НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Звіти до комп’ютерних практикумів з кредитного модуля

“Parallel computing”

**Виконав**

**Студенти групи ІТ-04**

**Гавриленко Я.С**

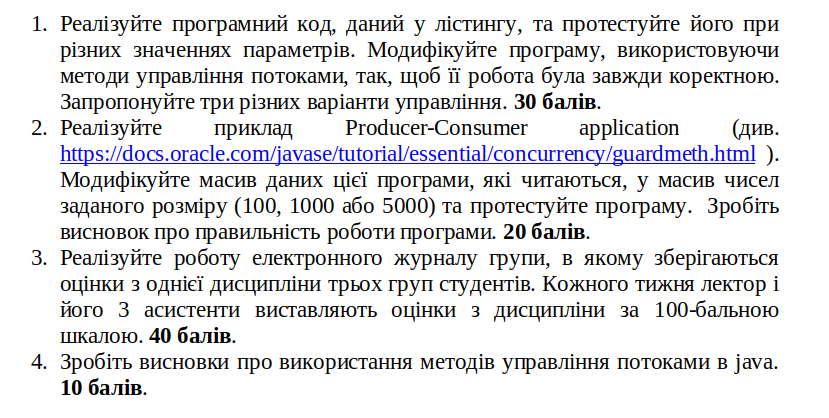
**Перевірив:**

Київ – 2022

**Комп‘ютерний практикум No 3**

**Розробка паралельних програм з використанням механізмів синхронізації: синхронізовані методи, локери, спеціальні типи**

ЗАВДАННЯ



**ВИКОНАННЯ**

Повний код лабораторної: <https://github.com/yan14171/parralel_calcullation_labs/tree/master/lab>3

**Завдання 1**

Реалізувавши програмний код, даний у лістингу, отримаємо результати, показані на рисунку 1.1.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.1 Результат роботи несинхронізованої програми

Як бачимо, результат суми в 100\_000 не зберігається, а з кількістю транзакцій тільки зменшується. Також кількість транзакцій не точно обчислюється.

Реалізуємо такі методи синхронізації, як синхронізований метод, синхронізований блок, локер та атомарні змінні. Повний код реалізації може бути знайдений в репозиторії гітхаб. Результат однаковий та представлений на рисунку 1.2.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 1.2 Результат роботи синхронізованої програми

**Завдання 2**

Реалізувавши програмний код, протестуємо його з двома прикладами: в першому передаватимемо від Producer-а до Consumer-а чисел, а в другому стрічок. Побачити часткову реалізацію тесту можна на двох методах run()

public void run() {  
 int[] importantInfo = new int[1000];  
 Arrays.*fill*(importantInfo, 5);  
  
 Random random = new Random();  
  
 for (int i = 0;  
 i < importantInfo.length;  
 i++) {  
 drop.put(importantInfo[i]);  
 try {  
 Thread.*sleep*(random.nextInt(5000));  
 } catch (InterruptedException e) {}  
 }  
  
 drop.put(0);  
}  
  
public void run() {  
 String importantInfo[] = {  
 "Mares eat oats",  
 "Does eat oats",  
 "Little lambs eat ivy",  
 "A kid will eat ivy too"  
 };  
 Random random = new Random();  
  
 for (int i = 0;  
 i < importantInfo.length;  
 i++) {  
 drops.put(importantInfo[i]);  
 try {  
 Thread.*sleep*(random.nextInt(5000));  
 } catch (InterruptedException e) {}  
 }  
 drops.put("DONE");  
}

Результати роботи чисельного на стрічкового прикладу можна побачити на рисунках 2.1 та 2.2 відповідно.

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 2.1 Реалізація числового прикладу

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 2.2 Реалізація стрічкового прикладу

**Завдання 3**

Реалізуємо поставлену задачу, використовуючи синхронізований блок в класі Teacher для синхронізації додавання оцінок до журналу.

Результат виконання:

Зображення, що містить календар

Автоматично згенерований опис

**ВИСНОВОК**

В рамках роботи, ми на практиці познайомились с механізмами синхронізації потоків в Java. Серед яких: synchronized метод, synchronized блок, Lockers, атомарні змінні.

Ми впевнились у незамінності високорівневих методів синхронізації потоків. Також, у багатьох випадках методи управління потоками є замінними між собою, у чому ми впевнилися при виконанні першого завдання. Серед методів присутні простіші, як наприклад синхронізований метод, який в той самий час має багато недоліків.

Також є більш складні методи, які потребують більше роботи від програміста, але надають значні переваги у питаннях налаштування синхронізації, уникання dead lock-ів, або зайвих витрат часу та ресурсів.