МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Кафедра системного проектування

**ЛАБОРАТОРНА РОБОТА №4**

з дисципліни: «ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ»

Виконав:

студент 3 курсу

групи ДА-81

Головін Б. О.

Варіант 5

Київ – 2021

***Лабораторна робота 4***

**Асинхронні обчислення.**

***Мета роботи:***

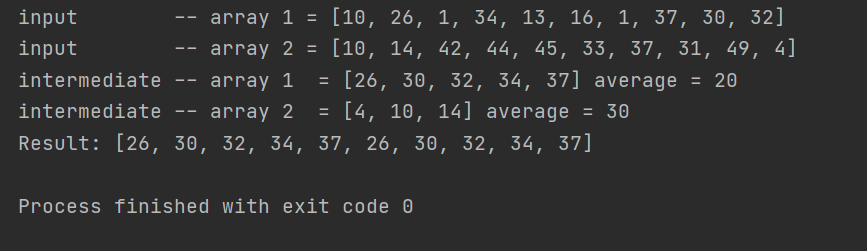
Розробка і реалізація паралельного алгоритму з використанням механізмів асинхронних обчислень

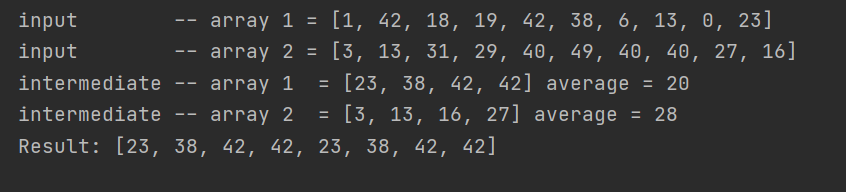
***Завдання*** Варіант 5

**5)** Створити 2 масиви (або колекції) з випадковими числами. У першому масиві - залишити елементи які більше середнього значення масиву, в другому - менше. Відсортувати масиви і злити в один відсортований масив ті елементи, які є в одному масиві і немає в іншому.

**Лістинг програми та її результати роботи**

import java.util.\*;  
import java.util.concurrent.CompletableFuture;  
import java.util.concurrent.ExecutionException;  
  
public class Main {  
 private static int *SIZE* = 10;  
  
 private static ArrayList<Integer> getRandom(int SIZE, int index) {  
 ArrayList<Integer> array = new ArrayList<>();  
 for (int i = 0; i < SIZE; i++) {  
 int randMin = 0;  
 int randMax = 50;  
 array.add(randMin + (int) (Math.*random*() \* randMax));  
 }  
 System.*out*.println("input -- array " + index + " = " + array);  
 return array;  
 }  
  
 public static void main(String[] args) {  
 // заполняем ArrayList`ы рандомными значениями от 0 до 50  
 final ArrayList<Integer> a1 = *getRandom*(*SIZE*, 1);  
 final ArrayList<Integer> a2 = *getRandom*(*SIZE*, 2);  
 // создаём итераторы для каждого массива для возможности изменения внутри лямбда функции  
 Iterator<Integer> iterator1 = a1.iterator();  
 Iterator<Integer> iterator2 = a2.iterator();  
  
 CompletableFuture<ArrayList<Integer>> firstFuture, secondFuture, resultFuture;  
  
 // асинхронно выполняем код лямбда-функции для данного фьючера  
 // вычисляем сумму и находим среднее значение  
 // находим элементы которые меньше или равны среднему и удаляем (с помощью итератора),  
 // получая массив с данными которые больше среднего  
 // далее сортируем  
 firstFuture = CompletableFuture.*supplyAsync*(() -> a1).thenApplyAsync(first -> {  
 int sum = 0, average = 0;  
 for (Integer element : a1) {  
 sum += element;  
 }  
 average = sum / a1.size();  
 while (iterator1.hasNext()) {  
 Integer item = iterator1.next();  
 if (item <= average) {  
 iterator1.remove();  
 }  
 }  
 Collections.*sort*(first);  
 System.*out*.println("intermediate -- array 1 = " + first + " average = " + average);  
 return first;  
 });  
  
 // асинхронно выполняем код лямбда-функции для данного фьючера  
 // вычисляем сумму и находим среднее значение  
 // находим элементы которые больше или равны среднему и удаляем (с помощью итератора),  
 // получая массив с данными которые меньше среднего  
 // далее сортируем  
 secondFuture = CompletableFuture.*supplyAsync*(() -> a2).thenApplyAsync(second -> {  
 int sum = 0, average = 0;  
 for (Integer element : a2) {  
 sum += element;  
 }  
 average = sum / a2.size();  
 while (iterator2.hasNext()) {  
 Integer item = iterator2.next();  
 if (item >= average) {  
 iterator2.remove();  
 }  
 }  
 Collections.*sort*(second);  
 System.*out*.println("intermediate -- array 2 = " + second + " average = " + average);  
 return second;  
 });  
  
 // находим разность массивов (множеств),  
 // проверяя есть ли данный элемент в другом массиве  
 // если нет, то добавляем  
 resultFuture = firstFuture.thenCombine(secondFuture, (first, second) -> {  
 ArrayList<Integer> a3 = new ArrayList<>(first);  
 for (Integer i : a1) {  
 if (!a2.contains(i))  
 a3.add(i);  
 }  
 return a3;  
 });  
 // выводим получившийся массив, при этом ославливая возможные ошибки  
 try {  
 System.*out*.println("Result: " + resultFuture.get());  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 } catch (ExecutionException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
}

****

****

**Опис роботи**

Для вирішення задачі за допомогою асинхронної роботи треба використати клас CompletableFuture

CompletableFuture - клас для асинхронної роботи, який дає можливість комбінувати кроки обробки, поєднуючи їх в ланцюжок. Клас містить близько 50 методів для виконання, об'єднання а так само обробки винятків.

С початку створюємо два ArrayList`и рандомними значеннями від 0 до 50. Та створюмо для ітератори для можливості зміни їх всередині лямбда функції. Також створюмо три об’єкти CompletableFuture<ArrayList<Integer>> firstFuture, secondFuture - для асинхроних обчислень, CompletableFuture<ArrayList<Integer>> resultFuture для комбінування результатів попердніх фьючерсів. Результат вийде як логічна різниця двох попередніх масивів.

Фьчерси firstFuture, secondFuture виконують работу по такому принципу:

* асинхронно виконуємо код лямбда-функції для даного Ф'ючер
* обчислюємо суму і знаходимо середнє значення
* знаходимо елементи які менше (більше) або дорівнюють середньому і видаляємо (за допомогою ітератора),
* отримуючи масив з даними які більше (менше) середнього
* далі сортуємо

Фьючер resultFuture виконує:

* знаходимо різницю масивів (множин),
* перевіряючи чи є даний елемент в іншому масиві
* якщо немає, то додаємо

**Висновки**

У даній лабораторній роботі ми ознайомитись з поняттями використання фьючерсів Java. А також проводилась розробка і реалізація програми яка моделює таку задачу:

* Створити 2 масиви (або колекції) з випадковими числами. У першому масиві - залишити елементи які більше середнього значення масиву, в другому - менше. Відсортувати масиви і злити в один відсортований масив ті елементи, які є в одному масиві і немає в іншому.