

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 №1

**Технології паралельних обчислень**

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

Київ 2023

# Изображение выглядит как текст, газета Автоматически созданное описаниеЗавдання:

Изображение выглядит как текст  Автоматически созданное описание

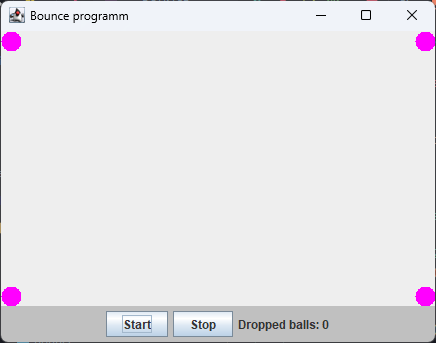
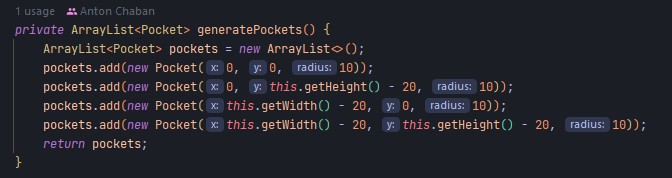
**Хід виконання:**

# [\*GitHub репозиторій\*](https://github.com/antonchaban/MultiThreading/tree/master/src/main/java/lab1)

1. Створимо клас Pocket який буде відповідати за характеристики наших луз на полі

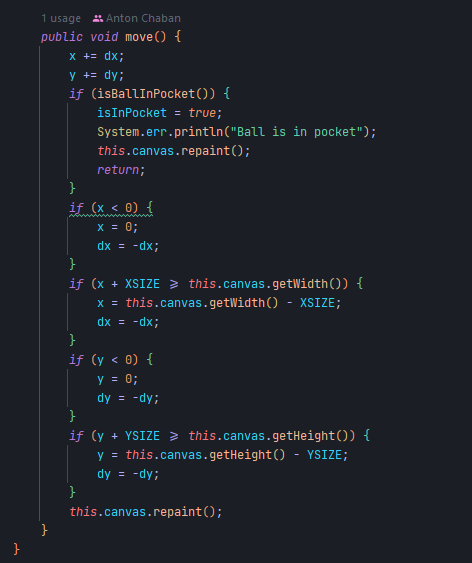
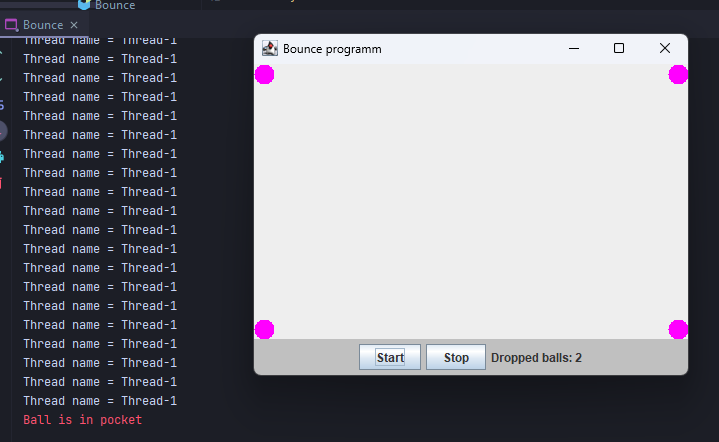
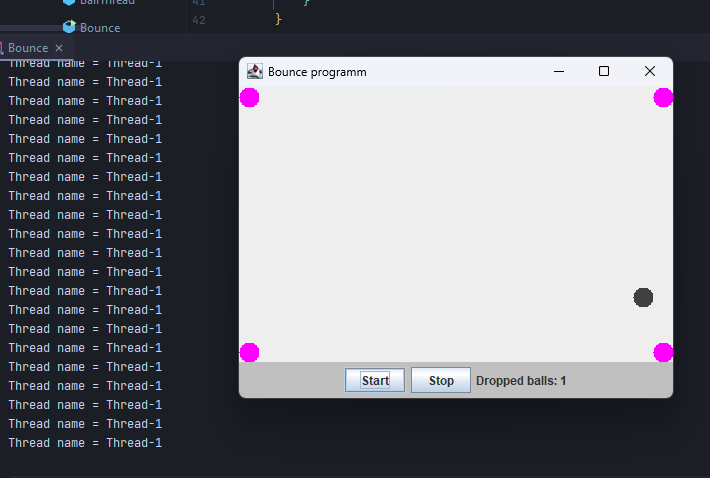


Методом isBallInPocket() виконуємо перевірку чи потрапила кулька в межі лузи.



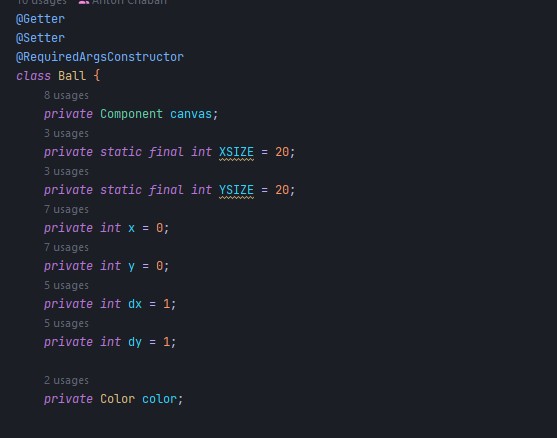
Наші лузи розміщені в 4 кутках ігрового поля.

Далі запустивши програму на виконання бачимо що кулька рухається по полю, процес відбувається у виділеному для неї потоці, як тільки кулька потрапляє у лузу – потік завершає свою роботу.



Якщо кулька потрапила в лузу зупиняємо рух і завершуємо наш потік.

1. Додаємо поле для ідентифікації кольору кульки



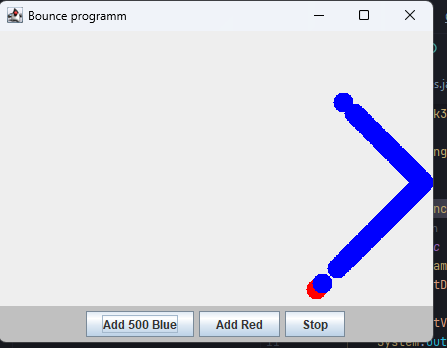
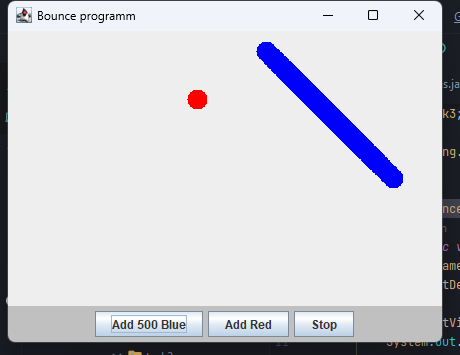
Для класу BallThread створимо конструктор в якому будемо задавати пріоритет потоку



Далі додамо кнопки які будуть створювати наші кульки (синіх будемо створювати одразу по 500), і встановлювати їм пріоритет, для синіх – 1 (найнижчий), а для червоних – 10 (найвищий).



Запустивши програму можна помітити що сині кульки з нижчим пріоритетом рухаються менш стабільно і «посмикуються» і з часом наздоганяють червону яка рухається більш стабільно.



1. Для демонстрації роботи методу join() створимо можливість зупиняти всі кульки які знаходяться в даний момент на більярдному столі. Сірі кульки будуть знаходитися в стані очікування, поки червона кулька не завершить

свій рух.

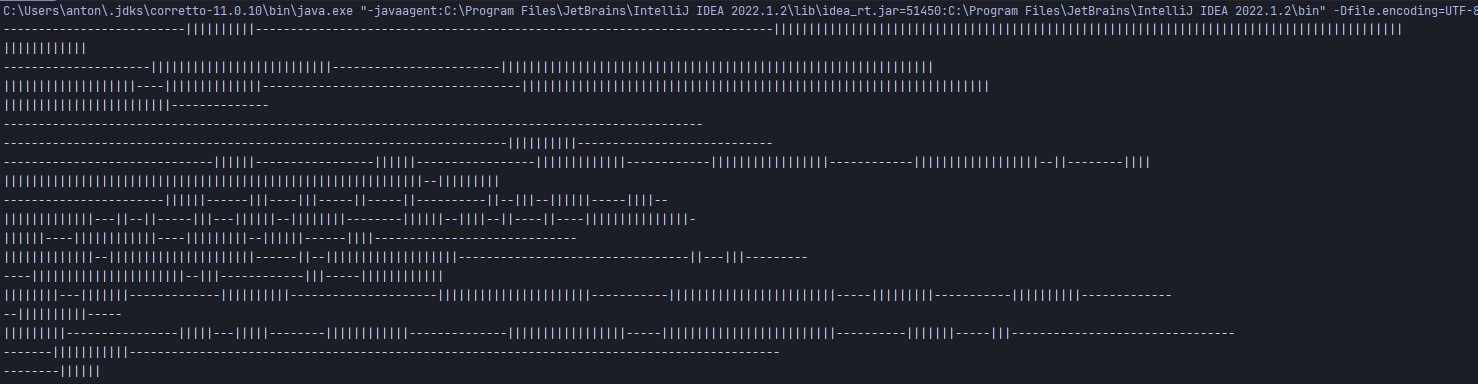
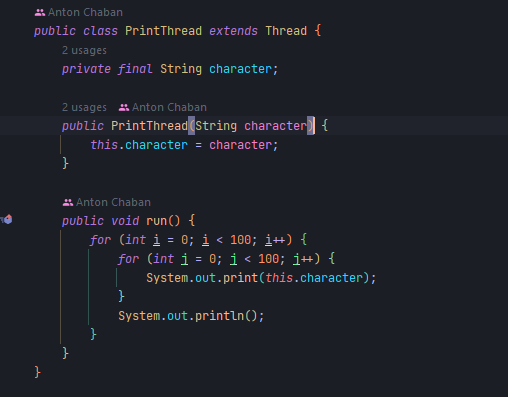


Після того як червона кулька завершить свій рух, сірі кульки продовжать рухатися.

1. Створимо 2 потоки, які будуть відповідати за друк символів «-» і «|» відповідно.

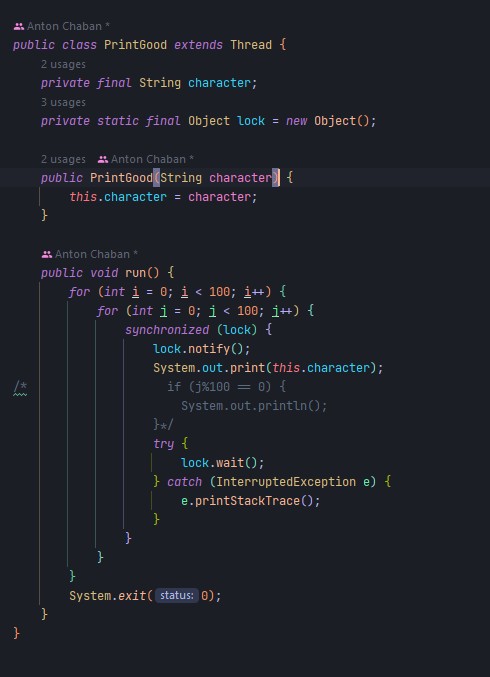


В першому способі не будемо використовувати ніякі методи синхронізації, і побачимо який вивід отримаємо



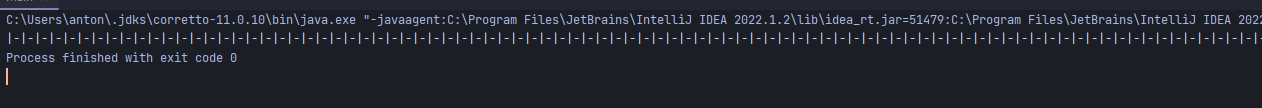
Як можна помітити символи друкуються у випадковому порядку, адже потоки ніяк не синхронізовані, а відповідно кожен символ друкувався тоді, коли його потік отримував цю можливість.

В другому способі створимо інший клас, який буде виводити ці символи в потрібному порядку «|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|»

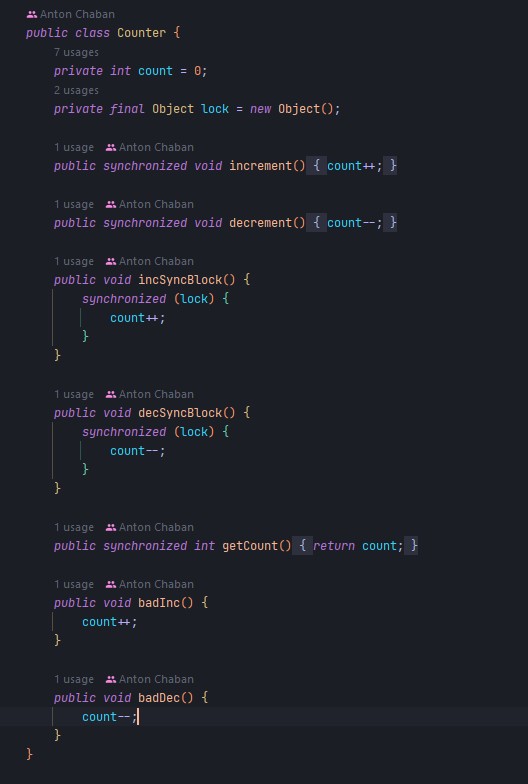


Створимо в середині класу об’єкт, на якому будемо викликати методи wait/notify, щоб передавати доступ до іншого потоку, для друку наступного символу.

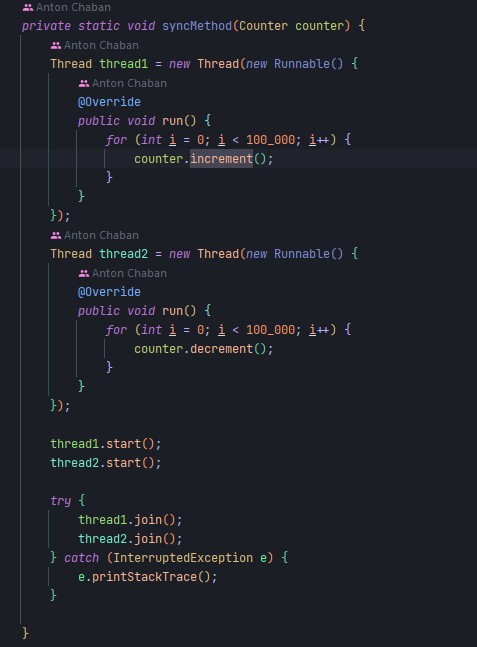
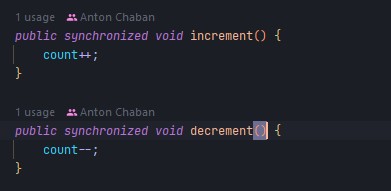
Як бачимо цього разу символи виводяться в потрібному порядку



1. Створимо клас Counter



Для виконання операцій в синхронізованому методі використаємо наступну реалізацію:



Використаємо ключове слово synchronized, для позначення того, що наші методи мають бути синхронізовані між собою.

Для виконання операцій у синхронізованому блоці використаємо наступні методи:



Для цього способу створимо Object на якому будемо синхронізуватися, і чекати його «звільнення» для виконання наступних дій.

Для виконання операцій з блокуванням об’єкту використаємо наступні методи:

Створимо ReentrantLock, на якому будемо викликати методи lock/unlock по завершенню операції



**Висновок:** Під час виконання даної лабораторної роботи я ознайомився та навчився використовувати на практиці основні методи керування потоками.