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

10/01/2021

PREPARED BY

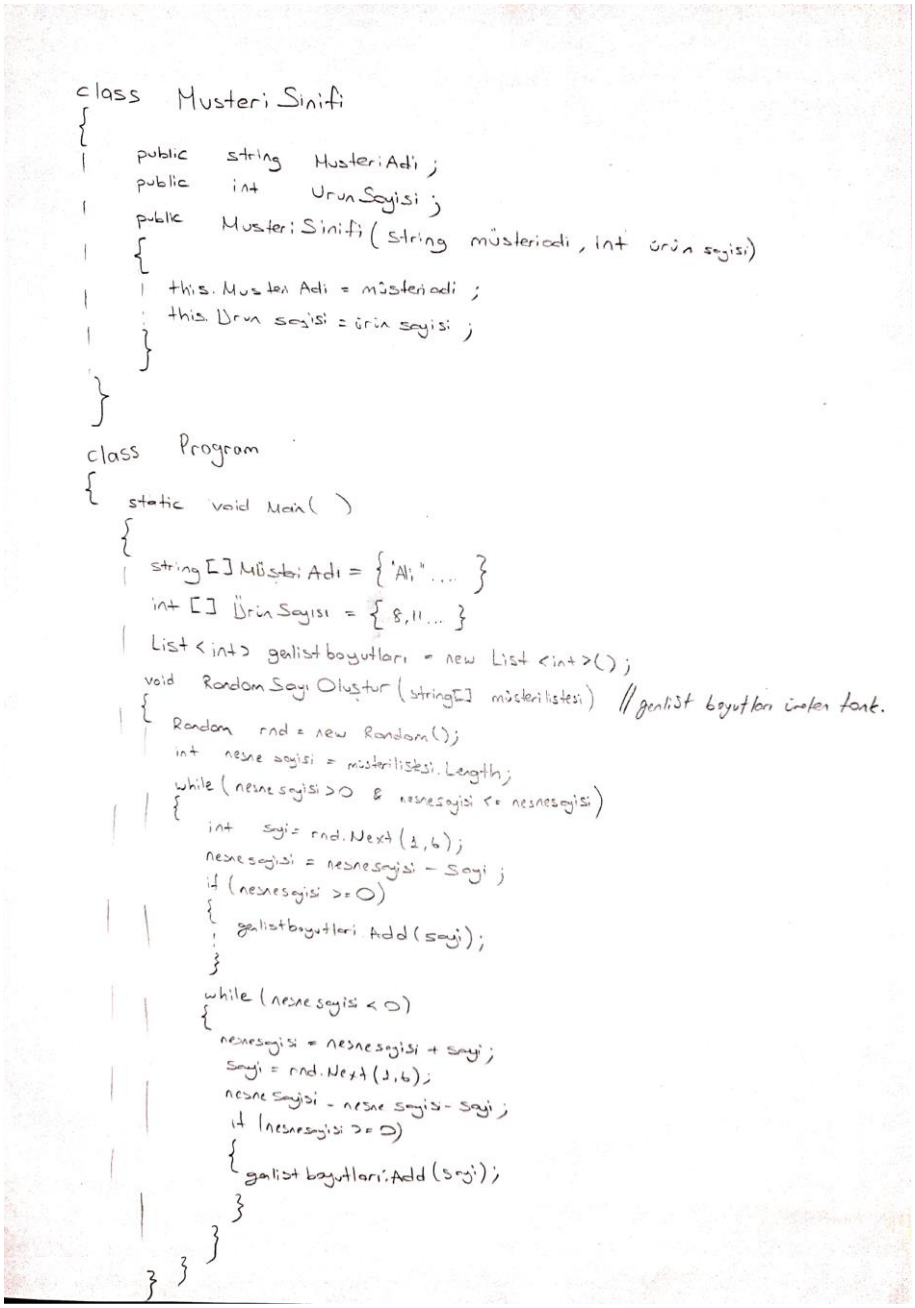
05180000113, Simge Merve Yaşbay

İçindekiler

LİSTE, YIĞIT, KUYRUK ve ÖNCELİKLİ KUYRUK VERİ YAPILARI

Visual Studio , 16.8.0 , C#

1.a Bileşik Veri Yapısı için Ön Çalışma



```
class Musteri Sinifi
{
    public string MusteriAdi ;
    public int UrunSayisi ;
    public Musteri Sinifi ( string müsteriadi , int ürün sayisi)
    {
        this.Musteri Adi = müsteriadi ;
        this.Urun sayisi = ürün sayisi ;
    }
}
class Program
{
    static void Main( )
    {
        string [] Müsteri Adi = { "Ali," ... }
        int [] Ürün Sayısı = { 8,11 ... }
        List < int > galist boyutları = new List <int >() ;
        void RandomSayı Oluştur (string[] müsterilistesi) // galist boyutları üreten fonk.
        {
            Random rnd = new Random();
            int nesne sayisi = müsterilistesi.Length;
            while ( nesnesayisi >0 & nesnesayisi <= nesnesayisi)
            {
                int sayi = rnd.Next (1,6);
                nesnesayisi = nesnesayisi - sayi ;
                if (nesnesayisi >=0)
                {
                galistboyutları Add (sayi);
                }
                while (nesnesayisi < 0)
                {
                nesnesayisi = nesnesayisi + sayi ;
                sayi = rnd.Next (1,6);
                nesne Sayisi - nesne sayisi - sayi ;
                if (nesnesayisi >= 0)
                {
                galist boyutları.Add (sayi);
                }
            }
        }
    }
}
```

(Main)

```
Random Sayı Oluştur (MüşteriAdi);
Arraylist arrayliste = new Arraylist();
int sayac = 0;
MusteriSiniti musteri siniti;
List<MusteriSiniti> genericliste;
while (sayac < MüşteriAdi.Length)
{
  for (int j=0; j<genlistbayutlari.Count ; j++)
  {
    genericliste = new List<MusteriSiniti>();
    int genericlisteLength = (genlistbayutlari)[j];
    for (int i=0 ; i < genericlisteLength ; i++)
    {
      musteri siniti = new MusteriSiniti(MüşteriAdi[sayac], ÜrünSayısı[sayac];
      generic liste.Add(musterisiniti);
      sayac ++;
      if (sayac == MüşteriAdi.Length)
        break;
      }
    arrayliste.Add(genericliste);
  }
}
```

1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma

1.b.1 Kaynak Kod

```
class Program
    {
        static void Main(string[] args)
        {
            string[] MüşteriAdı = { "Ali", "Merve", "Veli", "Gülay", "Okan", "Zekiye",
"Kemal", "Banu", "İlker", "Songül", "Nuri", "Deniz" };

            int[] ÜrünSayısı = { 8, 11, 16, 5, 15, 14, 19, 3, 18, 17, 13, 15 };
            List<int> genlistboyutlari = new List<int>(); //genericlistboyutlarını
saklamak için liste
            void RandomSayıOluştur(string[] müsterilistesi, int[] ürünlistesi)
//genericlist boyutları için gerekli random sayıları üretip bu sayıları saklayan
fonksiyon yazdım
            {

                Random rnd = new Random();
                int nesnesayisi = müsterilistesi.Length;   //random sayıların toplamı
nesne sayısı kadar olmalı
                while (nesnesayisi > 0 & nesnesayisi <= nesnesayisi)  //nesne sayısı 0
lanana kadar devam edecek
                {
                    int sayi = rnd.Next(1, 6);
                    nesnesayisi = nesnesayisi - sayi;  //random sayı ürettikten sonra
nesne sayısından çıkartıyoruz
                    if (nesnesayisi >= 0)   //0 dan büyükse doğru ilerliyoruzdur
                    {
                        genlistboyutlari.Add(sayi);  //random sayıyı bir listeye
atıyoruz
                    }
                    while (nesnesayisi < 0)    //nesne sayısı negatif olmaması gerek
bu yüzden burada nesne sayısını sıfırlayana kadar tekrar random sayılar üretiyoruz
                    {
                        nesnesayisi = nesnesayisi + sayi;
                        sayi = rnd.Next(1, 6);
                        nesnesayisi = nesnesayisi - sayi;
                        if (nesnesayisi >= 0)
                        {
                            genlistboyutlari.Add(sayi);

                        }
                    }
                }

            }
            RandomSayıOluştur(MüşteriAdı, ÜrünSayısı);  //fonksiyonu verilen listeler
için çağırdım

            ArrayList arrayliste = new ArrayList();
            int sayac = 0;
            MusteriSinifi musterisinifi;
            List<MusteriSinifi> genericliste;
            while (sayac < MüşteriAdı.Length)    //müşteriler bitene kadar genericlist
boyutlarına göre müşterileri gruplandırıp array listeye ekliyoruz
            {
                for (int j = 0; j < genlistboyutlari.Count; j++)
                {
                    genericliste = new List<MusteriSinifi>();
                    int genericlisteLength = (genlistboyutlari)[j];
```

```
                    for (int i = 0; i < genericlisteLength; i++)
                    {

                        musterisinifi = new MusteriSinifi(MüşteriAdı[sayac],
ÜrünSayısı[sayac]);
                        genericliste.Add(musterisinifi);
                        sayac++;
                        if (sayac == MüşteriAdı.Length)
                            break;

                    }
                    arrayliste.Add(genericliste);


                }


            }
            foreach (List<MusteriSinifi> item in arrayliste)  //yazdırma
            {
                foreach (MusteriSinifi item1 in item)
                {

                    Console.WriteLine("Müşteri Adı:\t" + item1.MusteriAdi + "\t Ürün
Sayısı: " + item1.UrunSayisi);

                }
                Console.WriteLine();
            }
```

1.b.2 Ekran görüntüleri

```
Müşteri Adı:    Ali     Ürün Sayısı: 8
Müşteri Adı:    Merve   Ürün Sayısı: 11
Müşteri Adı:    Veli    Ürün Sayısı: 16
Müşteri Adı:    Gülay   Ürün Sayısı: 5
Müşteri Adı:    Okan    Ürün Sayısı: 15

Müşteri Adı:    Zekiye  Ürün Sayısı: 14

Müşteri Adı:    Kemal   Ürün Sayısı: 19
Müşteri Adı:    Banu    Ürün Sayısı: 3

Müşteri Adı:    İlker   Ürün Sayısı: 18
Müşteri Adı:    Songül  Ürün Sayısı: 17

Müşteri Adı:    Nuri    Ürün Sayısı: 13

Müşteri Adı:    Deniz   Ürün Sayısı: 15
```

1.b.3 Veri Yapıları ve Açıklama

Bu bölümde List yapılarından GenericList ve ArrayList yapılarını kullandık.GenericList yapısı yukarıda bir satır boşlukla ayrılmış olan grupları tutuyor.ArrayList ise GenericList'lerden oluşan liste yapısıdır.Her GenericList ArrayList'in bir elemanıdır.

1.c Bileşik Veri Yapısı Bilgi Çıkarma

1.c.1 Kaynak Kod

```
            double listesayisi = arrayliste.Count;  //arrayliste sayısı
            Console.WriteLine("ArrayListe sayısı : " + listesayisi);
            Console.WriteLine("Listelerin Ortalama Eleman Sayısı: " +
(MüşteriAdı.Length / listesayisi)); //Her arrayde ortalama eleman sayısı
            Console.WriteLine();
```

1.c.2 Ekran görüntüleri

(Yukarıdaki örnek için geçerli sayılar)

```
ArrayListe sayısı : 6
Listelerin Ortalama Eleman Sayısı: 2
```

2.a Yığıt

2.a.1 Kaynak Kod

```
class StackX   //STACK sınıfı
    {
        private int maxsize;
        private MusteriSinifi[] stackarray;
        private int top;
        public StackX(int s)
        {
            maxsize = s;
            stackarray = new MusteriSinifi[maxsize];
            top = -1;
        }
        public void push(MusteriSinifi j)
        {
            stackarray[++top] = j;
        }
        public MusteriSinifi pop()
        {
            return stackarray[top--];
        }
        public MusteriSinifi peek()
        {
            return stackarray[top];
        }
        public Boolean isEmpty()
        {
            return (top == -1);
        }
        public Boolean isFull()
        {
            return (top == maxsize - 1);
        }
    }

//-------------YIĞIT CLASSINI ÇAĞIRIP YAZDIRMA(main'in içerisinde)----------------//


            Console.WriteLine("Yığıt Yazımı :");
            StackX stack = new StackX(MüşteriAdı.Length);
            foreach (List<MusteriSinifi> a in arrayliste) //arraydeki bilgilerimizi
yığıta ekliyoruz
```

```
            {
                foreach (MusteriSinifi b in a)
                {
                    stack.push(b);
                }

            }
            while (!stack.isEmpty())  //yığıt bitene kadar yığıttaki elemanları
yazdırıyoruz
            {
                MusteriSinifi value = stack.pop();
                Console.WriteLine("Müşteri Adı:\t " + value.MusteriAdi + "\t Ürün
Sayısı: " + value.UrunSayisi);
            }
```

2.a.2 Ekran görüntüleri

2.b Kuyruk

2.b.1 Kaynak Kod

```
class Queue //KUYRUK sınıfı
    {
        private int maxsize;
        private MusteriSinifi[] queArray;
        private int front;
        private int rear;
        private int nitems;
        public Queue(int s) //constructor
        {
            maxsize = s;
            queArray = new MusteriSinifi[maxsize];
            front = 0;
            rear = -1;
            nitems = 0;
        }
        public void insert(MusteriSinifi j) //itemi kuyruğun arkasına eklemek
        {
            if (rear == maxsize - 1)
            {
                rear = -1;
            }
            queArray[++rear] = j;
            nitems++; //item sayısı arttır
```

```
        }
        public MusteriSinifi remove() //kuyruğun başındaki itemi çıkarmak
        {
            MusteriSinifi temp = queArray[front++];
            if (front == maxsize)
            {
                front = 0;

            }
            nitems--; //item sayısı azalt
            return temp;
        }
        public MusteriSinifi peekFront() //ulaşma
        {
            return queArray[front];
        }
        public Boolean isEmpty()
        {
            return (nitems == 0);
        }
        public Boolean isFull()
        {
            return (nitems == maxsize);
        }
        public int size() //item sayısı
        {
            return nitems;
        }
    }

//----------------KUYRUK CLASSINI ÇAĞIRIP YAZDIRMA(main'in içerisinde)-------------//

            Console.WriteLine("Kuyruk Yazımı:");
            Queue theQueue = new Queue(MüşteriAdı.Length);
            Queue theQueuekopya = new Queue(MüşteriAdı.Length); //Ortalama işlem
süresi bulup yazdırma için kopya bir Queue tanımladım
            foreach (List<MusteriSinifi> c in arrayliste) //arraydaki bilgilerimizi
kuyruğa ekliyoruz
            {
                foreach (MusteriSinifi d in c)
                {
                    theQueue.insert(d);
                    theQueuekopya.insert(d);
                }
            }
            while (!theQueue.isEmpty()) //bilgileri çıkartıp yazdırma
            {
                MusteriSinifi bilgiler = theQueue.remove();
                Console.WriteLine("Müşteri Adı:\t" + bilgiler.MusteriAdi + "\t Ürün
Sayısı: " + bilgiler.UrunSayisi);
            }
```

2.b.2 Ekran görüntüleri

```
Kuyruk Yazımı:
Müşteri Adı:    Ali      Ürün Sayısı: 8
Müşteri Adı:    Merve    Ürün Sayısı: 11
Müşteri Adı:    Veli     Ürün Sayısı: 16
Müşteri Adı:    Gülay    Ürün Sayısı: 5
Müşteri Adı:    Okan     Ürün Sayısı: 15
Müşteri Adı:    Zekiye   Ürün Sayısı: 14
Müşteri Adı:    Kemal    Ürün Sayısı: 19
Müşteri Adı:    Banu     Ürün Sayısı: 3
Müşteri Adı:    İlker    Ürün Sayısı: 18
Müşteri Adı:    Songül   Ürün Sayısı: 17
Müşteri Adı:    Nuri     Ürün Sayısı: 13
Müşteri Adı:    Deniz    Ürün Sayısı: 15
```

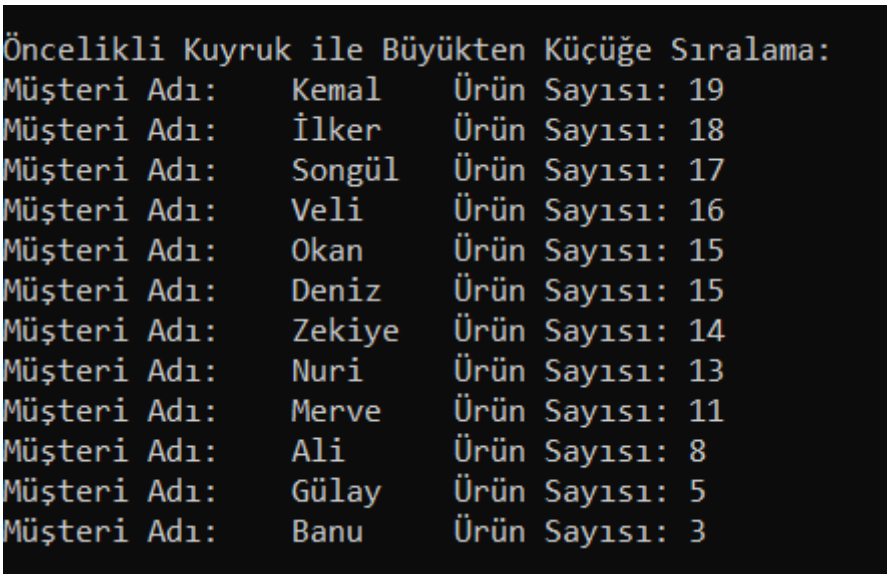
3.a Öncelikli Kuyruk Oluşturma

3.a.1 Kaynak Kod

```
class OncelikliQueue //ONCELİKLİ KUYRUK sınıfı
    {
        public List<MusteriSinifi> oncelikliqueuelist;
        public OncelikliQueue() //constructor
        {
            oncelikliqueuelist = new List<MusteriSinifi>();
        }
        public void ekle(MusteriSinifi j)
        {
            oncelikliqueuelist.Add(j);

        }
        public MusteriSinifi büyüktenküçüğesil() //öncelikli kuyrukta büyükten küçüğe
sıralaması için gerekli method
        {
            MusteriSinifi max = oncelikliqueuelist.ElementAt(0); //ilk elemanı max
alıyor
            int maxindex = 0;
            for (int i = 1; i < oncelikliqueuelist.Count; ++i) //burada listedeki
bütün elemanları karşılaştıracak
            {
                if (oncelikliqueuelist.ElementAt(i).UrunSayisi > max.UrunSayisi)
//gelen eleman öncekinden büyük ise buraya girip max değere atıyor
                {
                    max = oncelikliqueuelist.ElementAt(i);
                    maxindex = i;
                }
            }
            oncelikliqueuelist.RemoveAt(maxindex); //max değer hangi indeksde ise o
indeksdeki elemanı çıkartıyor ve döndürüyor
            return max;
        }
        public MusteriSinifi kucuktenbüyüğesil() //öncelikli kuyrukta küçükten büyüğe
sıralaması için gerekli method
        {
            MusteriSinifi min = oncelikliqueuelist.ElementAt(0); //ilk elemanı min
alıyor
            int minindex = 0;
```

```
            for (int j = 1; j < oncelikliqueuelist.Count; ++j) //listedeki bütün
elemanları karşılaştıracak
            {
                if (oncelikliqueuelist.ElementAt(j).UrunSayisi < min.UrunSayisi)
//gelen eleman öncekinden küçük ise buraya girip min değere atayacak
                {
                    min = oncelikliqueuelist.ElementAt(j);
                    minindex = j;
                }
            }
            oncelikliqueuelist.RemoveAt(minindex); //min değer hangi indeksde ise o
indeksdeki elemanı çıkartıyor ve döndürüyor
            return min;
        }
        public Boolean bosMu()
        {
            return oncelikliqueuelist.Count == 0;
        }
    }

//------ÖNCELİKLİ KUYRUK İLE SIRALAYIP YAZDIRMA İŞLEMLERİ(main'in içerisinde)-------//


            OncelikliQueue onceliklikuyrukbuyuktenkucuge = new OncelikliQueue();
//büyükten küçüğe kuyruk yapısı
            OncelikliQueue onceliklikuyrukkucuktenbuyuge = new OncelikliQueue();
//küçükten büyüğe kuyruk yapısı
            OncelikliQueue onceliklikuyrukkopya = new OncelikliQueue();
            foreach (List<MusteriSinifi> e in arrayliste)
            {
                foreach (MusteriSinifi f in e)
                {
                    onceliklikuyrukbuyuktenkucuge.ekle(f);  //arraydaki müşteri
bilgilerini öncelikli kuyruklara atıyor
                    onceliklikuyrukkucuktenbuyuge.ekle(f);

                }
            }
            Console.WriteLine("Öncelikli Kuyruk ile Büyükten Küçüğe Sıralama:");
            while (!onceliklikuyrukbuyuktenkucuge.bosMu()) //öncelikli kuyruktaki
elemanlar bitene kadar devam ediyor
            {
                MusteriSinifi degerler =
onceliklikuyrukbuyuktenkucuge.büyüktenküçüğesil();    //öncelikli kuyruktaki bütün
elemanları karşılaştırıp büyükten küçüğe sıralıyor
                onceliklikuyrukkopya.ekle(degerler);
                Console.WriteLine("Müşteri Adı:\t" + degerler.MusteriAdi + "\t Ürün
Sayısı: " + degerler.UrunSayisi);

            }
```

3.a.2 Ekran görüntüleri



3.b ArrayList ve Dizi altyapılarının karşılaştırılması

List yapısı dinamiktir fakat dizi yapısı statik ve süreklidir.Ayrıca bu örnekte dizi kullanılsaydı MüşteriAdı ve ÜrünSayısı dizileri ayrı ayrı çağırılmalıydı ve bu da aslında projenin istediğimiz pratikliğini sağlayamazdı.Int ve String tipinde iki dizi için işlemler yapılmak zorunda kalınır bu da daha çok iş yüküne yol açardı.List yapısında MüşteriAdı ve ÜrünSayısı dataları beraber tutuluyor.Bu da bize kolaylık sağlıyor.

4.a Öncelikli Kuyruk Güncelleme

(Bu kısım Öncelikli Kuyruk Sınıfı içerisinde bir methottur.Yukarıda 3.a.1 kısmında da eklenmiştir.)

```
        public MusteriSinifi kucuktenbüyüğesil() //öncelikli kuyrukta küçükten büyüğe
sıralaması için gerekli method
        {
            MusteriSinifi min = oncelikliqueuelist.ElementAt(0); //ilk elemanı min
alıyor
            int minindex = 0;
            for (int j = 1; j < oncelikliqueuelist.Count; ++j) //listedeki bütün
elemanları karşılaştıracak
            {
                if (oncelikliqueuelist.ElementAt(j).UrunSayisi < min.UrunSayisi)
//gelen eleman öncekinden küçük ise buraya girip min değere atayacak
                {
                    min = oncelikliqueuelist.ElementAt(j);
                    minindex = j;
                }
            }
            oncelikliqueuelist.RemoveAt(minindex); //min değer hangi indeksde ise o
indeksdeki elemanı çıkartıyor ve döndürüyor
            return min;
        }
```

4.b Ortalama İşlem Tamamlama Süresi

4.b.1 Kaynak Kod

```
//-------İşlem Süresi Yazdırma ve Ortalama İşlem Süresi Bulma(main'in içerisinde)---//
            double tekkasaiçintoplamişlemsüresi1 = 0;
            double müşteriişlemsüresi1 = 0;
            Console.WriteLine();
            Console.WriteLine("Kuyruk Kullanarak Hesaplanan İşlem Süreleri :");
            while (!theQueuekopya.isEmpty())
            {
                MusteriSinifi müşteriveişlemsüresi = theQueuekopya.remove();
                müşteriişlemsüresi1 = müşteriişlemsüresi1 +
müşteriveişlemsüresi.UrunSayisi;
                tekkasaiçintoplamişlemsüresi1 = tekkasaiçintoplamişlemsüresi1 +
müşteriişlemsüresi1;
                Console.WriteLine("Müşteri Adı:\t" + müşteriveişlemsüresi.MusteriAdi +
"\t İşlem Tamamlama Süresi: " + müşteriişlemsüresi1);

            }
            Console.WriteLine();
            Console.WriteLine("Tek Kasa için Ortalama İşlem Tamamlama Süresi(Kuyruk) :
" + tekkasaiçintoplamişlemsüresi1 / MüşteriAdı.Length);
            Console.WriteLine();

            Console.WriteLine("Öncelikli Kuyruk Kullanarak Hesaplanan İşlem
Süreleri:");
            double müşteriişlemsüresi2 = 0;
            double tekkasaiçintoplamişlemsüresi2 = 0;
            while (!onceliklikuyrukkopya.bosMu())
            {
                MusteriSinifi müşteriveişlemsüresi2 =
onceliklikuyrukkopya.kucuktenbüyüğesil();
                müşteriişlemsüresi2 = müşteriişlemsüresi2 +
müşteriveişlemsüresi2.UrunSayisi;
                tekkasaiçintoplamişlemsüresi2 = tekkasaiçintoplamişlemsüresi2 +
müşteriişlemsüresi2;
                Console.WriteLine("Müşteri Adı:\t" + müşteriveişlemsüresi2.MusteriAdi
+ "\t İşlem Tamamlama Süresi: " + müşteriişlemsüresi2);
            }
            Console.WriteLine();
            Console.WriteLine("Tek Kasa İçin Ortalama İşlem Tamamlama Süresi(PQ) : "+
tekkasaiçintoplamişlemsüresi2/MüşteriAdı.Length);
            Console.ReadKey();
```

4.b.2 Ekran görüntüleri

```
Kuyruk Kullanarak Hesaplanan İşlem Süreleri :
Müşteri Adı:    Ali      İşlem Tamamlama Süresi: 8
Müşteri Adı:    Merve    İşlem Tamamlama Süresi: 19
Müşteri Adı:    Veli     İşlem Tamamlama Süresi: 35
Müşteri Adı:    Gülay    İşlem Tamamlama Süresi: 40
Müşteri Adı:    Okan     İşlem Tamamlama Süresi: 55
Müşteri Adı:    Zekiye   İşlem Tamamlama Süresi: 69
Müşteri Adı:    Kemal    İşlem Tamamlama Süresi: 88
Müşteri Adı:    Banu     İşlem Tamamlama Süresi: 91
Müşteri Adı:    İlker    İşlem Tamamlama Süresi: 109
Müşteri Adı:    Songül   İşlem Tamamlama Süresi: 126
Müşteri Adı:    Nuri     İşlem Tamamlama Süresi: 139
Müşteri Adı:    Deniz    İşlem Tamamlama Süresi: 154

Tek Kasa için Ortalama İşlem Tamamlama Süresi(Kuyruk) : 77,75

Öncelikli Kuyruk Kullanarak Hesaplanan İşlem Süreleri:
Müşteri Adı:    Banu     İşlem Tamamlama Süresi: 3
Müşteri Adı:    Gülay    İşlem Tamamlama Süresi: 8
Müşteri Adı:    Ali      İşlem Tamamlama Süresi: 16
Müşteri Adı:    Merve    İşlem Tamamlama Süresi: 27
Müşteri Adı:    Nuri     İşlem Tamamlama Süresi: 40
Müşteri Adı:    Zekiye   İşlem Tamamlama Süresi: 54
Müşteri Adı:    Okan     İşlem Tamamlama Süresi: 69
Müşteri Adı:    Deniz    İşlem Tamamlama Süresi: 84
Müşteri Adı:    Veli     İşlem Tamamlama Süresi: 100
Müşteri Adı:    Songül   İşlem Tamamlama Süresi: 117
Müşteri Adı:    İlker    İşlem Tamamlama Süresi: 135
Müşteri Adı:    Kemal    İşlem Tamamlama Süresi: 154

Tek Kasa İçin Ortalama İşlem Tamamlama Süresi(PQ) : 67,25
```

4.b.3 Sözel olarak karşılaştırma

4.c Öncelikli Kuyruk Tartışma

Yukarıda görüldüğü üzere ÖncelikliKuyruk ile ortalama işlem tamamlama süresi 67.25 iken Kuyruk ile ortalama işlem tamamlama süresi 77.75 ‘dir.ÖncelikliKuyruk bu konuda daha verimlidir.Ancak ÖncelikliKuyruk ‘un küçükten büyüğe artan sırada kullanılması gerekir.Aksi durumda verim sağlanamaz.

4.d Öncelikli Kuyruk Öneri

Başka bir yöntem olarak müşterileri ikili gruplar halinde düşünürsek grupların ilk üyeleri sırayla işlem süreleri kısa olanlar,ikinci üyeleri de sırayla işlem süresi uzun olanlar şeklinde yaparsak verim elde edilebilir.Özetle küçükten büyüğe sıralanan listede bir yukarıdan bir aşağıdan eleman çekerek. Örneğin bu projede kullandığımız örnek için müşterilerin gelme sırası : Banu(3) ,Kemal(19),Gülay(5),İlker(18)…

Özdeğerlendirme Tablosu

Özdeğerlendirme Tablosu

Proje 2 Maddeleri	**Puan**	**Tahmini Not**	**Açıklama**
1 a) A4 Ön çalışma	**20**	**20**	Random genlistboyutları üreten fonksiyon ve arraylisti oluşturan döngüleri yazdım.
1 b) Kaynak kod, ekran görüntüsü, veri yapısının elemanlarının listelenmesi	**20**	**20**	Arraylisti başarılı şekilde ürettim,gerekenleri rapora ekledim.
1 c) Kaynak kodlar, Liste sayısı, listelerdeki ortalama eleman sayısı	**5**	**5**	İstenilen bilgiler hesaplandı ve yazdırıldı.Gerekenleri rapora ekledim.
2 a) Yığıt kaynak kod ve ekran görüntüleri	**5**	**5**	Yığıt sınıfı oluşturuldu.Main'in içinde bu sınıfın yapısında yığıt tanımladım.Gerektiği şekilde yazdırdım.Rapora ekledim.
2 b) Kuyruk kaynak kod ve ekran görüntüleri	**5**	**5**	Kuyruk sınıfını oluşturdum.Main'in içinde bu sınıf yapısında kuyruk tanımladım.Gerekenleri rapora ekledim.
3 a) Öncelikli Kuyruk kod ve ekran görüntüleri	**10**	**10**	ÖncelikliKuyruk sınıfını ve methotlarını oluşturdum.Main'in içinde bu yapıda önceliklikuyruk tanımladım.Yazdırdım.Gerekenleri rapora ekledim.
3 b) ArrayList ve Dizi altyapılarının karşılaştırılması	**5**	**5**	İstenilen şekilde yapıldı.
4) Kod, sonuçlar tablosu, ekran görüntüleri ve soruların cevapları.	**20**	**20**	Oluşturduğum Kuyruk ve ÖncelikliKuyruk yapısında kopyakuyruk ve kopyaÖncelikliKuyruk değişkenleri ile bilgileri tuttum.Bu kısımda

			onları kullanarak hesaplamaları yaptım ve yazdırdım.
5) Özdeğerlendirme Tablosu	**10**	**10**	İstenilen şekilde yapıldı.
Toplam	**100**	**100**	