

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**204 DATA STRUCTURES (3+1)**

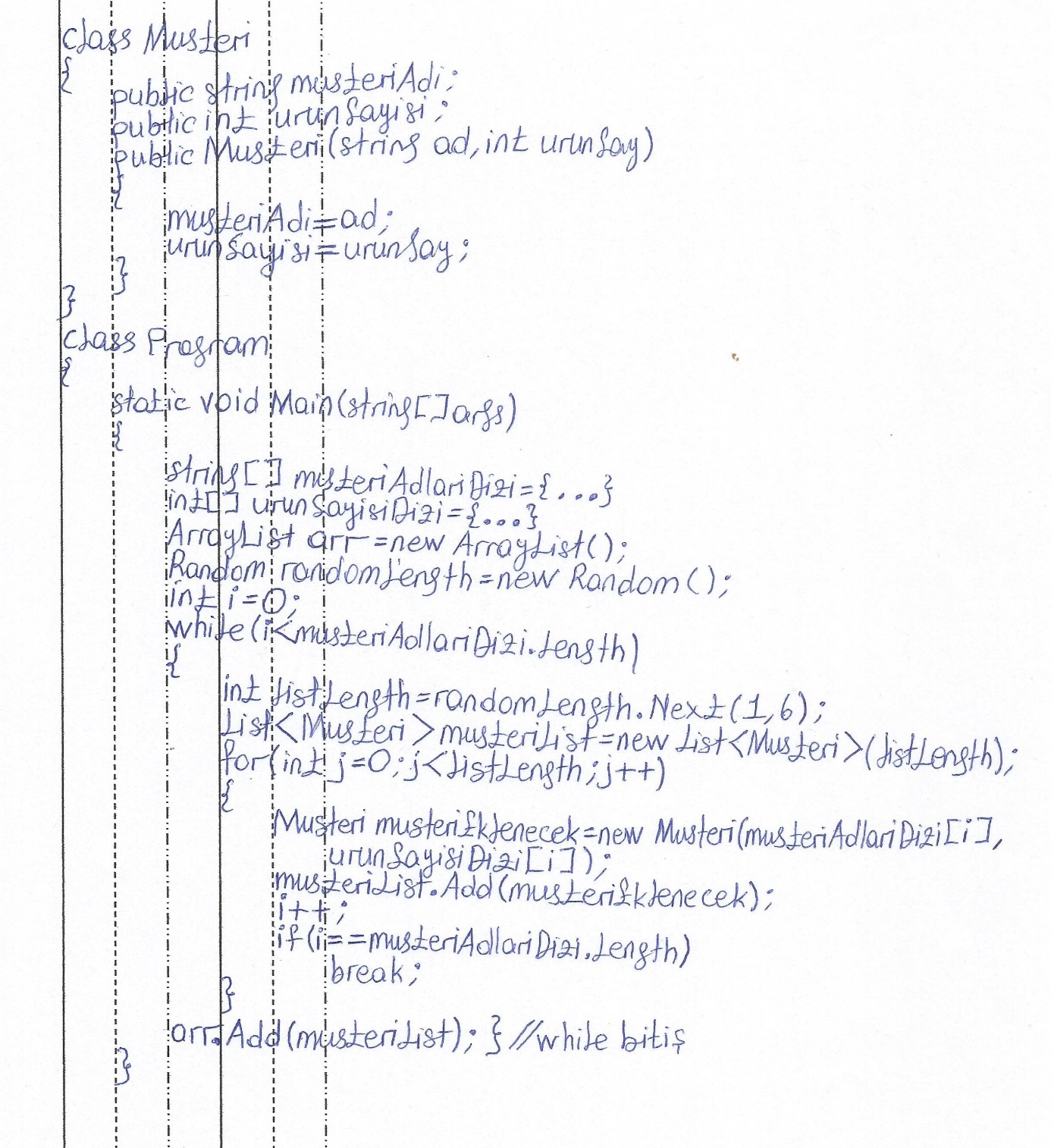
**2020–2021 FALL SEMESTER**

**PROJECT-2 REPORT**

**PREPARED BY**

05190000086,Ertuğrul Kantar

**1 a) Bileşik Veri Yapısı Oluşturma ve Eleman Ekleme Ön Çalışması ve Metodu**

****

**1 b) Kaynak kod ve Ekran Görüntüsü**

Kaynak Kod

class Musteri

{

public string musteriAdi;

public int urunSayisi;

public Musteri(string musteriAdi,int urunSayisi)

{

this.musteriAdi = musteriAdi;

this.urunSayisi = urunSayisi;

}

public void yazdir()

{

Console.WriteLine("Musteri Adi:: " + musteriAdi + " || Urun Sayisi:: " + urunSayisi);

}

}

Main

string[] musteriAdiDizi = {"Ali","Merve","Veli","Gülay","Okan","Zekiye","Kemal","Banu","İlker","Songül","Nuri","Deniz" };

int[] urunSayisiDizi = {8,11,16,5,15,14,19,3,18,17,13,15};

ArrayList genericListeleriTutanArrayList = new ArrayList();

Random listRandomLength = new Random();

int sayac = 0;

while (sayac < musteriAdiDizi.Length)

{

int listLength = listRandomLength.Next(1, 6);

List<Musteri> musteriGenericList = new List<Musteri>(listLength);

for(int i = 0; i < listLength; i++)

{

Musteri musteriEklenecek = new Musteri(musteriAdiDizi[sayac], urunSayisiDizi[sayac]);

musteriGenericList.Add(musteriEklenecek);

sayac++;

if (sayac == musteriAdiDizi.Length)

break;

}

genericListeleriTutanArrayList.Add(musteriGenericList);

}

Console.WriteLine("#genericListeleriTutanArrayList ve Icindekiler Yazdiriliyor...\n\*\*\*\*\*Parantez icindeki sayilar kacinci" +

" listede bulunuldugunu belirtir.\*\*\*\*\*");

for(int j = 0; j < genericListeleriTutanArrayList.Count; j++)

{

List<Musteri> yazdirilacakListe = (List<Musteri>)genericListeleriTutanArrayList[j];

Console.WriteLine("(" + (j + 1) + ")");

foreach (Musteri el in yazdirilacakListe)

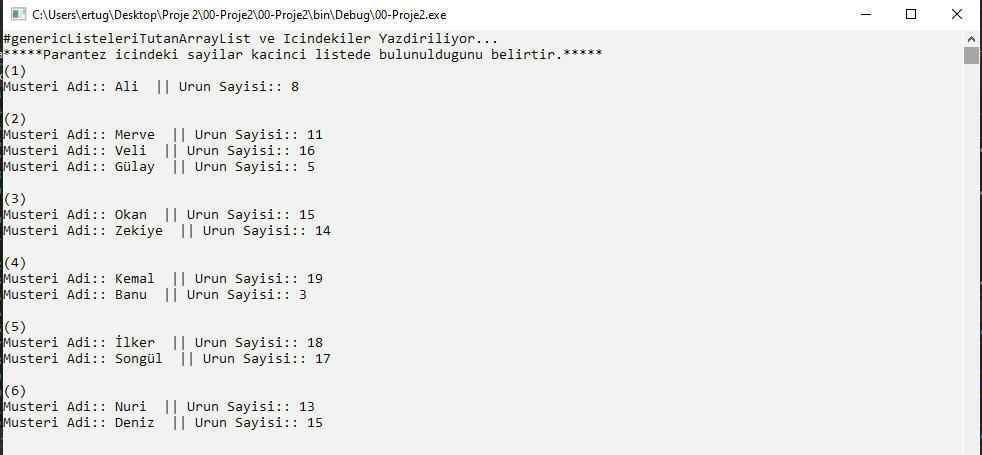
{

el.yazdir();

}

Console.WriteLine();

Ekran Görüntüsü



**1 c) Kaynak Kodlar, Liste Sayısı ve Listelerdeki Ortalama Eleman Sayısının Ekran Görüntüleri**

Kaynak Kod

Main

float musteriAdlariDizisiLengthFloat = musteriAdiDizi.Length;

float arrayListLength = genericListeleriTutanArrayList.Count;

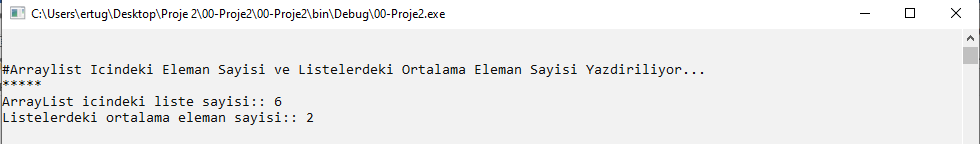
float ortalamaListeElemanSayisi = (musteriAdlariDizisiLengthFloat) / (arrayListLength);

Console.WriteLine("\n#Arraylist Icindeki Eleman Sayisi ve Listelerdeki Ortalama Eleman Sayisi Yazdiriliyor...\n\*\*\*\*\*");

Console.WriteLine("ArrayList icindeki liste sayisi:: " + arrayListLength);

Console.WriteLine("Listelerdeki ortalama eleman sayisi:: " + ortalamaListeElemanSayisi);

Ekran Görüntüsü



**2 a) Yığıt Kaynak Kod ve Ekran Görüntüleri**

Kaynak Kod

class StackM

{

public int maxSize;

public Musteri[] stackArray;

public int top;

public StackM(int boyut)

{

maxSize = boyut;

stackArray = new Musteri[maxSize];

top = -1;

}

public void push(Musteri musteriNesnesi)

{

stackArray[++top] = musteriNesnesi;

}

public Musteri pop()

{

return stackArray[top--];

}

public bool isEmpty()

{

return (top == -1);

}

}

Main

int musteriAdlariDizisiLengthInt = (int)musteriAdlariDizisiLengthFloat;

StackM musteriStack = new StackM(musteriAdlariDizisiLengthInt);

for(int say1 = 0; say1 < musteriAdlariDizisiLengthInt; say1++)

{

Musteri stackeEklenecekMusteriNesnesi = new Musteri(musteriAdiDizi[say1],urunSayisiDizi[say1]);

musteriStack.push(stackeEklenecekMusteriNesnesi);

}

Console.WriteLine("\n\n#musteriStack Yazdiriliyor...\n\*\*\*\*\*");

while(musteriStack.isEmpty()==false)

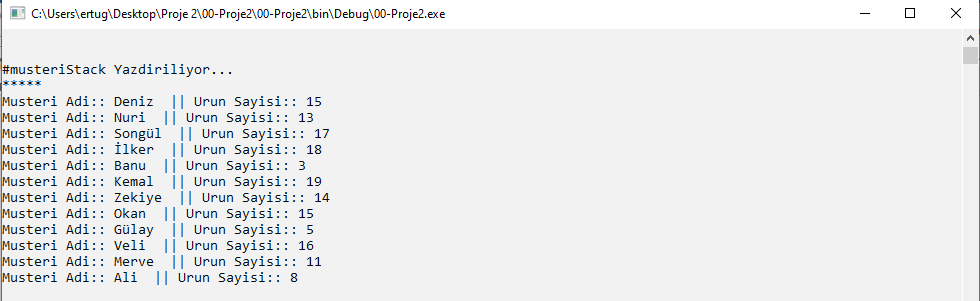
{

Musteri temp1 = musteriStack.pop();

temp1.yazdir();

}

Ekran Görüntüsü



**2 b) Kuyruk Kaynak Kod ve Ekran Görüntüleri**

Kaynak Kod

class QueueM

{

public int maxSize;

public Musteri[] queArray;

public int front;

public int rear;

public int nItems;

public QueueM(int s)

{

maxSize = s;

queArray = new Musteri[maxSize];

front = 0;

rear = -1;

nItems = 0;

}

public void insert(Musteri musteriNesnesi) // put item at rear of queue

{

if (rear == maxSize - 1)

rear = -1;

queArray[++rear] = musteriNesnesi;

nItems++;

}

public Musteri remove() // take item from front of queue

{

Musteri temp = queArray[front++];

if (front == maxSize)

front = 0;

nItems--;

return temp;

}

public bool isEmpty()

{

return (nItems == 0);

}

public QueueM(QueueM x) //Copy Constructor.

{

this.maxSize = x.maxSize;

this.queArray =(Musteri[])x.queArray.Clone();

this.front = x.front;

this.rear = x.rear;

this.nItems = x.nItems;

}

}

Main

QueueM musteriQueue = new QueueM(musteriAdiDizi.Length);

for(int say2 = 0; say2 < musteriAdlariDizisiLengthInt; say2++)

{

Musteri diziyeEklenecekMusteriNesnesi = new Musteri(musteriAdiDizi[say2], urunSayisiDizi[say2]);

musteriQueue.insert(diziyeEklenecekMusteriNesnesi);

}

QueueM ortalamaSureHesaplamadaKullanilacakmusteriQueue = new QueueM(musteriQueue);

Console.WriteLine("\n\n#musteriQueue Yazdiriliyor...\n\*\*\*\*\*");

while(musteriQueue.isEmpty()==false)

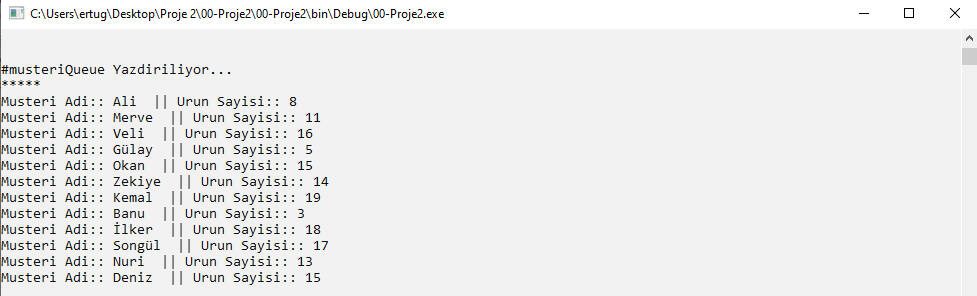
{

Musteri temp = musteriQueue.remove();

temp.yazdir();

}

Ekran Görüntüsü



**3 a) Öncelikli Kuyruk Kaynak Kod ve Ekran Görüntüleri**

Kaynak Kod

class OncelikliKuyrukAzalan

{

public List<Musteri> oncelikKuyrukAz;

public int nItems;

public OncelikliKuyrukAzalan()

{

nItems = 0;

oncelikKuyrukAz = new List<Musteri>();

}

public bool bosMu()

{

return (nItems == 0);

}

public void ekle(Musteri musteriEklenecek)

{

oncelikKuyrukAz.Add(musteriEklenecek);

nItems++;

}

public Musteri sil()

{

int silinecekIndex = 0;

int boyut = oncelikKuyrukAz.Count;

Musteri temp = oncelikKuyrukAz[0];

for (int i = 1; i < boyut; i++) //0. indexi tempe isaretledik. O sebeple donguyu 1'den baslatabiliriz.

{

if (temp.urunSayisi < oncelikKuyrukAz[i].urunSayisi)

{

temp = oncelikKuyrukAz[i];

silinecekIndex = i;

}

}

oncelikKuyrukAz.RemoveAt(silinecekIndex);

nItems--;

return temp;

}

}

Main

OncelikliKuyrukAzalan oncelikliKuyruk1 = new OncelikliKuyrukAzalan();

for(int say3 = 0; say3 < musteriAdlariDizisiLengthInt; say3++)

{

Musteri oncelikliKuyruk1EklenecekMusteriNesnesi = new Musteri(musteriAdiDizi[say3], urunSayisiDizi[say3]);

oncelikliKuyruk1.ekle(oncelikliKuyruk1EklenecekMusteriNesnesi);

}

Console.WriteLine("\n\n#oncelikliKuyruk1 (Azalan Sirada) Yazdiriliyor...\n\*\*\*\*\*");

while (oncelikliKuyruk1.bosMu() == false)

{

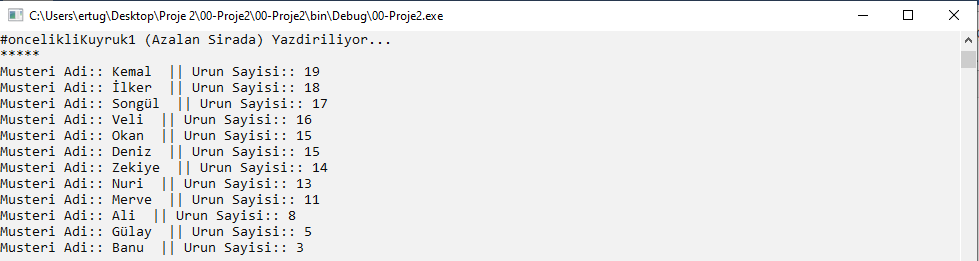
Musteri temp = oncelikliKuyruk1.sil();

temp.yazdir();

}

Console.WriteLine("\n");

Ekran Görüntüsü



**3 b) ArrayList ve Dizi Altyapılarının Karşılaştırılması**

Dizide kapasite sabit olduğundan, kapasitesi dolduğunda program işleyemeyecek. Kapasitesini, bu proje senaryosu özelinde, aşılamayacak kadar fazla tutsak dahi elemanları List yapısındaki kadar kolay çekip döndüremeyeceğiz çünkü dizilerde eleman silmek oldukça zahmetli. Diziyi ürün sayısına göre sıralayıp direkt olarak PQ’ya benzeterek bu zorluğu aşsak dahi, dizi üzerinden bu senaryoda çalışacak daha iyi modeller, statik yapısından dolayı, daha zor tasarlanabileceği için oldukça kısıtlayıcı bir durumla karşı karşıya kalacağız.

**4) Kod, Sonuçlar Tablosunun Ekran Görüntüleri ve Soruların Cevapları**

Kaynak Kod

**a)Güncellenmiş PQ**

class OncelikliKuyrukArtan

{

public List<Musteri> oncelikKuyrukArt;

public int nItems;

public OncelikliKuyrukArtan()

{

nItems = 0;

oncelikKuyrukArt = new List<Musteri>();

}

public bool bosMu()

{

return (nItems == 0);

}

public void ekle(Musteri musteriEklenecek)

{

oncelikKuyrukArt.Add(musteriEklenecek);

nItems++;

}

public Musteri sil()

{

int silinecekIndex = 0;

int boyut = oncelikKuyrukArt.Count;

Musteri temp = oncelikKuyrukArt[0];

for (int i = 1; i < boyut; i++) //0. indexi tempe isaretledik. O sebeple donguyu 1'den baslatabiliriz.

{

if (temp.urunSayisi > oncelikKuyrukArt[i].urunSayisi)

{

temp = oncelikKuyrukArt[i];

silinecekIndex = i;

}

}

oncelikKuyrukArt.RemoveAt(silinecekIndex);

nItems--;

return temp;

}

}

**b)PQ ve FIFO Kuyrukların Karşılaştırılması**

Main

OncelikliKuyrukArtan oncelikliKuyruk2 = new OncelikliKuyrukArtan();

for(int say4 = 0; say4 < musteriAdlariDizisiLengthInt; say4++)

{

Musteri oncelikKuyruk2EklenecekMusteriNesnesi = new Musteri(musteriAdiDizi[say4], urunSayisiDizi[say4]);

oncelikliKuyruk2.ekle(oncelikKuyruk2EklenecekMusteriNesnesi);

}

float kisilerinBekledigiSureFIFO = 0;

float kumulatifBeklemeSuresiFIFO = 0;

Console.WriteLine("#FIFO Yapisindaki Kuyrukta...\n\*\*\*\*\*");

while (ortalamaSureHesaplamadaKullanilacakmusteriQueue.isEmpty() == false)

{

Musteri temp = ortalamaSureHesaplamadaKullanilacakmusteriQueue.remove();

kisilerinBekledigiSureFIFO += temp.urunSayisi;

kumulatifBeklemeSuresiFIFO += kisilerinBekledigiSureFIFO;

Console.WriteLine("Musteri Adi:: " + temp.musteriAdi + " || Bekleyecegi Sure:: " + kisilerinBekledigiSureFIFO);

}

Console.WriteLine("#FIFO Kuyrukta Ortalama Bekleme Suresi:: "+kumulatifBeklemeSuresiFIFO + "/" + musteriAdlariDizisiLengthFloat + "= "+(kumulatifBeklemeSuresiFIFO) / musteriAdlariDizisiLengthFloat+"\n");

float kisilerinBekledigiSurePQ = 0;

float kumulatifBeklemeSuresiPQ = 0;

Console.WriteLine("\n#PQ (Azalan Sirada) Yapisindaki Kuyrukta...\n\*\*\*\*\*");

while (oncelikliKuyruk2.bosMu() == false)

{

Musteri temp = oncelikliKuyruk2.sil();

kisilerinBekledigiSurePQ += temp.urunSayisi;

kumulatifBeklemeSuresiPQ += kisilerinBekledigiSurePQ;

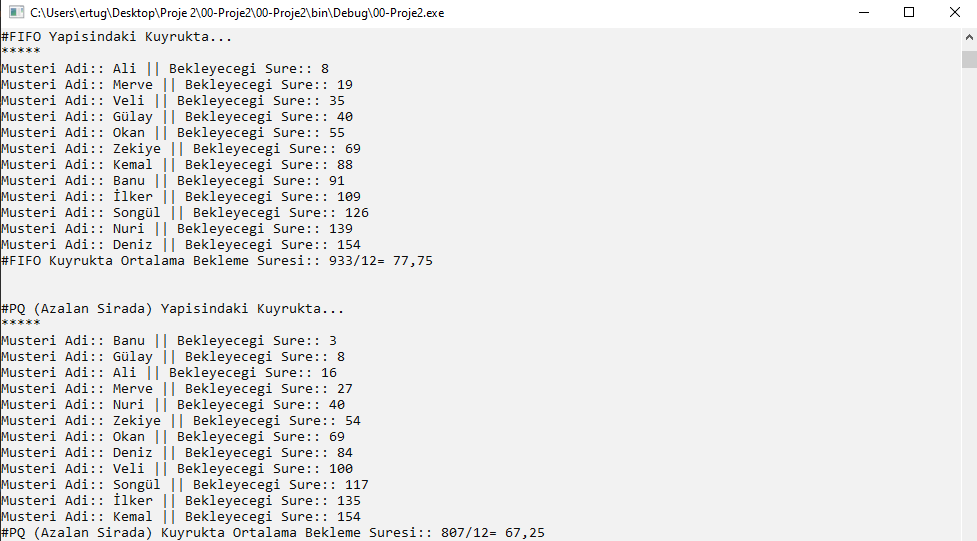
Console.WriteLine("Musteri Adi:: " + temp.musteriAdi + " || Bekleyecegi Sure:: " + kisilerinBekledigiSurePQ);

}

Console.WriteLine("#PQ (Azalan Sirada) Kuyrukta Ortalama Bekleme Suresi:: "+kumulatifBeklemeSuresiPQ+"/"+musteriAdlariDizisiLengthFloat+"= " + (kumulatifBeklemeSuresiPQ) / musteriAdlariDizisiLengthFloat + "\n");

Console.WriteLine("\n");

Ekran Görüntüsü



Karşılaştırma

PQ’da FIFO kuyruğa göre ortalama 10,5 birim zaman daha az bekleniyor.

**c)PQ’nun Dezavantajları ve Kullanılamayacağı Durumlar**

Senaryo özelinde kullanılamayacağı aşikar, çünkü ilk sırayı alanın hizmeti en son alma ihtimali mevcut. Teknik açıdan ise, eleman sayısının çok fazla olduğu durumlarda her seferinde en öncelikli elemanı aramak oldukça külfetli olacaktır. Sürekli olarak yeni verilerin eklendiği sistemlerde, önceliği en az olanlara hiç sıra gelmeyeceğinden, o tarz senaryolarda PQ kullanılamaz. Bu örnek için, 100 birim sürelik işi olan kişi sıraya girerse ve sıraya yeni giren her müşterinin işi 100 birim süreden az ise, 100 birim sürelik işi olan kişiye sıra gelmez.

**d)FIFO ve PQ’nun İyi Taraflarını Birleştiren Bir Yöntem**

FIFO yapısında tutulan elemanlar, baştan başlayarak daha küçük FIFO kuyruklara bölünsün. Daha sonra bu kuyruklar PQ dizaynı ile yeniden işlenirse, hem geliş sırası göz önüne alınmış olur hem de ortalama bekleme süresi azalır. Bu dizaynda küçük FIFO kuyrukların eleman sayısı, eldeki senaryoya göre optimize edilmelidir.

**5)Özdeğerlendirme Tablosu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 2 Maddeleri** | **Puan** | **Tahmini Not** | **Açıklama** |
| **1 a) A4 Ön çalışma** | **20** | **20** | Yapıldı, daha önce labda yapılmış bir örnekten faydalandım. |
| **1 b) Kaynak kod, ekran görüntüsü, veri yapısının elemanlarının listelenmesi** | **20** | **20** | Yapıldı,1-a daki taslağı koda döktüm. |
| **1 c) Kaynak kodlar, Liste sayısı, listelerdeki ortalama eleman sayısı** | **5** | **5** | Yapıldı, 1-b deki kodun üzerinden basit işlemler gerçekleştirildi. |
| **2 a) Yığıt kaynak kod ve ekran görüntüleri** | **5** | **5** | Yapıldı, proje dokümanında bahsi geçen kaynaktan yararlanıldı. |
| **2 b) Kuyruk kaynak kod ve ekran görüntüleri** | **5** | **5** | Yapıldı, proje dokümanında bahsi geçen kaynaktan yararlanıldı. |
| **3 a) Öncelikli Kuyruk kod ve ekran görüntüleri** | **10** | **10** | Yapıldı, proje dokümanındaki adımlar takip edilerek tasarlandı. |
| **3 b) ArrayList ve Dizi altyapılarının karşılaştırılması** | **5** | **5** | Yapıldı, bildiğim kadarıyla cevaplandırdım. |
| **4) Kod, sonuçlar tablosu, ekran görüntüleri ve soruların cevapları.** | **20** | **20** | Yapıldı, PQ yapısında küçük bir değişiklik yapıldı ve daha sonra istenen istatistikler hesaplandı. Sorular cevaplandırıldı. |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** | Projeyi elimden geldiğince düzgün yapmaya, raporu da derli toplu tutmaya çalıştım. Güzel bir sonuç alacağımı umuyorum. |
| **Toplam** | **100** | **100** |  |