Ejercicio Vector:

Llenar un vector de dimensión = 80 con valores (1,2,3,4) de forma aleatoria en la misma proporción, 20 veces cada número.

***Secuencial***

package Vector;// @Author: CABRERA PICOITA NAHOMI ASTRID

import java.util.Random;

// Llenar un vector de dimensión = 80 con valores (1,2,3,4) de forma aleatoria en la misma

// proporción, 20 veces cada numero.

public class Secuencial {

private static int[] vector = new int[80];

private static int[] comprueba = new int[4];

private static Random r = new Random();

public static void main(String[] args) {

llenarVector();

int uwu = 1;

for (int xd : vector) { // este for each permite que se imprima en filas de 20 el vector principal

System.out.print(xd + " - ");

if (uwu == 20 || uwu == 40 || uwu == 60) {

System.out.println();

}

uwu++;

}

}

public static void llenarVector() {

int n;

int indx = 0;

while (!verifica()) {

n = r.nextInt(4) + 1;

if (comprueba[n - 1] < 20) { // si es que el contador no ha llegado a 19 entonces se pueden seguir insertando

vector[indx] = n;

comprueba[n - 1]++;

indx++;

}

}

}

// verifica es la union de los contadores por cada número, la unión de estos contadores es un array de 4

// cuando todas los posiciones sean 20 decuelve un true indicando de que el vector principal tiene 20 veces

// cada uno de los números (1 2 3 4)

public static boolean verifica() {

for (int i = 0; i < comprueba.length; i++) {

if (comprueba[i] < 20)

return false;

}

return true;

}

}

***Hilos***

package Vector;// @Author: CABRERA PICOITA NAHOMI ASTRID

public class Hilos extends Thread {

private int n;

private static int vector[] = new int[80];

private static int cont = 0;

private int rango = 20;

public Hilos(int n) {

this.n = n;

}

// cada hilo se ejecutará 20 veces y agregarán un número en el vector que comparten los 4

// se irán agregando los números conforme vayan llegando

// lo único que recibe el hilo es el número

@Override

public void run() {

for (int i = 0; i < rango; i++) {

System.out.println("Agregado: " + n);

vector[cont] = n;

cont++;

}

}

public int getN() {

return n;

}

public static int[] getVector() {

return vector;

}

public static void main(String[] args) {

Hilos h1 = new Hilos(1);

Hilos h2 = new Hilos(2);

Hilos h3 = new Hilos(3);

Hilos h4 = new Hilos(4);

h1.start();

h2.start();

h3.start();

h4.start();

try { // se espera a que todos acaben de llenar el vector para imprimir de igual forma que en la clase secuencial

h1.join();

h2.join();

h3.join();

h4.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

int uwu = 1;

for (int xd : getVector()) {

System.out.print(xd + " - ");

if (uwu == 20 || uwu == 40 || uwu == 60) {

System.out.println();

}

uwu++;

}

}

}

Ejercicio Matrix:

Llenar una matriz de dimensión (n x m (par)) con valores de 1 a n de forma aleatoria en la proporción de veces cada número (si la matriz es 3x4 se debe llenar con 1,2,3 4 veces cada número de forma aleatoria)

***Secuencial***

package Matriz;// @Author: CABRERA PICOITA NAHOMI ASTRID

import java.util.Random;

import java.util.Scanner;

// Llenar una matriz de dimensión (n x m (par)) con valores de 1 a n de forma aleatoria en la proporcion de veces cada

// numero (si la matriz es 3x4 se debe llenar con 1,2,3 4 veces cada numero de forma aleatoria)

public class Secuencial {

private static int[] comprueba;

private static int[][] matriz;

private static Scanner sc = new Scanner(System.in);

private static Random r = new Random();

public static void main(String[] args) {

System.out.println("Ingrese las filas:");

int f = sc.nextInt();

System.out.println("Ingrese las columnas (número par):");

int c = sc.nextInt();

comprueba = new int[f];

matriz = new int[f][c];

if (c % 2 != 0) {

System.out.println("Las columnas deben ser un número par");

System.out.println("Adiós :D");

} else {

llenarMatriz();

for (int i = 0; i < f; i++) {

for (int j = 0; j < c; j++) {

System.out.print(matriz[i][j]);

}

System.out.println();

}

}

}

// lo que hay que tener en cuenta en este problema es que el número de veces que se van a repetir los números del

// 1 al numero de filas, serán el número de columnas, por eso el array de contadores debe

// llegar al número de columnas - 1

public static void llenarMatriz() {

int n;

int ixFila = 0;

int ixColum = 0;

while (!verifica()) {

n = r.nextInt(matriz.length) + 1; // es lo mismo que esto int n = 1 + r.nextInt(f);

if (comprueba[n - 1] < matriz[0].length) { // si es que el contador no ha llegado a num de columnas entonces se pueden seguir insertando

if (ixColum == matriz[0].length) {

ixColum = 0;

ixFila++;

}

matriz[ixFila][ixColum] = n;

comprueba[n - 1]++;

ixColum++;

}

}

}

public static boolean verifica() {

for (int i = 0; i < comprueba.length; i++) {

if (comprueba[i] < matriz[0].length)

return false;

}

return true;

}

}

***Hilos***

package Matriz;// @Author: CABRERA PICOITA NAHOMI ASTRID

import java.util.Scanner;

import java.util.concurrent.locks.ReentrantLock;

public class Hilos extends Thread {

private static int[][] matriz;

private static Scanner sc = new Scanner(System.in);

private static ReentrantLock lock = new ReentrantLock();

private static int ixFila = 0;

private static int ixColum = 0;

private int row;

private int f;

private int c;

private int n;

public Hilos(int row, int f, int c, int n) {

this.row = row;

this.f = f;

this.c = c;

this.n = n;

if (matriz == null) {

matriz = new int[f][c];

}

}

/\*

@Override

public void run() {

for (int i = 0; i < c; i++) {

lock.lock(); // esto es para evitar la desincronización o inconsistencia de datos

try {

System.out.println(getName() + " Agregado: " + n);

matriz[row][i] = n;

} finally {

lock.unlock();

}

}

}

ESTE RUN DE AQUÍ EVITA QUE LAS CARRERAS ENTRE HILOS PASEN Y ASÍ EVITA QUE LOS DATOS SE SOBREPONGAN UNO

SOBRE OTRO

\*/

@Override

public void run() {

for (int i = 0; i < c; i++) {

System.out.println(getName() + " Agregado: " + n);

matriz[ixFila][ixColum] = n;

ixFila++;

ixColum++;

if (ixFila == f) {

ixFila = 0;

}

if (ixColum == c) {

ixColum = 0;

}

}

}

public static int[][] getMatriz() {

return matriz;

}

public static void main(String[] args) {

System.out.println("Ingrese las filas:");

int f = sc.nextInt();

System.out.println("Ingrese las columnas (número par):");

int c = sc.nextInt();

if (c % 2 != 0) {

System.out.println("Las columnas deben ser un número par");

System.out.println("Adiós :D");

} else {

Thread[] threads = new Thread[f];

int numeros = 1;

for (int i = 0; i < threads.length; i++) { // se le envía la row por el run en el comentario

threads[i] = new Hilos(i, f, c, numeros); // se inicializan los hilos dentro del array y se corren

threads[i].start();

numeros++;

}

for (Thread xd : threads) { // se espera a que todos los hilos acaben

try {

xd.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

for (int i = 0; i < f; i++) { // se imprime

for (int j = 0; j < c; j++) {

System.out.print(getMatriz()[i][j] + " ");

}

System.out.println();

}

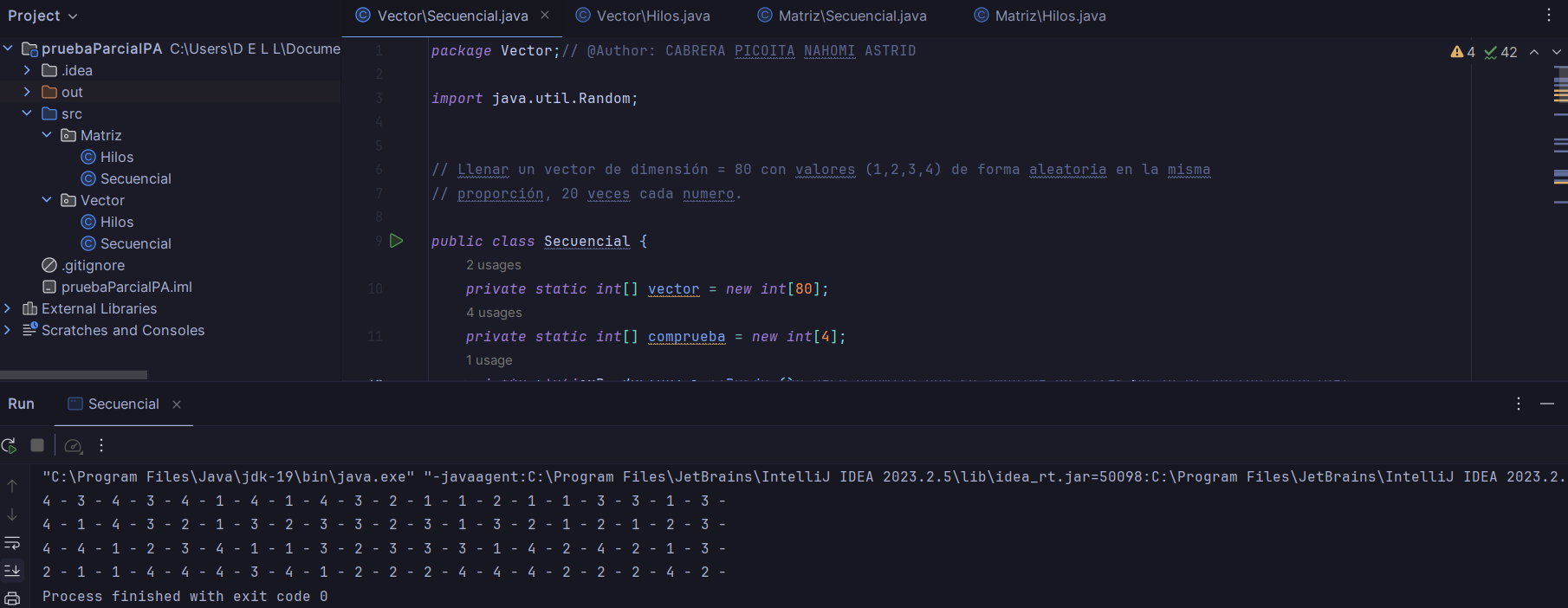
}

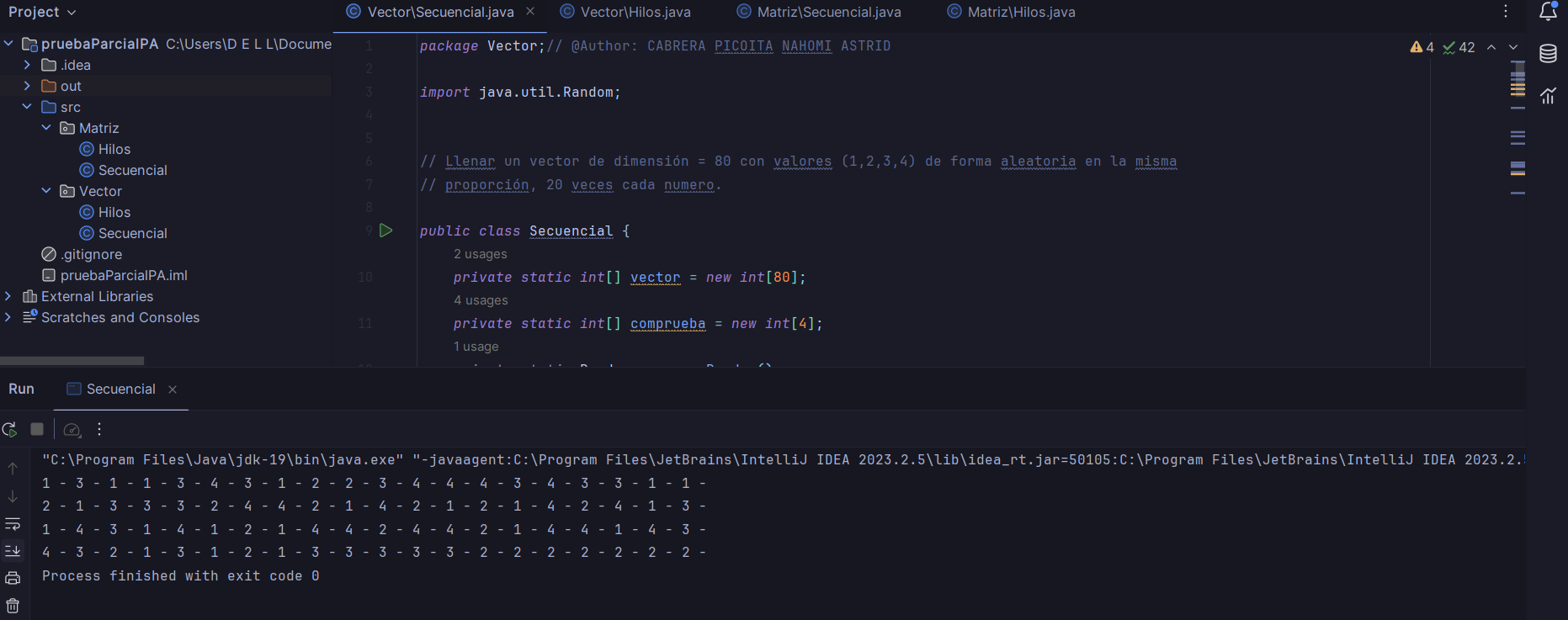
}

}

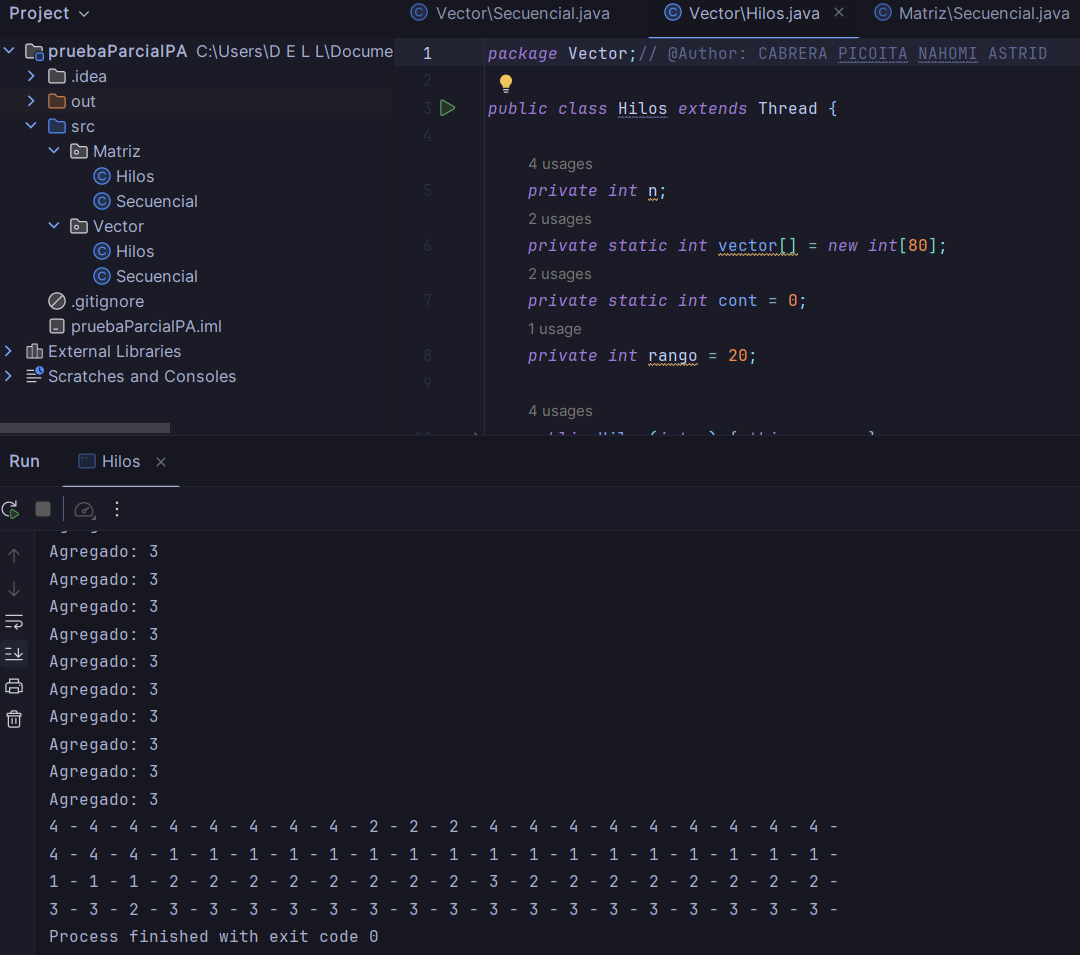
***Capturas:***

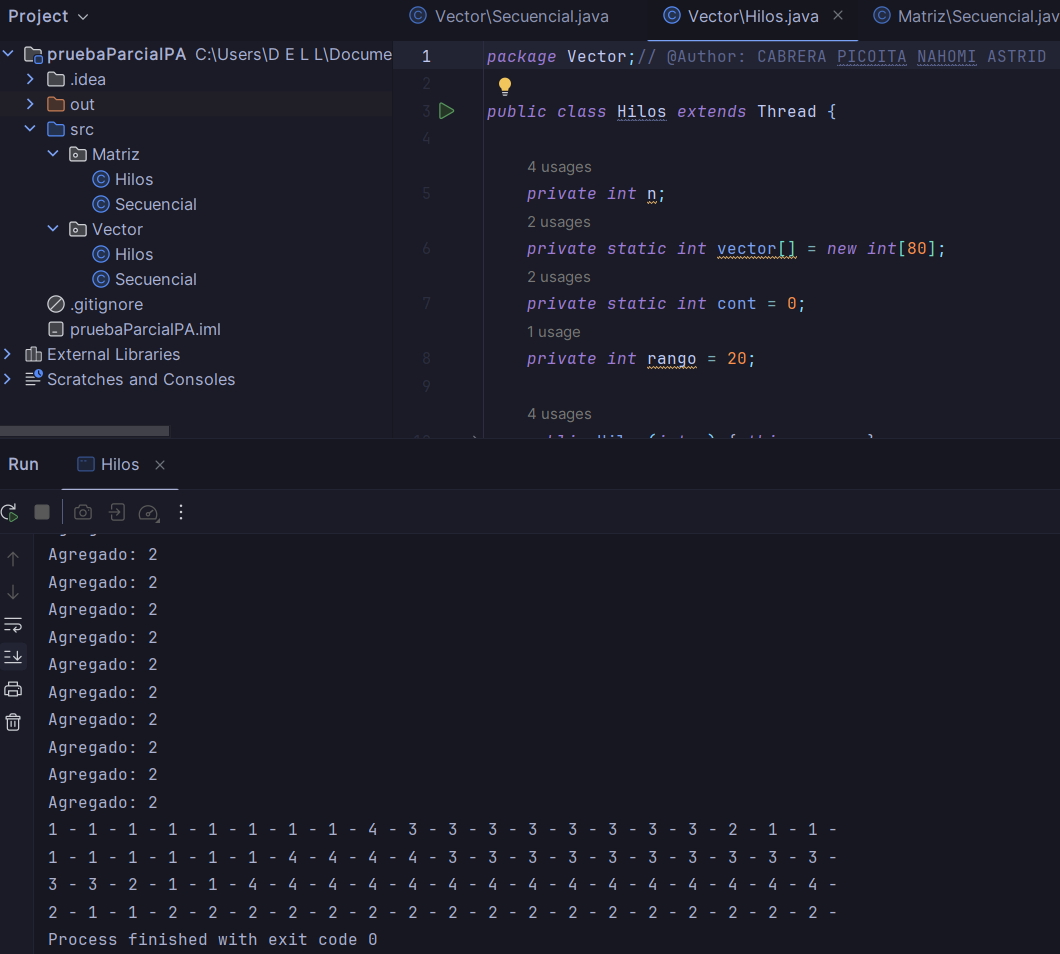
***Vector Secuencial***

******

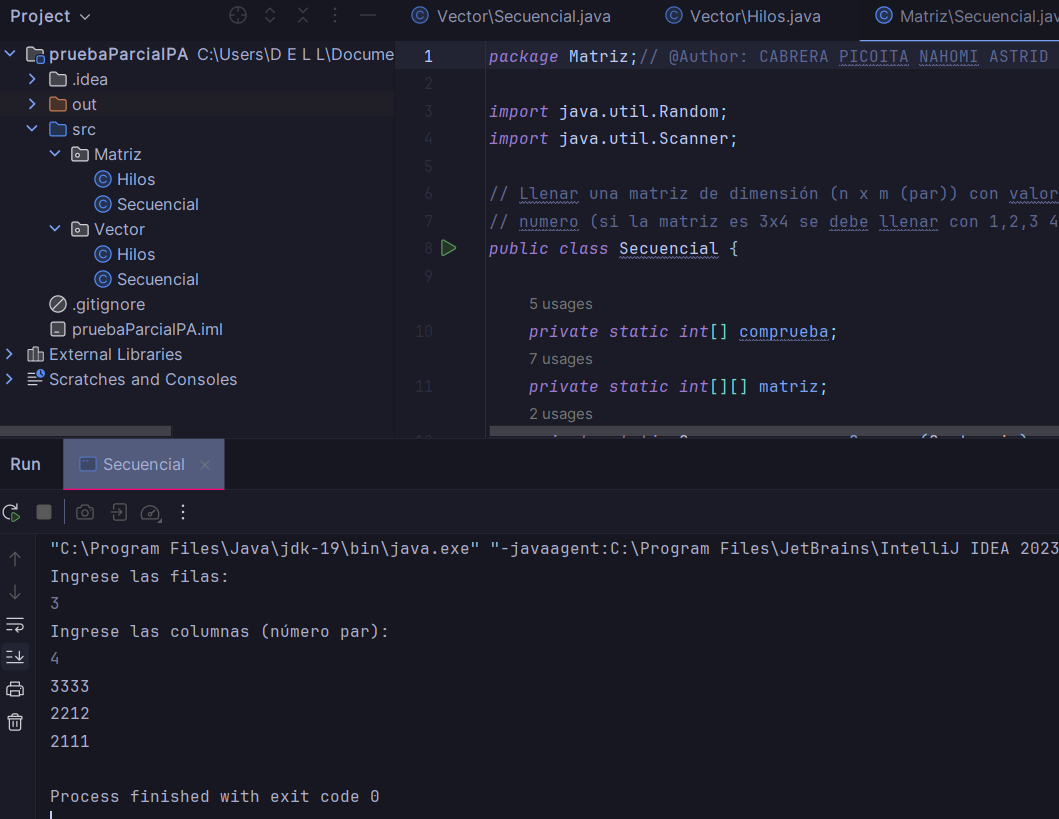
******

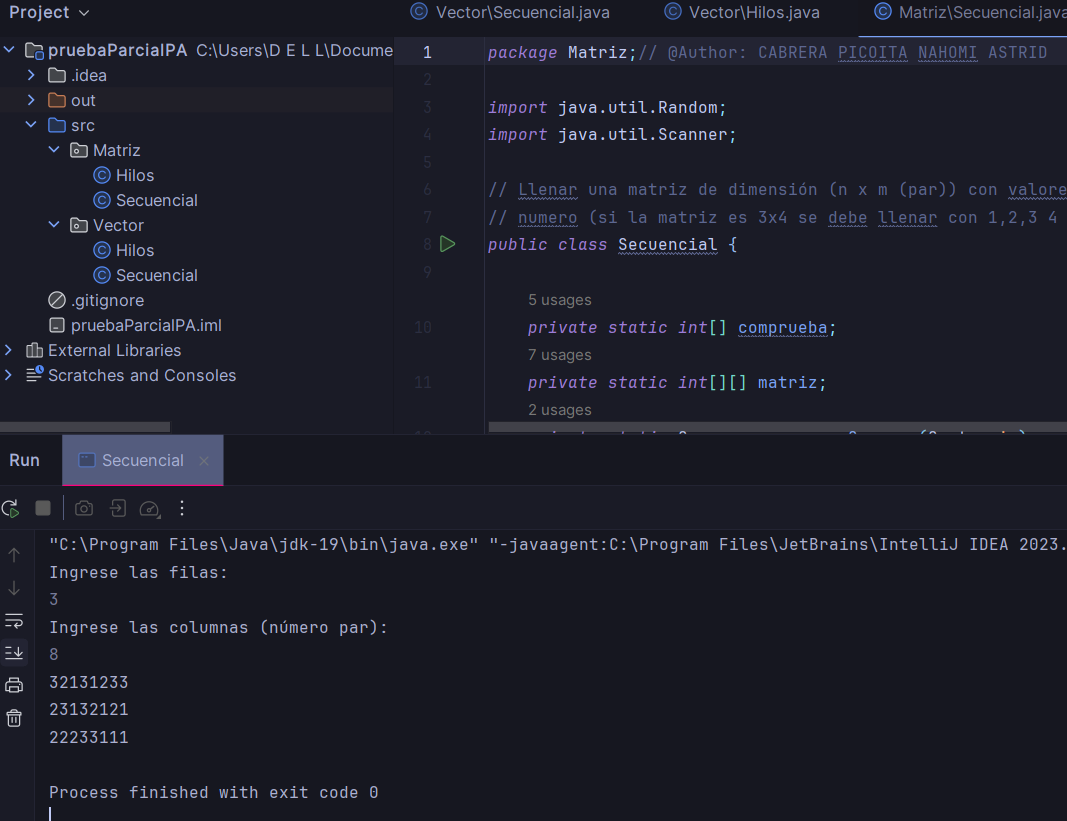
***Vector Hilos***

******

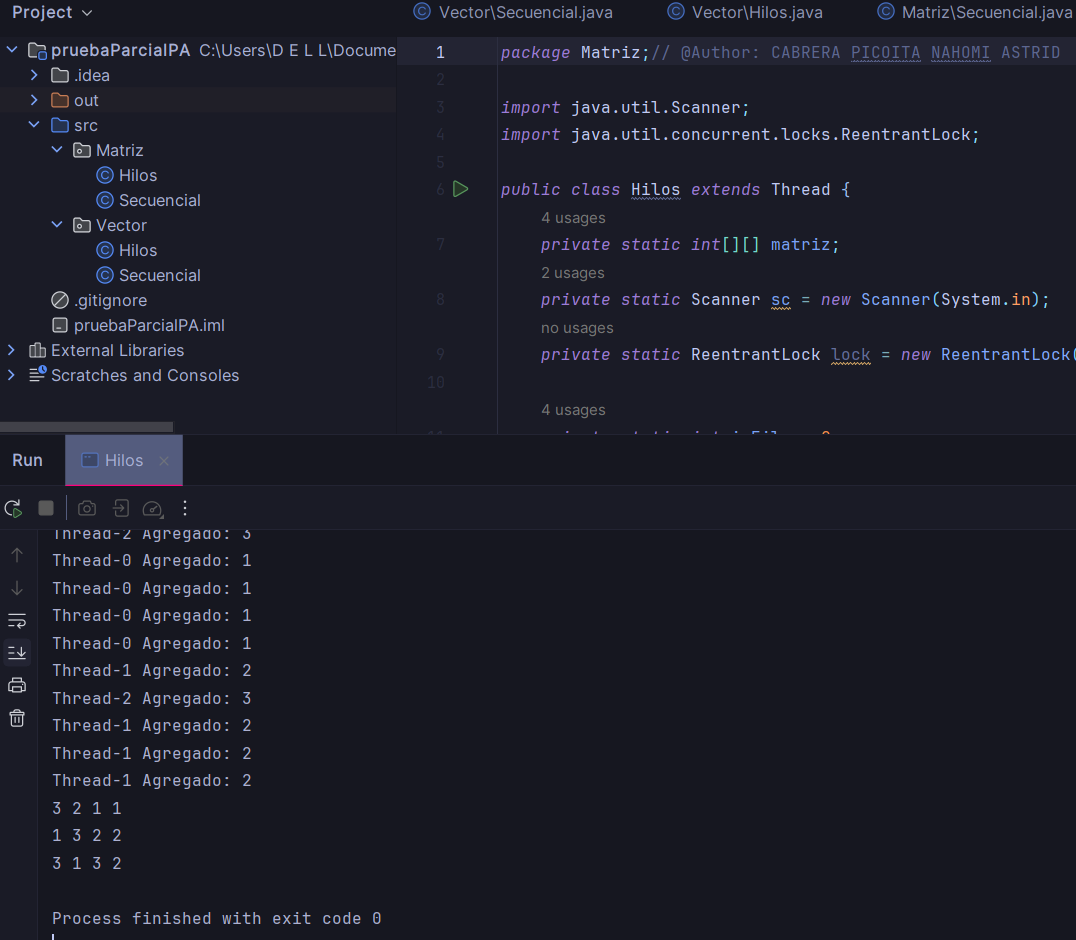
******

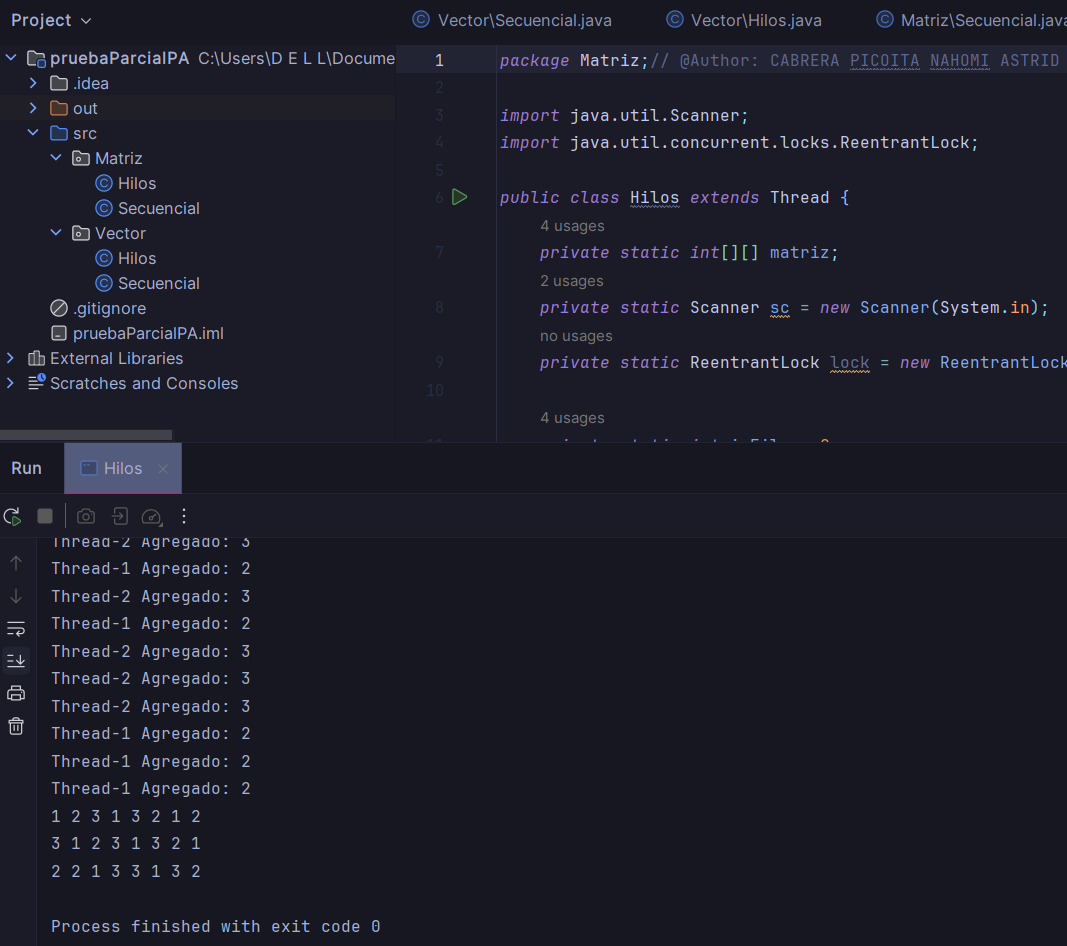
***Matriz Secuencial***

******

******

***Matriz Hilos***

******

******