

**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**

05190000043, Ahmet Hakan Demirel

İçindekiler

[1.a Bileşik Veri Yapısı için Ön Çalışma 2](#_Toc59980507)

[1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma 2](#_Toc59980508)

[1.b.1 Kaynak Kod 2](#_Toc59980509)

[1.b.2 Ekran görüntüleri 2](#_Toc59980510)

[1.b.3 Veri Yapıları ve Açıklama 2](#_Toc59980511)

[1.c Bileşik Veri Yapısı Bilgi Çıkarma 2](#_Toc59980512)

[1.c.1 Kaynak Kod 2](#_Toc59980513)

[1.c.2 Ekran görüntüleri 2](#_Toc59980514)

[2.a Yığıt 2](#_Toc59980515)

[2.a.1 Kaynak Kod 2](#_Toc59980516)

[2.a.2 Ekran görüntüleri 2](#_Toc59980517)

[2.b Kuyruk 2](#_Toc59980518)

[2.b.1 Kaynak Kod 2](#_Toc59980519)

[2.b.2 Ekran görüntüleri 2](#_Toc59980520)

[3.a Öncelikli Kuyruk Oluşturma 2](#_Toc59980521)

[3.a.1 Kaynak Kod 2](#_Toc59980522)

[3.a.2 Ekran görüntüleri 2](#_Toc59980523)

[3.b ArrayList ve Dizi altyapılarının karşılaştırılması 2](#_Toc59980524)

[4.a Öncelikli Kuyruk Güncelleme 3](#_Toc59980525)

[4.b Ortalama İşlem Tamamlama Süresi 3](#_Toc59980526)

[4.b.1 Kaynak Kod 3](#_Toc59980527)

[4.b.2 Ekran görüntüleri 3](#_Toc59980528)

[4.c Öncelikli Kuyruk Tartışma 3](#_Toc59980529)

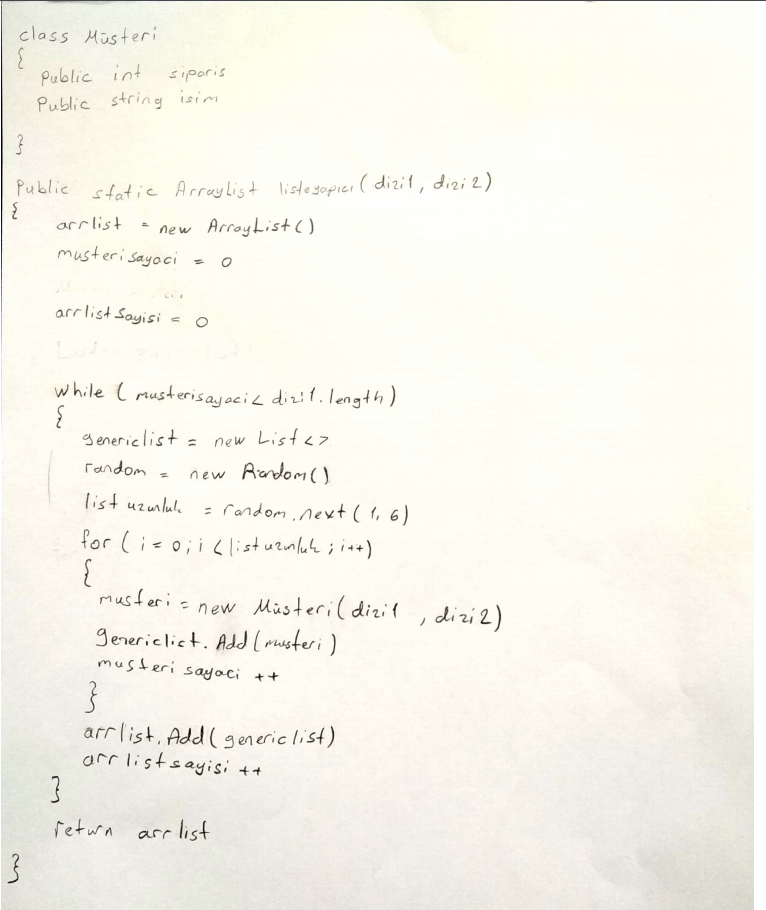
[4.d Öncelikli Kuyruk Öneri 3](#_Toc59980530)

[Özdeğerlendirme Tablosu 3](#_Toc59980531)

LİSTE, YIĞIT, KUYRUK ve ÖNCELİKLİ KUYRUK VERİ YAPILARI

//The platform, version, and programming language used

## 1.a Bileşik Veri Yapısı için Ön Çalışma



## 1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma

### 1.b.1 Kaynak Kod

public static ArrayList listGenerator(string[] dizi, int[] dizi2)

{

ArrayList arrlist = new ArrayList();

int musteriSayaci = 0;

Musteri musteri;

int arrlistSayisi = 0;

List<Musteri> genericlist;

while (musteriSayaci < dizi.Length)

{

genericlist = new List<Musteri>();

Random random = new Random();

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

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

{

musteri = new Musteri(dizi[musteriSayaci], dizi2[musteriSayaci]);

genericlist.Add(musteri);

musteriSayaci++;

if (musteriSayaci == dizi.Length)

break;

}

arrlist.Add(genericlist);

arrlistSayisi++;

}

Console.WriteLine("ArrayList sayısı: " +Convert.ToString(arrlistSayisi));

Console.WriteLine("Ortalama eleman sayısı: " +Convert.ToString(dizi.Length/arrlistSayisi));

Console.WriteLine();

return arrlist;

}

### 1.b.2 Ekran görüntüleri



### 1.b.3 Veri Yapıları ve Açıklama

ArrayList ve genericList kullandım.

## 1.c Bileşik Veri Yapısı Bilgi Çıkarma

### 1.c.1 Kaynak Kod

Console.WriteLine("ArrayList sayısı: " +Convert.ToString(arrlistSayisi));

Console.WriteLine("Ortalama eleman sayısı: " +Convert.ToString(dizi.Length/arrlistSayisi));

### .c.2 Ekran görüntüleri



## 2.a Yığıt

### 2.a.1 Kaynak Kod

class StackClass

{

public int maxSize;

public int top;

public Musteri[] stackArray;

public StackClass(int maxSize)

{

this.maxSize = maxSize;

stackArray = new Musteri[maxSize];

top = -1;

}

public void push(Musteri musteri)

{

stackArray[++top] = musteri;

}

public Musteri pop()

{

return stackArray[top--];

}

public Musteri peek()

{

return stackArray[top];

}

public bool isEmpty()

{

return (top == -1);

}

public bool isFull()

{

return (top == maxSize - 1);

}

}

static void stackYazdirma(string[] MusteriAdi, int[] ÜrünSayısı)

{

Musteri[] musteriList = new Musteri[MusteriAdi.Length];

for (int i = 0; i < MusteriAdi.Length; i++)

{

Musteri musteri = new Musteri(MusteriAdi[i], ÜrünSayısı[i]);

musteriList[i] = musteri;

}

StackClass stack = new StackClass(musteriList.Length);

for (int i = 0; i < musteriList.Length; i++)

{

stack.push(musteriList[i]);

}

while (!stack.isEmpty())

{

Musteri musteri = stack.pop();

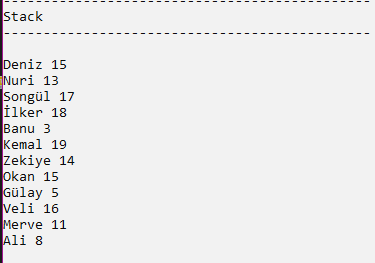
Console.WriteLine(musteri.musteriAdi + " " + musteri.siparisSayisi);

}

Console.WriteLine();

}

### 2.a.2 Ekran görüntüleri



## 2.b Kuyruk

### 2.b.1 Kaynak Kod

class Queue

{

public int maxSize;

public Musteri[] queArr;

public int front;

public int rear;

public int nItems;

public Queue(int maxSize)

{

this.maxSize = maxSize;

queArr = new Musteri[maxSize];

front = 0;

rear = -1;

nItems = 0;

}

public void Insert(Musteri musteri)

{

if (rear == maxSize - 1)

{

rear = -1;

}

queArr[++rear] = musteri;

nItems++;

}

public Musteri remove()

{

Musteri temp = queArr[front++];

if (front == maxSize)

front = 0;

nItems--;

return temp;

}

public Musteri peekFront()

{

return queArr[front];

}

public bool isEmpty()

{

return (nItems == 0);

}

public bool isFull()

{

return (nItems == maxSize);

}

public int size()

{

return nItems;

}

}

static Queue queueYazdirma(string[] MusteriAdi, int[] ÜrünSayısı)

{

Musteri[] musteriList = new Musteri[MusteriAdi.Length];

for (int i = 0; i < MusteriAdi.Length; i++)

{

Musteri musteri = new Musteri(MusteriAdi[i], ÜrünSayısı[i]);

musteriList[i] = musteri;

}

Queue queue = new Queue(musteriList.Length);

for (int i = 0; i < musteriList.Length; i++)

{

queue.Insert(musteriList[i]);

}

Queue temp = new Queue(queue.maxSize);

while (!queue.isEmpty())

{

Musteri musteri = queue.remove();

temp.Insert(musteri);

Console.WriteLine(musteri.musteriAdi + " " + musteri.siparisSayisi);

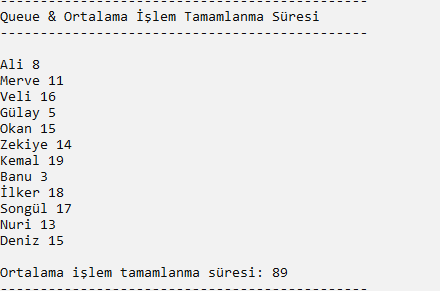
}

Console.WriteLine();

return temp;

}

### 2.b.2 Ekran görüntüleri



## 3.a Öncelikli Kuyruk Oluşturma

### 3.a.1 Kaynak Kod

class priorityQueue

{

public int maxSize;

public List<Musteri> pQueArr;

public int nItems;

public priorityQueue(int maxSize)

{

this.maxSize = maxSize;

pQueArr = new List<Musteri>();

nItems = 0;

}

public void Insert(Musteri musteri)

{

pQueArr.Add(musteri);

nItems++;

}

public Musteri remove()

{

int max = pQueArr.ElementAt(0).siparisSayisi;

int maxIndex = 0;

for (int i = 0; i < pQueArr.Count; i++)

{

if (pQueArr.ElementAt(i).siparisSayisi > max)

{

max = pQueArr.ElementAt(i).siparisSayisi;

maxIndex = i;

}

}

Musteri temp = pQueArr.ElementAt(maxIndex);

pQueArr.RemoveAt(maxIndex);

return temp;

}

public bool isEmpty()

{

return (nItems == 0);

}

}

static priorityQueue pQueueYazdirma(string[] MusteriAdi, int[] ÜrünSayısı)

{

Musteri[] musteriList = new Musteri[MusteriAdi.Length];

for (int i = 0; i < MusteriAdi.Length; i++)

{

Musteri musteri = new Musteri(MusteriAdi[i], ÜrünSayısı[i]);

musteriList[i] = musteri;

}

priorityQueue pQueue = new priorityQueue(musteriList.Length);

for (int i = 0; i < musteriList.Length; i++)

{

pQueue.Insert(musteriList[i]);

}

priorityQueue temp = new priorityQueue(pQueue.maxSize);

for (int i = 0; i < pQueue.maxSize; i++)

{

Musteri musteri = pQueue.remove();

temp.Insert(musteri);

Console.WriteLine(musteri.musteriAdi + " " + musteri.siparisSayisi);

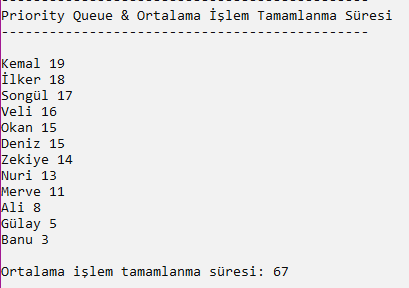
}

Console.WriteLine();

return temp;

}

### 3.a.2 Ekran görüntüleri



## 3.b ArrayList ve Dizi altyapılarının karşılaştırılması

Eleman sayısını bilmediğimizden dolayı Dizi kullanmamız sıkıntı oluşturur. Çok büyük bir dizi açarsak da RAMde çok fazla yer tutar.

## 4.a Öncelikli Kuyruk Güncelleme

class pQueueArtan

{

public int maxSize;

public List<Musteri> pQueArr;

public int nItems;

public pQueueArtan(int maxSize)

{

this.maxSize = maxSize;

pQueArr = new List<Musteri>();

nItems = 0;

}

public void Insert(Musteri musteri)

{

pQueArr.Add(musteri);

nItems++;

}

public Musteri remove()

{

int min = pQueArr.ElementAt(0).siparisSayisi;

int minIndex = 0;

for (int i = 1; i < pQueArr.Count; i++)

{

if (pQueArr.ElementAt(i).siparisSayisi < min)

{

min = pQueArr.ElementAt(i).siparisSayisi;

minIndex = i;

}

}

Musteri temp = pQueArr.ElementAt(minIndex);

pQueArr.RemoveAt(minIndex);

return temp;

}

public bool isEmpty()

{

return (nItems == 0);

}

}

static void pQueueArtanYazdirma(string[] MusteriAdi, int[] ÜrünSayısı)

{

Musteri[] musteriList = new Musteri[MusteriAdi.Length];

for (int i = 0; i < MusteriAdi.Length; i++)

{

Musteri musteri = new Musteri(MusteriAdi[i], ÜrünSayısı[i]);

musteriList[i] = musteri;

}

pQueueArtan pQueue = new pQueueArtan(musteriList.Length);

for (int i = 0; i < musteriList.Length; i++)

{

pQueue.Insert(musteriList[i]);

}

for (int i = 0; i < pQueue.maxSize; i++)

{

Musteri musteri = pQueue.remove();

Console.WriteLine(musteri.musteriAdi + " " + musteri.siparisSayisi);

}

Console.WriteLine();

}

## 4.b Ortalama İşlem Tamamlama Süresi

### 4.b.1 Kaynak Kod

static void ortalamaMusteriQueue(Queue queue)

{

int sonuc = 0;

int musteriSayisi = queue.maxSize;

for (int i = 1; i < queue.maxSize+1; i++)

{

Musteri musteri = queue.remove();

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

{

sonuc += musteri.siparisSayisi;

}

}

sonuc = sonuc / musteriSayisi;

Console.WriteLine("Ortalama işlem tamamlanma süresi: " + sonuc);

}

static void ortalamaMusteripQueue(priorityQueue pQueue)

{

int sonuc = 0;

int musteriSayisi = pQueue.maxSize;

for (int i = 1; i < pQueue.maxSize + 1; i++)

{

Musteri musteri = pQueue.remove();

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

{

sonuc += musteri.siparisSayisi;

}

}

sonuc = sonuc / musteriSayisi;

Console.WriteLine("Ortalama işlem tamamlanma süresi: "+sonuc);

}

### 4.b.2 Ekran görüntüleri



   
4.b.3 Sözel olarak karşılaştırma

## 4.c Öncelikli Kuyruk Tartışma

Öncelikli kuyruk daha hızlı olabilir fakat siparişi az olanlar daha fazla bekliyor. Bu sebepten dolayı birbirine yakın siparişler olduğunda kullanılması sıkıntı olabilir.

## 4.d Öncelikli Kuyruk Öneri

Kuyruktakileri kendi aralarında gruplarım. Daha sonra gruplar arasında seçim yaparım. Örnek olarak ilk ikisini alırım ve hangisinin daha önce gireceğine karar veririm. Sonra üç ve dördüncüye aynı işlemi yaparım. Böyle böyle sonra kadar giderim.

# 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** | **17** | **Hepsi yapıldı. Fakat 1c ile iç içe yapıldı.** |
| **1 c) Kaynak kodlar, Liste sayısı, listelerdeki ortalama eleman sayısı** | **5** | **5** | **Yapıldı. 1b nin içinde.** |
| **2 a) Yığıt kaynak kod ve ekran görüntüleri** | **5** | **5** | **Yapıldı.Kitaptan yardım aldım.** |
| **2 b) Kuyruk kaynak kod ve ekran görüntüleri** | **5** | **5** | **Yapıldı.Kitaptan yardım aldım.** |
| **3 a) Öncelikli Kuyruk kod ve ekran görüntüleri** | **10** | **10** | **Yapıldı.Verilen linkten ve İnternetten baktım.** |
| **3 b) ArrayList ve Dizi altyapılarının karşılaştırılması** | **5** | **5** |  |
| **4) Kod, sonuçlar tablosu, ekran görüntüleri ve soruların cevapları.** | **20** | **20** | **Yapıldı.** |
| **5) Özdeğerlendirme Tablosu** | **10** | **9** | **Yapıldı fakat tam anlamadığım için istenilenden farklı yapmış olabilirim.** |
| **Toplam** | **100** | **96** | **1b ve tablodan puanım kırılabilir.** |

**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.**