THREAD'LER

BÖLÜM - 1

Hazırlık-1: Bilgisayarınızda kaç adet işlemci bulunduğunu ekrana yazdırın. Örnek bir bilgisayarın verdiği sonuç:

Giris1.java

```
"Bilgisayarda toplam 4 tane işlemci bulunmaktadır."
```

Hazırlık-2: Verilen 2 sayı arasında kaç tane asal sayı olduğunu bulan ve çalışma süresini ekrana yazdıran bir program yazınız.

Giris2.java

```
Başlangıç: 1
Bitiş: 100
Toplam asal sayı adedi 25.
Toplam çalışma süresi 0.02 saniye.
```

Uygulama:

a) Verilen 2 sayı aralığının çok büyük olması durumu söz konusu. Örneğin 1.000.000 ile 5.000.000 arasında 270.015 adet asal sayı bulunmaktadır. Bilgisayarınız bu sonucu Thread kullanmadan ne kadar sürede hesaplar? Kendi bilgisayarınızın süresiyle karşılaştırın.

Bolum1.java (thread sayısı = 1 için)

```
"Toplam çalışma süresi: 821.885 saniye."
```

b) Aynı işi Thread'ler kullanarak yapın. Kaç tane Thread kullanılacağının girişi klavye ile yapılsın. Test edilecek sayı aralığını thread sayısına göre parçalara ayırarak aynı sonucu daha kısa sürede hesaplayınız. Sayıları eşit dağıttıktan sonra, kalan olursa onları da sonuncu thread'e veriniz.

Bolum1.java

```
Kaç tane thread kullanılsın  (en az 1 girilmeli) ?  3
1 ile 100 arasındaki asal sayıların adedi 3 tane Thread ile bulunuyor...

68 ve 100 arasında toplam 6 adet asal sayı var
2 ve 34 arasında toplam 11 adet asal sayı var
        yeni asal sayı adedi: 6
        yeni asal sayı adedi: 17
35 ve 67 arasında toplam 8 adet asal sayı var
        yeni asal sayı adedi: 25

Toplam asal sayı adedi 25.

                Toplam çalışma süresi:  0.022 saniye.
```

c) Aşağıda 1.000.000 ile 5.000.000 aralığı için çalışma süreleri verilmiştir. Kendi sonucunuzla karşılaştırın.

> A: "2 thread: Toplam çalışma süresi: 561.951 saniye."
> B: "4 thread: Toplam çalışma süresi: 416.935 saniye."
> C: "8 thread: Toplam çalışma süresi: 383.693 saniye."

Sonuçlarınızı aşağıdaki gibi tabloya giriniz ve karşılaştırınız:

	Lab PC	**Öğrenci PC**
Başlangıç	1.000.000	
Bitiş	5.000.000	
İşlemci Sayısı	4	
Asal sayı adedi	270.015	
1 Thread	821.885 sn	
2 Thread	561.951 sn	
4 Thread	416.935 sn	
8 Thread	383.693 sn	

BÖLÜM - 2

Bölüm-1'de verilen uygulamanın çözümlerini inceleyip denemeler yapınız. toplamaEkle() metotunun ***synchronized*** anahtar kelimesiyle başladığına dikkat ediniz.

synchronized anahtar kelimesini silerek geniş bir sayı aralığı için (Örn. 1 ile 1 milyon arasında) programı yeniden çalıştırın. Önceki sonuçtan farklı bir sonuç veriyor mu?

Sonuçlar doğru çıkmaya devam etmiş olabilir. Peki bu programa her zaman güvenebilir miyiz? Bu düşünceyle synchronized anahtar kelimesinin silindiği versiyon için programda bazı değişiklikler yapın ve Thread'lerin ortak kullandığı ***toplam*** değerini hatalı veren bir senaryo hazırlayın.

Örneğin, sayı asal olsun olmasın her seferinde toplam değerinin artmasını sağlayın. Bu şekilde farklı aralıklardaki sayılarla denemeler yapın. Sayı aralığı ne kadar büyüdüğünde program hatalı sonuçlar vermeye başladı? Aşağıda verilen hesaplamaları kendi bilgisayarınız için elde edip sonuçları karşılaştırınız.

Başlangıç	**Bitiş**	**Thread Sayısı**	**Sonuç –** synchronized **var**	**Sonuç –** synchronized **yok**
1	100	2	99	99
1000	7000	1	6000	6000
10	40000	3	39990	18125
1000000	2000000	6	1000000	660440
…	…	…	…	…

KAYNAKLAR

1- Schildt, Herbert. "Java: the complete reference." (2018).
2- Eck, David J. Introduction to programming using Java. David J. Eck, 2006.
3- http://math.hws.edu/javanotes/c12/index.html