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

3ВІТ з лабораторної роботи №2 з навчальної дисципліни «Економіка ІТ індустрії та підприємництво»

Тема:

МЕТРИКИ РОЗМІРУ. MIPA LINE OF CODE

Виконав:

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

Перевірив:

Родіонов П. В.

І. 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 С# та Java.
- 4. Захистити виконану роботу.

III. Результати виконання лабораторної роботи.

3.1. Вимірювання власного проєкту

Я використав написаний мною про ϵ кт курсової роботи, що ϵ грою на Java. У IntelliJ IDEA я скачав плагін Statistic, який надасть статистику LOC всього про ϵ кту.

Statistic Statistic										
© Refresh on selection │										
⊕ Overview ⇔ css ⇔ html ⇔ java ⇔ js ⇔ txt ⇔ xml										
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%			
	₿ 67	⊜ 54	⊜ 81%		⊜ 6%	⊜ 9	⊜ 13%			
AttackBuilding.java	€ 157	₿ 126	€ 80%		€ 1%	◎ 30	⊜ 19%			
Barrack.java ■ Barrack.java ■ Barrack.java ■ Barrack.java ■ □	₿ 66	₿ 54	₿ 82%		⊜ 2%	₿ 11				
BotConfiguration.java	⊜ 16		⊜ 69%		⊜ 19%					
BotManager.java ■	€ 1433		€ 85%	€ 90	₿ 6%	€ 123	9%			
🖰 Building.java	₿ 12	₿ 9	₿ 75%	₿ 0	₿ 0%		₿ 25%			
□ Camera.java	₿ 143	₿ 123	₿ 86%	₿ 0	⊜ 0%	₿ 20	⊜ 14%			
□ Castle.java	⊜ 67	₿ 54	₿ 81%		⊜ 3%		⊜ 16%			
Collision.java	₿ 269	₿ 189	◎ 70%	₿ 22	₿ 8%	§ 58	₿ 22%			
□ CollisionTest.java	₿ 65	₿ 56	₿ 86%	₿ 0	₿ 0%	₿ 9				
🖰 EntitiesManager.java		₿ 1014	₿ 85%	₿ 24	⊜ 2%	₿ 154				
□ Entity.java			◎ 79%		⊗ 4%	₿ 30				
□ EntityConfiguration.java	₿ 53	₿ 43	₿ 81%				₿ 6%			
□ EntityTest.java	₿ 23	₿ 18	₿ 78%		₿ 4%	₿ 4				
☐ Faction.java			€ 89%	⊗ 0	◎ 0%	₿ 15				
☐ FactionManager.java	₿ 32	₿ 19	€ 59%		9%	₿ 10	⊜ 31%			
□ Fighter.java	⊜ 66	₿ 55	⊜ 83%	₿ 0	⊜ 0%					
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) Effort = aa * size^bb, де aa, bb коефіцієнти, size – розмір продукту в KLOC. **Вартість** (грн.) Cost = Effort * salary, де salary – заробітна плата. **Час на розробку** Schedule = cb * Effort^db, де cb, db –коефіцієнти.

Таким чином,

Зусилля = $2.4 * 9.880^{1}.05 = 26.5892467013$ (людина/місяць) **Вартість** = 26.589 * 16000грн = 425427.94722 **Час на розробку** = $2.5 * 26.5892467013^{0}.38 = 8.69617561264$

3.3. Дослідження рівнів мов програмування С# 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:

```
bool swapped;
for (int i = 0; i < n - 1; i++)</pre>
                                                                                                                                                      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;
16
17
                                                                                                                                           11
12
13
14
15
18
19
20
21
22
23
24
25
26
27 -
28
29
30
31
32
}
                         if (!swapped)
                                                                                                                                                                        }
                                                                                                                                                                   if (!swapped) {
                                                                                                                                           18
19
20
21
22
             static void Main()
                  int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
int N = arr.Length;
BubbleSort(arr, N);
                                                                                                                                                       public static void main(String[] args) {
   int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
   bubbleSort(arr);
                                                                                                                                          23 °
24 °
25 °
26 °
27 °
28
                                                                                                                                                             System.out.println(Arrays.toString(arr));
                                                                                                             ₾ ⊕ @ ♣
                                                                                                                                                                                                                                                          ⊕ ⊕ □ ♣
```

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

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

```
E COMPILER EXPLORER
                                Add... ▼ More ▼ Templates
                                                                                                                                                                                                                                               Share ▼ Policies 🔔 ▼ Other ▼
                                                                                                                                □ jdk 21.0.2 (Editor #1) ℓ X
      source #1 Ø
A ▼ B Save/Load + Add new... ▼ Vim
                                                                                                                                    jdk 21.0.2
                                                                                                                                                                         ▼ 🗹 🤡 Compiler options...
                                                                                                                                                                                                                                                                                         *
          import java.util.Arrays;
                                                                                                                                      class BubbleSort {
               public static void bubbleSort(int[] arr) {
   int n = arr.length;
                                                                                                                                                          20: iconst 3
                                                                                                                                                          21: bipush
                    int n = arr.lengtr;
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 + 1];
            arr[j + 1] = temp;
            swapped = true;
}
                                                                                                                                                          25: iconst 4
                                                                                                                                                          26: bipush
                                                                                                                                                                                    22
                                                                                                                                                          28: iastore
29: dup
30: iconst_5
                                                                                                                                                          31: bipush
                                                                                                                                                                                    11
                                                                                                                                                               iastore
dup
  14
                                                                                                                                                          35: bipush
                                                                                                                                                          37: bipush
                           if (!swapped) {
                                                                                                                                                          39: iastore
40: astore_1
41: aload_1
                                                                                                                                        100
101
  19
20
                                                                                                                                        102
                                                                                                                                                          42: invokestatic
                                                                                                                                                                                                                 // Method bubbleSort:([I)V
                                                                                                                                                          45: getstatic #13
48: aload_1
49: invokestatic #19
52: invokevirtual #25
                                                                                                                                                                                                                 // Field java/lang/System
                                                                                                                                                                                                                                                     .out:Liava/io/Prim
               public static void main(String[] args) {
    int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
  22
                                                                                                                                                                                                                 // Method java/util/Arrays.toString:([I)Lja
// Method java/io/PrintStream.println:(Ljav
  23
                     bubbleSort(arr);
                      System.out.println(Arrays.toString(arr));
                                                                                                                                        107
```

Рисунок 3 — Байт-код алгоритму бульбашки на Java

```
EXPLORER Add... • More • Templates
                                                                                                                                                                                                                      Share ▼ Policies ♠ ▼ Other ▼
      urce #1 Ø
                                                                                                                  □ .NET CoreCLR 8.0 (Editor #1) Ø X
                                                                                      NET C#
                                                                                                               ▼ .NET CoreCLR 8.0
                                                                                                                                                 ▼ 🗹 🗸 Compiler options...
        using System;
                                                                                                                  class RubbleSort
            public static void Sort
int n = arr.Length;
                                                                                                                                                                                  ; int[]
                  bool swapped;
                                                                                                                                                    rdi, 0x7E6F90CF8328
                  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]){
                                                                                                                                                     CORINFO_HELP_NEWARR_1_VC
                                                                                                                                         mov rdi, 0x7E700FC2DCC8 ; const ptr
vmovdqu xmm0, xmmword ptr [rdi]
vmovdqu xmmoord ptr [rbx+0x10], xmm0
vmovdqu xmmoord ptr [rdi+0x0c]
vmovdqu xmmoord ptr [rdi+0x0c]
                                 int temp = arr[j];
arr[j] = arr[j + 1]
arr[j + 1] = temp;
swapped = true;
                                                                                                                                          vmovdqu xmmword ptr [rbx+0x10], xmm0
                                                                                                                                                     [BubbleSort:Sort(int[])]
rdi, 0x7E6F8C122624
 19
20
21
                                                                                                                                                     [System.String:JoinCore[int](System.ReadOnlySpan`1[ushort],System.Collections
                                                                                                                                          mov
call
             public static void Main(string[] args){
   int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
                                                                                                                                                    [System.Console:WriteLine(System.String)]
                   Console.WriteLine(string.Join(", ", arr));
                                                                                                                                                     rbp
```

Рисунок 4 — Байт-код алгоритму бульбашки на С#

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 С#.

Оскільки співвідношення байт-коду до вихідного коду на Java більше, це може вказувати на те, що Java-байт-код, у порівнянні з С#, 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 ϵ прямим показником швидкості виконання програми, більш компактний байт-код у С# 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 С# і Java на прикладі алгоритму сортування бульбашкою. Співвідношення рядків асемблера/байт-коду до вихідного коду склали 3,63 для С# 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 С#:
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));
}
```