

**Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ
з лабораторної роботи №2
з навчальної дисципліни «Економіка ІТ індустрії та підприємництво»**

Тема:

МЕТРИКИ РОЗМІРУ. MIRA LINE OF CODE

Виконав:

Студент 4 курсу кафедри ФІОТ,
Навчальної групи ІП-11
Головня О. Р.

Перевірив:

Родіонов П. В.

Київ 2024

I. Мета.

Мета роботи: Ознайомитися з загальними поняттями щодо вимірювань та метрикою розміру з мірою Lines of Code. Напрацювати вміння застосування засобів вимірювання метрики. Отримати загальні вміння щодо застосування метрики в економіці програмного забезпечення.

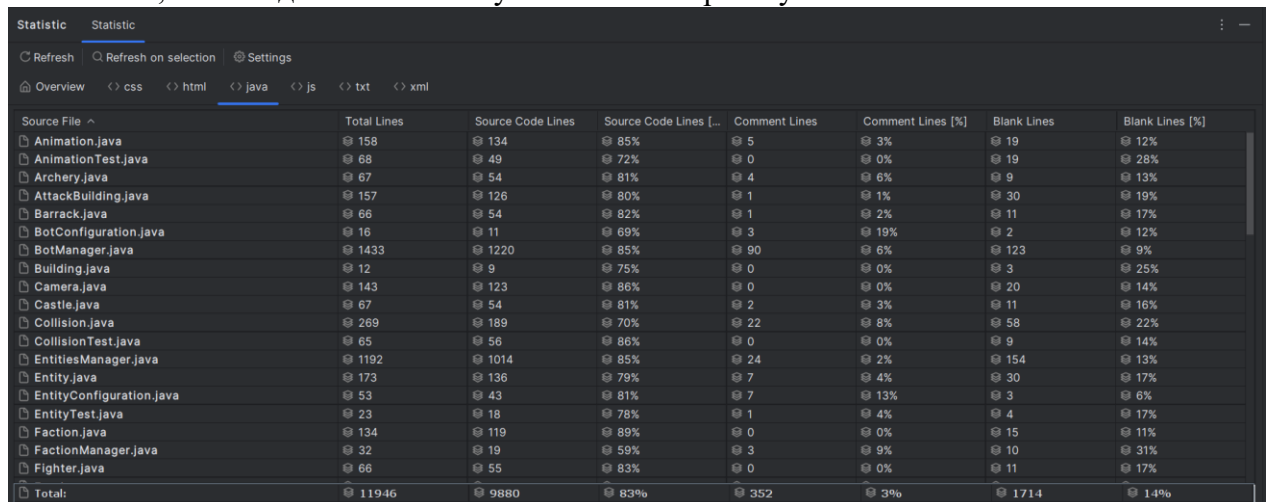
II. Завдання.

1. Застосовуючи вимірювачі у відповідних середовищах програмування (Visual Studio, Code Counter for Java, CodeCounter, та інші), на прикладі власних програмних текстів виконати вимірювання розміру.
2. Здійснити відповідні економічні розрахунки.
3. Дослідити рівні мов програмування C# та Java.
4. Захистити виконану роботу.

III. Результати виконання лабораторної роботи.

3.1. Вимірювання власного проєкту

Я використав написаний мною проєкт курсової роботи, що є грою на Java. У IntelliJ IDEA я скачав плагін Statistic, який надасть статистику LOC всього проєкту.



Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
Animation.java	158	134	85%	5	3%	19	12%
AnimationTest.java	68	49	72%	0	0%	19	28%
Archery.java	67	54	81%	4	6%	9	13%
AttackBuilding.java	157	126	80%	1	1%	30	19%
Barrack.java	66	54	82%	1	2%	11	17%
BotConfiguration.java	16	11	69%	3	19%	2	12%
BotManager.java	1433	1220	85%	90	6%	123	9%
Building.java	12	9	75%	0	0%	3	25%
Camera.java	143	123	86%	0	0%	20	14%
Castle.java	67	54	81%	2	3%	11	16%
Collision.java	269	189	70%	22	8%	58	22%
CollisionTest.java	65	56	86%	0	0%	9	14%
EntitiesManager.java	1192	1014	85%	24	2%	154	13%
Entity.java	173	136	79%	7	4%	30	17%
EntityConfiguration.java	53	43	81%	7	13%	3	6%
EntityTest.java	23	18	78%	1	4%	4	17%
Faction.java	134	119	89%	0	0%	15	11%
FactionManager.java	32	19	59%	3	9%	10	31%
Fighter.java	66	55	83%	0	0%	11	17%
Total:	11946	9880	83%	352	3%	1714	14%

Рисунок 1— Кількість LOC проєкту, Statistics

Я не враховував html/css та інші файли, тому що вони не зовсім відносяться до загального проєкту. Отже, загальна кількість рядків 9880

3.2. Відповідні економічні розрахунки

Оскільки проєкт відносно невеликий (до 25 тис. рядків коду), то його тип - «Organic»

Тип проєкту	ab	bb	cb	db
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Методика розв'язання типових задач:

Зусилля (людина/місяць) $\text{Effort} = aa * \text{size}^{bb}$,
де aa , bb коефіцієнти, size – розмір продукту в KLOC.

Вартість (грн.) $\text{Cost} = \text{Effort} * \text{salary}$,
де salary – заробітна плата.

Час на розробку $\text{Schedule} = cb * \text{Effort}^{db}$,
де cb , db – коефіцієнти.

Таким чином,

Зусилля $= 2.4 * 9.880^{1.05} = 26.5892467013$ (людина/місяць)

Вартість $= 26.589 * 16000 \text{ грн} = 425427.94722$

Час на розробку $= 2.5 * 26.5892467013^{0.38} = 8.69617561264$

3.3. Дослідження рівнів мов програмування C# та Java

Для початку я взяв за основу алгоритм сортування бульбашкою написаний на C++.
За допомогою сервісу codeconvert отримав код на C# та Java відповідно:

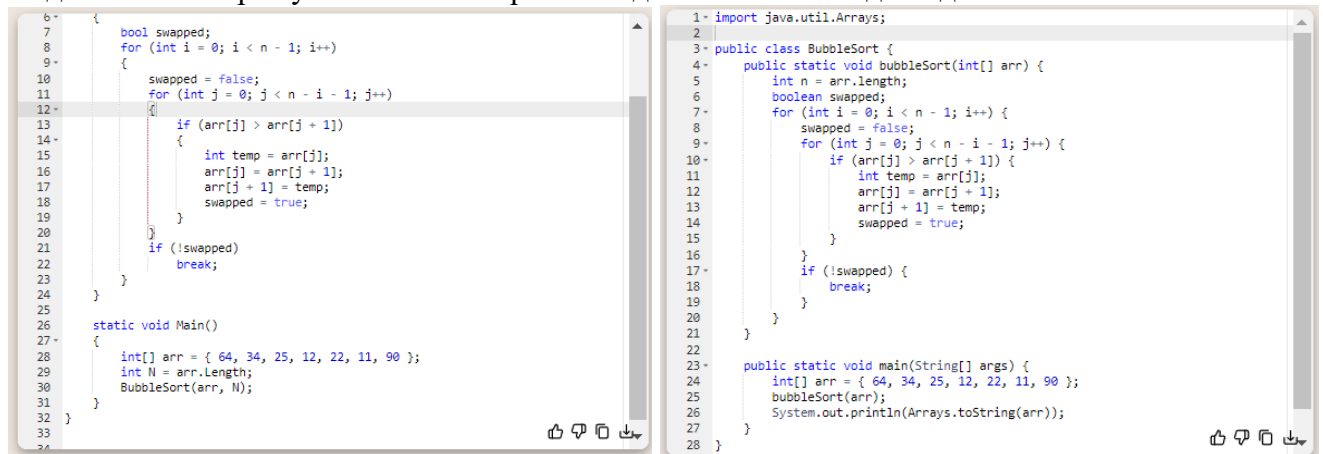


Рисунок 2 — Код алгоритму бульбашки на C# та Java відповідно

За допомогою сервісу CompilerExplorer подивився яка кількість рядків в байт-коді:

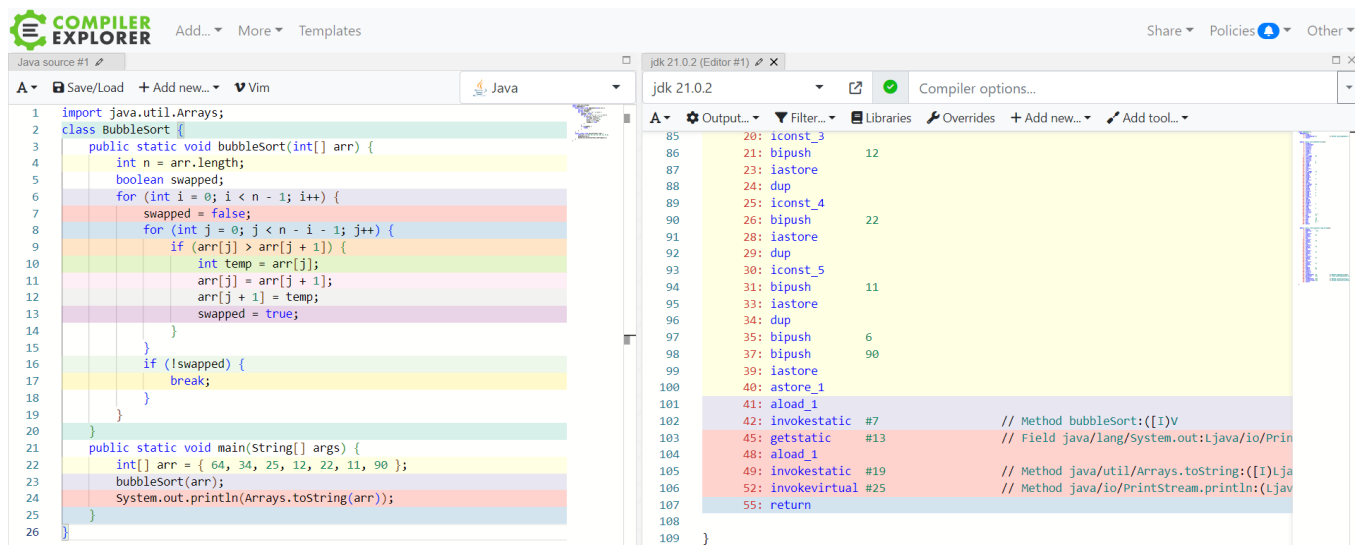


Рисунок 3 — Байт-код алгоритму бульбашки на Java

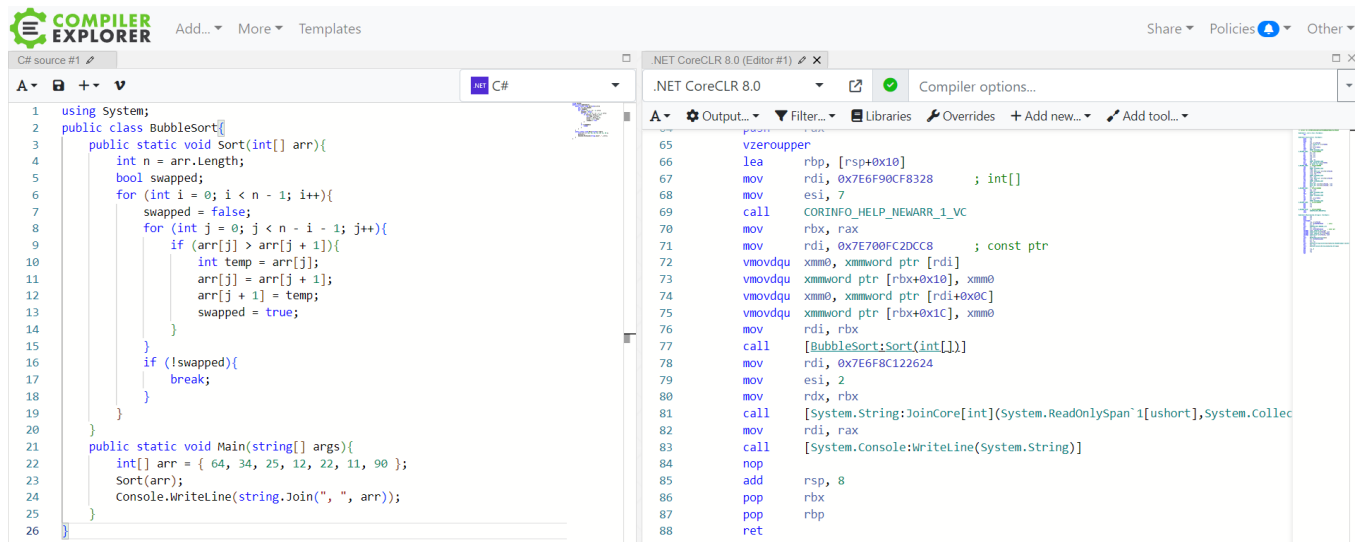


Рисунок 4 — Байт-код алгоритму бульбашки на C#

Таким чином, якщо розраховувати співвідношення рядків, отримаємо наступні результати:

Java: $109/26 = 4,19$

C#: $88/35 = 3,38$

Це означає, що алгоритм, написаний на Java, після компіляції генерує більше байт-коду на один рядок вихідного коду, ніж у випадку C#.

Оскільки співвідношення байт-коду до вихідного коду на Java більше, це може вказувати на те, що Java-байт-код, у порівнянні з C#, є більш деталізованим або займає більше місця для виконання тих самих інструкцій. Це також може бути наслідком того, як компілятори обробляють вихідний код, створюючи різну кількість проміжних інструкцій.

Хоча кількість байт-коду не є прямим показником швидкості виконання програми, більш компактний байт-код у C# може свідчити про дещо кращу продуктивність у певних умовах, оскільки менше байт-коду може означати менший об'єм роботи для віртуальної машини або рантайму.

IV. Висновки.

У ході практичної роботи було досліджено метрику LOC (Lines of Code) на власному проєкті. Було також проаналізовано мови C# і Java на прикладі алгоритму сортування бульбашкою. Співвідношення рядків асемблера/байт-коду до вихідного коду склали 3,63 для C# і 4,06 для Java. Різниця пов'язана з рівнем абстракції та оптимізаціями компіляторів. Практична робота дала змогу отримати навички оцінки розміру ПЗ, виявити обмеження метрики LOC, а також зрозуміти, як особливості мов програмування можуть впливати на результати.

Виконав: студент Олександр Головня

Лістинг Java:

```
import java.util.Arrays;

public class BubbleSort {

    public static void bubbleSort(int[] arr) {

        int n = arr.length;

        boolean swapped;

        for (int i = 0; i < n - 1; i++) {

            swapped = false;

            for (int j = 0; j < n - i - 1; j++) {

                if (arr[j] > arr[j + 1]) {

                    int temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                    swapped = true;

                }

            }

            if (!swapped) {

                break;

            }

        }

    }

    public static void main(String[] args) {

        int[] arr = { 64, 34, 25, 12, 22, 11, 90 };

        bubbleSort(arr);

        System.out.println(Arrays.toString(arr));

    }

}
```

Лістинг C#:

```
using System;

public class BubbleSort{

    public static void Sort(int[] arr){

        int n = arr.Length;

        bool swapped;

        for (int i = 0; i < n - 1; i++){

            swapped = false;

            for (int j = 0; j < n - i - 1; j++){

                if (arr[j] > arr[j + 1]){

                    int temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                    swapped = true;

                }

            }

            if (!swapped){

                break;

            }

        }

    }

    public static void Main(string[] args){

        int[] arr = { 64, 34, 25, 12, 22, 11, 90 };

        Sort(arr);

        Console.WriteLine(string.Join(" ", arr));

    }

}
```