**Yazılım Mühendisliği-2.Sınıf**

**Zeynep Nuriye Tekin**

**220609010**

**Kemal Çağrı Serdaroğlu**

**İşletim Sistemleri**

**Assignment 2**

**İÇİNDEKİLER:**

1. Java Konsol Kodları syf3

1.1. A Kodları syf3

1.2. B Kodları syf8

1.3. C Kodları syf14

1. Java Swing Arayüz Kodları syf19

2.1. A Kodları syf19

2.2. B Kodları syf22

2.3. C Kodları syf26

1. Race Condition Neden Önemli? Syf28
2. Lock’lar Neden Önemli? Syf29
3. A, B ve C Kodlarının Çalışma Zamanı Grafiği syf30

1. A, B ve C Kodlarının Çalışma Zamanı Tablosu syf30
2. Grafikte A ve B Birbirine Neden Çok Yakın? Syf30
3. System.nanoTime() Fonksiyonu syf31
4. **Java Konsol Kodları:**
   1. **A Kodu:**

package application;

import java.util.Random;

import java.util.Scanner;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

public class odev2\_A {

private static int[][] matrix; // Matrisi temsil eden dizisi

private static int matrixSize; // Matrisin boyutu

private static boolean negatifVar = false; // Matrisin içinde negatif sayı olup olmadığını belirler

private static int negatifSayisi = 0; // Matristeki negatif sayıların toplam sayısı

private static int pozitifSayisi = 0; // Matristeki pozitif sayıların toplam sayısı

private static int enBuyukSayi = Integer.MIN\_VALUE; // Matristeki en büyük sayı

private static int enKucukSayi = Integer.MAX\_VALUE; // Matristeki en küçük sayı

private static Lock lock = new ReentrantLock(); // ReentrantLock nesnesi, çoklu iş parçacığı arasında güvenliğini sağlamak için kullanılır

Ödevin A kısmı için yazılan kodda;

Çoklu iş parçacıklarının aynı veriye eriştiği senaryolarda oluşabilecek yarış koşullarını (race condition) önlemek için bir ReentrantLock nesnesi kullanıyor. Matris, matrisin boyutu, negatif sayacı tanımlanıyor. Ayrıca matristeki negatif/pozitif sayıların toplam sayısı ve en büyük/en küçük sayısı da tanımlanıyor.

ReentrantLock sınıfından lock adında nesne oluşturuluyor. Bu nesne çoklu iş sayısı arasında güvenli bir şekilde veri erişimi sağlamak için kullanılıyor.

public static void main(String[] args) {

getUserInput(); // Kullanıcıdan matris boyutunu al

initializeMatrix(matrixSize); // Matrisi rastgele sayılarla doldur

Thread[] threads = new Thread[matrixSize]; // Her satırı işleyecek iş parçacığı dizisi

// Her bir satır için bir iş parçacığı oluştur ve çalıştır

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

final int row = i;

threads[i] = new Thread(() -> {

processRow(row); // Satırı işle

});

threads[i].start(); // İş parçacığını başlat

}

// Tüm iş parçacıklarının tamamlanmasını bekleyelim

for (Thread thread : threads) {

try {

thread.join(); // İş parçacığının tamamlanmasını bekleyelim

} catch (InterruptedException e) {

e.printStackTrace();

}

}

// Sonuçları ve matrisi ekrana yazdır

System.out.println("Matris:");

printMatrix(matrix);

System.out.println("Negatif sayı var mı? " + negatifVar);

System.out.println("Negatif sayıların sayısı: " + negatifSayisi);

System.out.println("Pozitif sayıların sayısı: " + pozitifSayisi);

System.out.println("En büyük sayı: " + enBuyukSayi);

System.out.println("En küçük sayı: " + enKucukSayi);

}

Main fonksiyonu başlıyor. Kullanıcıdan matrisin boyutunu almak için “getUserInput” fonksiyonu çalıştırılıyor. MatrixSize boyutunda bir matris oluşturulup, rastgele sayılarla doldurmak için “initializeMatrix” fonksiyonu çağırılıyor

Her satırı işleyecek iş parçacıklarını içeren dizi oluşturuluyor. Her satıra bir iş parçacığı oluşturmak için for döngüsü başlatılıyor.

“final int row = i;” satırı iç içe lambda ifadelerinde kullanılmak üzere döngü değişkeninin bir nüshasını alıyor.

“ threads[i] = new Thread(() -> {processRow(row); }); ” ise her bir satır için bir iş parçacığı oluşturuyor ve bu iş parçacığının işleyeceği satırı belirliyor. Satırı işliyor. Oluşturulan iş parçacığı başlatılıyor, işlemler eşzamanlı gerçekleşiyor.

Tüm iş parçacıklarının tamamlanmasını beklemek için döngü başlatıyor. “Thread.join();” ise her iş parçacığının tamamlanmasını bekiyor böylece ana iş parçacığı diğerlerinin bitmesini bekliyor.

Gerekli fonksiyonlar çağırılarak matris, negatif var mı, en büyük sayı gibi sonuçlar ekrana çıktı veriliyor.

private static void getUserInput() {

Scanner scanner = new Scanner(System.in);

System.out.print("Matris boyutunu girin (10 ile 20000 arasında bir sayı): ");

while (true) {

try {

matrixSize = scanner.nextInt(); // Kullanıcıdan matris boyutunu al

if (matrixSize < 10 || matrixSize > 20000) {

System.out.print("Geçersiz boyut. Lütfen 10 ile 20000 arasında bir sayı girin: ");

} else {

break;

}

} catch (Exception e) {

System.out.print("Geçersiz giriş. Lütfen bir tam sayı girin: ");

scanner.next(); // Hatalı girişi temizle

}

}

}

Kullanıcıdan giriş almak için kullanılan “getUserInput” fonksiyonu çalıştırılıyor. Kullanıcıdan giriş almak için bir Scanner nesnesi oluşturuluyor. System.in giriş akışını tarıyor. Kullanıcıdan matris sayısını girmesi için mesaj yazdırılıyor.

Sonsuz döngü başlatılıyor. Kullanıcı doğru giriş yapana kadar bu döngü devam ediyor. Kullanıcı girişini kontrol etmek için “try-catch” bloğu başlatılıyor. Kullanıcıdan bir tam sayı girişi alınıyor ve matrixSize değişkenine atanıyor. İf bloğu ile Kullanıcının girdiği boyutun istenen dahilinde olup olmadığı kontrol ediliyor. Geçersiz giriş olduğu belirtiliyor ve tekrar giriş yapması isteniyor. Kullanıcı doğru giriş yapıyorsa else ile döngüden çıkılıyor. Catch ile kullanıcının girdiği değerin tam sayı olup olmadığı kontrol ediliyor. Hatalı giriş olursa scanner.next ile düzeltiliyor.

private static void initializeMatrix(int size) {

matrix = new int[size][size]; // Matrisi belirtilen boyutta oluştur

Random rand = new Random();

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

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

matrix[i][j] = rand.nextInt(11) - 5; // Matrisi rastgele sayılarla doldur (-5 ile 5 arasında)

}

}

}

Matrisin başlatılması için kullanılan “initializeMatrix” fonksiyonu başlatılıyor. Belirtilen boyutta bir matris oluşturuluyor. Boyutu size olan bir kare bir matris oluşturuluyor.

Rastgele sayı üretmek için bir Random nesnesi oluşturuluyor. Matrisin satırları üzerinde döngü başlıyor. Matrisin sütunları üzerinde döngü başlıyor. Matris rastgele sayılar ile dolduruluyor. Matrisin her elemanına -5 ile 5 aralığında sayılar atanıyor.

private static void processRow(int row) {

for (int num : matrix[row]) {

lock.lock(); // Kilit al

try {

// a) Negatif sayı kontrolü

if (num < 0) {

negatifVar = true;

}

// b) Negatif ve pozitif sayı sayımı

if (num < 0) {

negatifSayisi++;

} else {

pozitifSayisi++;

}

// c) En büyük ve en küçük sayı tespiti

if (num > enBuyukSayi) {

enBuyukSayi = num;

}

if (num < enKucukSayi) {

enKucukSayi = num;

}

} finally {

lock.unlock(); // Kilidi serbest bırak

}

}

}

private static void printMatrix(int[][] matrix) {

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

for (int j = 0; j < matrix[i].length; j++) {

System.out.print(matrix[i][j] + "\t");

}

System.out.println();

}

}

}

processRow fonksşyonu belirtilen bölümleri işlemek için kullanılıyor. For döngüsünde belirtilen bölümlerin elemanlarının döngüsü başlıyor. Kritik bölgelerden önce bir şifreleme parametreleri olan “lock” şeklinde kilitleme yapılıyor.

Try ile kritik bölgeye giriş hazırlıkları yapılıyor. Satırın her elemanı için negatif olup olmadığı kontrol ediliyor. Eleman negatifse negatifVar değişkeni true olarak değiştiriliyor. Negatif ve pozitif performansın sayımı yapılıyor. Eleman negatifse negatifSayısı artırılırsa pozitifSayısı artırılmıyor. Matrisin en büyük ve en küçük unsurları belirleniyor.

Eğer eleman mevcut en büyük sayıdan büyükse, enBuyukSayi bu eleman olarak güncelleniyor.

Eğer eleman mevcut en küçük sayıdan küçükse, enKucukSayi bu eleman olarak güncelleniyor.Kritik bölge çıkarıldığında, kilidin serbest bırakılması için unLock() yöntem çağrılıyor.

Matrisi yazdırmak için kullanılan printMatrix fonksiyonu başlatılıyor. Matirisin satırları üzerinde döngü başlıyor. Her sütunun sütunları üzerinde döngü başlıyor. Matrisin bileşenleri yazdırılıyor ve aralarına bir sekme karakteri ekleniyor.

A Çıktısı:

Matris boyutunu girin (10 ile 20000 arasında bir sayı): 12

Matris:

-4 -1 -4 1 0 3 -5 -1 -2 1 0 -4

3 2 0 -3 3 2 -5 0 3 0 -3 4

0 5 3 -5 5 3 -4 -5 -5 1 5 4

5 3 3 -1 -4 0 0 -5 4 5 -3 4

2 2 -5 4 5 5 3 -1 3 0 0 -4

-5 4 -4 0 3 -1 -2 1 1 -4 4 3

-4 1 3 4 1 -1 -3 1 0 3 -4 -3

2 0 -3 -3 -4 0 3 4 -4 0 1 4

-1 -3 -3 -4 1 5 5 3 -3 -5 3 -1

5 -2 -4 2 0 -4 -3 3 4 -5 3 -3

-4 -1 -2 4 1 -5 5 -4 -3 -5 0 2

2 -3 0 3 3 -4 4 -1 1 5 5 3

Negatif sayı var mı? true

Negatif sayıların sayısı: 58

Pozitif sayıların sayısı: 86

En büyük sayı: 5

En küçük sayı: -5

* 1. **B Kodu:**

package application;

import java.util.Random;

import java.util.Scanner;

import java.util.concurrent.Semaphore;

public class Odev2\_B {

private static int[][] matris; // Matrisimizi temsil eden 2D dizisi

private static int boyut; // Matrisin boyutu

private static boolean negatifVar = false; // Matrisin içinde negatif sayı olup olmadığını belirten bayrak

private static int negatifSayisi = 0; // Matristeki negatif sayıların toplam sayısı

private static int pozitifSayisi = 0; // Matristeki pozitif sayıların toplam sayısı

private static int enBuyukSayi = Integer.MIN\_VALUE; // Matristeki en büyük sayı

private static int enKucukSayi = Integer.MAX\_VALUE; // Matristeki en küçük sayı

private static Semaphore semaphore = new Semaphore(1); // Sadece bir thread'in aynı anda işlem yapmasına izin vermek için Semaphore nesnesi

public static void main(String[] args) {

boyutuAl(); // Kullanıcıdan matris boyutunu al

matrisiOlustur(boyut); // Matrisi rastgele sayılarla doldur

Thread[] threadler = new Thread[boyut]; // Her satırı işleyecek iş parçacığı dizisi

// Her bir satır için bir iş parçacığı oluştur ve çalıştır

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

final int satir = i;

threadler[i] = new Thread(() -> {

satirIsle(satir); // Satırı işle

});

threadler[i].start(); // İş parçacığını başlat

}

// Tüm iş parçacıklarının tamamlanmasını bekleyelim

for (Thread thread : threadler) {

try {

thread.join(); // İş parçacığının tamamlanmasını bekleyelim

} catch (InterruptedException e) {

e.printStackTrace();

}

}

// Sonuçları ve matrisi ekrana yazdır

System.out.println("Matris:");

matrisiYazdir(matris);

System.out.println("Negatif sayı var mı? " + negatifVar);

System.out.println("Negatif sayıların sayısı: " + negatifSayisi);

System.out.println("Pozitif sayıların sayısı: " + pozitifSayisi);

System.out.println("En büyük sayı: " + enBuyukSayi);

System.out.println("En küçük sayı: " + enKucukSayi);

}

Java programının giriş noktası olan main fonksiyonu başlıyor. “boyutuAl();”Kullanıcıdan matrisin boyutunu girmesini sağlayan boyutuAl fonksiyonu çağrılıyor. “matrisiOlustur(boyut);” Belirtilen boyutta bir matris oluşturup, rastgele sayılarla dolduran matrisiOlustur fonksiyonu çağrılıyor.

Thread[] threadler = new Thread[boyut]; Her satırı işleyecek iş parçacıklarını içeren bir dizi oluşturuluyor. Her bir satır için bir iş parçacığı oluşturmak için bir for döngüsü başlıyor.

“final int satir = i;” Güvenliği sağlamak için, iş parçacıklarının içinde kullanılacak olan satır numarası bir değişkene atanıyor.

Her bir iş parçacığı, belirli bir satırı işleyecek şekilde oluşturuluyor. “threadler[i].start();” Oluşturulan iş parçacığı başlatılıyor ve çalıştırılıyor. İş parçacıklarının tamamlanmasını beklemek için bir döngü başlıyor.

“thread.join();” Her iş parçacığının tamamlanmasını beklemek için join metodu çağrılıyor.

Matrisin ekrana yazdırılacağı bilgisini içeren bir çıktı oluşturuluyor. “matrisiYazdir(matris);” Oluşturulan matrisi ekrana yazdırmak için matrisiYazdir fonksiyonu çağrılıyor.

“System.out.println("Negatif sayı var mı? " + negatifVar);” Matriste negatif sayı olup olmadığını belirten bayrağın durumunu ekrana yazdıran bir çıktı oluşturuluyor.

“System.out.println("Negatif sayıların sayısı: " + negatifSayisi);” Matristeki negatif sayıların sayısını ekrana yazdıran bir çıktı oluşturuluyor.

System.out.println("Pozitif sayıların sayısı: " + pozitifSayisi);: Matristeki pozitif sayıların sayısını ekrana yazdıran bir çıktı oluşturuluyor.

“System.out.println("En büyük sayı: " + enBuyukSayi);” Matristeki en büyük sayıyı ekrana yazdıran bir çıktı oluşturuluyor.

“System.out.println("En küçük sayı: " + enKucukSayi);” Matristeki en küçük sayıyı ekrana yazdıran bir çıktı oluşturuluyor.

private static void boyutuAl() {

Scanner scanner = new Scanner(System.in);

System.out.print("Matris boyutunu girin (10 ile 20000 arasında bir sayı): ");

while (true) {

try {

boyut = scanner.nextInt(); // Kullanıcıdan matris boyutunu al

if (boyut < 10 || boyut > 20000) {

System.out.print("Geçersiz boyut. Lütfen 10 ile 20000 arasında bir sayı girin: ");

} else {

break;

}

} catch (Exception e) {

System.out.print("Geçersiz giriş. Lütfen bir tam sayı girin: ");

scanner.next(); // Hatalı girişi temizle

}

}

}

“private static void boyutuAl()” boyutuAl fonksiyonunu başlıyor. Kullanıcıdan girdi almak için bir Scanner nesnesi oluşturuluyor ve standart giriş (klavye) ile ilişkilendiriliyor. “System.out.print("Matris boyutunu girin (10 ile 20000 arasında bir sayı): ");” Kullanıcıya uygun bir matris boyutu girmesi için bir istek metni yazdırılıyor.

“while (true)” Sonsuz bir döngü başlatıyor. (Kullanıcı geçerli bir giriş yapana kadar devam edecek.) try ise Döngünün içinde bir try-catch bloğu başlıyor.

“boyut = scanner.nextInt();” Kullanıcıdan bir tamsayı girişi alınıyor ve boyut değişkenine atanıyor. if (boyut < 10 || boyut > 20000) {: Girilen boyutun geçerliliği kontrol ediliyor. Eğer girilen boyut 10 ile 20000 arasında değilse: “System.out.print("Geçersiz boyut. Lütfen 10 ile 20000 arasında bir sayı girin: ");” Kullanıcıya geçersiz bir boyut girdiği hatırlatılıyor ve yeni bir giriş isteniyor.

else { break; }: Eğer girilen boyut geçerli ise döngüden çıkılıyor.} catch (Exception e) {: Eğer kullanıcı geçerli olmayan bir giriş yaparsa (örneğin bir tamsayı değilse) bir istisna (exception) oluşuyor ve bu catch bloğu çalışıyor.

“System.out.print("Geçersiz giriş. Lütfen bir tam sayı girin: ");” Kullanıcıya geçersiz bir giriş yaptığı hatırlatılıyor ve yeni bir giriş isteniyor. “scanner.next();”Hatalı girişi temizlemek için next metodu çağrılıyor.

private static void matrisiOlustur(int boyut) {

matris = new int[boyut][boyut]; // Matrisi belirtilen boyutta oluştur

Random rand = new Random();

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

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

matris[i][j] = rand.nextInt(11) - 5; // Matrisi rastgele sayılarla doldur (-5 ile 5 arasında)

}

}

}

“private static void matrisiOlustur(int boyut)” matrisiOlustur` fonksiyonunun başlangıcıdır. Fonksiyon, bir tamsayı olan `boyut` parametresini alıyor. “matris = new int[boyut][boyut];” `matris` adında olan 2 boyutlu bir tamsayı dizisi oluşturuluyor. Bu dizinin boyutu, `boyut` parametresiyle belirtilen boyutta olacak şekilde ayarlanıyor. Yani `boyut`x`boyut` boyutunda bir kare matris oluşturuluyor.

“Random rand = new Random();” Rastgele sayılar oluşturmak için bir `Random` nesnesi oluşturuluyor.”for (int i = 0; i < boyut; i++)” İki katmanlı bir döngü başlatılıyor. Bu döngü, matrisin satırlarını (`i`) dolaşıyor. “for (int j = 0; j < boyut; j++)” İç içe bir döngü başlatılıyor. Bu döngü, matrisin sütunlarını (`j`) dolaşıyor. “matris[i][j] = rand.nextInt(11) - 5;” Her bir matris elemanına, -5 ile 5 arasında rastgele bir tamsayı atanıyor. `nextInt(11)` metodu, 0 ile 10 arasında (11 hariç) rastgele bir tamsayı üretiyor. Buna -5 eklenerek -5 ile 5 arasında bir değer elde ediliyor.

private static void satirIsle(int satir) {

for (int num : matris[satir]) {

try {

semaphore.acquire(); // Kritik bölgeye giriş

// a) Negatif sayı kontrolü

if (num < 0) {

negatifVar = true;

}

// b) Negatif ve pozitif sayı sayımı

if (num < 0) {

negatifSayisi++;

} else {

pozitifSayisi++;

}

// c) En büyük ve en küçük sayı tespiti

if (num > enBuyukSayi) {

enBuyukSayi = num;

}

if (num < enKucukSayi) {

enKucukSayi = num;

}

} catch (InterruptedException e) {

e.printStackTrace();

} finally {

semaphore.release(); // Kritik bölgeden çıkış

}

}

}

private static void matrisiYazdir(int[][] matris) {

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

for (int j = 0; j < matris[i].length; j++) {

System.out.print(matris[i][j] + "\t");

}

System.out.println();

}

}

}

“private static void satirIsle(int satir)” Bu metod, satır numarasını alıp bu satırdaki her bir elemanı işliyor. “for (int num : matris[satir])” Belirtilen `satir` numarasındaki matrisin elemanlarını (`num`) döngü ile dolaşıyor.”try { semaphore.acquire();” Semaphore kullanarak kritik bölgeye giriş yapıyor. (Kritik bölge, birden fazla iş parçacığının aynı anda erişimine izin verilmeyen kısımdır.)

“if (num < 0) { negatifVar = true;” Matrisin elemanı negatif ise, `negatifVar` değişkenini `true` olarak ayarlıyor. “if (num < 0) { negatifSayisi++; } else { pozitifSayisi++;” Negatif ve pozitif sayıları sayıyor. if Matrisin elemanlarının en büyüğünü belirliyor. Matrisin elemanlarının en küçüğünü belirliyor. Kesinti durumunda ekrana hata mesajı yazdırır. Kritik bölgeden çıkış yapıyor.

“private static void matrisiYazdir(int[][] matris) {`”

Matrisin satırlarını döngü ile dolaşıyor. Her bir satırın elemanlarını döngü ile dolaşıyor. Matrisin her bir elemanını ekrana yazdırıyor. Her satırın sonunda bir alt satıra geçiyor.

B Çıktısı:

Matris boyutunu girin (10 ile 20000 arasında bir sayı): 12

Matris:

1 -4 -1 4 -1 0 -4 1 -1 1 0 -2

3 -5 -4 3 3 -4 3 4 1 3 1 4

-3 1 5 2 -1 -3 -3 -3 1 5 -2 -1

2 -5 -3 1 2 -3 2 0 -2 5 2 2

-1 0 -1 -5 -4 4 -3 0 5 -3 -2 2

1 0 3 1 3 3 -4 -2 -2 -5 -2 3

1 3 4 0 -5 1 2 -4 -1 3 1 2

-5 -1 4 -4 -1 -4 -4 -2 -3 0 1 -2

2 2 1 -1 -3 -5 1 3 2 -1 4 -1

4 -5 4 0 -3 -1 -5 -2 -5 -2 -5 -4

0 5 -5 3 0 -1 -4 -2 -1 -1 -4 1

3 3 2 -4 -4 5 -1 4 -5 5 -3 -2

Negatif sayı var mı? true

Negatif sayıların sayısı: 71

Pozitif sayıların sayısı: 73

En büyük sayı: 5

En küçük sayı: -5

* 1. **C Kodu:**

package application;

import java.util.Random;

import java.util.Scanner;

public class odev2\_C {

    private static int[][] matris; // Matrisimizi temsil eden 2D dizisi

    private static int boyut; // Matrisin boyutu

    private static boolean negatifVar = false; // Matrisin içinde negatif sayı olup olmadığını belirten bayrak

    private static int negatifSayisi = 0; // Matristeki negatif sayıların toplam sayısı

    private static int pozitifSayisi = 0; // Matristeki pozitif sayıların toplam sayısı

    private static int enBuyukSayi = Integer.MIN\_VALUE; // Matristeki en büyük sayı

    private static int enKucukSayi = Integer.MAX\_VALUE; // Matristeki en küçük sayı

    public static void main(String[] args) {

        boyutuAl(); // Kullanıcıdan matris boyutunu al

        matrisiOlustur(boyut); // Matrisi rastgele sayılarla doldur

        islemleriYap(matris); // Matris üzerinde işlemleri gerçekleştir

    }

Matris tanımlanıyor. “boyutuAl():” ile kullanıcıdan matris boyutunu alınıyor.

“matrisiOlustur(int boyut):” Belirtilen boyutta bir matris oluşturuluyor ve rastgele sayılarla dolduruluyor. Oluşturulan matris üzerinde aşağıdaki işlemleri gerçekleştiryor;

Matrisin içinde negatif sayı olup olmadığını kontrol ediyor.

Matristeki negatif ve pozitif sayıların toplam sayılarını hesaplıyor.

Matristeki en büyük ve en küçük sayıları buluyor.

private static void boyutuAl() {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Matris boyutunu girin (10 ile 20000 arasında bir sayı): ");

        while (true) {

            try {

                boyut = scanner.nextInt(); // Kullanıcıdan matris boyutunu al

                if (boyut < 10 || boyut > 20000) {

                    System.out.print("Geçersiz boyut. Lütfen 10 ile 20000 arasında bir sayı girin: ");

                } else {

                    break;

                }

            } catch (Exception e) {

                System.out.print("Geçersiz giriş. Lütfen bir tam sayı girin: ");

                scanner.next(); // Hatalı girişi temizle

            }

        }

    }

private static void boyutuAl() {: Metodun başlangıcı. boyutuAl adında bir metodu tanımlıyoruz. Bu metot, matris boyutunu kullanıcıdan alıyor. “Scanner scanner = new Scanner(System.in);:” Kullanıcıdan girdi almak için bir Scanner nesnesi oluşturuyoruz ve standart giriş akışını (System.in) tarayıcıya bağlıyor.

“System.out.print("Matris boyutunu girin (10 ile 20000 arasında bir sayı): ");” Kullanıcıya matris boyutunu girmesini söyleyen bir ileti yazdırıyor. Sonsuz bir döngü başlatılır. Kullanıcı geçerli bir giriş yapana kadar bu döngü devam ediyor. Döngünün içinde bir try bloğu başlatılıyor. Bu, kullanıcının girdisinin doğru formatta olup olmadığını kontrol etmek için kullanılıyor. “boyut = scanner.nextInt();” Kullanıcıdan bir tamsayı girdisi alınır ve boyut değişkenine atanır.” if” Alınan boyutun 10 ile 20000 arasında olup olmadığı kontrol ediliyor.

“System.out.print("Geçersiz boyut. Lütfen 10 ile 20000 arasında bir sayı girin: ");” Eğer boyut geçerli aralıkta değilse, kullanıcıya geçersiz olduğu belirtiliyor ve yeni bir giriş yapması isteniyor. “else” Eğer boyut geçerli aralıkta ise, döngüden çıkılır ve kullanıcıdan giriş almayı durduruyor.} catch (Exception e) {: Eğer kullanıcının girdisi tamsayıya dönüştürülemezse veya giriş hatası olursa, bir istisna (exception) oluşur ve bu istisna Kullanıcıya geçersiz bir giriş yapıldığını belirten bir ileti gösterilir ve yeni bir giriş yapması isteniyor. “scanner.next();” Hatalı giriş temizleniyor. Bu, kullanıcının hatalı giriş yaptığında programın takılıp kalmasını engelliyor.

private static void matrisiOlustur(int boyut) {

        matris = new int[boyut][boyut]; // Matrisi belirtilen boyutta oluştur

        Random rand = new Random();

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

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

                matris[i][j] = rand.nextInt(11) - 5; // Matrisi rastgele sayılarla doldur (-5 ile 5 arasında)

            }

        }

    }

“private static void matrisiOlustur(int boyut)” Metodun başlangıcı. matrisiOlustur adında bir metodu tanımlıyoruz. Bu metot, belirtilen boyutta bir matris oluşturacak ve onu rastgele sayılarla dolduracak. matris = new int[boyut][boyut];: İki boyutlu bir diziyi (matris) belirtilen boyutta oluşturuyoruz. Boyut parametresi, matrisin satır ve sütun sayısını belirliyor. “Random rand = new Random();” Rastgele sayılar oluşturmak için bir Random nesnesi oluşturuyoruz.

“For” İki katmanlı bir döngü başlatıyoruz. Bu döngü, matrisin satırlarını (i değişkeni) dolaşıyor. “for (int j = 0; j < boyut; j++)” İkinci bir döngü daha başlatıyoruz. Bu döngü, her bir satırın sütunlarını (j değişkeni) dolaşacak. “matris[i][j] = rand.nextInt(11) - 5;” Matrisin belirli bir hücresine rastgele bir tam sayı atıyoruz. rand.nextInt(11) ifadesi, 0 ile 10 arasında bir rastgele sayı üretiyor. Buna 5 ekleyerek (range: 0-10) -5 ile 5 arasında bir sayı elde ediyoruz.

private static void islemleriYap(int[][] matris) {

        // Matris üzerinde işlemleri gerçekleştir

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

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

                int num = matris[i][j]; // Matrisin ilgili elemanını al

                // a) Negatif sayı kontrolü

                if (num < 0) {

                    negatifVar = true;

                }

                // b) Negatif ve pozitif sayı sayımı

                if (num < 0) {

                    negatifSayisi++;

                } else {

                    pozitifSayisi++;

                }

                // c) En büyük ve en küçük sayı tespiti

                if (num > enBuyukSayi) {

                    enBuyukSayi = num;

                }

                if (num < enKucukSayi) {

                    enKucukSayi = num;

                }

            }

        }

        // Sonuçları ekrana yazdırma

        System.out.println("Matris:");

        matrisiYazdir(matris);

        System.out.println("Negatif sayı var mı? " + negatifVar);

        System.out.println("Negatif sayıların sayısı: " + negatifSayisi);

        System.out.println("Pozitif sayıların sayısı: " + pozitifSayisi);

        System.out.println("En büyük sayı: " + enBuyukSayi);

        System.out.println("En küçük sayı: " + enKucukSayi);

    }

private static void islemleriYap(int[][] matris) {: Metodun başlangıcı. islemleriYap adında bir metodu tanımlıyoruz. Bu metod, belirtilen bir matris üzerinde çeşitli işlemleri gerçekleştirecek ve sonuçları ekrana yazdırıyor. for (int i = 0; i < boyut; i++) {: İki katmanlı bir döngü başlatıyor. Bu döngü, matrisin satırlarını (i değişkeni) dolaşacak. İkinci bir döngü daha başlatıyor. Bu döngü, her bir satırın sütunlarını (j değişkeni) dolaşıyor. “int num = matris[i][j];” Matrisin ilgili bir elemanını (num) alıyor.

if (num < 0) { negatifVar = true; }: Eğer eleman negatifse, negatifVar değişkenini true yaparak matriste negatif sayı olduğunu belirtiyoruz. if (num < 0) { negatifSayisi++; } else { pozitifSayisi++; }: Elemanın negatif olup olmadığını kontrol ederek, negatif ve pozitif sayıların sayısını güncelliyoruz. if (num > enBuyukSayi) { enBuyukSayi = num; }: Eleman, şu ana kadar bulunan en büyük sayıdan büyükse, enBuyukSayi değişkenini güncelliyoruz. if (num < enKucukSayi) { enKucukSayi = num; }: Eleman, şu ana kadar bulunan en küçük sayıdan küçükse, enKucukSayi değişkenini güncelliyoruz.

System.out.println("Matris:");: Matrisin başlangıcını belirten bir çıktı yazdırıyoruz. Gerekli fonksiyonları çağırarak ekrana değerlerin çıktıları veriliyor.

private static void matrisiYazdir(int[][] matris) {

        // Matrisi ekrana yazdır

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

            for (int j = 0; j < matris[i].length; j++) {

                System.out.print(matris[i][j] + "\t");

            }

            System.out.println();

        }

    }

}

matrisiYazdir adında bir metodu tanımlıyoruz. Bu metod, belirtilen bir matrisi ekrana yazdırıyor. “for” Matrisin satırlarını (i değişkeni) döngüyle dolaşıyoruz. “for (int j = 0; j < matris[i].length; j++)” Her bir satırın sütunlarını (j değişkeni) döngüyle dolaşıyoruz.

“System.out.print(matris[i][j] + "\t");” Her bir matris elemanını (matris[i][j]) ve bir tab boşluğu ("\t") ile birlikte ekrana yazdırıyor. “System.out.println();” Her satırın sonunda bir satır atlama işlemi yapıyoruz, böylece matris düzgün bir şekilde ekrana yazdırılıyor.

C Çıktısı:

Matris boyutunu girin (10 ile 20000 arasında bir sayı): 12

Matris:

1 -1 5 -4 -4 5 3 2 -1 4 2 -1

-3 5 1 0 0 -1 -2 -2 -2 -4 2 5

3 -1 1 4 4 -4 0 2 3 3 1 3

5 -1 -1 4 0 4 -3 -4 -4 2 5 1

-1 4 4 -5 4 3 4 4 4 -2 1 0

2 1 -4 2 2 -1 5 4 -3 -1 1 2

5 5 0 -3 3 3 1 -5 4 5 -4 0

-4 -2 -1 5 -2 -3 5 -4 -2 -3 -3 -3

5 1 1 -3 1 2 3 4 -4 -2 2 1

-2 -1 -1 -3 -4 3 -1 3 -1 -2 -3 -2

4 -3 5 -4 3 -2 5 -2 -4 -1 -4 5

5 1 0 1 -4 5 4 -1 3 -5 5 -4

Negatif sayı var mı? true

Negatif sayıların sayısı: 62

Pozitif sayıların sayısı: 82

En büyük sayı: 5

En küçük sayı: -5

1. **Java Swing Arayüz Kodları:**
   1. **A Kodları**

package swingex;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Random;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

public class A\_Frame extends JFrame {

    private int[][] matrix;

    private int matrixSize;

    private boolean negatifVar = false;

    private int negatifSayisi = 0;

    private int pozitifSayisi = 0;

    private int enBuyukSayi = Integer.MIN\_VALUE;

    private int enKucukSayi = Integer.MAX\_VALUE;

    private Lock lock = new ReentrantLock();

    private JTextArea textArea;

    public A\_Frame() {

        setTitle("Matris İşlemleri");

        setSize(600, 500);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new BorderLayout());

        textArea = new JTextArea();

        textArea.setEditable(false);

        JScrollPane scrollPane = new JScrollPane(textArea);

        JButton button = new JButton("Matris İşlemlerini Başlat");

        button.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                getUserInput();

                initializeMatrix(matrixSize);

                performOperations(matrix);

            }

        });

        panel.add(scrollPane, BorderLayout.CENTER);

        panel.add(button, BorderLayout.SOUTH);

        add(panel);

    }

    private void getUserInput() {

        String input = JOptionPane.showInputDialog("Matris boyutunu girin (10 ile 20000 arasında bir sayı): ");

        try {

            matrixSize = Integer.parseInt(input);

            if (matrixSize < 10 || matrixSize > 20000) {

                JOptionPane.showMessageDialog(null, "Geçersiz boyut. Lütfen 10 ile 20000 arasında bir sayı girin.");

                getUserInput();

            }

        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(null, "Geçersiz giriş. Lütfen bir tam sayı girin.");

            getUserInput();

        }

    }

    private void initializeMatrix(int size) {

        matrix = new int[size][size];

        Random rand = new Random();

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

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

                matrix[i][j] = rand.nextInt(11) - 5;

            }

        }

    }

    private void performOperations(int[][] matrix) {

        StringBuilder sb = new StringBuilder();

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

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

                int num = matrix[i][j];

                lock.lock();

                try {

                    if (num < 0) {

                        negatifVar = true;

                    }

                    if (num < 0) {

                        negatifSayisi++;

                    } else {

                        pozitifSayisi++;

                    }

                    if (num > enBuyukSayi) {

                        enBuyukSayi = num;

                    }

                    if (num < enKucukSayi) {

                        enKucukSayi = num;

                    }

                } finally {

                    lock.unlock();

                }

                sb.append(num).append("\t");

            }

            sb.append("\n");

        }

        sb.append("Negatif sayı var mı? ").append(negatifVar).append("\n");

        sb.append("Negatif sayıların sayısı: ").append(negatifSayisi).append("\n");

        sb.append("Pozitif sayıların sayısı: ").append(pozitifSayisi).append("\n");

        sb.append("En büyük sayı: ").append(enBuyukSayi).append("\n");

        sb.append("En küçük sayı: ").append(enKucukSayi).append("\n");

        textArea.setText("Matris:\n" + sb.toString());

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            public void run() {

                new A\_Frame().setVisible(true);

            }

        });

    }

}

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

* 1. **B Kodları**

package swingex;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Random;

import java.util.concurrent.Semaphore; // Semaphore için gerekli import

public class B\_Frame extends JFrame {

    private int[][] matris;

    private int boyut;

    private boolean negatifVar = false;

    private int negatifSayisi = 0;

    private int pozitifSayisi = 0;

    private int enBuyukSayi = Integer.MIN\_VALUE;

    private int enKucukSayi = Integer.MAX\_VALUE;

    private Semaphore semaphore = new Semaphore(1); // Semaphore tanımlaması

    private JTextArea textArea;

    public B\_Frame() {

        setTitle("Matris İşlemleri");

        setSize(600, 500);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new BorderLayout());

        textArea = new JTextArea();

        textArea.setEditable(false);

        JScrollPane scrollPane = new JScrollPane(textArea);

        JButton button = new JButton("Matris İşlemlerini Başlat");

        button.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                boyutuAl();

                matrisiOlustur(boyut);

                islemleriYap(matris);

            }

        });

        panel.add(scrollPane, BorderLayout.CENTER);

        panel.add(button, BorderLayout.SOUTH);

        add(panel);

    }

    private void boyutuAl() {

        String input = JOptionPane.showInputDialog("Matris boyutunu girin (10 ile 20000 arasında bir sayı): ");

        try {

            boyut = Integer.parseInt(input);

            if (boyut < 10 || boyut > 20000) {

                JOptionPane.showMessageDialog(null, "Geçersiz boyut. Lütfen 10 ile 20000 arasında bir sayı girin.");

                boyutuAl();

            }

        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(null, "Geçersiz giriş. Lütfen bir tam sayı girin.");

            boyutuAl();

        }

    }

    private void matrisiOlustur(int boyut) {

        matris = new int[boyut][boyut];

        Random rand = new Random();

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

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

                matris[i][j] = rand.nextInt(11) - 5;

            }

        }

    }

    private void islemleriYap(int[][] matris) {

        StringBuilder sb = new StringBuilder();

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

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

                int num = matris[i][j];

                try {

                    semaphore.acquire();

                    if (num < 0) {

                        negatifVar = true;

                    }

                    if (num < 0) {

                        negatifSayisi++;

                    } else {

                        pozitifSayisi++;

                    }

                    if (num > enBuyukSayi) {

                        enBuyukSayi = num;

                    }

                    if (num < enKucukSayi) {

                        enKucukSayi = num;

                    }

                } catch (InterruptedException e) {

                    e.printStackTrace();

                } finally {

                    semaphore.release();

                }

                sb.append(num).append("\t");

            }

            sb.append("\n");

        }

        sb.append("Negatif sayı var mı? ").append(negatifVar).append("\n");

        sb.append("Negatif sayıların sayısı: ").append(negatifSayisi).append("\n");

        sb.append("Pozitif sayıların sayısı: ").append(pozitifSayisi).append("\n");

        sb.append("En büyük sayı: ").append(enBuyukSayi).append("\n");

        sb.append("En küçük sayı: ").append(enKucukSayi).append("\n");

        textArea.setText("Matris:\n" + sb.toString());

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            public void run() {

                new B\_Frame().setVisible(true);

            }

        });

    }

}

* 1. **C Kodları**

package swingex;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Random;

//C şıkkı arayüz

public class C\_Frame extends JFrame {

    private int[][] matris;

    private int boyut;

    private boolean negatifVar = false;

    private int negatifSayisi = 0;

    private int pozitifSayisi = 0;

    private int enBuyukSayi = Integer.MIN\_VALUE;

    private int enKucukSayi = Integer.MAX\_VALUE;

    private JTextArea textArea;

    public C\_Frame() {

        setTitle("Matris İşlemleri");

        setSize(600, 500);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new BorderLayout());

        textArea = new JTextArea();

        textArea.setEditable(false);

        JScrollPane scrollPane = new JScrollPane(textArea);

        JButton button = new JButton("Matris İşlemlerini Başlat");

        button.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                boyutuAl();

                matrisiOlustur(boyut);

                islemleriYap(matris);

            }

        });

        panel.add(scrollPane, BorderLayout.CENTER);

        panel.add(button, BorderLayout.SOUTH);

        add(panel);

    }

    private void boyutuAl() {

        String input = JOptionPane.showInputDialog("Matris boyutunu girin (10 ile 20000 arasında bir sayı): ");

        try {

            boyut = Integer.parseInt(input);

            if (boyut < 10 || boyut > 20000) {

                JOptionPane.showMessageDialog(null, "Geçersiz boyut. Lütfen 10 ile 20000 arasında bir sayı girin.");

                boyutuAl();

            }

        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(null, "Geçersiz giriş. Lütfen bir tam sayı girin.");

            boyutuAl();

        }

    }

    private void matrisiOlustur(int boyut) {

        matris = new int[boyut][boyut];

        Random rand = new Random();

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

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

                matris[i][j] = rand.nextInt(11) - 5;

            }

        }

    }

    private void islemleriYap(int[][] matris) {

        StringBuilder sb = new StringBuilder();

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

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

                int num = matris[i][j];

                if (num < 0) {

                    negatifVar = true;

                }

                if (num < 0) {

                    negatifSayisi++;

                } else {

                    pozitifSayisi++;

                }

                if (num > enBuyukSayi) {

                    enBuyukSayi = num;

                }

                if (num < enKucukSayi) {

                    enKucukSayi = num;

                }

                sb.append(num).append("\t");

            }

            sb.append("\n");

        }

        sb.append("Negatif sayı var mı? ").append(negatifVar).append("\n");

        sb.append("Negatif sayıların sayısı: ").append(negatifSayisi).append("\n");

        sb.append("Pozitif sayıların sayısı: ").append(pozitifSayisi).append("\n");

        sb.append("En büyük sayı: ").append(enBuyukSayi).append("\n");

        sb.append("En küçük sayı: ").append(enKucukSayi).append("\n");

        textArea.setText("Matris:\n" + sb.toString());

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            public void run() {

                new C\_Frame().setVisible(true);

            }

        });

    }

}

1. **Race Condition Neden Önemli?**

Race condition, çoklu iş parçacığına sahip bir uygulamada birden fazla iş parçacığının paylaşılan bir kaynağa aynı anda erişmeye çalıştığı durumu ifade eder. Bu durum, senkronizasyon eksikliği veya uygun senkronizasyon tekniklerinin kullanılmaması durumunda ortaya çıkar.

Uygulamalarda race condition'ın önlenmesi genellikle önemlidir çünkü bu durum beklenmeyen sonuçlara ve hatalara yol açabilir. Örneğin, paylaşılan bir değişkeni aynı anda birden fazla iş parçacığı değiştirmeye çalışırsa, değişkenin son değeri beklenmeyebilir. Bu durumda programın tutarsız davranışlar sergilemesi mümkündür.

Race condition'ları önlemek için çeşitli senkronizasyon teknikleri kullanılabilir. Bu teknikler arasında kilitler (locks), senkronizasyon blokları, semaforlar ve monitörler gibi mekanizmalar bulunur. Bu teknikler, aynı anda yalnızca bir iş parçacığının paylaşılan kaynağa erişmesini sağlayarak race condition'ları engeller. Bu sayede programın tutarlılığı sağlanır ve beklenmeyen hatalar önlenmiş olur.

Lock, çoklu iş parçacığına sahip bir uygulamada senkronizasyonu sağlamak için kullanılan bir mekanizmadır. Bir lock, bir iş parçacığının bir kaynağa erişimini kilitleyerek diğer iş parçacıklarının o kaynağa erişimini engeller.

1. **Lock'lar Uygulamalarda Çeşitli Nedenlerle Gereklidir:**

1.Birden fazla iş parçacığının aynı anda paylaşılan bir kaynağa erişmeye çalışması durumunda, yarış koşulları (race conditions) ortaya çıkabilir. Bu durumda, beklenmeyen sonuçlar ve hatalar meydana gelebilir. Lock'lar bu yarış koşullarını önleyerek kaynaklara güvenli erişim sağlar.

2. Lock'lar, iş parçacıklarının belirli bir sırayla veya uygun bir şekilde çalışmasını sağlayarak senkronizasyonu sağlar. Örneğin, bir iş parçacığının belirli bir kaynağa erişimini bitirmeden diğer iş parçacıklarının beklemesini sağlayabilir.

3. Lock'lar, kritik bölge adı verilen kod bloklarını korumak için kullanılabilir. Kritik bölgeye giren bir iş parçacığı, ilgili lock'u alarak diğer iş parçacıklarının aynı kod bloğuna erişimini engeller. Bu şekilde, kritik bölge içindeki işlemler güvenli bir şekilde gerçekleştirilir.

4. İyi tasarlanmış lock kullanımı, deadlock'ları önlemeye yardımcı olabilir. Deadlock, iki veya daha fazla iş parçacığının birbirlerinin kilidini beklemesi ve sonuçta hiçbiri kilit alamadığı için sistemde bloke olması durumudur. Lock'ların düzgün kullanımı ve uygun kilit alma sırasının sağlanması, deadlock olasılığını azaltır.

Lock'lar genellikle Java gibi çoklu iş parçacıklı programlama dillerinde kullanılır ve paralel programlama ve eşzamanlılık konularında önemli bir rol oynarlar.

1. **A, B ve C Kodlarının Çalışma Zamanı Grafiği**
2. **A, B ve C Kodlarının Çalışma Zamanı Tablosu**

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **A\_süre(ns)** | **B\_süre(ns)** | **C\_süre(ns)** |
| 10 | 23074800 | 23898700 | 1005600 |
| 20 | 3645300 | 2366600 | 134200 |
| 30 | 4132300 | 2876300 | 482900 |
| 40 | 3765200 | 5258100 | 584600 |
| 50 | 4133500 | 4724100 | 678200 |
| 60 | 5667600 | 5831500 | 903500 |
| 70 | 7043500 | 8408700 | 1044900 |
| 80 | 6984800 | 6110500 | 775600 |
| 90 | 8623200 | 9335000 | 979700 |
| 100 | 9533200 | 9024600 | 1067500 |

1. **Grafikte A ve B Birbirine Neden Çok Yakın?**

A ve B'nin birbirine çok yakın olmasının nedenlerinden birkaçı:

İşlem süreleri, her çalıştırma için rastgele olabilir. Bu nedenle, A ve B'nin süreleri bazı durumlarda birbirine yakın olabilir.

İşlem yükü, iş parçacıklarının işlem sürelerini etkileyebilir. Eğer A ve B iş parçacıkları benzer yükler altında çalışıyorsa, sürelerinin birbirine yakın olması beklenir.

Java'nın JVM (Java Sanal Makinesi), iş parçacıklarının işlem sürelerini optimize etmek için çeşitli teknikler kullanabilir. Özellikle, belirli durumlarda iş parçacıkları arasında paralel işlem yapılabilir. Bu durumda, A ve B iş parçacıkları benzer şekilde optimize edilebilir ve sürelerinin birbirine yakın olması muhtemeldir.

İş parçacıklarının çalışma süreleri, sistem kaynaklarının kullanımına bağlı olabilir. Örneğin, CPU yükü, bellek kullanımı veya disk erişimi gibi faktörler, iş parçacıklarının sürelerini etkileyebilir ve A ve B iş parçacıklarının birbirine yakın olmasına neden olabilir.

1. **System.nanoTime() Fonksiyonu Kullanımı**

System.nanoTime(), Java'da bir zaman ölçüm mekanizması sağlayan bir metoddur. Bu metodun döndürdüğü değer, sistem saatinin başlangıcından bu yana geçen zamanı nanosaniye cinsinden temsil eder.

System.nanoTime(), genellikle performans ölçümlerinde veya zaman farklarını hassas bir şekilde belirlemek için kullanılır. Sistem saatine göre daha yüksek çözünürlüklü bir zaman ölçümü sağlar ve diğer zaman fonksiyonlarına kıyasla daha az maliyetlidir.