**SUNUCU İSTEK YOĞUNLUĞUNUN**

**MULTITHREAD İLE KONTROLÜ**

Onur GÜLER

*Bilgisayar Mühendisliği Bölümü*

*Kocaeli Üniversitesi*

*170202031*

[45onurguler45@gmail.com](mailto:45onurguler45@gmail.com)

**1.Özet**

Bu projenin amacı; herhangi bir sunucuya gelen isteklerin ,ana sunucuyu yormasını engellemek. Bunu da ana sunucuya gelen istekleri alt sunucular ile ana sunucudan çekip aynı anda en az 3 sunucuyla birlikte daha fazla isteğe cevap vermeye çalışarak yapmak.

**2.Giriş**

Bir sunucuya rastgele sayıda istekler gelmektedir. Bu ilk sunucu ana sunucu. En az iki tane alt sunucu olmalı ve bu alt sunucular ana sunucudan rastgele istekleri çekerek ana sunucuyu rahatlatmak ve daha fazla isteğe cevap vermeyi amaçlanmaktadır. Bu sunucular bir yandan istekleri alırken diğer yandan ise aldığı isteklere geri dönüş yapmak. Bir tane thread ile ise bu alt sunucuları kontrol edilmeli. Alt sunuculardan bir tanesinin kapasitesi yüzde yetmişi aşarsa bir tane daha yeni bir alt sunucu oluşturup yüzde yetmişi aşan alt sunucudaki kapasiteyi yarıya indirip yeni açılan sunucuya da yarısını geçirerek sunucuları rahatlatmak amaçlanmaktadır. Eğer alt sunucu sayısı üçten fazlaysa bu alt sunucuların kapasitelerinin sıfır olup olmadığını kontrol edip eğer sıfır olan bir alt sunucu varsa bu alt sunucuyu silmek amaçlanmaktadır.

Bu proje sayesinde projeyi yapacak kişinin Thread ve Multithread konusunda daha önce edinmiş olduğu teorik bilgileri pratiğe dönüştürmesi amaçlanmaktadır.

**3. Temel Bilgiler**

Proje gelişiminde;

Tümleşik geliştirme ortamı olarak “ NetBeans IDE ” kullanılmıştır.

**4. Sorunlar ve Yapılan Araştırmalar**

Projeyi Java programlama dilinde yazdığım için ilk önce threadlerin nasıl oluşturulacağını araştırdım. Thread oluşturmak için bir tane class oluşturup bu classa Thread kütüphanesini extends ettim.

**Threadler arasında veri aktarımı;**

Threadleri oluşturduktan sonra 2 farklı thread ile aynı veriye ulaşmam gerekti, int ve string gibi değişkenler üzerinde deneme yaptım ama bu threade gönderilen değişkenler sadece main thread de tanımlı olan değerleri alabiliyordu ve mainin içinde değişme olmuyordu. Bu problem için başta veri tabanında verileri tutmayı düşündüm daha sonra bir nesne oluşturup bu nesnelerin içinde verileri tutabileceğim aklıma geldi ama bundan tam emin olmadığım için biraz araştırma yaptım ve kaynakçada belirttiğim youtube adresinden nesneleri adresleri ile birlikte gönderdiğini öğrendim bu sayede mainin içinde oluşturduğum bir nesneyi başka threadin içinde değişiklik yaparsam, başka bir thread ile bu nesnenin içindeki veriye eriştiğimde yaptığım değişiklikler taşınabiliyordu.

**Threadler arasında senkronizasyon;**

İki ayrı thread bir nesnedeki veriye eriştiğinde her çalıştrımada çok farklı sonuçlar veriyordu biri işlem yapıp yazıyor, diğer thread de işlem yapıp yazıyor ama sadece son işlem kaydoluyordu. Javada bu durumu nasıl çözebileceğimi biraz araştırdım ve bu sorunun çözümünü de aynı youtube kanalından buldum. Bir thread bir veriye eriştiği anda diğer bu veriye erişmek isteyen threadleri beklemeye alan ‘synchronized’ bloğunu öğrendim. Bu blok içinde veriye erişip işlem yaptığımda daha sağlıklı çözümler gördüm.

**Thread silerken threadleri kapatmak;**

Bir sunucunun kapasitesi sıfıra düştüyse ve en az 3 tane alt sunucu varsa; kapasitesi sıfır olan alt sunucuyu silerken diğer sunucuları durdurup sunucuyu silip tekrar sunucuları başlatıyorum ama bir sunucu silindikten sonra isteklere dönüş yapan threadleri doğru şekilde kapatamadığımı gördüm ve bir sunucu için bir anda 2 3 tane isteklere dönüş yapan threadin çalıştığını gördüm. En çok takıldığım sorunlardan birisi buydu. Bu sorunun threadleri durdurduğum zaman bazı threadlerin o anda beklemede olduğunu fark ettim. Bunu için thredleri durdurduktan sonra tekrar işlemleri başlatmak için alt sunucuların isteklere dönüş yapma süresini çekip on milisaniye daha beklemelerini sağladıktan sonra doğru bir şekilde sunucu işlemlerinin sonlanıp başladığını gördüm.

# Java’da Değişkenler ve Veri Türleri

## Veri Türleri

Java’da değişkenler tanımlanırken herhangi bir veri türü belirtilmez. Hangi veri türü kullanıldığını değişkenin değerinden anlayabiliriz.

**ArrayList:**

Tıpkı Arra*y* gibi index yapısını kullanmaktadır. Ekleme, silme, sıralama, arama, ters çevirme gibi işlemler yapılabilir. ArrayList*leri* Array’den daha önde tutan dinamik bir yapısının olmasıdır. Array’lerde tanımlanma aşamasında dizinin boyutu belirtilmelidir, daha sonra program içinde dizinin boyutunu arttırmak zahmetli bir iştir. (Sürekli düzenli veya düzensiz bir şekilde arttırmak gerektirdiğinizi düşünürsek)**ArrayList** dinamik bir yapıya sahip olduğu için tanımlanırken boyutunun belirtilmesine gerek yoktur, çalışma anında dahi liste içine eleman eklenebilir, silinebilir, istenilen indise eleman eklenerek öteleme yapılabilir. Array’lerde bu mümkün değildir.

**ArrayList** listesinin elemanlarına erişebilmek için tıpkı Arraylerde olduğu gibi index numarası kullanılır. Ayrıca Arrayler sadece tanımlandıkları türde veri saklayabilirken, ArrayListler farklı türdeki verileri bir liste içinde barındırabilir.Dinamik olarak genişleyebilen tek boyutlu dizilere ArrayList denir.ArrayList kullanmak için kodumuzun en yukarısına

**import java.util.\*;**

eklememiz gerekmektedir.

**ArrayList dizi\_adi=new ArrayList();**

şeklinde kullanılır.

**String:** Metinsel veri türüdür.

|  |  |
| --- | --- |
| 1  2  3 | String deneme00 = "dedendeneme";      String deneme01 = "1234";      String deneme02 = '9Cdeneme'; |

Bir değişkeni string olarak tanımlayabilmek için = operatöründen sonra tırnak işareti içerisinde değişkenin değerini yazmalıyız. Yukarıdaki örnekte deneme00 adlı bir değişken tanımladık ve değerini zinzinzibidi şeklinde string olarak belirledik. deneme01 değişkeni de string'dir. Çünkü değeri sayısal olmasına rağmen tırnak işaretleri içinde yazdık. Tek tırnak ('') işareti içerisine yazılan değerler de string olarak algılanır.

**int ve float:** Sayısal veri türüdür.

|  |  |
| --- | --- |
| 1  2 | int deneme03 = 12;  // Tamsayı      floatdeneme04 = 2.3; // Ondalıklı sayı |

**Boolean:** Mantıksal veri türüdür.

|  |  |
| --- | --- |
| 1  2 | int deneme05 = true;      int deneme06 = false; |
|  |  |

Değişkenin değeri ya true (doğru) ya da false (yanlış) olabilir.

**Thread Kütüphanesi:**

Thread kütüphanesi javanın içinde bulunan bir kütüphanedir. Bu kütüphane sayesinde threadler oluşturup programımızın işlemci üzerinde birden fazla core üzerinde çalıştırılabilmesini ve böylelikle bir tarafı hata verdiğinde yinede programımız çalış durumda olabiliyor. Thread kütüphanesini bir classa extends edip run() fonksiyonun içine yazdığımzı kodları start() ile eş zamanlı çalıştırabilmekteyiz.

**run() :** Bu fonksiyonun için eşzamanlı olarak çalışmasını istediğimiz kodları yazıyoruz ve sanki yeni bir program çalıştırtmış oluyoruz.   
**start() :** Mainin içinde oluşturduğumuz bir thread nesnesini bu fonksiyon ile run() metodunu çalıştırıyoruz ve yeni bir program gibi diğerleriyle birlikte çalışmaya başlar..  
**Thread nesnesi oluşturma :**

Thrclas newthr=new Thrclas();

newthr.start();

**5. Kazanımlar**

**Thread**

Threadlerin nasıl çalıştığını, her bir threadin bir program gibi çalıştığını ve işlemci üzerinde paylaştırılmasını teorik olarak öğrenmiştim, bu proje sayesinde pratiğe döküp uygulama üstünde daha iyi anladım.

**Thread Kütüphanesi ve İşlemleri**

Java’da thread kütüphanesi ile birlikte yeni bir thread oluşturup bu threadleri arasında veri paylaşımının nesneler aracılığı ile yapılabildiğini, threadler arasında senkronizasyonun ‘synchronized’ bloğu ile yapıldığını öğrendim.

**6.Tasarım**

**a)Yalancı Kod(Pseudo)**

1. Main thread de önce ana sunucuyu oluştur. Daha sonra 2 tane alt sunuuc oluştur ve bu sunuculara daha iyi kontrol edebilmek için bir tane sunucu ArrayList’inde tut. Sıfırıncı indiste ana sunucuyu tut.

2. Daha sonra ana sunucu için önce Istekal thread nesnesini oluştur ve bu nesneyi start() metodu ile başlat.

2.1 Istekal classı run() fonksiyonunu çalıştır ve bu fonksiyonu her zaman döngü şeklinde çalıştır. İşlem sayısı limiti geçmediği sürece belirtilen sayıda işlem ile bir arasında rastgele işlem oluşturup bu işlem sayısını arttır. Belirtilen işlem kabul süresi kadar Thread.sleep() fonksiyonu ile threadi uyut.

3. Istekal nesnesi oluşturduktan sonra ana suncu için Istekdon thread nesnesini oluştur ve bu nesneyi start() metodu ile başlat.

3.1 Istekdon classı run() fonksiyonunu çalıştır ve bu fonksiyon sunucu nesnesinde bulunan ‘tdur’ boolean ifadesi ‘true’ olasıya kadar çalıştır. True olduğu zaman bu threadi sonlandır. İşlem sayısından belirtilen işlem dönüş sayısı ve bir arasında rastgele bir sayı üretip bu sayıyı çıkartıp işlem sayısına geri yaz eksiye düşerse 0 yap. Belirtilen işlem dönüş süresi kadar Thread.sleep() fonksiyonu ile threadi uyut.

4. Ana sunucu threadleri başlatıldıktan sonra Sunbas threadini başlat.

4.1 Alt sunucular için verileri alt sunucu işlem kabul saysı ile 1 arasında rastgele sayı üretip ana sunucu işlem sayısı 0 değilse, ana sunucudan alt sunucuya aktar.

4.2 3.1’deki işlem alt sunucular içinde çalıştır.

5.Ana sunucu ve alt sunucu threadleri çalıştırıldıktan sonra Sunkont threadini çalıştır. Bu thread de sunucuların kapasitelerini kontrol et.

5.1 Eğer en az 3 tane alt sunucu varsa kapasitelerini kontrol et , kapasitesi 0 olan sunucu varsa alt sunucu işlemlerini durdur ve bu sunucuyu sil, sonra thread işlem dönme süresinden on milisaniye kadar bekle ve 4. adımı çalıştır.

5.2 Eğer alt sunucu kapasitesi yüzde yetmişi geçerse alt sunucu işlemlerini durdur ve yeni bir alt sunucu oluşturup kapasiteleri bölüştür ve 4. adımı çalıştır.

6. En son Kapasite threadini çalıştır ve bu sunucların kapasitelerini kontrol edip Arayüz classı ile ekranda göster.

**b)Fonksiyonlar ve Nesneler**

**run() ve start()**

Oluşturduğum thread classlarındaki threadin yapması gereken işlemleri yaptırdığım yer. Thread kütüphanesindeki run fonksiyonunu override ediyorum. Start metodu ile oluşturduğumuz bu thread clasında oluşturulan bir thread nesnesinin run fonksiyonunu diğerleriyle eşzamanlı çalışabilmesini sağlıyoruz.

**Sunucu Nesnesi;**

Değişkenler:

islimit=10000; iskabul=200; issure=200; islem=0;

iskasu=200; ansu=null; kapas=0; tdur=false; Ve bu verilerin getter() ve setter() metodları.

**c) Yazılım Mimarisi**

Proje çalıştırıldığında ilk başta ana sunucuyu ve 2 tane alt sunucuyu oluştur ve rahat kontrol için sunucu ArrayList’inde tut. Daha sonra ana sunucu ve alt sunucular için istek alma ve istek döndürme threadlerini çalıştır. Bütün sunucular için threadler çalışırken bu alt sunucuları kontrol edecek sunucu kontrol threadini çalıştır. Bu threadde alt sunucuları kontrol et ve kapasitesi yüzde yetmişi aşan bir alt sunucu varsa alt sunucuları durdur ver bu alt sunucudaki işlem sayısını yarıya indir daha sonra yeni bir alt sunucu oluşturup işlemlerin yarısını bu yeni alt sunucuya geçir, eğer en az üç tane alt sunucu varsa sunucu kapasitelerini kontrol et ve kapasitesi sıfır olan bir alt sunucu varsa önce alt sunucuları durdur ve daha sonra bu alt sunucuyu sil daha sonra alt sunucuları tekrar başlat. Bu sunucuların kapasitelerini rahat kontrol etmek için ekranda bir arayüzde bu sunucuların kapasite doluluk oranlarını proses bar ile göster.

# 7. Kaynakça

Thread oluşturma ve metodları

🡪[https://ufukuzun.wordpress.com/2015/02/14/javada- multithreading-bolum-1-merhaba-thread/](https://ufukuzun.wordpress.com/2015/02/14/javada-%20%20%20multithreading-bolum-1-merhaba-thread/)

🡪<https://medium.com/gokhanyavas/javada-multithreading-bbc6a9181772>

Thread senkronizasyon ve veri paylaşımı

<https://www.youtube.com/watch?v=XTQs9OvFCAQ&lst=PLHfYetw_BGF-Gm_MsqKApw5nHPuHsytr3&index=77&t=1513s>

OpenCV fonksiyonları

<https://opencv-java-tutorials.readthedocs.io/en/latest/>

<https://docs.opencv.org/master/javadoc/index.html>

**8. Deneysel Sonuçlar**

