YAZILIM LABORATUVARI I

PROJE II:

SAMURAI SUDOKU SOLVER

SENA ÖKTEM

Kocaeli Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği(İÖ)

190202054@kocaeli.edu.tr

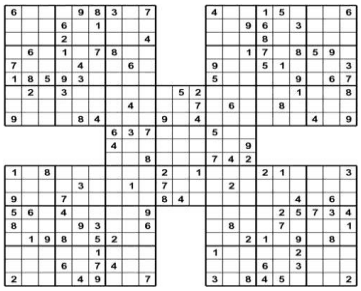
1. PROBLEM TANIMI :

Verilen .txt dosyası üzerinden sudokuyu oluşturacak olan rakamlar okunarak Samurai Sudoku oluşturulur. Oluşturulan sudoku 5'li ve 10'lu Thread kullanılarak çözdürülür. 5'li Thread'in sudokuyu çözme süresi ve 10'lu Thread'in sudokuyu çözme süresi hesaplanır. Bu Threadlerin ne kadar kutucuk çözdürebildiği hesaplanır. Elde edilen süre ve çözülen kutucuk sayısı süre-çözülen kare grafiği olarak arayüzde sunulur. Samurai Sudokunun çözümü arayüzde sunulur. Sudokunun çözüm aşamaları .txt dosyasında tutulur.

2. YAPILAN ARAŞTIRMALAR :

2.1. Samurai Sudoku:

Samurai Sudoku bulmacaları, örtüşen beş sudoku ızgarasından oluşur. Standart sudoku kuralları her 9 x 9 ızgara için geçerlidir. Her boş hücreye 1'den 9'a kadar rakamlar yerleştirilir. Her satır, her sütun ve her 3 x 3 kutu, her bir rakamdan birini içermelidir.



2.2. Tekli Sudoku Çözümü :

Sudokunun satır ve sütunlarındaki rakamları kontrol etmek ve kutucukların dolu mu boş mu olduğunu kotrol etmek için isValid() fonskiyonu üretilmiştir. Kutucukların boş olması durumunda kutucuğun içine 1-9 arası gelebilecek tüm değerlerin kontrolünü sağlaması için solve() fonksiyonu oluşturulmuştur. Üretilen algoritmada boş olan her kutucuk için 1-9 arası alabileceği değerler sırasıyla kutucuklara yazılıp denenir. Aynı satırda, sütunda ve ya ızgarada aynı rakam varsa solve() fonksiyonu false döner ve kutucuğun içi doldurulmaz. Eğer kontrol sonucunda solve() fonksiyonu true dönerse kutucuğun içine gelecek sayı bulunmuş olur. Bu işlemler tüm sudoku çözdürülene kadar recursive şekilde devam eder.

2.3 Thread ve Multi-Thread:

Thread:

Tek thread kullanılan algoritmalarda işlemle ilgili tüm görevleri thread yönetir. Çok iş parçacıklı bir yapı söz konusu değildir. Aynı anda farklı işler yapma gereği olmayan algoritmalarda kullanılır.

Multi-Thread:

Çok iş parçacıklı bir algoritmada aynı anda birden çok iş parçacığı yürütülür. Her iş parçacığı, kaynakları en iyi şekilde kullanarak aynı anda farklı görevleri yerine getirir. Böyle algoritmalarda Multi-Thread kullanılır.

3. GENEL YAPI :

Proje 6 classtan oluşmaktadır. 3 class JFrame, 1 class Main classıdır.

3.1. Map2() :

.txt ‘den harita okunması yapılmıştır. Okunan harita 5 farklı sudoku için 5 farklı matrise atanmıştır.

3.2.SudokuSolver() :

Map2() classında okunan sudokular, sudoku kurallarına uyacak şekilde kontrolü sağlanmış ve çözdürülmüştür.

3.3.DrawSudoku() :

JFrame classıdır. Graphics kütüphanesi kullanılmıştır. Map2() de okunan sudoku haritalarının birleşimi olan Samurai Sudokuyu ekrana çizdirir. SudokuSolve() classında çözülen Samurai Sudokuyu ekrana çizdirir.

3.4.GrafikCiz() :

JFrame classıdır. Graphics kütüphanesi kullanılmıştır. Çözülen kare-süre grafiğini ekrana çizdirir.

3.5.MenuPanel() :

JFrame classıdır. Proje çalıştırıldığında açılacak ilk ekrandır. 5’li Thread, 10’lu Thread ve Grafik isimli 3 adet butondan oluşmaktadır. Kullanıcının tercihine göre 5’li Thread butonu 5 thread kullanılarak sudokunun çözümünü göstermektedir. 10’lu Thread butonu 10 thread kullanılarak sudokunun çözümünü göstermektedir. Grafik butonu Çözülen kare-süre grafiğini göstermektedir.

3.6.Main() :

Projenin çalıştırıldığı ana classtır.

4.YAZILIM MİMARİSİ :

```
private boolean solve(int[][] board) {

    map2= new Map2();
    map2.dosyaOku();
    for(int i=0 ; i<9;i++) {
        for(int j=0;j<9;j++) {
            if(board[i][j] == 0) {
                for(int c = 1; c<=9;c++) {
                    if(isValid(board,i,j,c)) {
                        board[i][j]=c;
                        if(solve(board)) {
                        writeFile(i, j, c);
                        solvedNumber1++;

                            return true;
                        }
                        else {
                            board[i][j]=0;
                        }


                    }
                }
                return false;
            }
        }

    }
    return true;
```

solve fonksiyonu sudokudaki boş kutucuklara gelebilecek rakamları dener ve uygun olanını bulur.

```
private boolean isValid(int [][] board,int row,int col,int c) {
    for(int i=0;i<9;i++) {
        if(board[i][col]!=0 && board[i][col] == c) {
            return false;
        }
        if(board[row][i] != 0 && board[row][i]==c) {
            return false;
        }
        if(board[3*(row/3)+i/3][3*(col/3)+i%3]!=0 && board[3*(row/3)+i/3][3*(col/3)+i%3]==c) {
            return false;
        }
    }
    return true;
}
```

İsValid fonksiyonu sudokudaki kutucuğun boş olup olmadığının kontrolünü sağlar.

```
public void writeFile(int i,int j,int c) {
      try {

           fWriter = new FileWriter("C:\\Users\\mehmet\\Desktop\\data.txt", true);
            bufferedWriter = new BufferedWriter(fWriter);
              bufferedWriter.write(" sudokunun "+(i+1)+" . satiri ve " +(j+1)+" . sutununa "+ c +" yazdi");
              bufferedWriter.newLine();

      //  fWriter=new FileWriter("C:\\Users\\mehmet\\Desktop\\data.txt");
      //  pWriter=new PrintWriter(fWriter);
  //  pWriter.printf("sudokunun %d . satiri ve %d . sutununa  %d yazdi",i+1,j+1,c);

  //  System.out.println("sudokunun"+(i+1)+" . satiri ve " +(j+1)+" . sutununa "+ c +" yazdi");
      bufferedWriter.close();
      }catch (IOException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
      }

      return;
  }
```

WriterFile fonksiyonu sudokunun çözüm adımlarını .txt’de depolar.

```
public void paint(Graphics g) {
    MenuPanel menuPanel = new MenuPanel();
    SudokuSolver solver = new SudokuSolver();

    Graphics2D g2d = (Graphics2D) g;
    super.paintComponents(g2d);
      g2d.drawLine(100, 50, 100, 250);
      g2d.drawLine(100, 250,300, 250);
      g2d.setColor(Color.green);
    g2d.drawString("millisecond", 305, 250);
    g2d.drawString("SolvedBox", 100, 45);
    g2d.setColor(Color.blue);
     g2d.drawLine(430, 80,450, 80);
     g2d.drawString("T5",455,80);

     g2d.setColor(Color.magenta);
     g2d.drawLine(430, 95,450, 95);
     g2d.drawString("T10",455,95);
     g2d.setColor(Color.green);
     for(int i=0;i<=200;i+=50)
        {
             g2d.drawOval(100+i,250,3,3);
             g2d.drawString(String.valueOf((i/50)*1000),
             g2d.drawOval(100,50+i,3,3);
             g2d.drawString(String.valueOf((i/50)*200),
        }

     for(int i=0;i<200;i+=50) {
            g2d.setColor(Color.blue);
          QuadCurve2D q = new QuadCurve2D.Float();
    q.setCurve(100,250, 600,i/50*menuPanel.timeElapsed
    g2d.draw(q);
    g2d.setColor(Color.MAGENTA);
    q.setCurve(100,250, 300,i/50*menuPanel.timeElapsed2
    g2d.draw(q);
     }
```

Paint fonksiyonu grafiğin koordinat düzlemini, koordinatlarının ismini ve koordinat üzerindeki sayıları ekrana çizdirir.

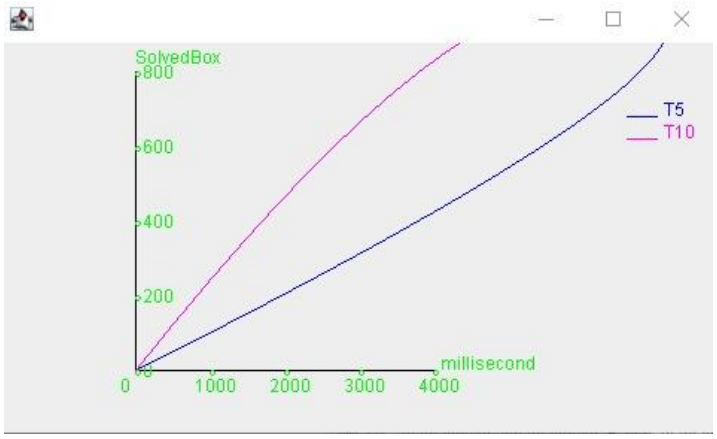
```
public void paint(Graphics g) {
Graphics2D g2d = (Graphics2D) g;

//sudoku1  sudoku2  sudoku5 sudoku3 sudoku4
for(int i=0;i<9;i++) {
    for(int j=0;j<9;j++) {

          g2d.drawRect(40+j*30,40+i*30,30,30);
          g2d.drawRect(40+((j+12)*30),40+i*30,30,30)
          g2d.drawRect(40+((j+6)*30),40+((i+6)*30),3
          g2d.drawRect(40+j*30,40+((i+12)*30),30,30)
         g2d.drawRect(40+((j+12)*30),40+((i+12)*30),
```

Paint fonksiyonu sudokuyu ve sudokunun çözümünü ekrana çizdirir.

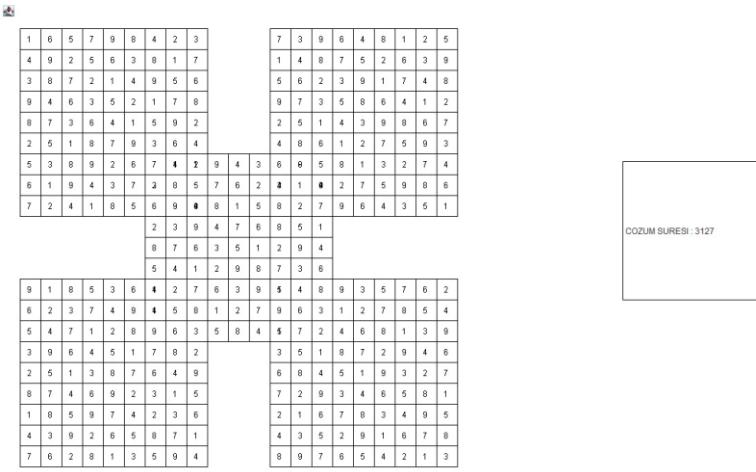
5. ARAYÜZ VE TASARIM

JFrame classı kullanılmıştır. Graphics kütüphanesi kullanılmıştır.

İlk açılan frame'de 3 tane buton kullanılmıştır. Butonlara göre 5'li,10'lu thread ve grafik çizimi frameleri gösterilecektir.



Grafik çizimi için Graphics kütüphanesi kullanılmıştır. .drawLine() fonksiyonu ile yatay ve dikey olmak üzere iki çizgiyle koordinat sistemi çizdirilmiştir. .drawOval() fonksiyonu ile koordinat üzerindeki sayılar yerleştirilmiştir. .drawString() fonksiyonu ile x ve y ekseninin neyi simgelediği yazdırılmıştır.



Samurai Sudoku çizimi için Graphics kütüphanesi kullanılmıştır. .drawRect() fonksiyonu ile sudokunun kutucukları çizdirilmiştir. .drawString() fonksiyonu ile kutucukların içi çözülen değerlerle doldurulmuştur.

6. REFERANSLAR :

https://www.youtube.com/watch?v=mcXc8Mva2bA

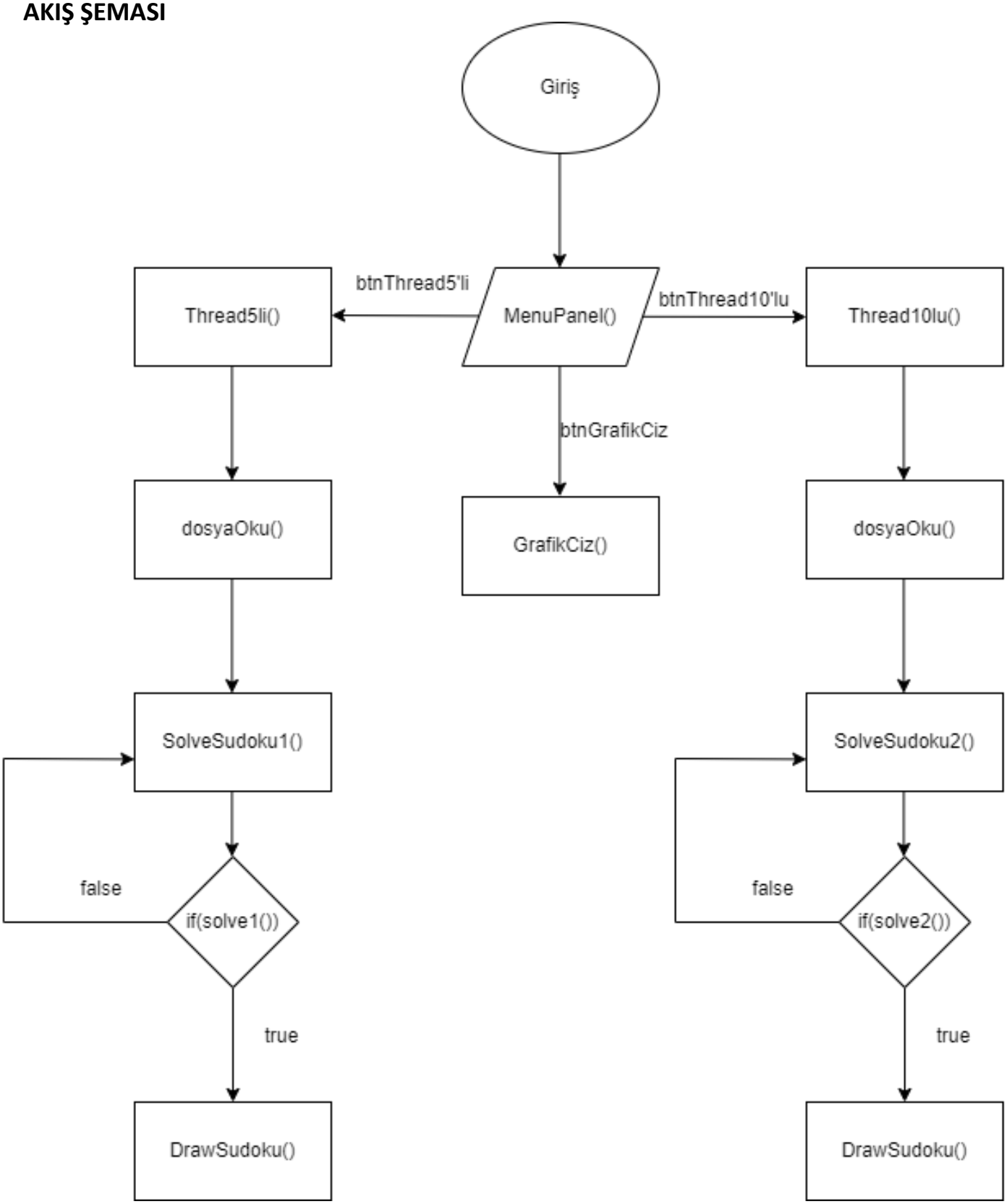
https://www.youtube.com/watch?v=r_MbozD32eo

https://www.youtube.com/watch?v=fw3QIvD7-aY

https://www.youtube.com/watch?v=dTBvbH3xAmc

https://www.geeksforgeeks.org/multithreading-in-java/

AKIŞ ŞEMASI


Giriş
MenuPanel()
btnThread5'li
btnThread10'lu
Thread5li()
Thread10lu()
btnGrafikCiz
GrafikCiz()
dosyaOku()
dosyaOku()
SolveSudoku1()
SolveSudoku2()
false
if(solve1())
false
if(solve2())
true
true
DrawSudoku()
DrawSudoku()