Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 4 REPORT

Seyit Ahmet KARACA
141044084

Course Assistant:Nur Banu ALBAYRAK

(Q1)

1. Test Cases

StackA,B,C ve D için her birine test.csv deki satirlari tek tek okutmak için her birini ArrayList oluşturdum.Arraylistin her elemanindeki stackler bir satiri tutacak şekilde kodlandıktan sonra içerlerindeki bilgileri stringbuilder ile birleştirip dosyaya yazdırdım.

Dosyadan arraylist'i okuyan ve push methodunun kullanıldığı method :

```
/**
 * test.csv yi stackA ya okur
 * @param alsd
 * @throws IOException
 */
public static void readToStackA(ArrayList<StackA> alsd) throws IOException {
    FileReader fr = new FileReader(new File( pathname: "test.csv"));
    BufferedReader br = new BufferedReader(fr);
    StackA sd;
    String line;
    String linePart[];
    while((line = br.readLine()) != null ){
        sd = new StackA();
        linePart = line.split( regex: ",");
        for(int i = 0 ; i < linePart.length;i++){
            sd.push(linePart[i]);
        }
        alsd.add(sd);
    }

    br.close();
    fr.close();
}
```

Arraylisti methoda gonderip cagrilmasi :

```
/*  methodlarin push lari burada kullanildi */
startTime = System.nanoTime();
    readToStackD(sdal);
stopTime = System.nanoTime();
result = stopTime-startTime;
System.out.println("StackD okuma sure :"+result+" nano saniye");
```

Stack pop-get-remove methodlarinin kullanildigi blok :

```
/*   Stacklerin get-pop methodlari       */
startTime = System.nanoTime();

for (int i = 0; i < saal.size(); i++) {
    sb.append(saal.get(i).size() + ",");
    sizeStack = saal.get(i).size();

    for (int j = 0; j < sizeStack; j++) {
        sb.append(saal.get(i).pop());
        if (j != sizeStack - 1)
            sb.append(",");
    }
    sb.append("\n");
}

stopTime = System.nanoTime();
result = stopTime-startTime;
System.out.println("StackA pop-get sure :"+result+" nano saniye");
/*-----------------------------------------------*/
```

2. Running and Results

main.java | testResult_1.csv | StackD.java | StackC.java | StackB.java | StackA.java | IStack.java

```
7,7,6,5,4,3,2,1
5,4.0,6.7,4.5,2.3,1.2
7,f,d,t,j,h,g,a
3,defter,kalem,kitap
7,7,6,5,4,3,2,1
5,4.0,6.7,4.5,2.3,1.2
7,f,d,t,j,h,g,a
3,defter,kalem,kitap
7,7,6,5,4,3,2,1
5,4.0,6.7,4.5,2.3,1.2
7,f,d,t,j,h,g,a
3,defter,kalem,kitap
7,7,6,5,4,3,2,1
5,4.0,6.7,4.5,2.3,1.2
7,f,d,t,j,h,g,a
3,defter,kalem,kitap
```

Programın çıktısı resimde de görüldüğü gibi her stack elemanlarını push,pop , remove ve get methodlarını kullanarak elemanlar dosyadan okunup , herbiri için dosyaya yazdırırldı.
Her birinde aynı methodlar olmasına karşın verilerin tutuldukları yapılar farklı olduğu için implementlerinde ufak farklılıklar olsa da benzer yapılar olduklarından implement edilmesi kolay oldu.

3.Analysis

Analiz olarak her bir stack için nano saniye cinsinden okuma sürelerini ve verilerini birleştirdikleri blokların sürelerini tuttum.Bu süreleri tutarken dosyadan okurken push methodunu kullandım.Dosyaya yazdırmak için pop , get methodları kullanıldım ve bunları her bir stack için yaptım.Bundan dolayı aynı methodları kullandıkları kod bloklarının sürelerini tutarak aralarında bir farklılık olduğunu anladım.Aşağıda 3 farklı çalıştırmada çalışma süreleri bulunuyor.Bu süreler nano saniye cinsinden aldım.Yaptıkları işlemler çok hızlı olduğundan saniye cinsinden tuttuğumda hepsi aynı değeri gösteriyorlardı bu yüzden daha detaylı görmek için nano saniye cinsnden aldım.
Push işlemini en hızlı yapan stackB oldu ve en yavaşı ise stackD oldu.
Aralarındaki fark ise stackB arraylist objesinde tutuyor elemanlar.StackA da arraylistten extend edilsede bir üst sınıfın methodlarını kullandığım için StackB kadar hızlı olamadı.
StackC node yapısı kullandığı için sürekli yeni node oluştur sonraki node'u eskisine ekle işlemleri yaparken diğerlerine göre daha fazla işlem yaptığından işlem süresi uzun çıktı.
StackD ise bir queue objesi tutacak ve bunun üzerinden işlem yapacaktı.Bu objeyi linkedlist ile ilişkilendirip öyle kullandım ve StackC deki node yapısı bunda da olduğu için ve bir obje üzerinden çağırdığım için süresi en fazla olan Stack bu oldu.

Verilerini bir stringbuiler'da birleştirirken her bir stackin diğer methodlarını kullandım ve get ve pop methodlarının implement edilmelerinden kaynaklı farklılıklar oluştu.Ortalamaları alındığında hemen hemen hepsinin çalışma süreleri push ve get methodlarında aynı çıktılar.

Analiz 1 :

```
StackD okuma sure :1173215 nano saniye
StackC okuma sure :876754 nano saniye
StackB okuma sure :500156 nano saniye
StackA okuma sure :516736 nano saniye


StackA pop-get sure :38292 nano saniye
StackB pop-get sure :35528 nano saniye
StackC pop-get sure :26843 nano saniye
StackD pop-get sure :53292 nano saniye
```

Analiz 2:

```
StackD okuma sure :1131766 nano saniye
StackC okuma sure :869648 nano saniye
StackB okuma sure :486339 nano saniye
StackA okuma sure :813198 nano saniye


StackA pop-get sure :79346 nano saniye
StackB pop-get sure :57240 nano saniye
StackC pop-get sure :46186 nano saniye
StackD pop-get sure :85267 nano saniye
```

Analiz 3:

```
StackD okuma sure :1083606 nano saniye
StackC okuma sure :939914 nano saniye
StackB okuma sure :478444 nano saniye
StackA okuma sure :543185 nano saniye


StackA pop-get sure :37896 nano saniye
StackB pop-get sure :34344 nano saniye
StackC pop-get sure :27238 nano saniye
StackD pop-get sure :51319 nano saniye
```

Q2. Test Cases

MyQueue sınıfım için her bir satırın tutulması için myQueue oluşturup onda her bir satırı tutan myQueue yapısı oluşturdum.Her bir satırı tutan myQueue objemi reverse methodu ile ters cevirip elemanlarınıda döngü ile ters cevirerek dosyaya yazılacak hale getirdim.Aynı şekilde queue objeleri tutan bir queue objemide okuma fonksiyonuna gönderip satırları queue objesinde tutup bunları queue'ya ekleyip reverseQueue methodu ile myQueue üzerinden ters çevirip dosyaya yazılacak hale getirdim.

MyQueue ters çevrilmesi :

```
        myqueue.reverse();
        for (int i = 0; i < myqueue.size(); i++) {
            tempMyQueue = (myQueue) myqueue.get(i);
            tempMyQueue.reverse();
            size2 = tempMyQueue.size();
            for (int j = 0; j < size2; j++) {
                sbMyqueue.append(tempMyQueue.remove());
                if (j != size2 - 1)
                    sbMyqueue.append(",");
            }
            sbMyqueue.append("\n");
        }
```

Queue ters çevrilmesi :

```
    if (fileReader(ku)) {

        myku.reverseQueue(ku);
        size = ku.size();
        for (int i = 0; i < size; i++) {
            ku2 = (Queue) ku.remove();
            myku.reverseQueue(ku2);
            size2 = ku2.size();
            for (int j = 0; j < size2; j++) {
                sb.append(ku2.remove());
                if (j != size2 - 1)
                    sb.append(",");
            }
            sb.append("\n");
        }
```

MyQueue tutan myQueue objesi okunması :

```
public static boolean fileReader(myQueue _myqueue) throws IOException {
    myQueue temp;
    FileReader fileReader = new FileReader(new File( pathname: "test.csv"));
    BufferedReader bufferedReader= new BufferedReader(fileReader);
    String line;
    String lineParts[];
    while((line = bufferedReader.readLine()) != null){
        temp = new myQueue();
        lineParts =  line.split( regex: ",");
        for(int i = 0 ; i < lineParts.length;i++){
            temp.add(lineParts[i]);
        }
        _myqueue.add(temp);
    }

    fileReader.close();
    bufferedReader.close();
    if(_myqueue.size() > 0)
        return true;
    else
        return false;

}
```

Queue tutan queue objesi okunması :

```
public static boolean fileReader(Queue _que) throws IOException {
    Queue temp;
    FileReader fileReader = new FileReader(new File( pathname: "test.csv"));
    BufferedReader bufferedReader= new BufferedReader(fileReader);
    String line;
    String lineParts[];
    while((line = bufferedReader.readLine()) != null){
        temp = new LinkedList();
        lineParts =  line.split( regex: ",");
        for(int i = 0 ; i < lineParts.length;i++){
            temp.add(lineParts[i]);
        }
        _que.add(temp);
    }

    fileReader.close();
    bufferedReader.close();
    if(_que.size() > 0)
        return true;
    else
        return false;
}
```

2- Running and Results

Recursive olarak Queue parametresi alan reverseQueue methoduna dosyadan okutup doldurudğum objeyi myQueue üzerinden reverseQueue çağırdım.Reverse fonksiyonu myQueue üzerinden parametre almadan çağırdım.Dosyaya yazılıcak stringbuiler ı dosyaya her seferinde append ederek yazdırdığım için bu çıktı program her çalıştırıldığında üzerine eklenerek devam ediyor.

Programın çıktısı

```
defter,kalem,kitap
f,d,t,j,h,g,a
4.0,6.7,4.5,2.3,1.2
7,6,5,4,3,2,1
defter,kalem,kitap
f,d,t,j,h,g,a
4.0,6.7,4.5,2.3,1.2
7,6,5,4,3,2,1
```

Github : https://github.com/SeyitAhmetKARACA/141044084_HW04