<Item> items = produce100_Items();

    public static void main(String[] args) {
        int choice;
        Scanner scan = new Scanner(System.in);

        System.out.println("1. 7 watkow (4 produceMe, 3 consumeMe). ");
        System.out.println("2. parallelStream. ");
        System.out.println("3. Pula watkow. ");
        System.out.println("Wybierz.");
        choice = scan.nextInt();

        switch (choice) {
            case 1:
                SingleThreadsOperations.startSingleThreads(items);
                break;
            case 2:
                double previous = System.nanoTime();
                MultiThreadsOperations.runParallelStream(items);
                double now = System.nanoTime();
                T0.showActionTime(now, previous);
                break;
            case 3:
                TPO.runFixedThreadPool(items);
                break;
            default:
                System.out.println("Switch error");
                break;
        }
    }
}
```

Utworzenie 100 obiektów klasy Item:

```
private static List<Item> produce100_Items() {
    return Stream
        .generate(Item::new)
        .limit(100)
        .collect(Collectors.toList());
}
```

Utworzenie 7 wątków (4 produceMe i 3 consumeMe)

```
public class SingleThreadsOperations {
    private static TimeOperations TO = new TimeOperations();
    private static double now = 0;
    private static Thread tp;
    private static Thread tc;

    private static void singleThreadProduce(int singleThreadID, List<Item> items) {
        for (int i = singleThreadID; i < items.size(); i += 4)
            items.get(i).produceMe();
    }

    private static void singleThreadConsume(int singleThreadID, List<Item> items) {
        for (int i = singleThreadID; i < items.size(); i += 3)
            items.get(i).consumeMe();
    }

    public static void startSingleThreads(List<Item> items) {
        double previous = System.nanoTime();

        for (int i = 0; i < 4; i++) {
            int finalI1 = i;
            tp = new Thread(() -> singleThreadProduce(finalI1, items));
            tp.start();

            if (i < 3) {
                int finalI = i;
                tc = new Thread(() -> singleThreadConsume(finalI, items));
                tc.start();
            }
        }

        while (tp.isAlive() || tc.isAlive())
            now = System.nanoTime();

        TO.showActionTime(now, previous);
    }
}
```

Stream Api (parallelStream)

```
public class MultiThreadsOperations {

    public static void runParallelStream(List<Item> items) {
        runParallelProducer(items);
        runParallelConsumer(items);
    }

    private static void runParallelProducer(List<Item> items) { items.parallelStream().forEach(Item::produceMe); }

    private static void runParallelConsumer(List<Item> items) { items.parallelStream().forEach(Item::consumeMe); }
```

Pule wątków

```
public class ThreadPoolOperations {
    private static double previous;
    private static int poolSize = 12;
    private int working = 0;
    private int producerId = 0;
    private static int consumerId = 0;
    private static TimeOperations t0 = new TimeOperations();

    public void runFixedThreadPool(List<Item> items) {
        setThreadPool();
        previous = System.nanoTime();
        ExecutorService threadsManager = Executors.newFixedThreadPool(poolSize);

        for (int i = 0; i < poolSize; i++) {
            threadsManager.execute(new Thread(() -> {
                try {
                    manageThreads(items, poolSize);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            })
        );
        threadsManager.shutdown();
    }

    private void setThreadPool() {
        Scanner scan = new Scanner(System.in);
        System.out.println("Please write amount of threads in thread pool, but please take a even number:");
        int pool = scan.nextInt();

        while (pool % 2 != 0) {
            System.out.println("Please take a even number: ");
            pool = scan.nextInt();
        }
        poolSize = pool;
    }
}
```

Wyniki:

```
1. 7 watkow (4 produceMe, 3 consumeMe).  
2. parallelStream.  
3. Pula watkow.  
Wybierz.|
```

Wybierz.

1

```
Producing: Items.Item-0  
Producing: Items.Item-1  
Producing: Items.Item-2  
Producing: Items.Item-3  
Produced: Items.Item-0  
Produced: Items.Item-2  
Producing: Items.Item-6  
Produced: Items.Item-1  
Producing: Items.Item-5  
Produced: Items.Item-3  
Producing: Items.Item-7  
Consuming: Items.Item-1
```

Wybierz.

2

```
Producing: Items.Item-65  
Producing: Items.Item-31  
Producing: Items.Item-90  
Producing: Items.Item-81  
Producing: Items.Item-15  
Producing: Items.Item-6  
Producing: Items.Item-3  
Producing: Items.Item-43
```

Wybierz.

3

podaj ilosc watkow w puli (parzyste):

4

```
Producing: Items.Item-0  
Producing: Items.Item-1  
Produced: Items.Item-0  
Producing: Items.Item-2  
Consuming: Items.Item-0  
Produced: Items.Item-1  
Producing: Items.Item-3  
Consuming: Items.Item-1
```