

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт

З лабораторної роботи № 3 з дисципліни
«Технології паралельних обчислень»

Тема: «Розробка паралельних програм з використанням механізмів
синхронізації: синхронізовані методи, локери, спеціальні типи»

Виконав(ла)

ІП-14 Бабіч Денис

(шифр, прізвище, ім'я, по батькові)

Перевірив

Дифучина О. Ю.

(шифр, прізвище, ім'я, по батькові)

Київ 2024

ОСНОВНА ЧАСТИНА

Мета роботи: 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.



Рисунок 1.1 – Результат роботи несинхронізованого підходу

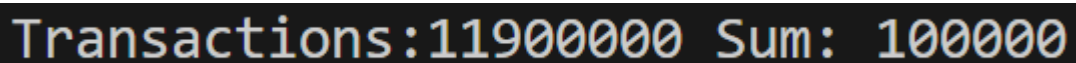


Рисунок 1.2 – Результат роботи синхронізованого підходу

Спосіб обробки фінансів не є синхронізованим, тому, якщо кілька потоків намагаються виконати транзакцію одночасно, це може призвести до проблеми Race Condition, коли баланс рахунку може стати від'ємним або загальна сума всіх рахунків може не залишатися постійною.

- 1.1. Синхронізація за допомогою локера.

Лістинг методу `transferWithLock`

```
public void transferWithLock(int senderIndex, int receiverIndex, int amount) {
    this.lock.lock();
    try {
        this.accounts[senderIndex] -= amount;
        this.accounts[receiverIndex] += amount;

        ++this.transactionsCount;
```

```

        if (this.transactionsCount % NTEST == 0) {
            test();
        }
    }
    finally {
        this.lock.unlock();
    }
}

```

1.2. Синхронізація за допомогою синхронізованого блока.

Лістинг методу **transferWithSynchronizedBlock**

```

public void transferWithSynchronizedBlock(int senderIndex, int receiverIndex, int
amount) {
    synchronized(this)
    {
        this.accounts[senderIndex] -= amount;
        this.accounts[receiverIndex] += amount;

        ++this.transactionsCount;

        if (this.transactionsCount % NTEST == 0) {
            test();
        }
    }
}

```

1.3. Синхронізація за допомогою синхронізованого методу.

Лістинг методу **transferWithSynchronizedMethod**

```
public synchronized void transferWithSynchronizedMethod(int senderIndex, int
receiverIndex, int amount)
{
    this.accounts[senderIndex] -= amount;
    this.accounts[receiverIndex] += amount;

    ++this.transactionsCount;

    if (this.transactionsCount % NTEST == 0)
    {
        test();
    }
}
```

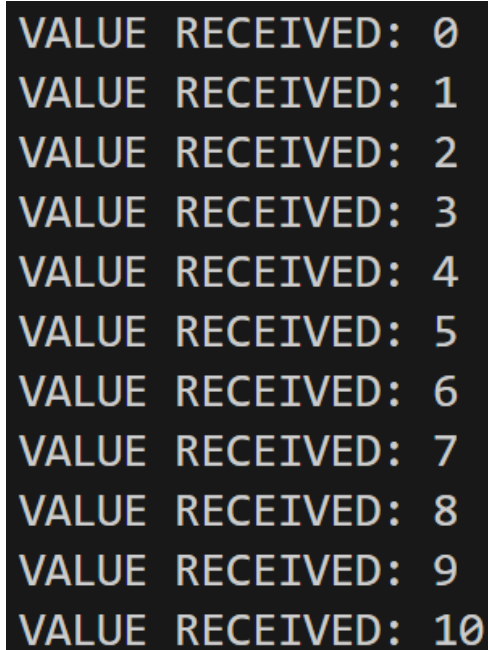
2. Реалізуйте приклад Producer-Consumer application. Модифікуйте масив даних цієї програми, які читаються, у масив чисел заданого розміру (100, 1000 або 5000) та протестуйте програму. Зробіть висновок про правильність роботи програми.

Лістинг класу **Producer**

```
final class Producer implements Runnable {
    public static final int INTERRUPT_VALUE = -1;
    private Drop drop;
    private int[] dataBuffer;

    public Producer(Drop drop, int dataBufferSize) {
        this.drop = drop;
```

```
this.dataBuffer = new int[dataBufferSize];  
}  
  
public void run() {  
    Random random = new Random();  
    for (int i = 0; i < this.dataBuffer.length; ++i) {  
        this.dataBuffer[i] = i;  
        this.drop.put(this.dataBuffer[i]);  
        try {  
            Thread.sleep(random.nextInt(5000));  
        } catch (InterruptedException e) {  
            System.out.println(e.getStackTrace());  
        }  
    }  
    this.drop.put(Producer.INTERRUPT_VALUE);  
}
```



```
VALUE RECEIVED: 0  
VALUE RECEIVED: 1  
VALUE RECEIVED: 2  
VALUE RECEIVED: 3  
VALUE RECEIVED: 4  
VALUE RECEIVED: 5  
VALUE RECEIVED: 6  
VALUE RECEIVED: 7  
VALUE RECEIVED: 8  
VALUE RECEIVED: 9  
VALUE RECEIVED: 10
```

Рисунок 1.3 – Результат виконання програми

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.

Лістинг класу Group

```
final class Group {
    private static int idCounter = 0;
    private final int ID;
    private final String NAME;
    private final ArrayList<Student> STUDENTS;

    public Group(String name, Student... students) {
        this.ID = ++Group.idCounter;

        this.NAME = name;
        this.STUDENTS = new ArrayList<>(Arrays.asList(students));
    }

    public static int getGroupsCount() {
        return Group.idCounter;
    }

    public long getId() {
        return this.ID;
    }

    public String getName() {
        return this.NAME;
    }
}
```

```
public int getStudentsCount() {  
    return this.STUDENTS.size();  
}
```

```
public Student getStudent(int index) {  
    if (index >= this.STUDENTS.size() || index < 0) {  
        throw new IllegalArgumentException("Sequence number is out of bounds.");  
    }  
    return this.STUDENTS.get(index);  
}
```

```
public int getStudentIndex(Student student) {  
    if (student == null) {  
        throw new IllegalArgumentException("Student cannot be null");  
    }  
    final int RESULT_FAILURE = -1;  
    int index = this.STUDENTS.indexOf(student);  
    return (index != RESULT_FAILURE) ? index : RESULT_FAILURE;  
}
```

```
public boolean containsStudent(Student student) {  
    if (student == null) {  
        throw new IllegalArgumentException("Student cannot be null");  
    }  
    return this.STUDENTS.contains(student);  
}  
}
```

Лістинг класу Journal

```
final class Journal
{
    public static final int GRADE_MIN_VALUE = 0;
    public static final int GRADE_MAX_VALUE = 100;

    private static int idCounter = 0;

    private final int ID;
    private final String NAME;

    private final int WEEKS_COUNT;

    private final ConcurrentHashMap<Student, ReentrantLock[]> SHEET_LOCKS;
    private final HashMap<Group, ConcurrentHashMap<Student, Integer[]>> SHEET;

    public Journal(String name, int weeksCount, Group... groups)
    {
        this.ID = ++Journal.idCounter;

        this.NAME = name;

        this.WEEKS_COUNT = weeksCount;

        this.SHEET = new HashMap<>();
        this.SHEET_LOCKS = new ConcurrentHashMap<>();

        for (Group group : groups)
        {
```



```

        for (int i = 0; i < group.getStudentsCount(); ++i)
        {
            this.SHEET_LOCKS.computeIfAbsent(group.getStudent(i), k ->
IntStream.range(0,          weeksCount).mapToObj(j          ->          new
ReentrantLock()).toArray(ReentrantLock[]::new));

            this.SHEET.computeIfAbsent(group, k -> new
ConcurrentHashMap<>()).put(group.getStudent(i),          IntStream.range(0,
weeksCount).mapToObj(j -> 0).toArray(Integer[]::new));
        }
    }
}

```

```

public int getId()
{
    return this.ID;
}

```

```

public String getName()
{
    return this.NAME;
}

```

```

public int getWeeksCount()
{
    return this.WEEKS_COUNT;
}

```

```

public void putGrade(Student student, int weekIndex, int grade)
{

```

```

if (student == null)
{
    throw new IllegalArgumentException("Student cannot be null");
}

if (weekIndex < 0 || weekIndex >= this.WEEKS_COUNT)
{
    throw new IllegalArgumentException("Invalid weekIndex value.");
}

        if (grade < Journal.GRADE_MIN_VALUE || grade >
Journal.GRADE_MAX_VALUE)
{
    throw new IllegalArgumentException("Grade value is invalid.");
}

ReentrantLock lock = this.SHEET_LOCKS.get(student)[weekIndex];

lock.lock();

try
{
    for (Group group : this.SHEET.keySet())
    {
        if (group.containsStudent(student)) {
            this.SHEET.get(group).get(student)[weekIndex] += grade;
            break;
        }
    }
}

```

```

    } finally {
        lock.unlock();
    }
}

public void printSheet() {

    List<Group> sortedGroups = new
    ArrayList<>(this.SHEET.keySet()).stream().sorted(Comparator.comparing(Group::ge
    tName)).collect(Collectors.toList());

    for (Group group : sortedGroups) {
        System.out.println(group.getName());

    List<Student> sortedStudents = new
    ArrayList<>(this.SHEET.get(group).keySet()).stream().sorted(Comparator.comparin
    g(Student::getFullName)).collect(Collectors.toList());

    for (Student student : sortedStudents) {
        System.out.printf("%-10s: ", student.getFullName());
        for (int j = 0; j < this.WEEKS_COUNT; ++j) {
            System.out.printf("%-3d|", this.SHEET.get(group).get(student)[j]);
        }
        System.out.println();
    }

    System.out.println();
}
}
}

```

Лістинг класу Student

```
package task3;

final class Student
{
    private static int idCounter = 0;

    private final int ID;

    private String name;
    private String surname;
    private String patronymic;
    private String fullName;

    public Student(String name, String surname, String patronymic)
    {
        this.ID = ++Student.idCounter;

        this.name = name;
        this.surname = surname;
        this.patronymic = patronymic;
        this.fullName = String.format("%s %s %s", surname, name, patronymic);
    }

    public static int getStudentsCount()
    {
        return Student.idCounter;
    }
}
```

```
public long getId()
{
    return this.ID;
}
```

```
public String getName()
{
    return this.name;
}
```

```
public String getSurname()
{
    return this.surname;
}
```

```
public String getPatronymic()
{
    return this.patronymic;
}
```

```
public String getFullName()
{
    return this.fullName;
}
}
```

Лістинг класу Teacher

```
final class Teacher {  
    private static int idCounter = 0;  
  
    private final int ID;  
  
    private String name;  
    private String surname;  
    private String patronymic;  
    private String fullName;  
  
    public Teacher(String name, String surname, String patronymic) {  
        this.ID = ++Teacher.idCounter;  
  
        this.name = name;  
        this.surname = surname;  
        this.patronymic = patronymic;  
        this.fullName = String.format("%s %s %s", surname, name, patronymic);  
    }  
  
    public static int getAcademicsCount() {  
        return Teacher.idCounter;  
    }  
  
    public long getId() {  
        return this.ID;  
    }  
  
    public String getName() {
```

```
        return this.name;
    }

    public String getSurname() {
        return this.surname;
    }

    public String getPatronymic() {
        return this.patronymic;
    }

    public String getFullName() {
        return this.fullName;
    }

    public int generateGrade(int min, int max) {
        if (min > max)
        {
            throw new IllegalArgumentException("min must be less than or equal to max");
        }

        return min + (int)(Math.random() * ((max - min) + 1));
    }
}
```

Лістинг класу Main

```

package task3;

import java.util.ArrayList;

final class Main
{
    public static void main(String[] args)
    {
        Group group1 = new Group("Group 1", Main.generateStudentsArray(25));
        Group group2 = new Group("Group 2", Main.generateStudentsArray(30));
        Group group3 = new Group("Group 3", Main.generateStudentsArray(27));

        Journal journal = new Journal("Journal", 10, group1, group2, group3);

        Teacher lecturer = new Teacher("Lecturer", "Lecturer", "Lecturer");
        Teacher assistant1 = new Teacher("Assistant 1", "Assistant 1", "Assistant 1");
        Teacher assistant2 = new Teacher("Assistant 2", "Assistant 2", "Assistant 2");
        Teacher assistant3 = new Teacher("Assistant 3", "Assistant 3", "Assistant 3");

        Group[] groups = { group1, group2, group3 };
        Teacher[] teachers = { lecturer, assistant1, assistant2, assistant3 };

        ArrayList<Thread> threads = new ArrayList<>();

        for (Teacher teacher : teachers)
        {
            threads.addLast(new Thread(() ->
            {

```



```

    for (Group group : groups)
    {
        for (int i = 0; i < group.getStudentsCount(); ++i)
        {
            for (int weekIndex = 0; weekIndex < journal.getWeeksCount();
++weekIndex)
            {
                journal.putGrade(group.getStudent(i), weekIndex,
teacher.generateGrade(Journal.GRADE_MIN_VALUE,
Journal.GRADE_MAX_VALUE));
            }
        }
    }
}

```

```

for (Thread thread : threads)
{
    thread.start();
}

```

```

for (Thread thread : threads)
{
    try
    {
        thread.join();
    }
    catch (InterruptedException ex)
    {

```

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

journal.printSheet();
}

private static Student[] generateStudentsArray(int count)
{
    Student[] students = new Student[count];

    String placeholder;

    for (int i = 0; i < count; ++i)
    {
        placeholder = Integer.toString(i);

        students[i] = new Student(placeholder, placeholder, placeholder);
    }

    return students;
}
}
```

Group 1												
0 0 0	:	122	241	250	128	153	199	182	155	201	143	
1 1 1	:	260	262	165	109	280	169	104	74	217	271	
10 10 10	:	117	270	235	195	258	118	222	129	229	172	
11 11 11	:	269	241	270	209	178	259	134	223	270	142	
12 12 12	:	265	198	141	278	249	117	165	210	215	281	
13 13 13	:	198	107	143	283	122	143	237	151	212	175	
14 14 14	:	187	114	169	213	219	269	165	210	241	99	
15 15 15	:	203	228	79	218	212	167	247	194	99	258	
16 16 16	:	222	229	122	178	125	227	172	112	256	217	
17 17 17	:	291	237	212	237	198	217	185	334	181	176	
18 18 18	:	259	124	137	276	254	246	151	302	289	306	
19 19 19	:	185	168	264	238	159	270	178	166	180	190	
2 2 2	:	240	327	167	169	250	180	217	192	179	315	
20 20 20	:	226	168	181	183	258	152	156	120	146	194	
21 21 21	:	162	316	126	228	234	188	290	197	202	195	
22 22 22	:	330	176	264	170	128	267	163	299	154	186	
23 23 23	:	178	125	177	206	239	239	276	103	218	144	
24 24 24	:	246	271	117	132	68	243	118	165	274	232	
3 3 3	:	182	225	251	131	223	230	198	146	136	147	
4 4 4	:	184	246	207	165	211	205	302	236	168	238	
5 5 5	:	147	190	290	137	251	236	213	138	250	207	
6 6 6	:	123	163	223	133	291	77	173	196	244	225	
7 7 7	:	284	210	253	93	177	194	101	227	165	241	
8 8 8	:	214	206	230	237	188	247	223	310	144	209	
9 9 9	:	130	218	195	157	191	269	202	166	207	298	

Рисунок 1.4 – Отримані оцінки для першої групи

Group 2												
0 0 0	:	228	66	159	118	173	208	168	155	150	255	
1 1 1	:	194	167	223	247	147	115	85	113	174	194	
10 10 10	:	243	254	247	238	270	204	232	215	98	327	
11 11 11	:	209	245	94	130	155	199	208	241	165	189	
12 12 12	:	289	185	204	227	185	224	241	308	148	296	
13 13 13	:	195	155	133	241	314	196	294	295	192	233	
14 14 14	:	266	232	184	153	220	247	197	337	144	306	
15 15 15	:	168	285	205	122	258	153	228	173	179	231	
16 16 16	:	307	162	189	212	148	233	220	13	239	256	
17 17 17	:	201	102	246	131	256	173	129	171	180	313	
18 18 18	:	227	210	193	197	130	225	279	228	58	224	
19 19 19	:	162	194	197	264	212	242	257	199	207	298	
2 2 2	:	184	180	206	220	277	285	165	206	269	144	
20 20 20	:	313	261	264	194	305	190	244	306	158	255	
21 21 21	:	190	224	101	223	249	158	187	129	243	206	
22 22 22	:	171	219	150	258	303	179	83	167	131	197	
23 23 23	:	111	108	259	135	211	184	174	248	291	302	
24 24 24	:	247	183	259	213	162	267	221	227	171	261	
25 25 25	:	170	189	218	294	238	282	251	196	186	284	
26 26 26	:	145	332	194	232	164	220	212	217	255	169	
27 27 27	:	200	146	146	157	100	210	115	266	268	200	
28 28 28	:	161	131	198	146	140	168	232	219	172	100	
29 29 29	:	284	235	200	183	195	282	282	135	208	108	
3 3 3	:	265	88	43	115	180	153	141	221	210	189	
4 4 4	:	253	184	93	192	221	245	296	180	221	206	
5 5 5	:	162	215	213	200	149	202	165	236	226	220	
6 6 6	:	195	158	210	203	265	209	227	259	157	179	
7 7 7	:	293	268	192	159	175	252	234	236	236	244	
8 8 8	:	125	203	274	141	250	220	247	139	195	179	
9 9 9	:	244	143	256	206	135	319	195	100	227	130	

Рисунок 1.5 – Отримані оцінки для другої групи

Group 3												
0 0 0	:	99		128		268		130		233		248
1 1 1	:	247		162		173		253		142		212
10 10 10	:	186		164		172		313		114		163
11 11 11	:	159		213		204		137		149		256
12 12 12	:	337		156		287		177		192		178
13 13 13	:	225		140		167		138		117		244
14 14 14	:	208		268		258		143		130		153
15 15 15	:	144		143		160		309		204		211
16 16 16	:	204		199		181		201		113		227
17 17 17	:	194		176		203		141		261		185
18 18 18	:	255		135		201		260		272		98
19 19 19	:	246		218		245		269		109		143
2 2 2	:	196		258		324		169		126		97
20 20 20	:	199		182		127		279		171		146
21 21 21	:	243		291		139		213		138		188
22 22 22	:	187		103		145		253		234		173
23 23 23	:	148		199		242		230		132		319
24 24 24	:	121		157		205		308		141		248
25 25 25	:	246		227		215		83		315		148
26 26 26	:	75		258		93		149		223		151
3 3 3	:	203		195		224		232		220		209
4 4 4	:	256		292		160		196		268		179
5 5 5	:	130		177		221		141		179		256
6 6 6	:	210		294		221		201		298		269
7 7 7	:	207		177		203		157		273		165
8 8 8	:	199		172		233		237		340		241
9 9 9	:	305		162		276		103		234		196

Рисунок 1.6 – Отримані оцінки для третьої групи

ВИСНОВКИ

Під час виконання даної роботи було розглянуто та реалізовано різні методи управління потоками на прикладі різних завдань у Java. Було використано синхронізацію, блокування для забезпечення коректної роботи програм. В першому завданні було реалізовано програмний код з різними методами управління потоками, що дозволило забезпечити коректну роботу програми симуляції банкової системи. У другому завданні було реалізовано приклад Producer-Consumer application, де було модифіковано масив даних, який передається між відправником та споживачем. Третє завдання вимагало реалізації електронного журналу групи. Завдяки використанню потоків, було можливо ефективно виставляти оцінки студентам відразу багатьом викладачам. Таким чином, використання потоків допомагає забезпечити коректність роботи програм, оптимізацію використання ресурсів системи та покращують продуктивність.