**SUNUCU İSTEK YOĞUNLUĞUNUN MULTITHREAD İLE KONTROLÜ**

*170202122-Fatih KARAMAN-170202124- Enver ÇALÇOBAN*

Bilgisayar Mühendisliği Bölümü Kocaeli Üniversitesi

[f4t1hkrmn@gmail.com](mailto:f4t1hkrmn@gmail.com) [envercalcoban](mailto:oguzturker8@gmail.com)@gmail.com

# Özet ( Genel Yapı )

Uygulama, projede bir sunucuya gelen isteklerdeki aşırı yoğunluğu, multithread kullanarak alt sunucularla birlikte azaltmalktır. Projede sunum ve kullanım kolaylığını sağlamak için java dili swing kütüphanesi kulanılarak arayüz tasarlanmıştır. Bu arayüz ile sisteme belirtilen sayıda istek girilmekte ve sistem de sürekli aktif çalışmakta olan main sunucu girilen istekten 1-100 arası random üretilen sayı adedi kadar gelen istekten sayı düşmekte ve aktif alt sunucularda main sunucunun kuyruğuna eklenen sayılardan(isteklerden) 1-50 arası random üretilen sayı adedi kadar istek düşmektedir. Tabi bu işlemler gerçekleşiyorken senkronize tasarlanmış cevap threadleri random üretilen sayı adedi kadar main sunucu isteginden düşmektedir. Main sunucu kapasitesi 10.000 olarak alt sunucu kapasitesi 5000 olarak sınırlandırılmıştır. Ve main kuyruktan istek alan alt sunucu %70 kapasitesine ulaştıgında yeni sunucu oluşturulup mevcut üzerindeki yükün yarısını yeni oluşturulan sunucuya aktarmaktadır. Oluşan tüm sunucu senkronize çalışmakta olup yeni oluşturulan sunucular üzerindeki yük sıfırlandıgında sunucular sistemden çıkmatadır.

# Giriş

Kullanıcı uygulamayı ilk çalıştırdığında giriş ekranı ile karşılaşır

Ana Sunucu:10000 istek kapasitesine sahiptir. 500 ms zaman aralıklarıyla [1-100] arasında rastgele sayıda istek kabul etmektedir. İstek olduğu sürece 200 ms zaman aralıklarıyla [1-50] arasında rastgele sayıda isteğe geri dönüş yapmaktadır.

Main sunucu 10000 istek kapasite kontrolü vardır

Alt Sunucu:5000 istek kapasitesine sahiptir. 500 ms zaman aralıklarıyla 1-50 arasında rastgele sayıda ana sunucudan istek almaktadır. İstek olduğu sürece 300 ms zaman aralıklarıyla 1-50 arasında rastgele sayıda isteğe geri dönüş yapmaktadır.

Alt sunucu oluşturucu:Mevcut olan alt sunucuları kontrol eder. Eğer herhangi bir alt sunucunun kapasitesi %70 ve üzerinde ise yeni bir alt sunucu oluşturur ve kapasitenin **yarısını** yeni oluşturduğu alt sunucuya gönderir. Eğer herhangi bir alt sunucunun kapasitesi %0 a ulaşır ise mevcut olan alt sunucu sistemden kaldırılır. En az iki alt sunucu sürekli çalışır durumda kalıyor

Sunucu takip :Sistemde mevcut olan tüm sunucuların bilgilerini (Sunucu/Thread sayısı, ve kapasiteleri(%)) canlı olarak göstermektedir.

Uygulama çalıştırıldığında 1 adet ana sunucu ve 2 adet alt sunucu bulunmaktadır.

Ana sunucuya 500 ms aralıklarla istekler gelmektedir.

Alt sunucular 500 ms aralıkla ana sunucudan istekleri alıp kendileri geri dönüş yapmaktadır.

Alt sunucu oluşturucu, alt sunucuları kontrol ederek %70 ve üzerinde kapasitesi olması durumunda yeni bir alt sunucu oluşturarak kapasitesi %70 in üzerine çıkan sunucudaki isteklerin yarısını yeni sunucuya aktarmaktadır. Yeni oluşturulan alt sunucunun kapasitesi %0 a düşerse alt sunucu ortadan kalkmaktadır.

Ayrıca Sunucu Takibinde canlı olarak sunucuların kapasite bilgileri ekrana yazdırılmaktadır

# Temel Bilgiler

Proje gelişiminde;

Java nesneye yönelik programlama dili kullanılmıştır.

# 2.0 !!!!!!!!!!!!!!!!!!!!!!!!!!!!

# 2.1 Problem Tanımı

1). Aynı anda birden fazla çalışan Threadlerin senkronize çalışması.

2). Aynı anda birden fazla Threadin aynı değişken üzerinde çalışması.

3). Kapasite %70 geçmesi durumunda yeni oluşan Thread nesnesı.

# Yapılan Araştırmalar

1). Projemizde ana sunucu 50 ms aralıklarla istek kabul ederken 150 ms aralıklar kuyrukta bekleyen isteklere cevap vermesi gerekmektedir. İstek kabul ve istek cevapları(threadler) aynı anda çalışmaya başlıyacakları zaman zaman istek değişkeni üzerinde çakışmaya sebeb olacaktır. Bu problemin önüne geçebilmek için synchronized (lock), wait(),notify() özellikleri kullanılmıştır.Proje aşağıdaki başlıklar altında geliştirilmiştir.

2). Projemizde ana sunucu kuyrugunda bekleyen isteklerin mevcut olması durumunda sabit alt sunucular ve bunların %70 performansın üzerine çıkması durumunda geçici olarak oluşan geçici alt sunucular main sunucu kuyrugundaki istekleri kendi kuyruklarına eklemektedir. Tabi bu süreçte main sunucu kuyrugundaki istek sayısını ifade eden değişkene birçok thread aynı anda erişebilmekte ve aynı anda okuma/ yazma işlemi yapabilme duruma sebeb olmaktadır, bu durumun sonuçunda isteklere verilen cevaplar ile istek kuyrugunda mevcut istek sayı degerleri tutmama duruma sebeb olabilmektedir. Bu problemin önüne geçmek için lock = new ReentrantLock(); anahtar üretip bu anahtar ile

**lock.lock();**

**mainThreadIstek = mainThreadIstek + value4;**

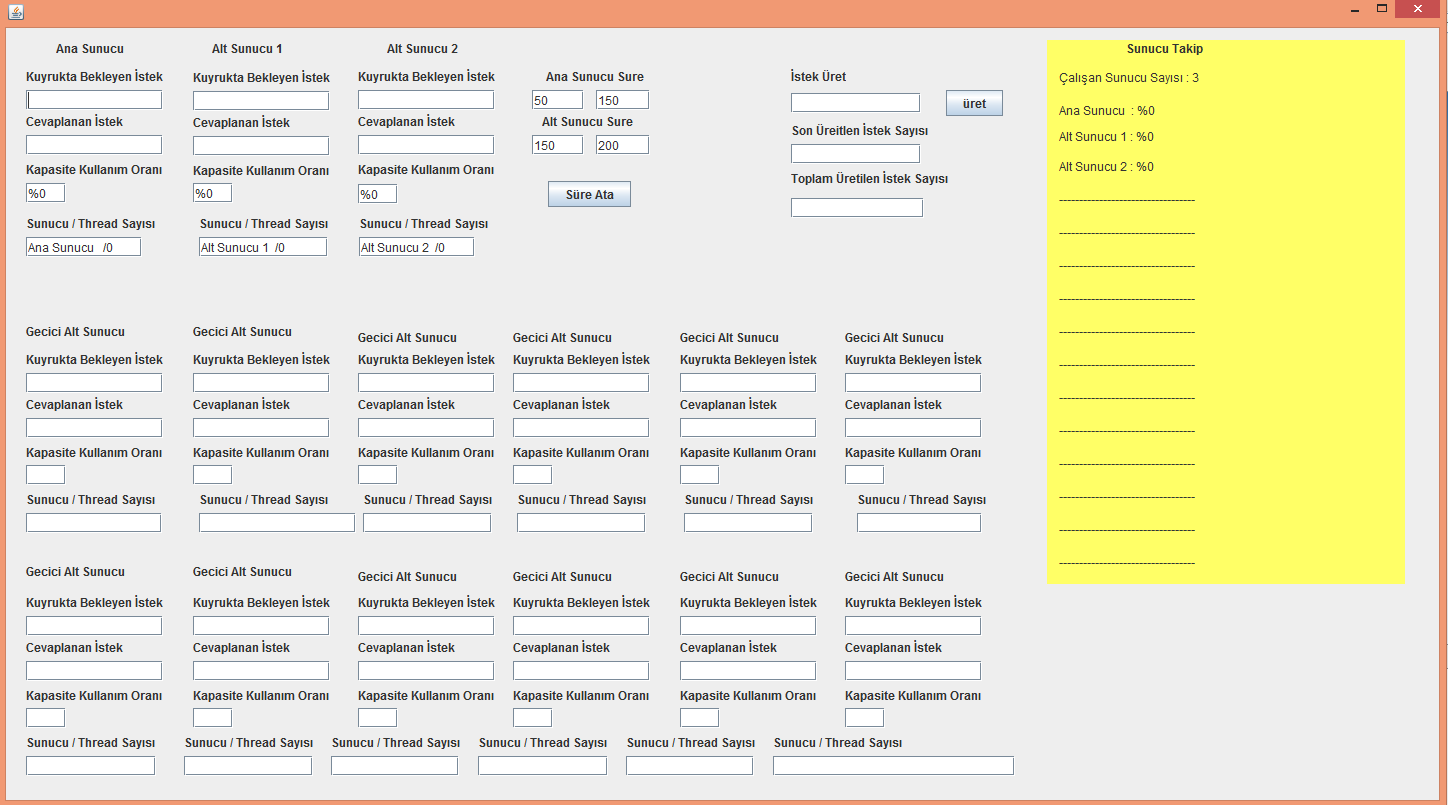
**lock.unlock();**

şeklinde mainTreadIstek değişkenini kapsülleyerek mainThread üzerinde threadlerin sıralı işlem yapması saglanarak problemin önüne geçilmiştir.

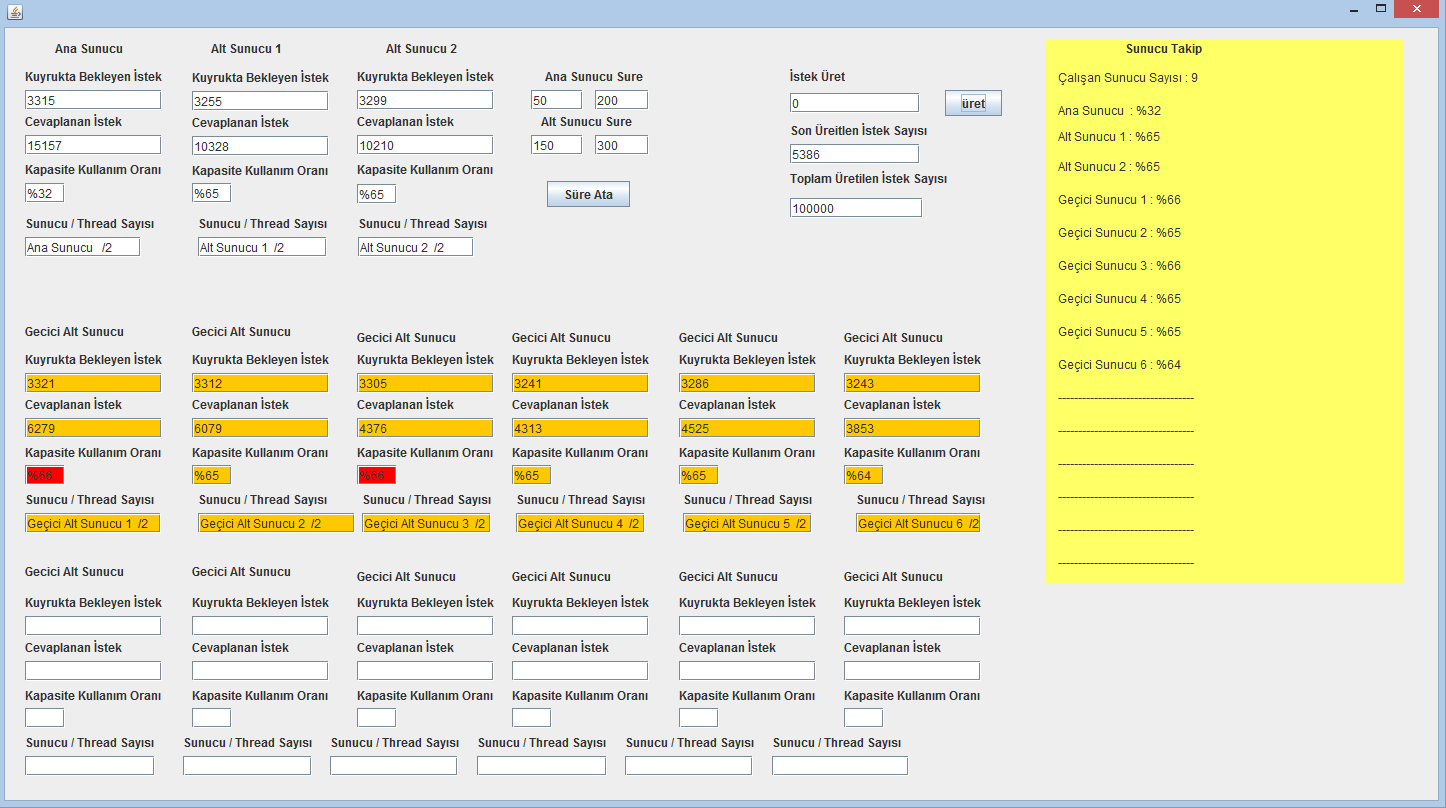
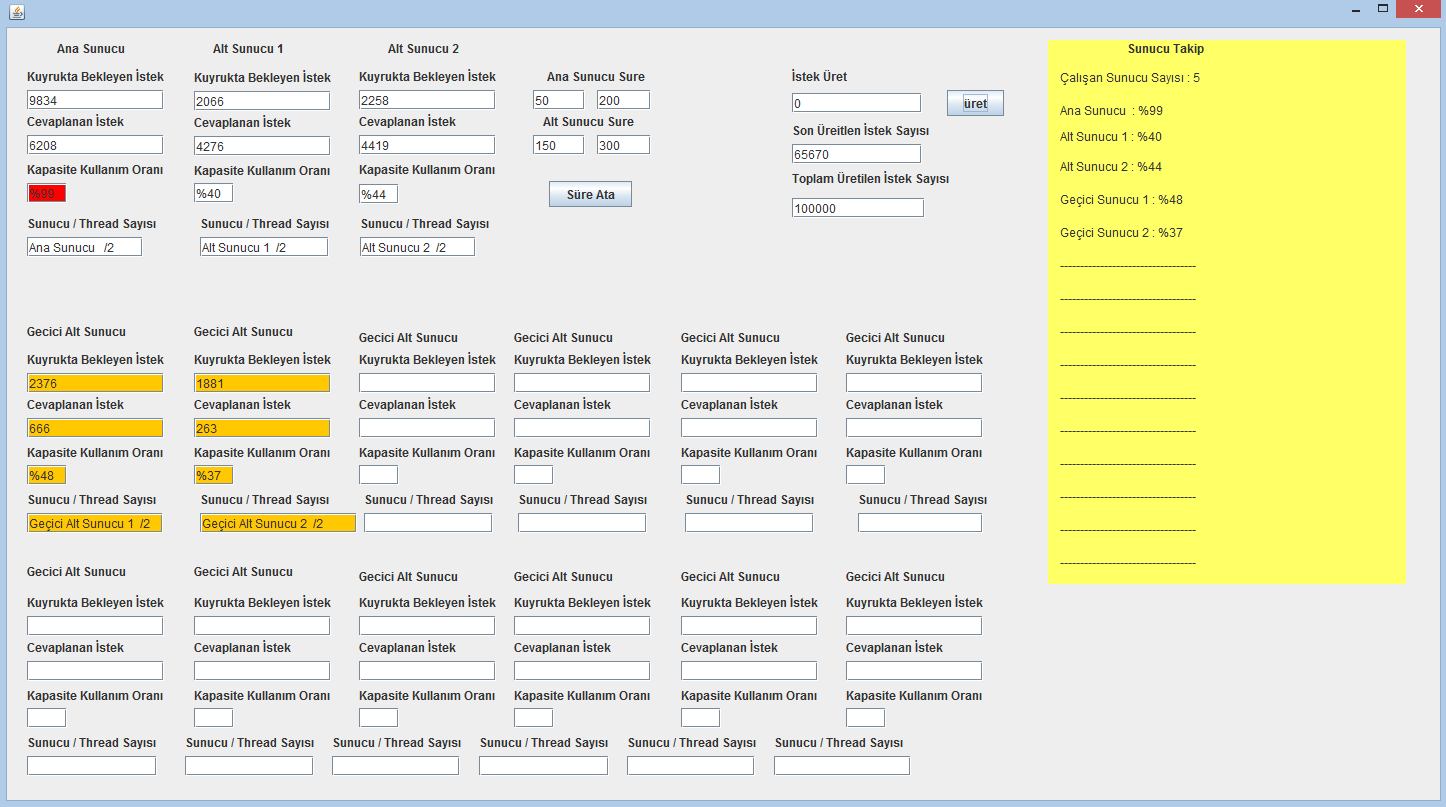
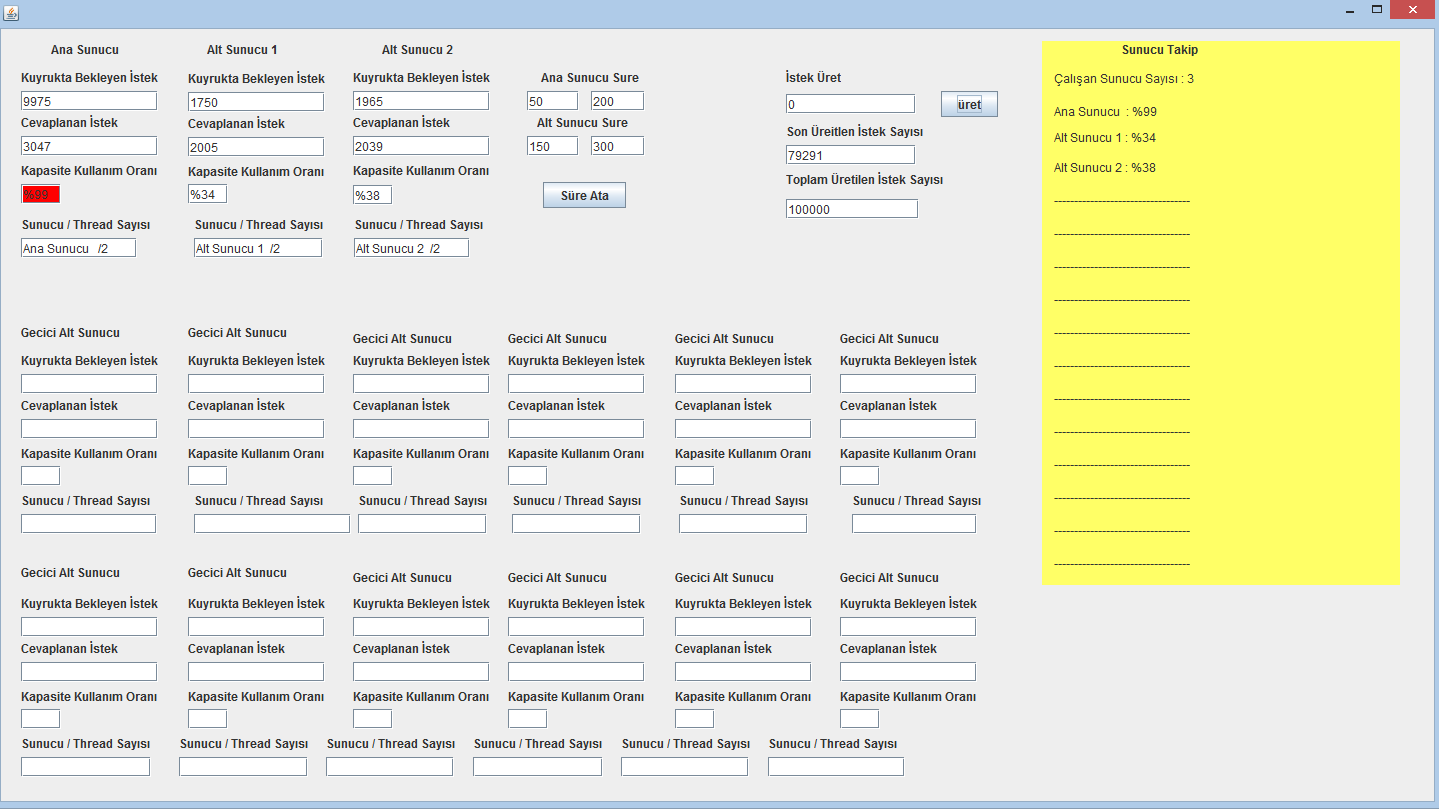
3). Mevcut alt sunucular 5000 kapasite limitinin %70 ‘ine geldiğinde yeni thread oluşturup oluşan bu thread’e mevcut thread üzerindeki yükün yarısının aktarılması ve yeni oluşan bu thread üzerindeki yük miktarı bittiginde threadin sonlandırılması işlemi yapılırken mevcut alt sunucu üzerindeki yükün yarısı yeni sunucu üzerine aktarılması, Yeni threadler için bir **recursive** sınıf oluşturmamızı gerktirdi. Bu geçici alt sunucular için tasarlanmış recursive sınıf sayesinde herbi oluşan alt sunucuda üzerindeki yükün yarısını yeni oluşacak alt sunucuya aktarabilme özelliğine sahip olmuş ve yeni oluşan geçici alt sunucularında kapasitesi %70 ‘üzerine çıktıgında tekrar yeni bir geçici alt sunucu oluşturmamıza çözüm oldu.

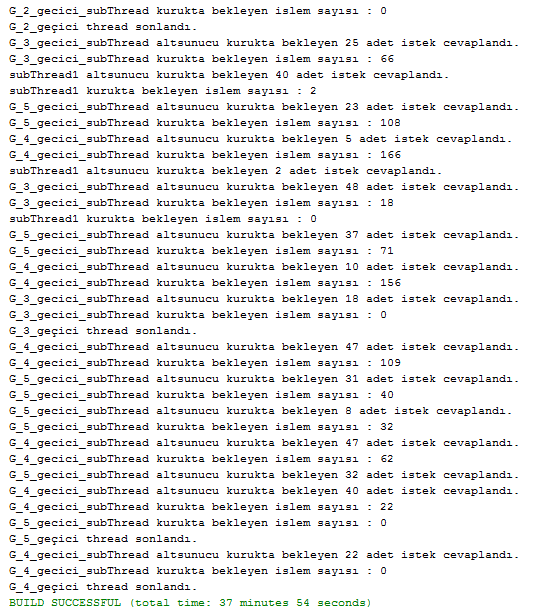
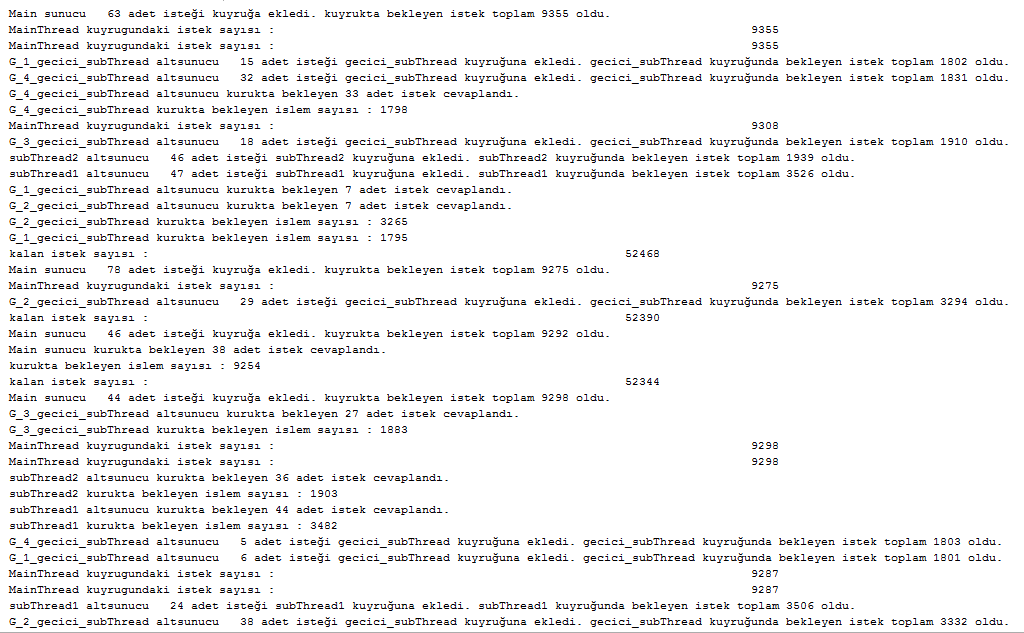
# Tasarım

Programın bazı ekran görüntüleri;







* 1. **Kaynaklar**

1. https://aykutt.com/java-1-rastgele-sayi-uretme-fonksiyonu-random/
2. https://www.w3schools.com/java/java\_conditions.asp
3. https://ufukuzun.wordpress.com/2015/05/04/javada-multithreading-bolum-8-beklet-ve-bildir-wait-and-notify/
4. https://nuriuzunoglu.com/2016/03/12/veri-yapilari-kuyruk-queue/
5. https://www.geeksforgeeks.org/array-implementation-of-queue-simple/
6. https://www.geeksforgeeks.org/queue-remove-method-in-java/
7. https://medium.com/gokhanyavas/java-collections-5-queues-f5c3abc28dcf
8. https://medium.com/gokhanyavas/javada-multithreading-bbc6a9181772
9. https://emrahmete.wordpress.com/2011/10/06/javada-thread-yapisi-ve-kullanimi-hakkinda-ipuclari/
10. https://kod5.org/javada-static-ve-final-anahtar-sozcukleri/
11. https://ufukuzun.wordpress.com/2015/05/05/javada-multithreading-bolum-9-yeniden-girilir-kilitler-re-entrant-locks/
12. https://stackoverflow.com/questions/45256409/objects-of-multiple-class-in-java
13. https://ufukuzun.wordpress.com/2015/05/07/javada-multithreading-bolum-13-threadlerin-yarida-kesilmesi-interrupting/
14. https://gelecegiyazanlar.turkcell.com.tr/soru/thread-durdurma
15. https://yazdoldur.com/programlama/java/java-thread-kavrami-multithreading-ve-olusturma-yontemleri/
16. https://dzone.com/articles/java-thread-tutorial-creating-threads-and-multithr
17. https://stackoverflow.com/questions/42350400/multi-threading-in-recursive-methods
18. https://stackoverflow.com/questions/34213400/jtextfield-how-to-set-background-color
19. https://www.w3schools.com/java/java\_switch.asp
20. https://www.mehmetkirazli.com/java-random-sinifi/
21. https://ufukuzun.wordpress.com/2015/02/14/javada-multithreading-bolum-1-merhaba-thread/ (Access date: 14 ŞUBAT 2015)
22. https://www.youtube.com/watch?v=Xj1uYKa8rIw (Access date: 24 Kas 2016)
23. https://www.youtube.com/watch?v=vCDrGJWqR8w (19 Şub 2015)
24. https://medium.com/gokhanyavas/javada-multithreading-bbc6a9181772(Nisan 20, 2017)
25. https://www.udemy.com/course/sifirdan-ileri-seviyeye-komple-java-gelistirici-kursu/learn/lecture/8523436#overview
26. https://www.udemy.com/course/sifirdan-ileri-seviyeye-komple-java-gelistirici-kursu/learn/lecture/8523460#overview
27. <https://www.udemy.com/course/sifirdan-ileri-seviyeye-komple-java-gelistirici-kursu/learn/lecture/8523472#overview>
    1. **Kazanımlar:** Geliştirdiğimiz bu proje ile thread yapısını threadlerin bir arada senkronize çalışmasını ve thread sınıfları oluşturabilme yeteneği kazandık. Threadler ile farklı iş parçacıkları oluşturarak parelel bir şekilde işlemler yapabilmekteyiz.