GEBZE TECHNICAL UNIVERSITY

CSE443 – Object Oriented Analysis and Design

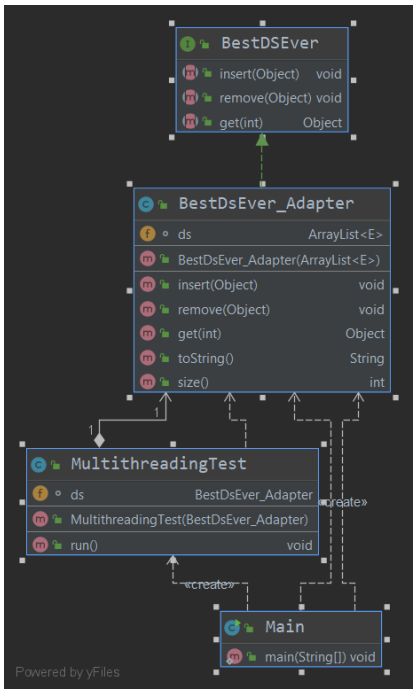
Homework 3 Report

İslam Göktan SELÇUK – 141044071

Part 1

İlk part için verilen veri yapısı için implementasyon detayını bilmeden sınıfın operasyonlarının thread safe hale getirilmesi istenmişti. Bu problem için Java'nın sahip olduğu ArrayList BestDSEver için kullanıldı. Adapter örüntüsü kullanılarak bu veri yapısı thread safe hale getirildi. Metotlar için synchronized keyword'ü kullanılarak birden fazla thread'in aynı veri yapısı üzerinde gerçekleştirdiği operasyonların çakışması engellenmiş oldu.

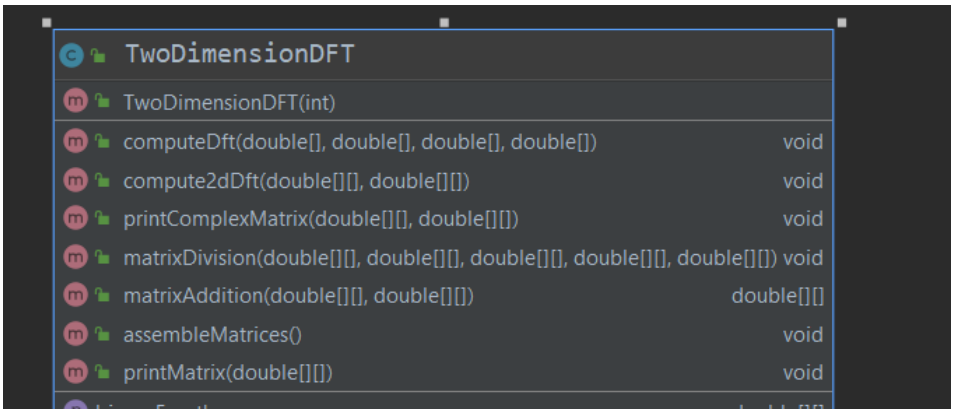
```java
public class BestDsEver_Adapter<E> implements BestDSEver {
    ArrayList<E> ds;

    public BestDsEver_Adapter(ArrayList<E> ds) { this.ds = ds; }

    @Override
    public synchronized void insert(Object o) {
        ds.add((E)o);
    }

    @Override
    public synchronized void remove(Object o) { ds.remove((E)o); }

    @Override
    public synchronized Object get(int index) { return ds.get(index); }

    @Override
    public synchronized String toString() { return ds.toString(); }

    public int size() { return ds.size(); }
}
```



Part 2

Öncelikle 2D Dft hesabı için gerekli olan matris operasyonlarının yapılacağı ve bu operasyonların tutulacağı iki boyutlu diziler için TwoDimensionDFT sınıfı oluşturuldu. Bu sınıf içerisinde matrisin 4'e bölünmesi, matris toplanması ve dft'yi hesaplayan metotlar implement edildi.

İki boyutlu DFT hesabı için tek boyutlu dft'yi hesaplayan metot kullanıldı.



A + B operasyonu için dört adet matris oluşturuldu. Karmaşık sayının reel kısmı için ayrı sanal kısmı için ayrı matrisler oluşturuldu. Bu oluşturulan matrisler, her bir thread'te eş zamanlı işlem yapılabilmesi için dörde bölündü.

Toplama işlemi için 4 adet thread oluşturuldu. Toplama işlemleri için wait() ve notifyAll() metotlarıyla tüm thread'lerin işlemlerinin bitirildiğinden emin olunarak implementasyon yapıldı.

Tüm thread'lerin kontrolü için synchronizedList kullanıldı. İşi biten her thread bu listeye spesifik bir string ekledi. Böylelikle listedeki eleman sayısı kontrol edilerek wait() metotu için bir bariyer oluşturulması sağlandı.

Toplama işleminden sonra dörde ayrılmış olan matrisler birleştirilerek ekrana yazıldı.

Toplama işlemi bittikten sonra buradaki benzer süreç dft hesabında tekrarlandı. Dörde bölünen matrisler için thread'ler paralel olarak dft hesabını gerçekleştirdi.

4096x4096 boyutlu matris için programın çıktı üretmesi çok uzun sürdüğü için 512x512 boyutlu matris kullanıldı. Aşağıdan boyut değiştirilebilir.

```
public void start() throws InterruptedException {

    dft = new TwoDimensionDFT( n: 512);
```

```java
for (int i = 0; i < 4; i++)
{
    Thread object = new Thread(new MatrixAdditionThread(dft, i, this.additionSyncList));
    object.start();
}
System.out.println("Waiting for Addition Operation...");
waitForAddition();

System.out.println("<<< End of addition >>>");
dft.assembleMatrices();
dft.printComplexMatrix(dft.getArealInput(), dft.getAimagInput());

for (int i = 0; i < 4; i++)
{
    Thread object = new Thread(new MatrixDftThread(dft, i, this.dftSynchedList));
    object.start();
}
System.out.println("Waiting for Calculation of Dft...");
waitForDft();

dft.assembleMatrices();
System.out.println("<<< End of Dft >>>");
```

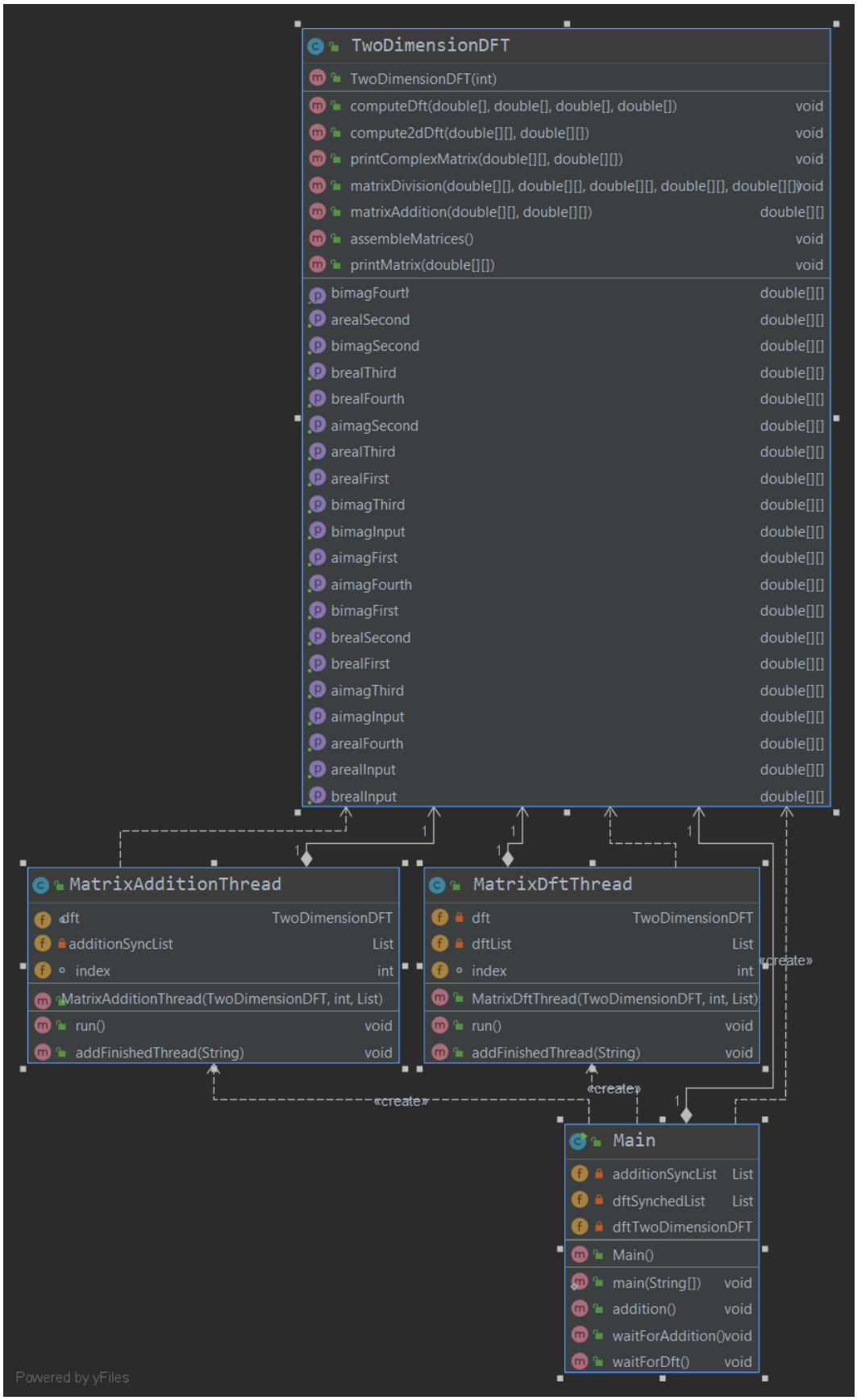
Toplama işleminin gerçekleştiren thread'lerin oluşturulması

Tüm thread'ler için bekleyen bariyer.

DFT işlemini gerçekleştiren thread'lerin oluşturulması

```java
public MatrixAdditionThread(TwoDimensionDFT dft, int index, List synchedList) {
    this.dft = dft;
    this.index = index;
    this.additionSyncList = synchedList;
}

@Override
public void run() {
    try
    {
        switch (index) {
            case 0:
                dft.matrixAddition(dft.getArealFirst(), dft.getBrealFirst());
                dft.matrixAddition(dft.getAimagFirst(), dft.getBimagFirst());
                addFinishedThread( element: "first");
                break;
            case 1:
                dft.matrixAddition(dft.getArealSecond(), dft.getBrealSecond());
                dft.matrixAddition(dft.getAimagSecond(), dft.getBimagSecond());
                addFinishedThread( element: "second");
                break;
```

```java
public void waitForAddition() throws InterruptedException {
    synchronized (additionSyncList) {
        while (additionSyncList.size() < 4) {
            additionSyncList.wait();
        }
    }
}

public void waitForDft() throws InterruptedException {
    synchronized (dftSynchedList) {
        while (dftSynchedList.size() < 4) {
            dftSynchedList.wait();
        }
    }
}
```


TwoDimensionDFT
TwoDimensionDFT(int)
computeDft(double[], double[], double[], double[]) void
compute2dDft(double[][], double[][]) void
printComplexMatrix(double[][], double[][]) void
matrixDivision(double[][], double[][], double[][], double[][], double[][]) void
matrixAddition(double[][], double[][]) double[][]
assembleMatrices() void
printMatrix(double[][]) void
bimagFourth double[][]
arealSecond double[][]
bimagSecond double[][]
brealThird double[][]
brealFourth double[][]
aimagSecond double[][]
arealThird double[][]
arealFirst double[][]
bimagThird double[][]
bimagInput double[][]
aimagFirst double[][]
aimagFourth double[][]
bimagFirst double[][]
brealSecond double[][]
brealFirst double[][]
aimagThird double[][]
aimagInput double[][]
arealFourth double[][]
arealInput double[][]
brealInput double[][]
1
1
1
1
1
MatrixAdditionThread
dft TwoDimensionDFT
additionSyncList List
index int
MatrixAdditionThread(TwoDimensionDFT, int, List)
run() void
addFinishedThread(String) void
MatrixDftThread
dft TwoDimensionDFT
dftList List
index int
MatrixDftThread(TwoDimensionDFT, int, List)
run() void
addFinishedThread(String) void
«create»
«create»
«create»
1
Main
additionSyncList List
dftSynchedList List
dftTwoDimensionDFT
Main()
main(String[]) void
addition() void
waitForAddition() void
waitForDft() void
Powered by yFiles