Відповіді на питання

# Який розмір вхідних даних алгоритму?

* int n =100000

# Яке призначення першого циклу for?

* Заповнює масив рандомними числами які помнажаються на n(100000). Діапазон Math.random() від 0.0 до 1.0 за замовчуванням в типі double, тому ми помнажаемо на **n** = 100000, тобто переносимо кому на 5 цифр вперед, а потім переводимо у int, щоб прибрати дробову частину

# Яке призначення другого циклу for?

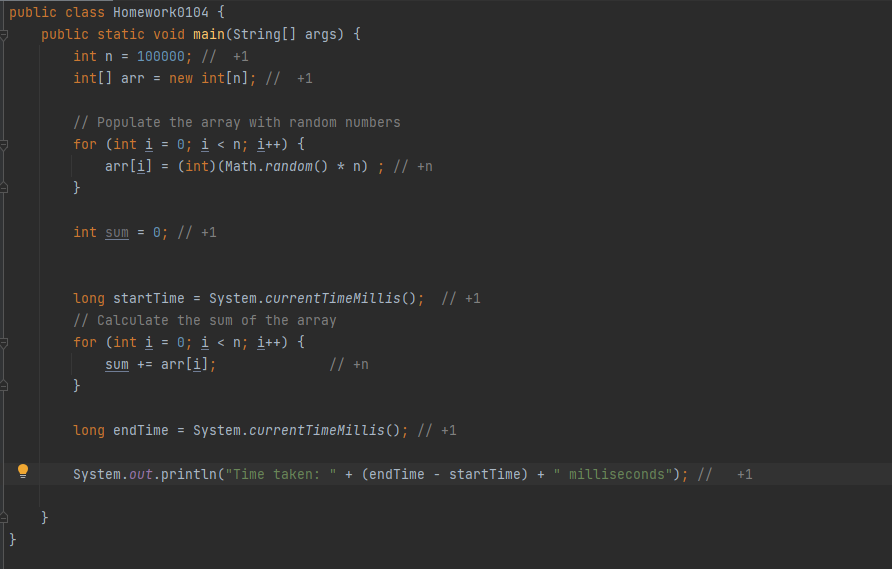
* Обчислює суму всіх чисел в масиві

# Яка часова складність у найгіршому випадку?

* Big –O

O(6 + 2n) = O(2n)

Відносно добре від Good(графік презент.)

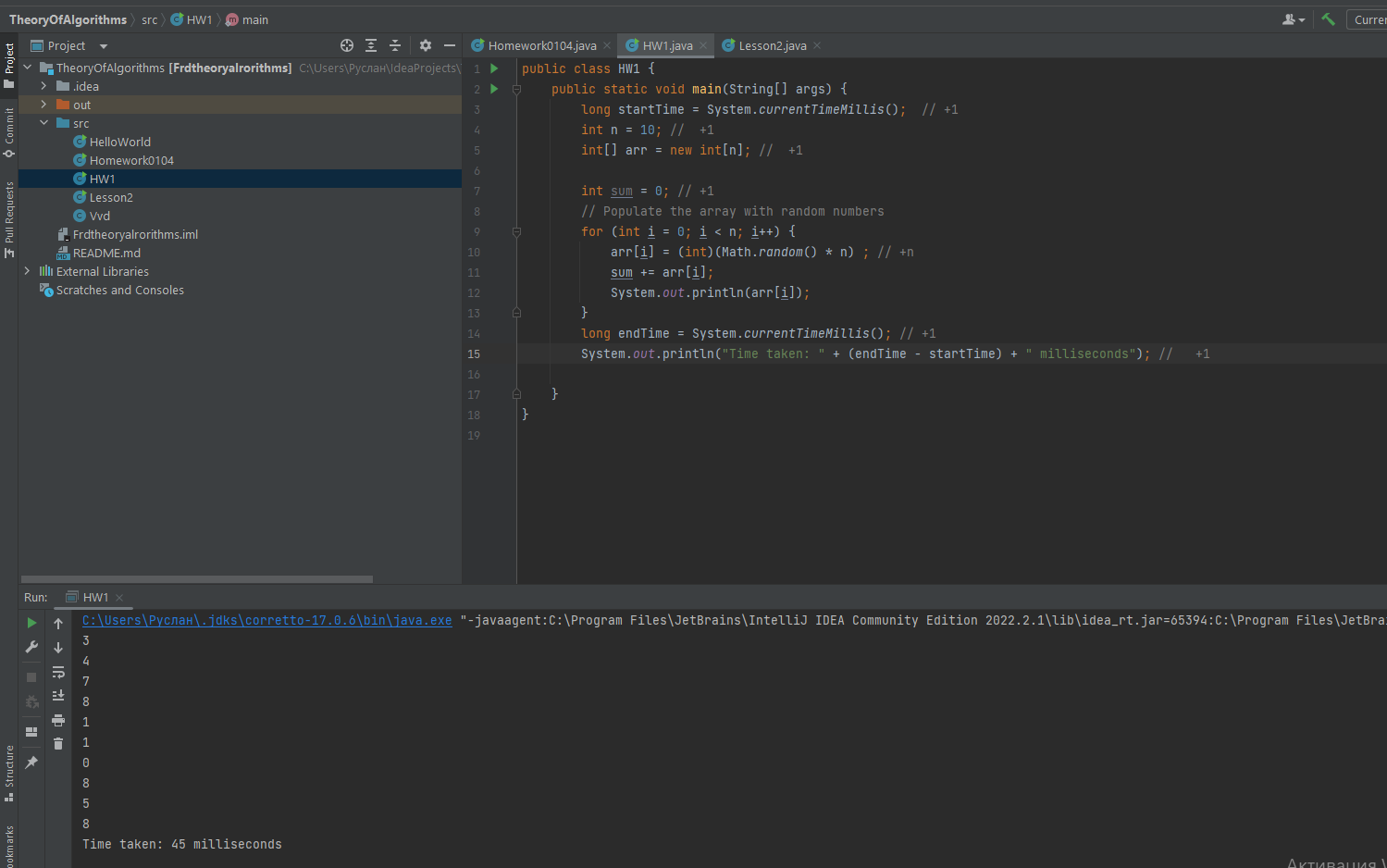


# Чи можна покращити часову складність цього алгоритму? Якщо так, то як?

* Одиночні строки коду, не має великого значення.
* Цикл в циклі робити безглузно так як O(n^2),

а це погано.

* Можна спростити алгоритм, прибравши один цикл, а його тіло циклу перемістити у перший. І виходить O(n), що дуже добре.



(прибрал 100000 так як лагав пк)

# Яка мета асимптотичного аналізу?

* визначити найкращий, найгірший і середній час, необхідний для виконання заданого завдання.

# Які два типи аналізу існують в асимптотичному аналізі?

* **Часова складність:** вимірює, скільки часу потрібно алгоритму для виконання
* **Просторова складність:** вимірює, скільки пам'яті використовує алгоритм

# Що таке часова складність?

* **Часова складність:** вимірює, скільки часу потрібно алгоритму для виконання

# Які фактори впливають на складність часу?

* 1. Розмір вхідних даних
  2. Кількість операцій, які виконує алгоритм
  3. Використання пам'яті алгоритмом

# Для чого призначена нотація Big-O?

* Нотація **Big O** визначає верхню межу будь-якого алгоритму, тобто ваш алгоритм не може зайняти більше часу, ніж цей час. Іншими словами, можна сказати, що велике **О** позначає максимальний час, який займає алгоритм, або найгіршу часову складність алгоритму. Отже, запис великого **О** є найпоширенішим для позначення часової складності алгоритму. Так, якщо функція має вигляд **g(n),** то велике представлення **g(n)** записується як **O(g(n)**) .