EGE UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

204 DATA STRUCTURES (3+1)

2020–2021 FALL SEMESTER

PROJECT-2 REPORT

(List, Stack, Queue, PQ – Priority Queue Data Structures)

DELIVERY DATE

11/01/2021

PREPARED BY

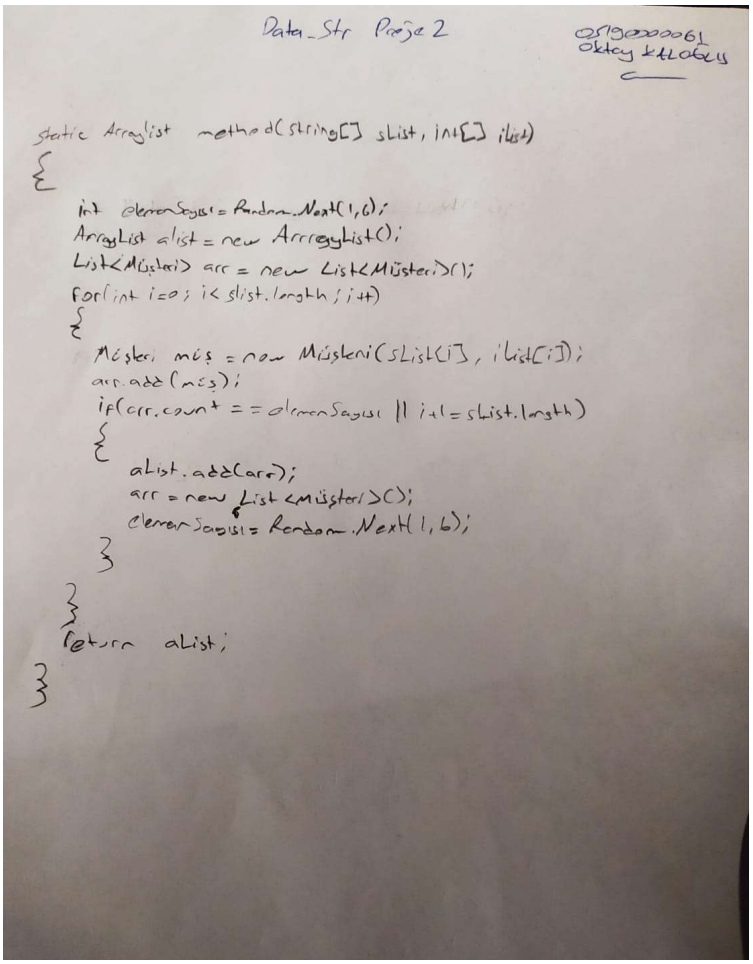
05190000061 Oktay Kaloğlu

İçindekiler

LİSTE, YIĞIT, KUYRUK ve ÖNCELİKLİ KUYRUK VERİ YAPILARI

Visual Studio Community 2019, 16.8.1 and C# used to develop this program.

1.a Bileşik Veri Yapısı için Ön Çalışma



1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma

1.b.1 Kaynak Kod

```
static ArrayList BileşikVeriYapısıOluşturmaVeElemanEklemeMetodu(string[] sList, int[] iList)
        {
            Random rand = new Random();//birden fazla random sayı üretilecekse nesne oluşturulması sayının tekrarlanma hatasını önlüyor
            int randNum = rand.Next(1, 6);

            ArrayList aList = new ArrayList();
            List<Müşteri> arr = new List<Müşteri>();

            for (int i = 0; i < sList.Length; i++)
            {
                Müşteri müşteri = new Müşteri(sList[i], iList[i]);
                arr.Add(müşteri);

                if ((arr.Count == randNum)||(i + 1 == sList.Length) )//liste yeteri kadar eleman sayısına ulaşmış veya giriş dizisinin
son itemine ulaşılmış. artık liste arrayliste eklenebilir
                {
                    aList.Add(arr);
                    randNum = rand.Next(1, 6);
                    arr = new List<Müşteri>();//yeni bir liste oluşturulup referansı ileride arrayliste verilecek

                }

            }
            return aList;
        }
```

```
static void BileşikVeriYapısındakiElemanlarıYazdırınız(ArrayList aList)
{
    foreach (List<Müşteri> müşlis in aList)
    {
        foreach (Müşteri müş in müşlis)
        {
            Console.Write(müş.data() + " , ");
        }
        Console.WriteLine();
    }
    Console.WriteLine(" ");
}
```

1.b.2 Ekran görüntüleri

//1-b maddesi için üretilen konsol/ekran görüntüsünü buraya ekleyiniz

```
Ali 8 , Merve 11 , Veli 16 ,
Gülay 5 , Okan 15 ,
Zekiye 14 , Kemal 19 ,
Banu 3 , İlker 18 , Songül 17 ,
Nuri 13 , Deniz 15 ,
```

1.b.3 Veri Yapıları ve Açıklama

//1-b maddesi için kullanmış olduğunuz veri yapılarını burada listeleyip kısaca açıklayınız

ArrayList(aList),List<T>(arr),dizi(sList,iList),nesne(Müşteri) gibi veri yapıları kullanılmıştır.

ArrayList birden fazla türde nesne içerebileceği için tercih edilmiştir. Oluşturulmasında ve kullanımında bir eleman limiti belirtilmesine, çıkartma işlemlerinde indexlerin düzenlenmesine gerek kalmamaktadır.

List aynı türden elemanları içerir ve arraylistin getirdiği kullanım kollaylıklarını barındırır.

Dizi için bellekte belirli bir elaman sayısı için alan ayrılır ve ayrılan bu alan tek türden elemanları tutar. Ekleme ancak boş olan alana yapılabilir. Çıkartmada ise ayrılan bellek alanı geri verilmez tutulmaya devam eder sadece içeriği sıfırlanır.

Müşteri basit bir şekilde tüketici adı ve ürün sayısını tutar.

1.c Bileşik Veri Yapısı Bilgi Çıkarma

1.c.1 Kaynak Kod

//1c maddesi için yazmış olduğunuz kodları buraya ekleyiniz

```
Console.WriteLine("ArrayList içerisindeki liste sayısı : "+aList.Count);
Console.WriteLine("Listelerin ortalama elaman sayısı : " + (float)MüşteriAdı.Length/ (float)aList.Count);
Console.WriteLine(" " );
```

1.c.2 Ekran görüntüleri

```
Ali 8 , Merve 11 , Veli 16 , Gülay 5 ,
Okan 15 , Zekiye 14 , Kemal 19 , Banu 3 , İlker 18 ,
Songül 17 ,
Nuri 13 , Deniz 15 ,

ArrayList içerisindeki liste sayısı : 4
Listelerin ortalama elaman sayısı : 3
```

2.a Yığıt

2.a.1 Kaynak Kod

//Ders kitabındaki kodu güncelliyerek kullandım.

```
class Yığıt
    {
        private int maxSize; // size of stack array
        private Müşteri[] stackArray;
        private int top; // top of stack
                         //--------------------------------------------------------------
        public Yığıt(int size) // constructor
        {
            this.maxSize = size; // set array size
            this.stackArray = new Müşteri[maxSize]; // create array
            this.top = -1; // no items yet
        }
        public void push(Müşteri j) // put item on top of stack
        {
            stackArray[++top] = j; // increment top, insert item
        }

        public Müşteri pop() // take item from top of stack
        {
            return stackArray[top--]; // access item, decrement top

        }
        public Müşteri peek() // peek at top of stack
        {
            return stackArray[top];
        }
        public bool isEmpty() // true if stack is empty
        {
            return (top == -1);
        }
        public bool isFull() // true if stack is full
        {
            return (top == maxSize - 1);
        }


        public int getMaxSize()
        {
            return maxSize;
        }

    }

static void for2A(ArrayList aList,int say) {

            Yığıt müşteriler = new Yığıt(say);

            foreach (List<Müşteri> müşlis in aList)
            {
                foreach (Müşteri müş in müşlis)//arraylist içerisindeki nesnelere tek tek ulaşmak
                {
                    müşteriler.push(müş);//yığıta ekleme
                }
            }

            for (int i =0; i<müşteriler.getMaxSize();i++) {

                Console.WriteLine(müşteriler.pop().data());//yığıtın en üstündeki nesnelerin yığıttan çıkartılıp içeriğindeki bilgilerin
yazdırılması
            }
            Console.WriteLine("------");
        }
```

2.a.2 Ekran görüntüleri

```
2-a-)
Deniz 15
Nuri 13
Songül 17
İlker 18
Banu 3
Kemal 19
Zekiye 14
Okan 15
Gülay 5
Veli 16
Merve 11
Ali 8
------
```

2.b Kuyruk

2.b.1 Kaynak Kod

//Ders kitabındaki kodu güncelliyerek kullandım.

```
    class Kuyruk
    {
        private int maxSize;
        private Müşteri[] queArray;
        private int front;
        private int rear;
        public Kuyruk(int s) // constructor
        {
            maxSize = s + 1; // array is 1 cell larger
            queArray = new Müşteri[maxSize]; // than requested
            front = 0;
            rear = -1;
        }
        public void insert(Müşteri j) // put item at rear of queue
        {
            if (rear == maxSize - 1)
                rear = -1;
            queArray[++rear] = j;
        }
        public Müşteri remove() // take item from front of queue
        {
            Müşteri temp = queArray[front++];
            if (front == maxSize)
                front = 0;
            return temp;
        }
        public Müşteri peek() // peek at front of queue
        {
            return queArray[front];
        }
        public bool isEmpty() // true if queue is empty
        {
            return (rear + 1 == front || (front + maxSize - 1 == rear));
        }
        public bool isFull() // true if queue is full
        {
            return (rear + 2 == front || (front + maxSize - 2 == rear));
        }
        public int size() // (assumes queue not empty)
        {
            if (rear >= front) // contiguous sequence
                return rear - front + 1;
            else // broken sequence
                return (maxSize - front) + (rear + 1);
        }

    }
static void for2B(ArrayList aList, int say)
        {

            Kuyruk müşteriler = new Kuyruk(say);

            foreach (List<Müşteri> müşlis in aList)
            {
                foreach (Müşteri müş in müşlis)//arraylist içerisindeki nesnelere tek tek ulaşmak
                {
                    müşteriler.insert(müş);//sıraya ekleme
                }
            }
            for (int i = 0; i < say; i++)
            {

                Console.WriteLine(müşteriler.remove().data());//sıranın en sonundaki nesnelerin sıradan çıkartılıp içeriğindeki
bilgilerin yazdırılması
            }
            Console.WriteLine("------");

        }
```

2.b.2 Ekran görüntüleri

```
2-b-)
Ali 8
Merve 11
Veli 16
Gülay 5
Okan 15
Zekiye 14
Kemal 19
Banu 3
İlker 18
Songül 17
Nuri 13
Deniz 15
```

3.a Öncelikli Kuyruk Oluşturma

3.a.1 Kaynak Kod

//3-a maddesi için yazmış olduğunuz kodları buraya ekleyiniz

```
    class ÖncelikliKuyruk
    {
        private List<Müşteri> queArr;

        public ÖncelikliKuyruk()
        {
            queArr = new List<Müşteri>();

        }

        public bool bosMu()//sıra boş ise doğru döndürücek
        {
            return (0==queArr.Count);
        }
        public void ekle(Müşteri müş)// sıranın sonuna ekler
        {
            queArr.Add(müş);
        }
        public Müşteri sil()//eğer sıra boş ise adı 0 ürün sayısı 0 olan nesne döndürecektir.
        {
            Müşteri müşteriDöndür= new Müşteri("0",0);
            if (!bosMu())//ancak sıra boş değilse çıkarma işlemi yapılabilir
            {

                int indexOfBiggest = 0;
                for (int i = 0; i < queArr.Count; i++) {
                    if (queArr[indexOfBiggest].ÜrünSayısı < queArr[i].ÜrünSayısı)//en çok ürün sayısına sahip nesnenin indexinin
bulunması ve tutulması
                    {
                        indexOfBiggest = i;

                    }
                }
                müşteriDöndür = queArr[indexOfBiggest];
                queArr.RemoveAt(indexOfBiggest);

            }

            return müşteriDöndür;
        }

        public Müşteri artanSıradaSil()//eğer sıra boş ise adı 0 ürün sayısı 0 olan nesne döndürecektir.
        {
            Müşteri müşteriDöndür = new Müşteri("0", 0);
            if (!bosMu())//ancak sıra boş değilse çıkarma işlemi yapılabilir
            {

                int indexOfSmaller = 0;
                for (int i = 0; i < queArr.Count; i++)
                {
                    if (queArr[indexOfSmaller].ÜrünSayısı > queArr[i].ÜrünSayısı)//en az ürün sayısına sahip nesnenin indexinin bulunması
ve tutulması
                    {
                        indexOfSmaller = i;

                    }
                }
                müşteriDöndür = queArr[indexOfSmaller];
                queArr.RemoveAt(indexOfSmaller);

            }
```

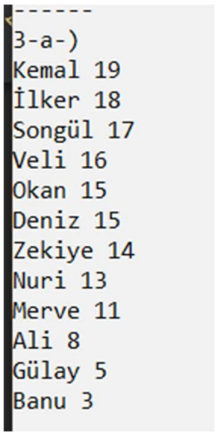
```
            return müşteriDöndür;
        }

    }

static void for3A(ArrayList aList)
        {
            ÖncelikliKuyruk müşteriler = new ÖncelikliKuyruk();
            foreach (List<Müşteri> müşlis in aList)
            {
                foreach (Müşteri müş in müşlis)//arraylist içerisindeki nesnelere tek tek ulaşmak
                {
                    müşteriler.ekle(müş);//sıraya ekleme
                }
            }

            while (!müşteriler.bosMu()){
                Console.WriteLine(müşteriler.sil().data());
            }
            Console.WriteLine("------");
        }
```

3.a.2 Ekran görüntüleri



3.b ArrayList ve Dizi altyapılarının karşılaştırılması

Bir dizi kullanılabilmesi için öncelikle sıraya alınabilecek maksimum insan sayısı belirlenmeli. Sıraya kaç kişi alınacağı belirtilmediği veya bilinmediği durumlarda dizi oluşturulamaz. Dizilerde aradan ya da baştan çıkartma işlemlerinden sonra dizide geri kalan itemlerin aradaki boşluğun kapatılması için indexlerinin 1 azaltılması gerekir dizideki eleman sayısı büyüdükçe bu işlem pahalılaşmaya başlar list kullanımında remove methotları ile listeden çıkartılan itemler için index kaydırma işlemlerini kendi methodları ile hızlı ve verimli halletmektedir.

4.a Öncelikli Kuyruk Güncelleme

//Öncelikli kuyruk nesnesinin bir methotu olarak yazdım.

```
        public Müşteri artanSıradaSil()//eğer sıra boş ise adı 0 ürün saıyısı 0 olan nesne döndürecektir.
        {
            Müşteri müşteriDöndür = new Müşteri("0", 0);
            if (!bosMu())//ancak sıra boş değilse çıkarma işlemi yapılabilir
            {

                int indexOfSmaller = 0;
                for (int i = 0; i < queArr.Count; i++)
                {
                    if (queArr[indexOfSmaller].ÜrünSayısı > queArr[i].ÜrünSayısı)//en az ürün sayısına sahip nesnenin indexinin bulunması
ve tutulması
                    {
                        indexOfSmaller = i;

                    }
                }
                müşteriDöndür = queArr[indexOfSmaller];
                queArr.RemoveAt(indexOfSmaller);

            }
```

```
            return müşteriDöndür;
        }
```

4.b Ortalama İşlem Tamamlama Süresi

4.b.1 Kaynak Kod

```
static void for4B(ArrayList aList, string[] MüşteriAdı) {
            Kuyruk müşteriler = new Kuyruk(MüşteriAdı.Length);
            ÖncelikliKuyruk öncelikliMüşterilerArtan = new ÖncelikliKuyruk();

            foreach (List<Müşteri> müşlis in aList)
            {
                foreach (Müşteri müş in müşlis)//arraylist içerisindeki nesnelere tek tek ulaşmak
                {
                    müşteriler.insert(müş);//sıraya ekleme
                    öncelikliMüşterilerArtan.ekle(müş);
                }
            }

            int öncelikliİşlemTamamlanmaSüresi = 0;
            int öncekiMüşteriTamamlamaSüresi = 0;
            Müşteri dönenMüşteri;
            while (!öncelikliMüşterilerArtan.bosMu())
            {
                dönenMüşteri = öncelikliMüşterilerArtan.artanSıradaSil();
                öncekiMüşteriTamamlamaSüresi+= dönenMüşteri.ÜrünSayısı;//her bir müşteri için toplam bekleme ve işlem süresi
                Console.WriteLine(dönenMüşteri.data()+" müşterisi için işlem bekleme süresi : "+öncekiMüşteriTamamlamaSüresi);
                öncelikliİşlemTamamlanmaSüresi += öncekiMüşteriTamamlamaSüresi;//bütün müşterilerin toplam bekleme süresi ve işlem süresi
            }
            Console.WriteLine("Önceli sıradaki müşterilerin ortalama bekleme süresi : " + (float)öncelikliİşlemTamamlanmaSüresi /
(float)MüşteriAdı.Length);

            Console.WriteLine("-----");
            öncekiMüşteriTamamlamaSüresi = 0;
            int normalKuyrukTamamlanmaSüresi = 0;
            while (!müşteriler.isEmpty())
            {
                dönenMüşteri = müşteriler.remove();
                öncekiMüşteriTamamlamaSüresi+= dönenMüşteri.ÜrünSayısı;
                Console.WriteLine(dönenMüşteri.data() + " müşterisi için işlem bekleme süresi : " + öncekiMüşteriTamamlamaSüresi);
                normalKuyrukTamamlanmaSüresi += öncekiMüşteriTamamlamaSüresi;
            }
            Console.WriteLine("kuyruktaki müşterilerin ortalama bekleme süresi : " + (float)normalKuyrukTamamlanmaSüresi /
(float)MüşteriAdı.Length);

        }
```

4.b.2 Ekran görüntüleri

```
------
Banu 3 müşterisi için işlem bekleme süresi : 3
Gülay 5 müşterisi için işlem bekleme süresi : 8
Ali 8 müşterisi için işlem bekleme süresi : 16
Merve 11 müşterisi için işlem bekleme süresi : 27
Nuri 13 müşterisi için işlem bekleme süresi : 40
Zekiye 14 müşterisi için işlem bekleme süresi : 54
Okan 15 müşterisi için işlem bekleme süresi : 69
Deniz 15 müşterisi için işlem bekleme süresi : 84
Veli 16 müşterisi için işlem bekleme süresi : 100
Songül 17 müşterisi için işlem bekleme süresi : 117
İlker 18 müşterisi için işlem bekleme süresi : 135
Kemal 19 müşterisi için işlem bekleme süresi : 154
Önceli sıradaki müşterilerin ortalama bekleme süresi : 67.25
-----
Ali 8 müşterisi için işlem bekleme süresi : 8
Merve 11 müşterisi için işlem bekleme süresi : 19
Veli 16 müşterisi için işlem bekleme süresi : 35
Gülay 5 müşterisi için işlem bekleme süresi : 40
Okan 15 müşterisi için işlem bekleme süresi : 55
Zekiye 14 müşterisi için işlem bekleme süresi : 69
Kemal 19 müşterisi için işlem bekleme süresi : 88
Banu 3 müşterisi için işlem bekleme süresi : 91
İlker 18 müşterisi için işlem bekleme süresi : 109
Songül 17 müşterisi için işlem bekleme süresi : 126
Nuri 13 müşterisi için işlem bekleme süresi : 139
Deniz 15 müşterisi için işlem bekleme süresi : 154
kuyruktaki müşterilerin ortalama bekleme süresi : 77.75
```

4.b.3 Sözel olarak karşılaştırma

FİFO tarzında insanlar sıraya girdiğinde ürün sayısı fazla olan insanlar sıranın başlarındaysalar arkada kalan az sayıda ürüne sahip insanlar fazladan sırada beklemek zorunda kalmaktadırlar ve katlanarak artmaktadır. Bu nedenle FİFO'nun ortalama bekleme süresi öncelikli kuyruğa göre fazladır.

4.c Öncelikli Kuyruk Tartışma

Her zaman kasanın sırası dolu olmayabilir, bütün insanların beklenmesi sıraya önce gelen insanlar için vakit kaybıdır. Eğer kasada işlem gören yoksa ve sıra boş ise ilk gelenin işlemlerine direk başlanması gerekir. Gerçek hayatta sıradaki insanların yer değiştirmeleri de bir süre alacağı unutulmamalıdır.

4.d Öncelikli Kuyruk Öneri

Sıra boş ise ilk gelenin işlemlerine hemen başlanmalı. Eğer kasa bir müşterinin işlemlerini yapıyorsa ve sırada birden fazla müşteri varsa ürün sayısı azdan fazlaya doğru kendi aralarında sıralanmaları gerekmektedir. Eğer sıra varsa ve kasadaki müşterinin işlemleri bittiği anda hemen sıradaki müşterinin işlemine başlanmalıdır. Sıraya yeni bir müşterinin geldiği anda kasanın işlemi sonlanırsa ve bu yeni gelen müşteri sıranın en az ürününe sahipse ilk önce bu müşterinin işlemi tamamlanmalıdır.

Özdeğerlendirme Tablosu

Özdeğerlendirme Tablosu

Proje 2 Maddeleri	**Puan**	**Tahmini Not**	**Açıklama**
1 a) A4 Ön çalışma	**20**	**20**	**Yapıldı**
1 b) Kaynak kod, ekran görüntüsü, veri yapısının elemanlarının listelenmesi	**20**	**20**	**Yapıldı**
1 c) Kaynak kodlar, Liste sayısı, listelerdeki ortalama eleman sayısı	**5**	**5**	**Yapıldı**
2 a) Yığıt kaynak kod ve ekran görüntüleri	**5**	**5**	**Yapıldı**
2 b) Kuyruk kaynak kod ve ekran görüntüleri	**5**	**5**	**Yapıldı**
3 a) Öncelikli Kuyruk kod ve ekran görüntüleri	**10**	**10**	**Yapıldı**
3 b) ArrayList ve Dizi altyapılarının karşılaştırılması	**5**	**5**	**Yapıldı**
4) Kod, sonuçlar tablosu, ekran görüntüleri ve soruların cevapları.	**20**	**20**	**Yapıldı**
5) Özdeğerlendirme Tablosu	**10**	**10**	**Yapıldı**
Toplam	**100**	**100**	

Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı (ve nelerin yapılmadığı / yapılamadığı) yazılmalıdır. Tahmini not kısmına da ilgili maddeden kaç almayı beklediğinizi yazmalısınız.