



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Лабораторна робота №3

Технології паралельних обчислень

Виконав студент групи ІТ-03: Чабан А.Є.		Перевірив:
		Дифучина О.Ю
		Дата:
		Оцінка:

Київ 2023

Завдання:

5.3 Завдання до комп'ютерного практикуму 3 «Розробка паралельних програм з використанням механізмів синхронізації: синхронізовані методи, локери, спеціальні типи»

1. Реалізуйте програмний код, даний у лістингу, та протестуйте його при різних значеннях параметрів. Модифікуйте програму, використовуючи методи управління потоками, так, щоб її робота була завжди коректною. Запропонуйте три різних варіанти управління. **30 балів.**
2. Реалізуйте приклад Producer-Consumer application (див. <https://docs.oracle.com/javase/tutorial/essential/concurrency/guardmeth.html>).

Модифікуйте масив даних цієї програми, які читаються, у масив чисел заданого розміру (100, 1000 або 5000) та протестуйте програму. Зробіть висновок про правильність роботи програми. **20 балів.**

3. Реалізуйте роботу електронного журналу групи, в якому зберігаються оцінки з однієї дисципліни трьох груп студентів. Кожного тижня лектор і його 3 асистенти виставляють оцінки з дисципліни за 100-бальною шкалою. **40 балів.**
4. Зробіть висновки про використання методів управління потоками в java. **10 балів.**

Хід виконання:

Завдання 1:

Надано лістинг коду для симуляції переказу коштів між акаунтами банку, необхідно синхронізувати транзакції, аби списання і зарахування з рахунку на рахунок проходило без втрат.

Перший спосіб – зробити метод `transfer()` `synchronized`

```
1 usage  Anton Chaban
public synchronized void transfer(int from, int to, int amount) { // synchronized version 1
    accounts[from] -= amount;
    accounts[to] += amount;
    ntransacts++;
    if (ntransacts % NTEST == 0)
        test();
}
```

Другий спосіб – наша транзакція має проходити повністю, тому ми можемо обгорнути її в синхронізований блок

```

1 usage  Anton Chaban
public void transfer(int from, int to, int amount) {
    synchronized (this) {
        accounts[from] -= amount;
        accounts[to] += amount;
        ntransacts++;
        if (ntransacts % NTEST == 0)
            test();
    }
}

```

Третій спосіб – використати локери

```

2 usages
private Lock lock = new ReentrantLock();

```

```

1 usage  Anton Chaban
public void transfer(int from, int to, int amount) { //
    lock.lock();
    try {
        accounts[from] -= amount;
        accounts[to] += amount;
        ntransacts++;
        if (ntransacts % NTEST == 0)
            test();
    } finally {
        lock.unlock();
    }
}

```

Четвертий – використати Atomic змінну, але цього не достатньо, так як атомік змінні дають гарантії безпеки для операції над об'єктом, а не над усією транзакцією, тому всеодно потрібно використовувати synchronized блок, або інший спосіб.

```

1 usage  Anton Chaban
public void transfer(int from, int to, int amount) { // ver 2 Atomic
    synchronized (this) {
        accounts.addAndGet(from, -amount);
        accounts.addAndGet(to, amount);
        if (ntransacts.incrementAndGet() % NTEST == 0) {
            test();
            Thread.currentThread().interrupt();
        }
    }
}
}

```

Завдання 2:

```

1  package lab3.task2;
2
3  import java.util.Random;
4
5  1 usage  Anton Chaban *
public class Consumer implements Runnable {
    3 usages
6  private Drop drop;
7
8  1 usage  Anton Chaban
    public Consumer(Drop drop) {
9      this.drop = drop;
10 }
11
12  Anton Chaban *
    public void run() {
13        Random random = new Random();
14        for (int message = drop.take(); message != -1; message = drop.take()) {
15            System.out.format("MESSAGE RECEIVED: %s\n", message);
16            try {
17                Thread.sleep(10/*random.nextInt(5000)*/);
18            } catch (InterruptedException e) {
19            }
20        }
21    }
22 }

```

```

1 package lab3.task2;
2
3 import java.util.Random;
4
5 1 usage  Anton Chaban *
6 public class Producer implements Runnable {
7     3 usages
8     private Drop drop;
9
10    1 usage  Anton Chaban
11    > public Producer(Drop drop) { this.drop = drop; }
12
13    Anton Chaban *
14    public void run() {
15        var size = 500;
16        var importantInfo = new int[size];
17        for (int i = 0; i < importantInfo.length; i++) {
18            importantInfo[i] = i + 1;
19        }
20        Random random = new Random();
21        for (int i = 0; i < importantInfo.length; i++) {
22            drop.put(importantInfo[i]);
23            try {
24                Thread.sleep(10/*random.nextInt(5000)*/);
25            } catch (InterruptedException e) {
26            }
27        }
28        drop.put( numMsg: -1);
29    }
30 }

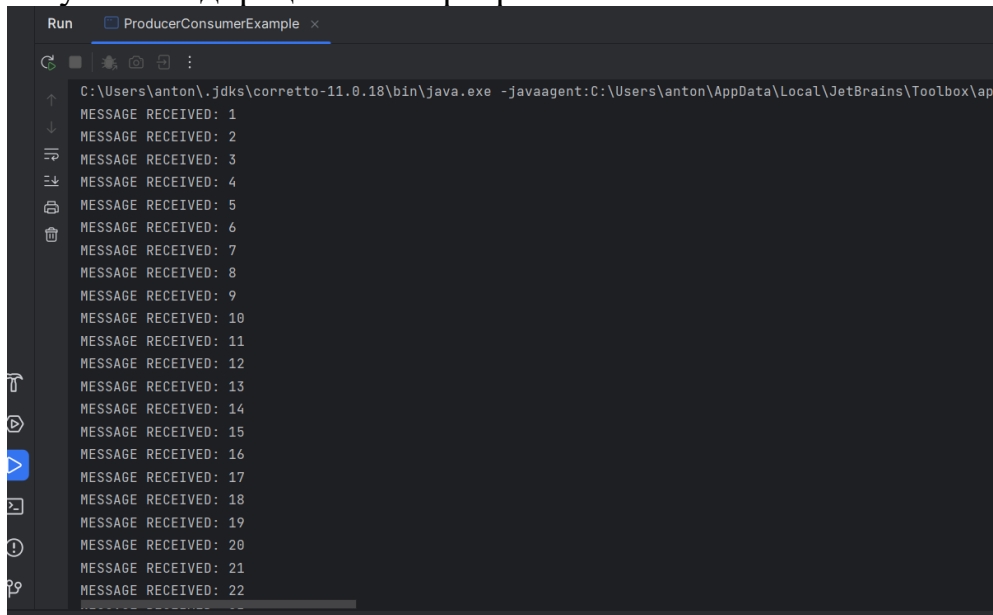
```

```

3 public class Drop {
    2 usages
4     private int numMsg;
    4 usages
5     private boolean empty = true;
6
    2 usages Anton Chaban *
7     public synchronized int take() {
8         while (empty) {
9             try {
10                 wait();
11             } catch (InterruptedException e) {
12             }
13         }
14         empty = true;
15         notifyAll();
16         return numMsg;
17     }
18
    2 usages Anton Chaban *
19     public synchronized void put(int numMsg) {
20         while (!empty) {
21             try {
22                 wait();
23             } catch (InterruptedException e) {
24             }
25         }
26         empty = false;
27         this.numMsg = numMsg;
28         notifyAll();
29     }
30 }

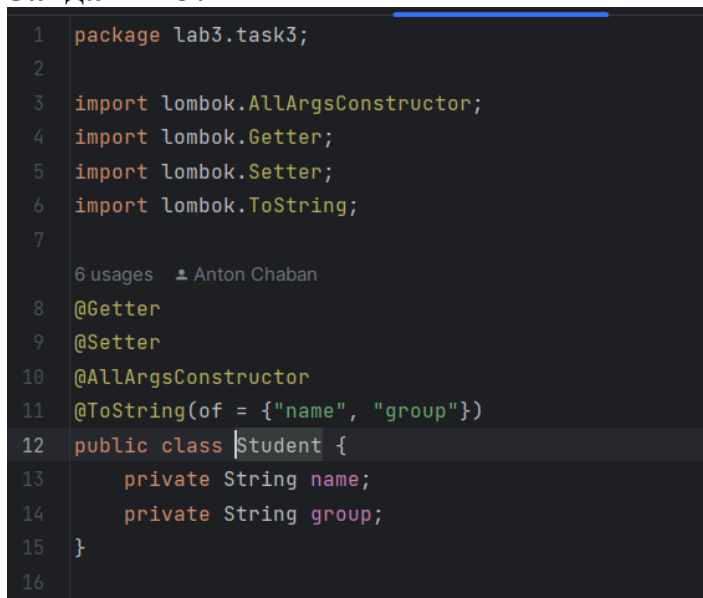
```

Результат відпрацювання програми:



```
Run ProducerConsumerExample x
C:\Users\anton\jdk\corretto-11.0.18\bin\java.exe -javaagent:C:\Users\anton\AppData\Local\JetBrains\Toolbox\app
MESSAGE RECEIVED: 1
MESSAGE RECEIVED: 2
MESSAGE RECEIVED: 3
MESSAGE RECEIVED: 4
MESSAGE RECEIVED: 5
MESSAGE RECEIVED: 6
MESSAGE RECEIVED: 7
MESSAGE RECEIVED: 8
MESSAGE RECEIVED: 9
MESSAGE RECEIVED: 10
MESSAGE RECEIVED: 11
MESSAGE RECEIVED: 12
MESSAGE RECEIVED: 13
MESSAGE RECEIVED: 14
MESSAGE RECEIVED: 15
MESSAGE RECEIVED: 16
MESSAGE RECEIVED: 17
MESSAGE RECEIVED: 18
MESSAGE RECEIVED: 19
MESSAGE RECEIVED: 20
MESSAGE RECEIVED: 21
MESSAGE RECEIVED: 22
```

Завдання 3:



```
1 package lab3.task3;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Getter;
5 import lombok.Setter;
6 import lombok.ToString;
7
8 @Getter
9 @Setter
10 @AllArgsConstructor
11 @ToString(of = {"name", "group"})
12 public class Student {
13     private String name;
14     private String group;
15 }
16
```

```

1 package lab3.task3;
2
3 import lombok.Getter;
4
5 import java.util.ArrayList;
6 import java.util.HashMap;
7 import java.util.List;
8 import java.util.Map;
9
10 1 usage Anton Chaban
11 @Getter
12 public class Journal {
13     private Map<Student, ArrayList<Integer>> journal = new HashMap<>();
14
15     1 usage Anton Chaban
16     public synchronized void setMark(Student student, Integer mark) {
17         if (journal.containsKey(student)) {
18             journal.get(student).add(mark);
19         } else {
20             journal.put(student, new ArrayList<>(List.of(mark)));
21         }
22     }
23
24     1 usage Anton Chaban
25     public synchronized void printJournal() {
26         for (var entry : journal.entrySet()) {
27             System.out.println(entry.getKey() + " " + entry.getValue());
28         }
29     }
30 }

```



```

Anton Chaban *
7 public static void main(String[] args) throws InterruptedException {
8     var students = List.of(
9         new Student(name: "Anton", group: "IT-03"),
10        new Student(name: "Petro", group: "IT-01"),
11        new Student(name: "Sergiy", group: "IT-04"),
12        new Student(name: "Oleksii", group: "IT-01")
13    );
14
15    var journal = new Journal();
16
17    var threads = new ArrayList<Thread>();
18
19    for (int i = 0; i < 4; i++) {
20        final Thread t = new Thread(() -> {
21            for (int j = 0; j < students.size(); j++) {
22                var mark = (int) (Math.random() * 100) + 1;
23                journal.setMark(students.get(j), mark);
24                System.out.println("Thread " + Thread.currentThread().getName() + " set mark for "
25                    + students.get(j) + " =" + mark);
26            }
27        });
28        t.start();
29        threads.add(t);
30    }
31    for (Thread thread : threads) {
32        thread.join();
33    }
34    journal.printJournal();
35 }
36 }
37

```

Результат:

```

C:\Users\anton\.jdk\corretto-11.0.18\bin\java.exe -javaagent:C:\Users\anton\AppData\Lo
Thread Thread-3 set mark for Student(name=Anton, group=IT-03) =8
Thread Thread-3 set mark for Student(name=Petro, group=IT-01) =69
Thread Thread-1 set mark for Student(name=Anton, group=IT-03) =68
Thread Thread-2 set mark for Student(name=Anton, group=IT-03) =34
Thread Thread-1 set mark for Student(name=Petro, group=IT-01) =11
Thread Thread-3 set mark for Student(name=Sergiy, group=IT-04) =77
Thread Thread-1 set mark for Student(name=Sergiy, group=IT-04) =99
Thread Thread-2 set mark for Student(name=Petro, group=IT-01) =84
Thread Thread-1 set mark for Student(name=Oleksii, group=IT-01) =53
Thread Thread-0 set mark for Student(name=Anton, group=IT-03) =66
Thread Thread-0 set mark for Student(name=Petro, group=IT-01) =71
Thread Thread-3 set mark for Student(name=Oleksii, group=IT-01) =37
Thread Thread-0 set mark for Student(name=Sergiy, group=IT-04) =78
Thread Thread-2 set mark for Student(name=Sergiy, group=IT-04) =62
Thread Thread-2 set mark for Student(name=Oleksii, group=IT-01) =99
Thread Thread-0 set mark for Student(name=Oleksii, group=IT-01) =24
Student(name=Sergiy, group=IT-04) [77, 99, 62, 78]
Student(name=Oleksii, group=IT-01) [37, 53, 24, 99]
Student(name=Anton, group=IT-03) [68, 34, 8, 66]
Student(name=Petro, group=IT-01) [69, 11, 84, 71]

Process finished with exit code 0

```

Використаємо синхронізовані методи `setMark()`, для того щоб запобігти ситуації з гонкою даних та некоретного запису оцінок до журналу.