KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAZILIM LAB. I 2021-2022 GÜZ DÖNEMİ
PROJE 2

180201018-Ayşenur Akpınar
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
aysenurakpinar98@gmail.com

190201100-Rukiye Canlı
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
rukiyecanli435@gmail.com

1.Problem Tanımı

Bu projede bizden samurai sudoku içindeki her bir sudoku için bir başlangıç noktası seçerek 5 thread ile çözüme ulaşmanız beklenmektedir. Ayrıca projeyi her bir sudoku için iki ayrı başlangıç noktası seçerek toplam 10 thread ile de gerçekleştirmemiz gerekmektedir. Threadler eşzamanlı olarak çözüme başlamalı, senkronizasyon problemi göz önünde bulundurulmalıdır. İşlemleri gerçekleştirdikten sonra bu 2 ister arasında zaman ve bulduğu çözüm karesi arasında ilişki grafiği çizdirilecektir. Y eksenini bulduğu kare sayısı X ekseni geçen süre olarak düşünülmelidir.2 ister aynı grafik içinde farklı renklerle çizdirilecektir. Ayrıca her bir modelin hücrelerindeki değişimleri veri tabanında veya text dosyasında tutmamız gerekmektedir.

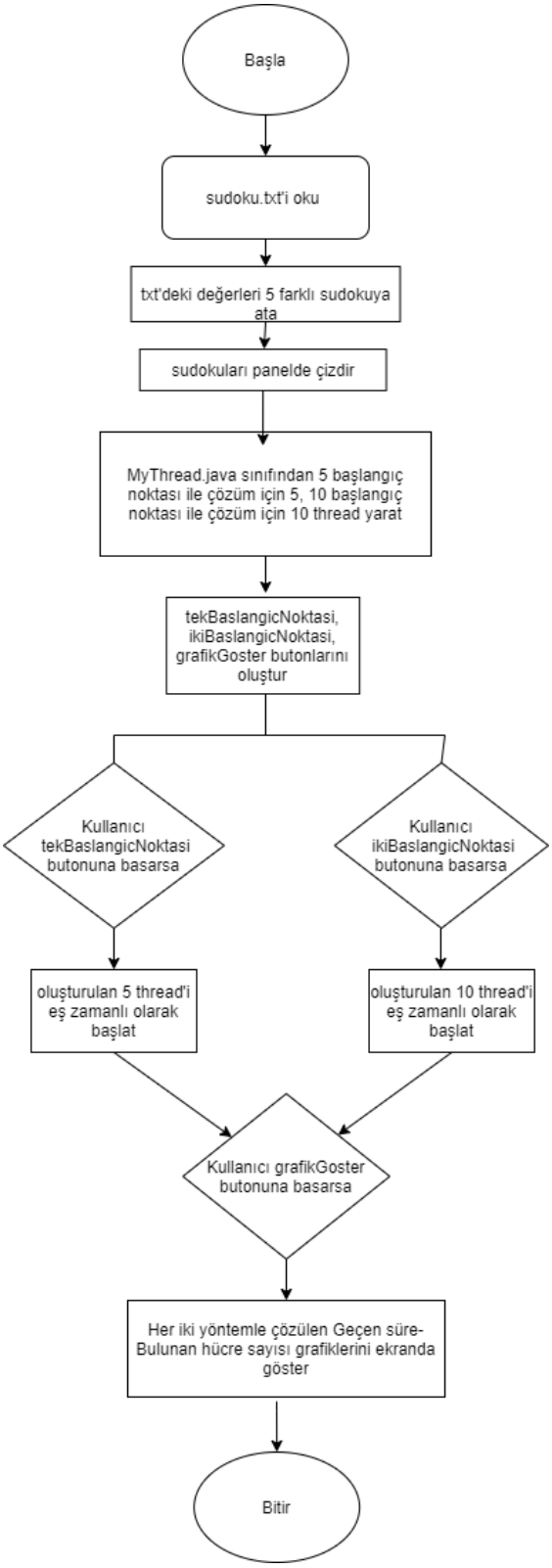
2.Yapılan Araştırmalar

Projede temel nokta sudokuları çözdürmek olduğundan öncelikle yazılımsal olarak sudoku nasıl çözülür diye araştırmalar yapmaya başladık. Samurai sudoku çözümü için neredeyse hiç kaynak bulamamış olsak da 9x9'luk sudoku çözüm algoritmalarını inceleyebildik. Sudoku çözümü için iki farklı yaklaşım gördük. Bunlardan birincisi bir sitede gördüğümüz Brute Force yöntemiydi. Bu yöntemle her bir hücreye tek tek değerler konulup sudoku dolduğunda bu çözüm uygun mu değil mi diye kontrol ediliyordu. Bu yöntem çok fazla zaman gerektirdiğinden biz ikinci yaklaşım olan backtracking algortimasını kullanmaya karar verdik. Bu yaklaşımda her bir hücreye bir değer konulup hemen ardından bulunduğu satır, sutun ve blok için uygun mu değil mi diye kontrol yapılır; uygunsa o değer yerleştirilir ve bir sonraki hücreye geçilir. Eğer uygun değilse bir önceki hücreye gidilir ve stack'te tuttuğu değerin bir fazlası denenir. Bu işlemler tüm boş hücreler çözülene kadar devam eder. Algoritma seçim kısmını hallettikten sonra çözüm algoritmasını kodlama aşamasına geçtik. Bu kısımda bazı internet sitelerinden faydalandık. Satır, sutun, blok kontrolü için fonksiyonlar tanımladık. Çözüm fonksiyonunun backtracking algoritmasını uygulayabilmesi için fonksiyonu rekürsif olarak kodladık. Daha sonra bu sudokuları ekranda gösterme aşamasına geçtik.Bunun için bir sitede gördüğümüz Java Applet sınıfı ile oluşturulmuş sudoku arayüzünden faydalanmaya karar verdik. Bu arayüzü samurai sudoku tasarımına

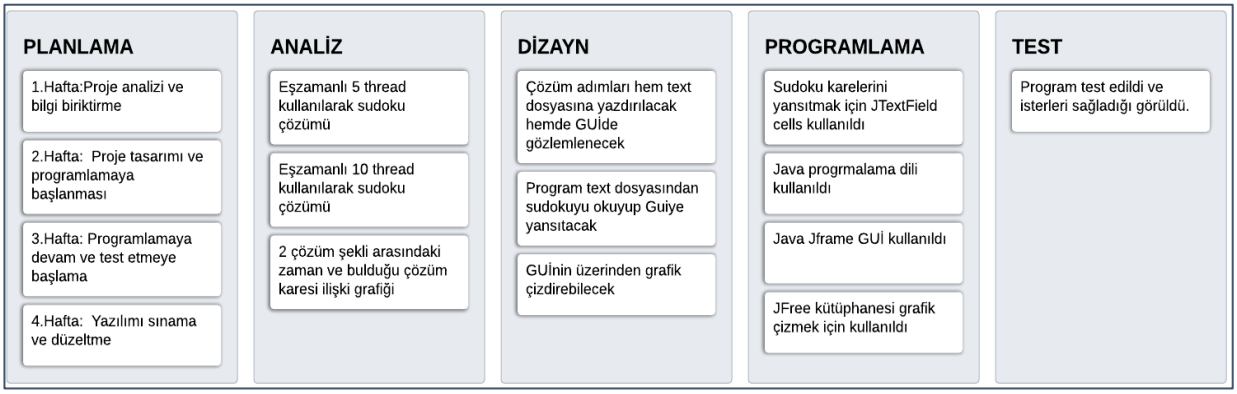
uyarlayabildik fakat sudokuları çözdürme aşamasında threadler kullandığımız ve thread kullanımında kilit nokta her bir thread için farklı bir obje kullanılması olduğundan bu applet'le oluşturulmuş tek bir panelde multithread işlemlerini yaptıramadık bu yüzden de JFrame sınıfını kullanarak her bir sudoku için farklı panel oluşturup threadlere farklı objeler tanımlamış olduk. JFrame ile tasarlama kısmında bazı web sitelerindeki 9x9'luk sudoku arayüzlerinden yararlandık. Sudokuları thread'ler ile çözdürebilmek için thread konusunu araştırmaya başladık. Bazı sitelerden ve videolardan bu konuyu anlayıp projemizde uygulamaya geçtik. Sudokuları tek bir başlangıç noktasından başlatıp çözdürebildik fakat her bir sudoku için iki thread gerektiğinden iki başlangıç noktasıyla çözüm isterini başlangıçta uygulayamadık. Bu yüzden de çoklu thread'lerde senkronizasyon işlemlerini araştırmaya başladık. Senkronizasyon problemini threadlerin aynı obje için ortak kullandığı solve() fonksiyonuna "senkronized" kelimesini ekleyerek halledebildik. Ortadaki sudokuyu diğer sudokularla beraber çözdürmeyi bir türlü başaramadığımız için projemizi 5 ayrı sudoku şeklinde tasarladık.Son olarak da bu sudokuların çözüm aşamalarını text dosyasına kaydetmek için Java'da thread adımlarını dosyaya yazdırma ve bu iki isterin grafiklenmesi için Java'da grafik çizdirme konularında araştırmalar yaptık. Bu şekilde bizden istenen tüm isterleri gerçekleştirmiş olduk.

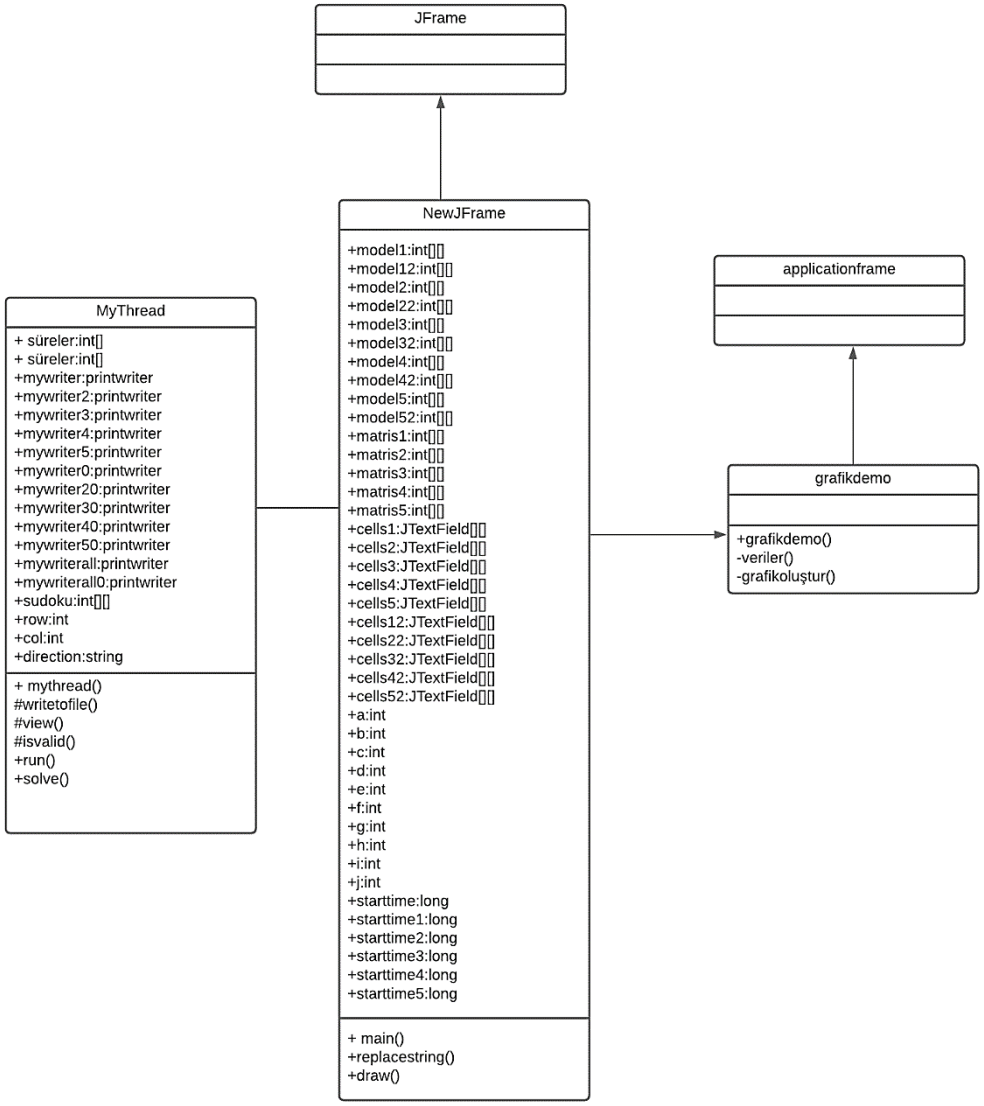
3.Tasarım

3.1. Akış Şeması



3.2 Yazılım Mimarisi





4.Genel Yapı

Programlama dili olarak daha çok tecrübemiz olan java dilini kullandık. GUİ olarak kullanmak amacıyla Java Jframe sınıfı oluşturduk.Bu sınıfta öncelikle 5 ayrı sudoku modeli için 5 ayrı JPanel oluşturduk.Bu panellere örnek verilen text dosyasındaki sudokuları atamak için FileReader ve BufferedReader kullandık.Sonraki aşamalarda sorun oluşturmaması için text dosyasındaki '*' karakterlerini replaceString() fonksiyonunu kullanarak 0'a dönüştürdük.Aldığımız değerleri 5 ayrı matrise atadık.Daha sonra bu matrisleri ve önceden tanımladığımız 5 ayrı JTextField hücre matrislerini draw() fonksiyonu ile 5 ayrı panele çizdirdik.Sudoku tasarım kısmından sonra sudokuyu çözen threadler oluşturmak için MyThread sınıfı açtık.MyThread sınıfında sudoku,row,col,direction isimli değişkenler tanımladık.Bu değişkenleri constructordaki parametrelere atadık.Sudoku çözmek için solve() fonksiyonunu oluşturduk.Bu fonksiyonda fonksiyona gönderilen matris, directiona bağlı olarak ileri veya geri yönde çözdürülebilir.Eğer direction olarak ileri geldiyse bu hücrenin değerinin 0'dan farklı olup olmadığı kontrol edilir.Hücrede 0'dan farklı bir değer tanımlıysa direk olarak bir sonraki hücreye gider.Eğer 0 ise hücreye 1 ile 10 arasında sırasıyla değerler verilmeye başlanır.Her bir değerin isvalid() fonksiyonu ile satır ,sütun,ve 3x3 blok kontrolü yapılır.Eğer değer bukunduğu satır,sütun veya 3x3 blokta bulunmuyorsa true döndürülür ve değer o hücreye atanır.Atandıktan sonra hücrenin yeni değeri view() fonksiyonu ile GUİ ekranında güncellenir.Ve bir sonraki hücreye gider.Hücre 0 olduğu halde 1 ile 10 arasında hiçbir değer uygun değilse hücreye tekrar 0 değeri atanır ve backtracking yöntemi ile bir önceki hücreye denediği değerler dışında değerler verilip isvalid()fonksiyonuna gönderilmeye devam edilir.Uygun değer bulunursa tekrar bir sonraki hücreye gidilir,bulunmazsa backtracking yöntemi tekrar eder. Satırdaki hücre sayısı 8'den büyük olduğu zaman bir sonraki

satıra geçilir. Satır sayısı 8'den büyük olursa "sudoku çözüldü" diye bir istisna fırlatılır. Bu, sudokunun doğru bir şekilde çözüldüğü anlamına gelir.Solve() fonksiyonunun direction parametresine "geri" değeri gelirse fonksiyon sudokuyu geriye doğru çözmeye başlar. Yukarıdaki işlemler burda da aynen devam eder fakat yönler farklıdır. Satırdaki hücrenin değeri "0" olduğu zaman bir önceki satıra geçilir. Gelinen satırın değeri "0"dan küçük olursa sudoku çözülmüş olur ve bir istisna fırlatılır.MyThread sınıfımızdan thread üreteceğimiz için bu sınıfta run() fonksiyonunu tanımlamamız gerekir. Oluşturacağımız threadler ile sudokuyu çözdüreceğimizden bu fonksiyonun içinde solve() fonksiyonunu çağırıyoruz. Solve() fonksiyonuna iki başlangıç noktasıyla sudoku çözdürme işlemi de yaptırmamız istendiğinden threadlerin çakışması problemini ve bilgi kayıplarını önlemek için bu fonksiyonu "synchronized void solve()" şeklinde tanımladık. Projenin isterlerinden biri de 5 thead ve 10 thread ile çözülmüş olan sudokular arasında zaman ve bulduğu çözüm karesi arasında ilişki grafiği çizdirmekti.Bu aşamada solve() fonksiyonunda direction="ileri" için satır sayısı 8'den büyük, direction="geri" için satır sayısı 0'dan küçük olduğunda her bir model için ayrı ayrı süre hesaplamaları yaptık. Modellerin süreleri için bir dizi tanımladık ve her bir modelin çözüm süresini bu süreleri diziye atadık. Bu diziyi daha sonrasında GrafikeDemo adlı sınıfta kullanacağımız için static olarak tanımladık. Grafik için süre değerlerini hesaplattıktan sonra çözülmüş olan hücre sayıları için sudoku.txt dosyasındaki her bir modelin "0" içeren hücre sayılarını hesaplattık ve bu değerleri grafikolustur fonksiyonunda kullandık. Ayrıca sudoku çözüm aşamalarını text veya veri tabanına kaydetmemiz gerektiğinden her bir sudoku için yeni dosya açıp bu dosyalara hücre değişimlerini matris matris kaydettik. Her bir sudoku çözüme ulaştığında onun için tanımlamış olduğumuz dosyayı kapattık ve tüm sudokular çözüme ulaştığında bu dosyaları çözümFull adlı dosyaya yazdırdık.

5.Referanslar

Web site

https://www.geeksforgeeks.org/sudoku-backtracking-7/

https://www.heimetli.ch/ffh/simplifiedsudoku.html

https://causlayer.orgs.hk/OscarHChung/GUI-Sudoku/blob/master/GuiSudoku.java

https://stackoverflow.com/questions/850866/multi-threaded-algorithm-for-solving-sudoku

https://stackoverflow.com/questions/63338960/java-how-to-set-up-a-grid-for-sudoku-solver-using-swing-and-super-paintchildren

https://coderanch.com/t/665173/java/implement-multi-threading-sudoku-solver

https://www.techwithtim.net/tutorials/python-programming/sudoku-solver-backtracking/

https://ufukuzun.wordpress.com/2015/02/26/javada-multithreading-bolum-3-synchronized-anahtar-kelimesi/

https://www.geeksforgeeks.org/java-program-merge-two-files-third-file/

https://stackoverflow.com/questions/34958829/how-to-save-a-2d-array-into-a-text-file-with-bufferedwriter

https://www.jfree.org/jfreechart/