Yazılım Mühendisliği-2.Sınıf

Zeynep Nuriye Tekin

220609010

Kemal Çağrı Serdaroğlu

İşletim Sistemleri

Assignment 1

1. **A Bölümümün Kod Açıklaması:**

Ödevin A kısmı için yazılan kod üzerinde, kullanıcıdan N ve M değerleri istenir. İş parçacığı(thread) kullanılarak N boyutlu rastgele dizi oluşturur. Oluşturulan dizi M adet alt kümlere bölünür ve aritmetik işlemler uygulanır.

package application;

import java.util.Scanner;

class MWorkerThread extends Thread {

private int[] array;

private int startIndex;

private int endIndex;

“MWorkerThread“ adında sınıf oluşturulur ve bu sınıf Thread sınıfından türetilmiştir. MWorkerThread sınıfının “array” adında bir dizi, “stardIndex” adında alt kümenin başlangıç indisi ve “endIndex” olarak tanımlanmış alt kümenin bitiş indisi olmak üzere belli özellikler tanımlanmıştır.

public MWorkerThread(int[] array, int startIndex, int endIndex) {

this.array = array;

this.startIndex = startIndex;

this.endIndex = endIndex;

}

MWorkerThread sınıfının kurucu metodu kullanılır. Bu metod belirlediğimiz array, startIndex ve endIndex değişkenlerini başlatmaya yarar.

public void run() {

int max = array[startIndex];

int min = array[startIndex];

int sum = 0;

System.out.println("Alt Kume Elemanlari:");

for (int i = startIndex; i <= endIndex; i++) {

if (array[i] > max) {

max = array[i];

}

if (array[i] < min) {

min = array[i];

}

sum += array[i];

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

}

System.out.println();

double average = (double) sum / (endIndex - startIndex + 1);

“run” olarak oluşturulan metodun içerisinde, en küçük, en büyük, toplam ve ortalama değişkenleri tanımlanır. For döngüsü açılarak alt küme elemanlarından min, max ve sum değerleri bulunur. Bu değerlere göre average hesaplanır.

synchronized (System.out) {

System.out.println("Alt Kume Bilgileri:");

System.out.println("En Buyuk: " + max);

System.out.println("En Kucuk: " + min);

System.out.println("Toplam: " + sum);

System.out.println("Ortalama: " + average);

System.out.println();

}

}

}

Run metosunun devamında, “Synchronized(System.out)” ile hesaplanan değerler ekrana çıktı verir. Synchronized anahtar kelimesini kullanmazsak eğer çıktı doğru olmaz. Her alt küme için verilecek çıktı cümleleri üst üste denk gelir. Bu durum düzensizliğe ve karmaşaya sebep olur. Synchronized kelimesi yazılan kodun birden fazla thread tarafından aynı anda erişilmesini önler. Yazılan kod sırası ile konsolda görünmüş olur.

public class MainA {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Dizinin boyutunu girin (N):");

int N = scanner.nextInt();

System.out.println("Her alt kumenin eleman sayisini girin (M):");

int M = scanner.nextInt();

int[] array = generateRandomArray(N);

“MainA” adındaki ana metod açılır. Kullanıcıdan sayı alınacağı için Scanner sınıfı içer aktarılır. Bu sınıf System.in aracılığı ile veri okumayı sağlar. “scanner” nesnesi oluşturulur. Kullanıcıdan mesaj yoluyla dizinin boyutu için N değeri ve dizinin alt küme eleman sayısı için M değerini girmesi istenir. Kullanıcının girdiği tam sayı değeri okunur. Okunan değer tanımlanan N, M değişkenlerine atanır.

Rastgele elemanların oluşturulması için generateRandomArray fonksiyonu çağırılır ve array dizisine atanır.

for (int i = 0; i < N; i += M) {

int startIndex = i;

int endIndex = Math.min(i + M - 1, N - 1);

MWorkerThread thread = new MWorkerThread(array, startIndex, endIndex);

thread.start();

try {

thread.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

scanner.close();

}

Dizi üzerinde işlemler gerçekleştirebilmek için çoklu iş parçacıkları(threads) kullanılır. İlk önce dizi elemanlarını alt küme halinde işlemek için for döngüsü açılır. Döngünün her adımında M kadar elemanlık alt küme işlenir. startIndex alt kümenin başlangıç indisini, endIndex ise bitiş indisini belirler. Son alt kümenin dizinin son elemanını geçmemesi için “Math.min” yöntemi kullanılır.

Her alt küme için bir MWorkerThread nesnesi oluşturulur. Bu sayede her bir alt küme kendi iş parçacığında işlenir. Join() yöntemi her alt küme işleminin tamamlanmasını bekler. İş parçacıklarının senkronize şekilde çalışmasını sağlar. Hata olması duruma karşı “InterruptedException” bu hatayı ayıklamak amacıyla ekrana yazdırır.

private static int[] generateRandomArray(int size) {

int[] array = new int[size];

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

array[i] = (int) (Math.random() \* 10) + 1;

}

return array;

}

}

generateRandomArray metodu, belirtilen boyutta rastgele bir dizi oluşturur ve bu diziyi döndürür. Oluşturulan rastgele dizinin elemanları 1 ile 10 arasında olacaktır.

1. **B Bölümünün Kod Açıklaması:**

Ödevin B kısmı için yazılan kod üzerine, verilen boyuttaki matrisin her bir satırının maksimum minimun toplam ve ortalaması hesaplanır. Bu hesaplama iş parçacıkları ile yapılır. Her bir iş parçası verilen işlemleri yapar ve sonuçları döndürür.

package application;

import java.util.Random;

class NWorkerThread extends Thread {

private int[][] matrix;

private int row;

private int max;

private int min;

private int sum;

private double average;

Rastgele sayı üretmek için "java.util.Random” kütüphanesi tanımlanır. “NWorkerThread” adında sınıf oluşturulur ve bu sınıf Thread sınıfından türetilmiştir. NWorkerThread sınıfının “matrix” adında bir matris, “row” adında işlenecek satırın indisi, “max” adında en büyük değer, “min” adında en küçük değer, “sum” adında satır elemanları toplamı, “average” adında satırdaki elemanların ortalamasını ifade eden değişkenler tanımlanır. Bu değişkenler NWorkerThread sınıfının tanımlanan özellikleridir.

public NWorkerThread(int[][] matrix, int row) {

this.matrix = matrix;

this.row = row;

this.max = Integer.MIN\_VALUE;

this.min = Integer.MAX\_VALUE;

this.sum = 0;

}

Burada NWorkerThread sınıfının kurucu metodu oluşturulur. Kurucu metod, bir iş parçası oluşturulduğunda çağrılır ve iş parçasının özelliklerini çağırır. Max değişkenine Integer.MIN\_VALUE atanmasının sebebi işlenecek değerlerin her zaman bu değerden büyük olacağı anlamına gelir. Min değişkenine Integer.MAX\_VALUE atanmasının sebebi işlenecek değerlerin her zaman bu değerden küçük olacağı anlamına gelir. 0 sayısı sum değerine atanır.

public void run() {

int[] rowArray = matrix[row];

for (int num : rowArray) {

if (num > max) {

max = num;

}

if (num < min) {

min = num;

}

sum += num;

}

average = (double) sum / rowArray.length;

}

“run” adında oluşturduğumuz metod, iş parçasının çalışma mantığını açıklar. matrix[row] ifadesiyle, iş parçasının işleyeceği matristen belirtilen satırın elemanlarını alır ve rowArray dizisine atar.

For döngüsü içerisinde sırası ile satırda elemanların en büyüğü ve en küçüğü bulunur. Bulmak için if koşulundan faydalanılır. Bu döngü içinde sum değeri elemanlar ile toplanır. Döngü kapatılır. Average; sum ve uzunluk metodu ile hesaplanır.

public int getMax() {

return max;

}

public int getMin() {

return min;

}

public int getSum() {

return sum;

}

public double getAverage() {

return average;

}

}

NWorkerThread sınıfının dışarıya erişimi açık olan metodlarını tanımlar. Bu metotlar, iş parçasının hesapladığı değerlere dışarıdan erişimi sağlar.

getMax(), metodu iş parçasının hesapladığı belirtilen satırdaki en büyük değeri döndürür.

getMin(), metodu iş parçasının hesapladığı belirtilen satırdaki en küçük değeri döndürür.

getSum(), metodu iş parçasının hesapladığı belirtilen satırdaki deyıların toplam değerini döndürür.

getAverage(), metodu iş parçasının hesapladığı belirtilen satırdaki elemanların ortalama değerini döndürür.

public class MainB {

public static void main(String[] args) {

int N = 10;

int[][] matrix = generateRandomMatrix(N);

NWorkerThread[] threads = new NWorkerThread[N];

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

threads[i] = new NWorkerThread(matrix, i);

threads[i].start();

}

İş parçacıklarını oluşturur ve başlatır. İş parçacıkları belirtilen matrisin her bir satırını işlemek için oluşturulur. Oluşturulacak matrisin boyutunu belirlemek için N değişkenine 10 değeri atanır. Fonksiyon çağırılarak boyutu N olan rastgele matris oluşturulur ve matrix değişkenine atanır.

NWorkerThread tipinde threads dizisi oluşturulur. Bu dizide her bir iş parçacığı için bir öğe bulunur. For döngüsü her bir satır için bir iş parçacığı oluşturur ve başaltır. Döngünün her bir adımında i indisindeki satır için iş parçacığı oluşturulur. “NWorkerThread(matrix, i)” ile ilgili satır ve matris iş parçacığına iletilir. Start() metodu ile oluşturulan iş parçacığı çağrılır.

try {

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

threads[i].join();

}

}

catch (InterruptedException e) {

e.printStackTrace();

}

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

System.out.println("Satır " + i + " için bilgiler:");

System.out.println("Alt Kume Elemanlari: " + arrayToString(matrix[i]));

System.out.println("En Büyük: " + threads[i].getMax());

System.out.println("En Küçük: " + threads[i].getMin());

System.out.println("Toplam: " + threads[i].getSum());

System.out.println("Ortalama: " + threads[i].getAverage());

System.out.println();

}

}

for döngüsü içindeki join() metodu, her bir iş parçacığının tamamlanmasını bekler. Bu şekilde, iş parçacıkları paralel olarak çalışır ve tamamlanmalarını beklenir. Hata ayıklayıcı için “e.printStackTrace();” metodu kullanılır. Tüm iş parçacıkları tamamlandıktan sonra for döngüsü ile her bir iş parçacığının hesapladığı değerler ekrana yazdırılır.

private static int[][] generateRandomMatrix(int N) {

int[][] matrix = new int[N][N];

Random random = new Random();

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

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

matrix[i][j] = random.nextInt(10) + 1;

}

}

return matrix;

}

“generateRandomMatrix” metodu parametre olarak N değerini alır, boyutunu belirler. N boyutunda bir matris oluşturur. Random sınıfından bir nesne oluşturularak random adında bir değişkene atanır. Bu nesne rastgele sayılar üretmemizi sağlar. İki tane for döngüsü iç içe açılarak dıştaki döngü matrisin satırlarını ifade ederken içteki döngü sütunlarını ifade eder. random.nextInt(10) + 1 ifadesiyle, 1 ile 10 arasında rastgele bir sayı elde edilir ve bu sayı ilgili matris elemanına atanır. Burada nextInt(10) ifadesi ile 0 ile 9 arasında bir rastgele sayı elde edilir ve buna 1 eklenerek 1 ile 10 arasında bir rastgele sayı elde edilmiş olur. Son olarak oluşturulan matris döndürülür.

private static String arrayToString(int[] array) {

StringBuilder sb = new StringBuilder();

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

sb.append(array[i]);

if (i != array.length - 1) {

sb.append(", ");

}

}

return sb.toString();

}

}

Bu metod bir int dizisi alarak bu diziyi bir string e dönüştürür. StringBuilder sınıfından bir sb adına bir nesne oluşturulur. Bu nesne string ifadeleri birleştirmek için kullanılır. For döngüsü oluşturulur. Bu döngü dizinin elemanlarında tek tek gezinir. Son olarak StringBuilder nesnesi diziyi tekrardan string veri türüne dönüştürür ve bu değeri döndürür.

1. **C Bölümünün A Kod Açıklaması:**

package application;

import java.util.Scanner;

public class MainCA {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Dizinin boyutunu girin (N):");

int N = scanner.nextInt();

System.out.println("Her alt kumenin eleman sayisini girin (M):");

int M = scanner.nextInt();

int[] array = generateRandomArray(N);

int chunkSize = M;

Bu kod bloğu, kullanıcıdan bir dizi boyutu (N) ve her bir alt kümenin eleman sayısını (M) alır. Daha sonra, N boyutunda rastgele bir dizi oluşturmak için generateRandomArray metodunu çağırır.

chunkSize değişkenine M değeri atanır, ancak bu değişken kodun geri kalanında kullanılmaz. Muhtemelen ilerleyen satırlarda kullanılmak üzere bir işarettir, ancak şu anki kod parçasında kullanılmamaktadır.

for (int i = 0; i < N; i += M) {

int startIndex = i;

int endIndex = Math.min(i + M - 1, N - 1);

System.out.println("Alt Kume Elemanlari:");

for (int j = startIndex; j <= endIndex; j++) {

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

}

System.out.println();

processChunk(array, startIndex, endIndex);

}

scanner.close();

}

Bu kod bloğu, bir diziyi belirli boyuttaki alt kümelerine böler ve her bir alt küme için belirli bir işlemi gerçekleştirir. Dıştaki `for` döngüsü, diziyi tarar ve alt kümeler oluşturur. İçteki döngü, her bir alt kümenin elemanlarını yazdırır.

Her bir alt küme için belirli bir işlem, `processChunk` metodunda gerçekleştirilir. Son olarak, girişten alınan veri akışı kapatılır.

private static void processChunk(int[] array, int startIndex, int endIndex) {

int max = array[startIndex];

int min = array[startIndex];

int sum = 0;

for (int i = startIndex; i <= endIndex; i++) {

if (array[i] > max) {

max = array[i];

}

if (array[i] < min) {

min = array[i];

}

sum += array[i];

}

Bu “processChunk”metodu, belirli bir alt kümenin elemanlarını işler. Başlangıç ve bitiş indisleri arasındaki elemanlar taranarak, en büyük ve en küçük değerler belirlenir, ayrıca elemanların toplamı hesaplanır.

Bu sayede alt kümenin en büyük ve en küçük elemanları ile toplam değeri elde edilir. Bu işlemler, belirli bir alt küme için yapılır ve sonuçlar ilgili değişkenlere atanır.

double average = (double) sum / (endIndex - startIndex + 1);

System.out.println("Alt Kume Bilgileri:");

System.out.println("En Buyuk: " + max);

System.out.println("En Kucuk: " + min);

System.out.println("Toplam: " + sum);

System.out.println("Ortalama: " + average);

System.out.println();

}

Bu bölümde, belirli bir alt kümenin işlenmesinden sonra hesaplanan bilgiler ekrana yazdırılır. İlk olarak, o alt kümedeki sayıların ortalaması hesaplanır ve ardından bu bilgiler alt kümeyle ilgili olarak ekrana basılır. En büyük, en küçük, toplam ve ortalama değerler alt kümenin karakteristik özelliklerini yansıtır ve kullanıcıya sunulur. Her bir alt küme işleminin sonucu ayrı ayrı görüntülenir.

private static int[] generateRandomArray(int N) {

int[] array = new int[N];

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

array[i] = (int) (Math.random() \* 10) + 1;

}

return array;

}

}

Bu metod, belirtilen boyutta (N) rastgele bir dizi oluşturur. Her bir dizi elemanı, 1 ile 10 arasında rastgele bir tam sayıdır. Bu işlem, dizi boyutu kadar tekrarlanır ve her bir eleman rastgele bir değer alır.

Sonuç olarak, oluşturulan rastgele dizi geri döndürülür. Bu metot, çeşitli algoritmaların ve işlemlerin denemeleri için kullanılabilir veya rastgele veri gerektiren uygulamalarda kullanılabilir.

1. **C Bölümünün B Kod Açıklaması:**

//İşletim sistemi ödevi-C

package application;

import java.util.Random;

public class MainCB {

public static void main(String[] args) {

int N = 10;

int[][] matrix = generateRandomMatrix(N);

10x10 boyutunda rastgele bir matris oluşturur. `generateRandomMatrix` metodunu çağırarak bu matrisi oluşturur. Matrisin her bir elemanı, 1 ile 10 arasında rastgele bir tam sayı değeridir. Bu matris, genellikle çeşitli algoritmaların ve işlemlerin test edilmesi veya simülasyonlarında kullanılabilir. Oluşturulan matris, daha sonra işlenmek veya başka hesaplamalar için kullanılabilir.

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

System.out.println("Satır " + i + " için bilgiler:");

System.out.println("Alt Kume Elemanlari: " + arrayToString(matrix[i]));

System.out.println("En Büyük: " + findMax(matrix[i]));

System.out.println("En Küçük: " + findMin(matrix[i]));

System.out.println("Toplam: " + calculateSum(matrix[i]));

System.out.println("Ortalama: " + calculateAverage(matrix[i]));

System.out.println();

}

}

For döngüsü, oluşturulan matrisin her bir satırı için işlem yapar. Her iterasyonda, bir satırın bilgileri ekrana yazdırılır. Bu bilgiler arasında, satırın elemanları, en büyük ve en küçük değerleri, toplamı ve ortalama değeri bulunur. `arrayToString`, `findMax`, `findMin`, `calculateSum` ve `calculateAverage` metotları kullanılarak bu hesaplamalar gerçekleştirilir. Bu döngü, matrisin her satırının istatistiklerini ekrana yazdırmak için kullanılır.

private static int[][] generateRandomMatrix(int N) {

int[][] matrix = new int[N][N];

Random random = new Random();

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

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

matrix[i][j] = random.nextInt(10) + 1;

}

}

return matrix;

}

Bu metod, belirtilen boyutta (N x N) rastgele değerler içeren bir matris oluşturur. İki katmanlı bir döngü kullanarak, her satır ve sütuna rastgele değerler atanır. `Random` sınıfından bir nesne oluşturularak rastgele sayılar elde edilir ve bu sayılar 1 ile 10 arasında olacak şekilde ayarlanır. Oluşturulan matris, metot tarafından döndürülerek çağrıldığı yere iletilir. Bu metodun kullanılması, belirli bir boyutta rastgele değerlerle doldurulmuş bir matris elde etmek için tercih edilir.

private static int findMax(int[] array) {

int max = Integer.MIN\_VALUE;

for (int num : array) {

if (num > max) {

max = num;

}

}

return max;

}

Bu metod, bir tam sayı dizisinin en büyük değerini bulmamızı sağlar. Başlangıçta `max` değişkeni en küçük değer olarak atanır. Daha sonra dizi elemanları üzerinde bir döngü kullanılarak, dizi elemanları tek tek kontrol edilir. Eğer eleman, mevcut maksimum değerden büyükse, `max` değişkeni bu elemana atanır ve döngü devam eder. Bu şekilde döngü tamamlandığında, dizinin en büyük elemanı bulunur ve `max` değeri döndürülür. Bu metodun kullanımı, bir dizinin en büyük elemanını belirlemek için pratik bir yol sağlar.

private static int findMin(int[] array) {

int min = Integer.MAX\_VALUE;

for (int num : array) {

if (num < min) {

min = num;

}

}

return min;

}

Bu metod, bir tam sayı dizisinin en küçük değerini bulmamızı sağlar. Başlangıçta `min` değişkeni en büyük olası değer olarak atanır. Daha sonra dizi elemanları üzerinde bir döngü kullanılarak, dizi elemanları tek tek kontrol edilir. Eğer eleman, mevcut minimum değerden küçükse, `min` değişkeni bu elemana atanır ve döngü devam eder. Bu şekilde döngü tamamlandığında, dizinin en küçük elemanı bulunur ve `min` değeri döndürülür. Bu metodun kullanımı, bir dizinin en küçük elemanını belirlemek için pratik bir yol sağlar.

private static int calculateSum(int[] array) {

int sum = 0;

for (int num : array) {

sum += num;

}

return sum;

}

Bu metod, bir tam sayı dizisinin elemanlarının toplamını hesaplamamızı sağlar. Başlangıçta `sum` değişkeni sıfır olarak atanır. Daha sonra dizi elemanları üzerinde bir döngü kullanılarak, her eleman `sum` değişkenine eklenir. Bu şekilde döngü tamamlandığında, dizinin tüm elemanlarının toplamı `sum` değişkeninde saklanır ve bu değer döndürülür. Bu metodun kullanımı, bir dizinin elemanlarının toplamını bulmak için pratik bir yol sağlar.

private static double calculateAverage(int[] array) {

int sum = calculateSum(array);

return (double) sum / array.length;

}

Bu metod, bir tam sayı dizisinin elemanlarının ortalamasını hesaplamamızı sağlar. İlk olarak, `calculateSum` metodunu kullanarak dizinin elemanlarının toplamını hesaplar. Daha sonra, bu toplam değeri dizinin uzunluğine bölünerek ortalama hesaplanır. Sonuç olarak, ortalama değeri ondalıklı bir sayı olarak döndürülür. Bu metod, bir dizinin elemanlarının ortalamasını bulmak için kullanılabilir ve hesaplama işlemini basitleştirir.

private static String arrayToString(int[] array) {

StringBuilder sb = new StringBuilder();

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

sb.append(array[i]);

if (i != array.length - 1) {

sb.append(", ");

}

}

return sb.toString();

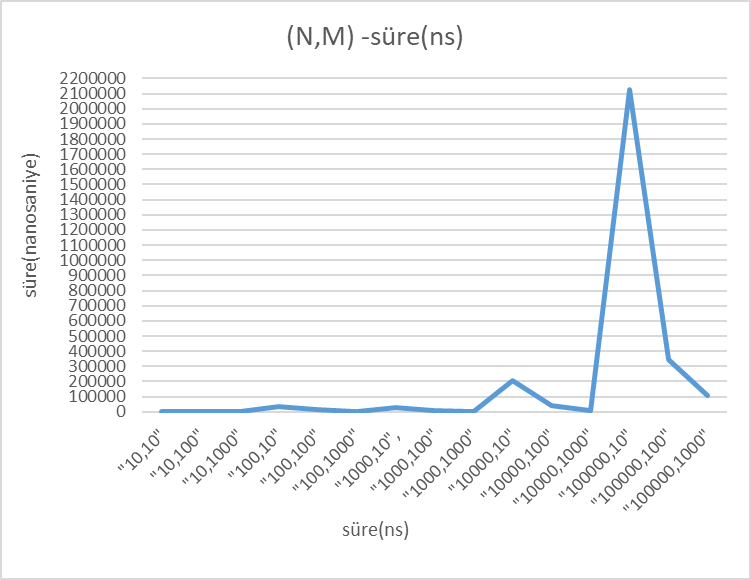
}

}

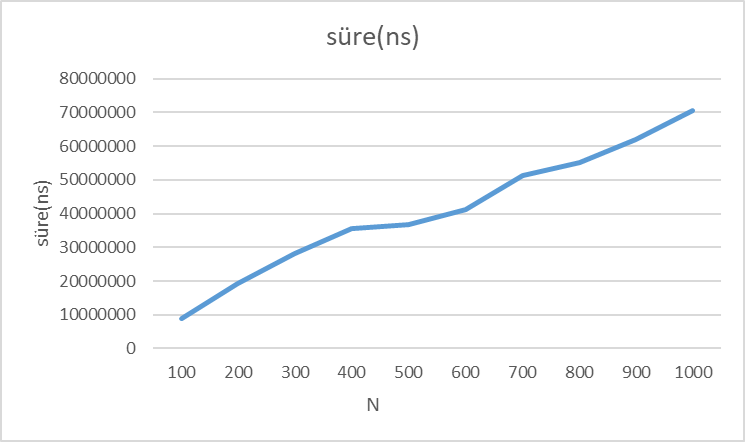
Bu metod, bir tam sayı dizisini bir metin dizesine dönüştürmemizi sağlar. Metod, bir StringBuilder nesnesi oluşturur ve dizinin her elemanını bu nesneye ekler. Her elemanın ardından bir virgül ve bir boşluk ekler, ancak son elemandan sonra virgül eklenmez.

Son olarak, StringBuilder'daki içeriği bir dize olarak döndürür. Bu şekilde, dizinin elemanları virgülle ayrılarak tek bir metin dizesi haline getirilir. Bu metot genellikle dizi elemanlarını yazdırmak veya göstermek için kullanılır.

1. **A Bölümünün Grafiği:**



1. **B Bölümünün Grafiği:**



1. **C Bölümünün A ile Karşılaştırılması:**

|  |  |  |
| --- | --- | --- |
| **"(N,M)"** | **normal A süre** | **thread'sız A süre** |
| "10,10" | 2650 | 2900 |
| "100,10" | 35790 | 680.0 |
| "100,100" | 12200 | 6400.0 |
| "1000,10", | 28060 | 1690.0 |
| "1000,100" | 7610 | 5610.0 |
| "1000,1000" | 1050 | 33200.0 |
| "10000,10" | 208150 | 474.2 |
| "10000,100" | 38080 | 2991.0 |
| "10000,1000" | 10220 | 31380.0 |
| "100000,10" | 2124890 | 276.16 |

1. **C Bölümünün B ile Karşılaştırılması:**

|  |  |  |
| --- | --- | --- |
| **N** | **normal B süre** | **thread'sız B**  **süre** |
| 100 | 9040600 | 2079500 |
| 200 | 19301900 | 1755600 |
| 300 | 28304100 | 2798200 |
| 400 | 35436800 | 2798200 |
| 500 | 36808900 | 5544200 |
| 600 | 41170300 | 6737400 |
| 700 | 51321900 | 9912000 |
| 800 | 55167900 | 12187900 |
| 900 | 62013200 | 15947700 |
| 1000 | 70638600 | 17335100 |

1. **System.nanoTime() fonksiyonunun kullanımı:**

nanoTime() metodu, java.lang.System sınıfının bir parçası olarak bulunur ve zamanı nanosaniye cinsinden döndürür. Bu, Java Sanal Makinesinin (JVM) çalışma zamanındaki geçerli zamanı döndürür. Bu değer, nanosaniye hassasiyeti sağlar ve genellikle geçen süreyi ölçmek için kullanılır.

nanoTime() metodu, sistem veya saat zamanı ile ilgili değildir. Sabit ancak keyfi bir başlangıç zamanına kadar olan nanosaniye değerlerini gösterir. Bu değerler, aynı JVM örneği içinde elde edilen iki değer arasındaki farklar hesaplandığında anlamlı hale gelir. Örneğin, bir kodun ne kadar sürede tamamlandığını kontrol edebiliriz.

\*NOT: Yorum satırları kodlarda bulunmaktadır. Ancak rapora yorum satırları değil, kod satırları yazılmış ve açıklanmıştır.

KAYNAKÇA  
<https://medium.com/@yusufcancelik/javada-thread-lere-giriş-2-synchronized-anahtar-kelimesi-c028ebe874e6>

https://medium.com/@mhmtonar1/java-synchroni̇zed-da117a782e9c

<https://www-educative-io.translate.goog/answers/what-is-the-systemnanotime-function-in-java?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=tr&_x_tr_hist=true>