

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2021–2022 FALL SEMESTER**

**PROJECT-2 REPORT**

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

**DELIVERY DATE**

04/01/2022

**PREPARED BY**

05200000045, Şevval GÖNÜL

İçindekiler

[1.a Bileşik Veri Yapısı için Ön Çalışma 2](#_Toc91168778)

[1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma 2](#_Toc91168779)

[1.b.1 Kaynak Kod 2](#_Toc91168780),3

[1.b.2 Ekran görüntüleri 4](#_Toc91168781)

[1.b.3 Veri Yapıları ve Açıklama 4](#_Toc91168782)

[1.c Bileşik Veri Yapısı Bilgi Çıkarma 4](#_Toc91168783)

[1.c.1 Kaynak Kod 4](#_Toc91168784)

[1.c.2 Ekran görüntüleri 4](#_Toc91168785)

[2.a Yığıt 4](#_Toc91168786)

[2.a.1 Kaynak Kod 4,5](#_Toc91168787)

[2.a.2 Ekran görüntüleri 5](#_Toc91168788)

[2.b Kuyruk 5](#_Toc91168789)

[2.b.1 Kaynak Kod 5,6](#_Toc91168790)

[2.b.2 Ekran görüntüleri 6](#_Toc91168791)

[3.a Öncelikli Kuyruk Oluşturma 6](#_Toc91168792)

[3.a.1 Kaynak Kod 6,7](#_Toc91168793)

[3.a.2 Ekran görüntüleri 7](#_Toc91168794)

[3.b ArrayList ve Dizi altyapılarının karşılaştırılması 8](#_Toc91168795)

[4.a Öncelikli Kuyruk Güncelleme 8,9](#_Toc91168796)

[4.b Ortalama İşlem Tamamlama Süresi 9](#_Toc91168797)

[4.b.1 Kaynak Kod 9,10](#_Toc91168798)

[4.b.2 Ekran görüntüleri 10](#_Toc91168799)

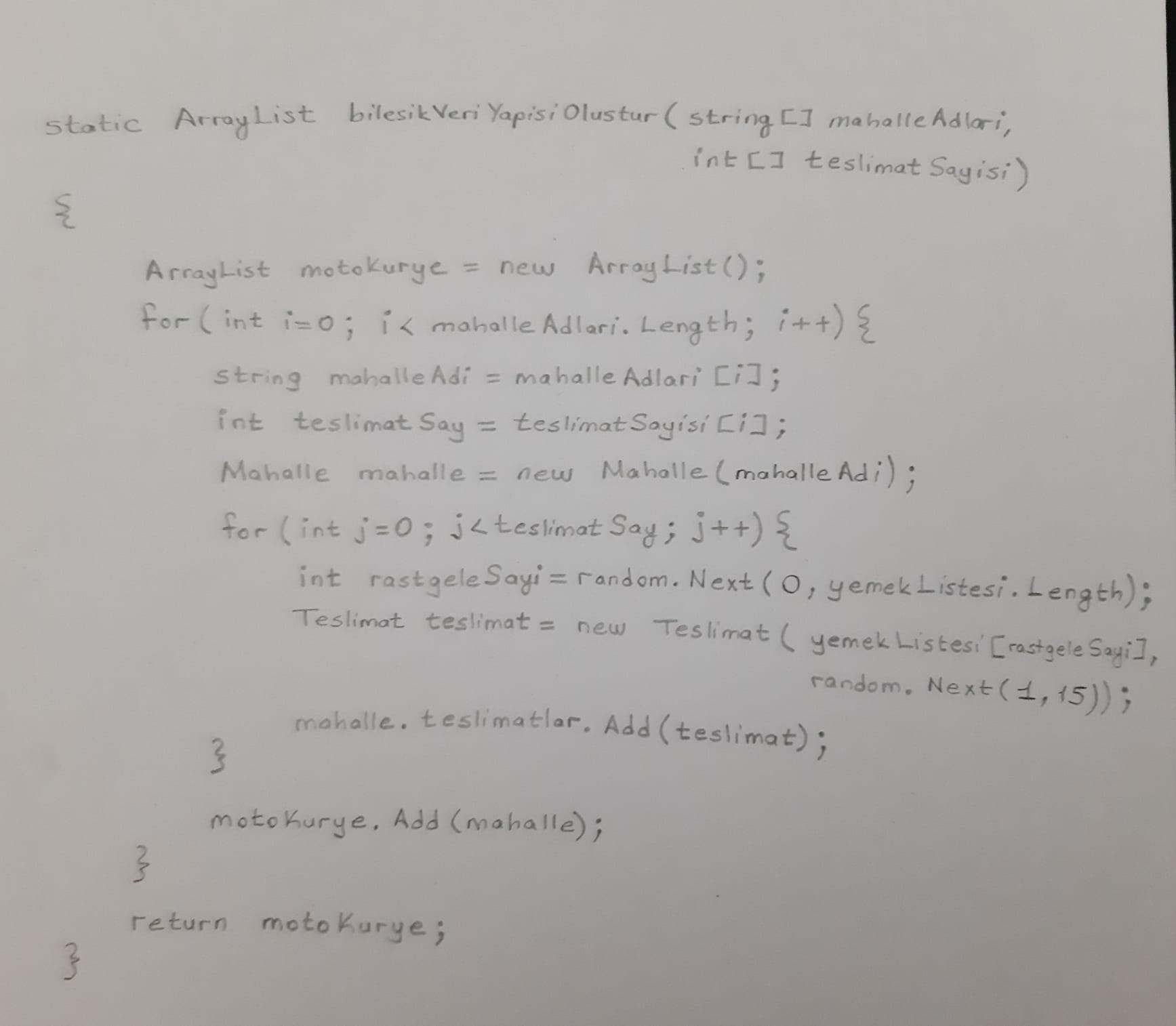
[4.c Öncelikli Kuyruk Tartışma 10](#_Toc91168800)

[5. Özdeğerlendirme Tablosu 11](#_Toc91168801)

LİSTE, YIĞIT, KUYRUK ve ÖNCELİKLİ KUYRUK VERİ YAPILARI

Visual Studio, 2019, C# used

## 1.a Bileşik Veri Yapısı için Ön Çalışma



## 1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma

### 1.b.1 Kaynak Kod

class Teslimat

{

public string yemekAdi;

public int adet;

public Teslimat(string yemekAdi, int adet)

{

this.yemekAdi = yemekAdi;

this.adet = adet;

}

}

class Mahalle

{

public string mahalleAdi;

public List<Teslimat> teslimatlar;

public Mahalle(string mahalleAdi)

{

this.mahalleAdi = mahalleAdi;

teslimatlar = new List<Teslimat>();

}

public override string ToString()

{

string str = "Mahalle Adı: " + mahalleAdi + ", Teslimatlar: ";

foreach(Teslimat teslimat in teslimatlar)

{

str += String.Format("[{0}, {1}] ", teslimat.yemekAdi, teslimat.adet);

}

return str;

}

}

class Program

{

static string[] yemekListesi = { "İçli Köfte", "Pizza", "Hamburger", "Mantı", "Döner", "Börek", "Pilav", "Türlü", "Simit", "Kızartma" };

static Random random = new Random();

static ArrayList bilesikVeriYapisiOlustur(string[] mahalleAdlari, int[] teslimatSayisi)

{

ArrayList motoKurye = new ArrayList();

for(int i = 0; i < mahalleAdlari.Length; i++) // Herbir mahalle için:

{

string mahalleAdi = mahalleAdlari[i];

int teslimatSay = teslimatSayisi[i];

Mahalle mahalle = new Mahalle(mahalleAdi); // Yeni bir mahalle ve o mahalledeki teslimatları tutmak için bir GenericList oluşturdu.

for(int j = 0; j < teslimatSay; j++) // O mahalledeki herbir teslimat için:

{

int rastgeleSayi = random.Next(0, yemekListesi.Length); // 0 ile yemek listesinin uzunluğu arasında rastgele bir sayi

Teslimat teslimat = new Teslimat(yemekListesi[rastgeleSayi], random.Next(1,15)); // yemekListesinde rastgeleSayiya karşılık gelen yemek ile 1-15 arasında bir adette oluşturulan teslimat

mahalle.teslimatlar.Add(teslimat); // mahalle nesnesinin içindeki teslimatlar GenericList'ine teslimat nesnesi ekleme

}

motoKurye.Add(mahalle);

}

return motoKurye;

}

static void yazdir(ArrayList motoKurye)

{

int teslimatSay = 0; // Toplam teslimat sayısı

foreach (Mahalle mahalle in motoKurye)

{

Console.Write("Mahalle Adı: " + mahalle.mahalleAdi + ", ");

Console.Write("Teslimatlar: ");

foreach (Teslimat teslimat in mahalle.teslimatlar)

{

Console.Write("[{0}, {1}] ", teslimat.yemekAdi, teslimat.adet);

teslimatSay += 1;

}

Console.WriteLine();

}

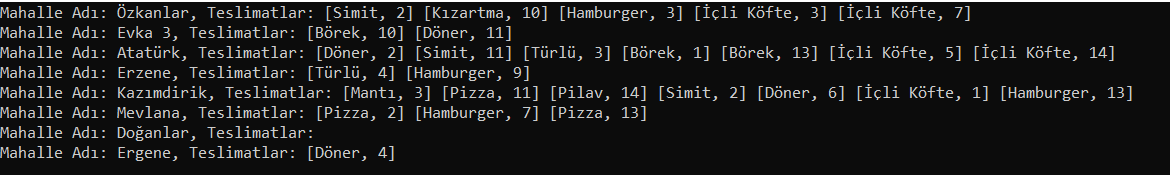
// Toplam liste ve teslimat sayısı yazdırma:

Console.WriteLine("\nArrayList içindeki GenericList Sayısı (Mahalle Sayısı) : " + motoKurye.Count);

Console.WriteLine("Toplam teslimat sayısı: " + teslimatSay + "\n");

}

### 1.b.2 Ekran görüntüleri



### 1.b.3 Veri Yapıları ve Açıklama

Mahalle adlarını, herbir mahalleye teslim edilecek teslimat sayılarını ve yemek listesini tutmak için:

string[] mahalleAdlari , int[] teslimatSayisi, string[] yemekListesi adında diziler kullandım. Mahalle nesnelerini tutabilmek için motoKurye adında bir ArrayList oluşturdum ve herbir mahalle nesnesinin içinde o mahalleye yapılacak olan teslimatları tutmak için teslimatlar adında bir Generic List kullandım.

## 1.c Bileşik Veri Yapısı Bilgi Çıkarma

### 1.c.1 Kaynak Kod

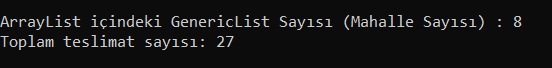
// Toplam liste ve teslimat sayısı yazdırma:

Console.WriteLine("\nArrayList içindeki GenericList Sayısı (Mahalle Sayısı) : " + motoKurye.Count);

Console.WriteLine("Toplam teslimat sayısı: " + teslimatSay + "\n");

}

### 1.c.2 Ekran görüntüleri



## 2.a Yığıt

### 2.a.1 Kaynak Kod

class Stack

{

private int maxSize;

private Mahalle[] stackArray; // Mahalle tipindeki nesneleri tutacak olan Stack

private int top;

public Stack(int size)

{

maxSize = size;

stackArray = new Mahalle[maxSize];

top = -1;

}

public void push(Mahalle newItem)

{

stackArray[++top] = newItem;

}

public Mahalle pop()

{

return stackArray[top--];

}

public bool isEmpty()

{

return (top == -1);

}

}

… main

// Yığıt, kuyruk ve öncelikli kuyruk oluşturma ve tüm mahalleleri ArrayList'ten çekerek bu veri yapılarına eklema:

Stack stack = new Stack(mahalleAdlari.Length);

Queue queue = new Queue(mahalleAdlari.Length);

OncelikliKuyruk priorityQueue = new OncelikliKuyruk();

foreach (Mahalle mahalle in motoKurye)

{

stack.push(mahalle);

queue.insert(mahalle);

priorityQueue.ekle(mahalle);

}

// Yığıttaki elemanları ekrana yazdırma:

Console.WriteLine("Yığıt:");

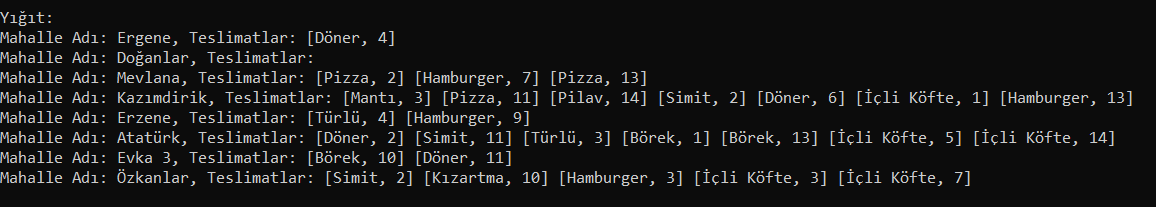
while(!stack.isEmpty())

{

Console.WriteLine(stack.pop().ToString());

}

### 2.a.2 Ekran görüntüleri



## 2.b Kuyruk

### 2.b.1 Kaynak Kod

class Queue

{

private int maxSize;

private Mahalle[] queueArray; // Mahalle tipindeki nesneleri tutacak olan Queue

private int front; private int rear; private int nItems;

public Queue(int size)

{

maxSize = size;

queueArray = new Mahalle[maxSize];

front = 0;

rear = -1;

nItems = 0;

}

public void insert(Mahalle newItem) // Kuyruğa eleman ekleme

{

if (rear == maxSize - 1)

rear = -1;

queueArray[++rear] = newItem;

nItems++;

}

public Mahalle remove()

{

Mahalle temp = queueArray[front++];

if (front == maxSize)

front = 0;

nItems--;

return temp;

}

public bool isEmpty()

{

return (nItems == 0);

}

}

… main

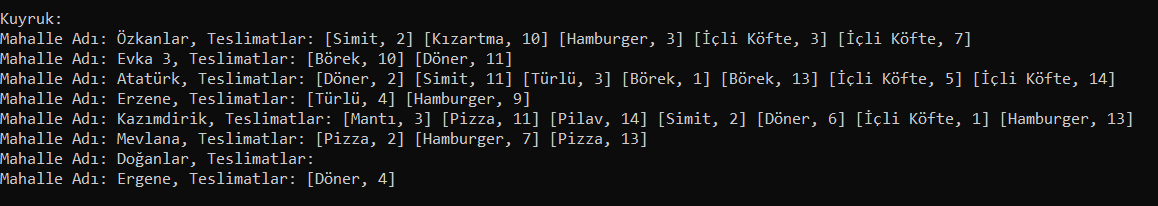
Console.WriteLine("\nKuyruk: ");

// Kuyruktaki elemanları ekrana yazdırma:

while(!queue.isEmpty())

Console.WriteLine(queue.remove().ToString());

### 2.b.2 Ekran görüntüleri



## 3.a Öncelikli Kuyruk Oluşturma

### 3.a.1 Kaynak Kod

class OncelikliKuyruk

{

List<Mahalle> pq; // Priority Queue

public OncelikliKuyruk()

{

pq = new List<Mahalle>();

}

public void ekle(Mahalle yeniMahalle) // Eleman kuyruğun sonuna eklenir

{

pq.Add(yeniMahalle); // Add metodu yeni gelen elemanı sona ekler

}

public Mahalle sil() // Azalan Öncelik Kuyruğu olduğu için önce en fazla teslimat yapılan mahalleyi silecek olan metod

{

int maxTeslimatSay = 0; // Initialize etmek için max değişkenine olamayacak kadar küçük bir değer veriyorum

Mahalle maxTeslimatliMahalle = pq[0]; // Silinecek olan elemanı kuyruktaki ilk eleman olarak belirledim. Döngüde güncellenecek.

foreach (Mahalle mahalle in pq)

{

int teslimatSay = mahalle.teslimatlar.Count; // O mahalledeki teslimat sayısı

if (teslimatSay > maxTeslimatSay)

{

maxTeslimatSay = teslimatSay;

maxTeslimatliMahalle = mahalle;

}

}

// Döngüden çıkınca silinecek olan eleman yani en fazla teslimat yapılacak olan mahalle bulunmuş olur.

pq.Remove(maxTeslimatliMahalle); // List classının Remove metoduna argüman olarak silinecek olan eleman verilir ve o eleman listeden kaldırılmış olur

return maxTeslimatliMahalle; // Silinen eleman geri döndürülür

}

public bool bosMu()

{

return (pq.Count == 0);

}

}

… main

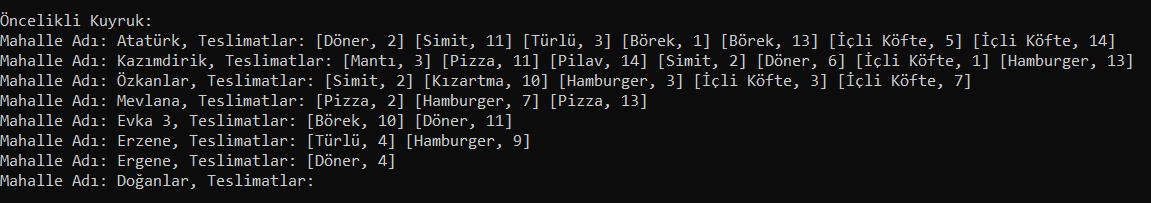
Console.WriteLine("\nÖncelikli Kuyruk: ");

// Öncelikli Kuyruktaki elemanları ekrana yazdırma:

while (!priorityQueue.bosMu())

Console.WriteLine(priorityQueue.sil().ToString());

### 3.a.2 Ekran görüntüleri



## 

## 3.b ArrayList ve Dizi altyapılarının karşılaştırılması

Öncelik kuyruğunda elemanları sıralı tutmadığımız için eklemeyi sondan yapıyoruz fakat silme işlemi ortadan baştan sondan olabilir. List dinamik bir veri yapısı olduğu için aradan eleman çıkarmak kolay ve etkindir. Elemanları kaydırmak için kod yazmaya gerek kalmaz. Fakat dizi kullansaydık eleman sildiğimizde boşlukları doldurmak için elemanları kaydırmamız gerekecekti. Ayrıca dizi kullansaydık sabit uzunlukta olacaktı. Tanımlandıktan sonra büyültülüp küçültülemezdi. List’te ise böyle bir kısıtlama yok. Eleman eklediğimiz sürece boyutu artacak ve çıkarttığımızda azalacak. Bu sayede hem boş yere bellek harcanmayacak hem de taşma gibi bir sorun olmayacak.

## 4.a Öncelikli Kuyruk Güncelleme

class Kuyruk

{

List<int> queueList;

public Kuyruk()

{

queueList = new List<int>();

}

public void ekle(int yeniEleman) // Kuyruk yapısında elemanlar sondan eklenir

{

queueList.Add(yeniEleman); // Add metodu sondan ekleme yapar

}

public int sil() // Kuyruk yapısında elemanlar baştan silinir.

{

int silinen = queueList[0];

queueList.RemoveAt(0); // En baştaki (0.indexteki) elemanı silme

return silinen;

}

public bool bosMu()

{

return (queueList.Count == 0);

}

}

class Program

{

static int okutmaSuresi = 3; // Herbir ürünün okutulması 3 saniye sürmektedir.

static void Main(string[] args)

{

int[] urunSay = { 6, 7, 2, 1, 12, 5, 3, 7, 4, 2 }; // Kasada bekleyen herbir müşterinin ürün sayıları

int musteriSay = urunSay.Length;

Kuyruk kuyruk = new Kuyruk();

OncelikliKuyruk pq = new OncelikliKuyruk();

for (int i = 0; i < musteriSay; i ++) // Herbir müşterinin ürün sayıları kuyruk yapılarına eklenir.

{

kuyruk.ekle(urunSay[i]);

pq.ekle(urunSay[i]);

}

int[] kuyrukIslemSureleri = new int[musteriSay]; // Kuyruk yapısındaki herbir müsterinin işlem tamamlanma sürelerini tutacak olan dizi

int[] pqIslemSureleri = new int[musteriSay]; // Öncelikli Kuyruk yapısındaki herbir müsterinin işlem tamamlanma sürelerini tutacak olan dizi

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

{

if (i == 0) // Bu kuyruklardaki ilk müştesi ise, işlem süresi kendi ürünlerini okutma süresine eşittir

{

kuyrukIslemSureleri[i] = kuyruk.sil() \* okutmaSuresi;

pqIslemSureleri[i] = pq.oncelikliSil() \* okutmaSuresi;

}

else // Diğer müşteriler için işlem tamamlanma süresi, kendinden önceki müşterinin işlem tamamlanma süresi ile kendisinin ürünlerini okutma süresi toplamına eşittir.

{

kuyrukIslemSureleri[i] = kuyrukIslemSureleri[i - 1] + kuyruk.sil() \* okutmaSuresi;

pqIslemSureleri[i] = pqIslemSureleri[i - 1] + pq.oncelikliSil() \* okutmaSuresi;

}

}

int kuyrukİslemToplam = 0;

Console.Write("Kuyruktaki herbir müşterinin işlem tamamlanma süreleri: ");

foreach (int sure in kuyrukIslemSureleri)

{

Console.Write(sure + " ");

kuyrukİslemToplam += sure;

}

Console.WriteLine("\nKuyruk yapısı kullanılan kasada ortalama işlem tamamlanma süresi: " + (double)kuyrukİslemToplam/musteriSay);

## 4.b Ortalama İşlem Tamamlama Süresi

### 4.b.1 Kaynak Kod

class OncelikliKuyruk

{

List<int> pq;

public OncelikliKuyruk()

{

pq = new List<int>();

}

public void ekle(int yeniEleman)

{

pq.Add(yeniEleman); // Gelen eleman kuyruğun sonuna eklenir

}

public int oncelikliSil() // Artan sırada öncelik kuyruğu olduğu için önce en küçük olan elemanı silecek olan metod

{

int min = pq[0]; // Öncelik kuyruğunun ilk elemanını minimum olarak initialize ettim, döngüde güncellenecek.

foreach(int sayi in pq)

{

if (sayi < min)

min = sayi;

}

// Döngüden çıkınca en küçük eleman bulunmuş olur ve remove metoduyla o elemanı listeden kaldırıyorum:

pq.Remove(min);

return min; // Öncelikli kuyrukta en az sayıda ürünü olan müşteri silinir ve ürünlerinin sayısı döndürülür.

}

public bool bosMu()

{

return (pq.Count == 0);

}

}

… main

Console.Write("\nÖncelikli Kuyruktaki herbir müşterinin işlem tamamlanma süreleri: ");

int pqİslemToplam = 0;

foreach (int sure in pqIslemSureleri)

{

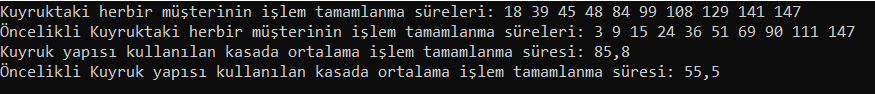
Console.Write(sure + " ");

pqİslemToplam += sure;

}

Console.WriteLine("Öncelikli Kuyruk yapısı kullanılan kasada ortalama işlem tamamlanma süresi: " + (double)pqİslemToplam/musteriSay);

### 4.b.2 Ekran görüntüleri



4.b.3 Sözel olarak karşılaştırma

## 4.c Öncelikli Kuyruk Tartışma

Kuyruk yapısı kullanılan kasada ortalama işlem tamamlanma süresi 85.8 çıkarken öncelikli kuyruk yapısı kullanınca bu süre 55.5’e düşmektedir. Bu genel memnuniyet açısından bir fayda sağlar ve işlem tamamlanma süresi açısından daha verimli olmaktadır fakat en çok ürün alan kişi en uzun süre beklemek zorunda kalacağından adaletsizliğe yol açacaktır. Dolayısıyla gerçek bir öncelik durumunun (Ör: Hastanede öncelik sırası değişebilir, uçakların piste inişinde öncelik durumları olabilir. Her zaman ilk gelenin işlemini ilk bitirmesi gerekmez) olmadığı durumlarda adaletsizliğe yol açacağından kullanılamaz.

# 5. 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, veri yapısındaki toplam teslimat 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, karşılaştırma ve soruların cevapları.** | **20** | **20** | **Yapıldı.** |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** | **Yapıldı.** |
| **Toplam** | **100** | **100** | **Yapıldı.** |

**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.**