

**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**

05180000091, Kutay Avcı

İçindekiler

[1.a Bileşik Veri Yapısı için Ön Çalışma 3](#_Toc59980507)

[1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma 4](#_Toc59980508)

[1.b.1 Kaynak Kod 6](#_Toc59980509)

[1.b.2 Ekran görüntüleri 7](#_Toc59980510)

[1.b.3 Veri Yapıları ve Açıklama 7](#_Toc59980511)

[1.c Bileşik Veri Yapısı Bilgi Çıkarma 7](#_Toc59980512)

[1.c.1 Kaynak Kod 7](#_Toc59980513)

[1.c.2 Ekran görüntüleri 8](#_Toc59980514)

[2.a Yığıt 8](#_Toc59980515)

[2.a.1 Kaynak Kod 8](#_Toc59980516)

[2.a.2 Ekran görüntüleri 10](#_Toc59980517)

[2.b Kuyruk 10](#_Toc59980518)

[2.b.1 Kaynak Kod 10](#_Toc59980519)

[2.b.2 Ekran görüntüleri 10](#_Toc59980520)

[3.a Öncelikli Kuyruk Oluşturma 12](#_Toc59980521)

[3.a.1 Kaynak Kod 12](#_Toc59980522)

[3.a.2 Ekran görüntüleri 12](#_Toc59980523)

[3.b ArrayList ve Dizi altyapılarının karşılaştırılması 14](#_Toc59980524)

[4.a Öncelikli Kuyruk Güncelleme 15](#_Toc59980525)

[4.b Ortalama İşlem Tamamlama Süresi 15](#_Toc59980526)

[4.b.1 Kaynak Kod 15](#_Toc59980527)

[4.b.2 Ekran görüntüleri 15](#_Toc59980528)

[4.c Öncelikli Kuyruk Tartışma 16](#_Toc59980529)

[4.d Öncelikli Kuyruk Öneri 16](#_Toc59980530)

[Özdeğerlendirme Tablosu 17](#_Toc59980531)

LİSTE, YIĞIT, KUYRUK ve ÖNCELİKLİ KUYRUK VERİ YAPILARI

//Visual Studio, 2019 16.8.0, C#

## 1.a Bileşik Veri Yapısı için Ön Çalışma

## 

## 1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma

### 1.b.1 Kaynak Kod

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 };

int lenght = MüşteriAdı.Length;

int count = 0;

ArrayList arrayList = new ArrayList();

int start = 0;

Random random = new Random();

while (true)

{

int sayı = random.Next(1,5);

count += sayı;

List<Müşteri> genlist = new List<Müşteri>();

for(int i = start; i<count; i++)

{

if (i >= lenght)

break;

genlist.Add(new Müşteri(MüşteriAdı[i], ÜrünSayısı[i]));

}

if (genlist.Count == 0)

break;

start = count;

arrayList.Add(genlist);

if (count >= lenght)

break; }

int listcount = 1;

foreach(List<Müşteri> l in arrayList)

{

Console.WriteLine("Liste" + listcount + ": ");

foreach(Müşteri m in l)

{

Console.WriteLine("Müşteri : " + m);

}

listcount++;

}

### 1.b.2 Ekran görüntüleri

### 1.b.3 Veri Yapıları ve Açıklama

ArrayList ve Generic List kullanılmıştır.Generic list müşteri sınıfı nesnelerini , ArrayList ise Generic list nesnelerini tutmaktadır.

## 1.c Bileşik Veri Yapısı Bilgi Çıkarma

### 1.c.1 Kaynak Kod

double ortalama = (double)lenght / (double)arrayList.Count;

Console.WriteLine("ArrayList eleman sayısı : " + arrayList.Count);

Console.WriteLine("Ortalama eleman sayısı : " + ortalama);

### 1.c.2 Ekran görüntüleri

## 2.a Yığıt

### 2.a.1 Kaynak Kod

class Stack

{

private int maxSize; // size of stack array

private Müşteri[] stackArray;

private int top; // top of stack

//--------------------------------------------------------------

public Stack(int s) // constructor

{

maxSize = s; // set array size

stackArray = new Müşteri[maxSize]; // create array

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));

}

//--------------------------------------------------------------

} // end class StackX

//Main metodu gerçekleştirimi egederse yüklenen kaynak kodlar için bulunmaktadır…

### 2.a.2 Ekran görüntüleri

## 

## 2.b Kuyruk

### 2.b.1 Kaynak Kod

### class Queue

### {

### private int maxSize;

### private Müşteri[] queArray;

### private int front;

### private int rear;

### private int nItems;

### //--------------------------------------------------------------

### public Queue(int s) // constructor

### {

### maxSize = s;

### queArray = new Müşteri[maxSize];

### front = 0;

### rear = -1;

### nItems = 0;

### }

### //--------------------------------------------------------------

### public void insert(Müşteri j) // put item at rear of queue

### {

### if (rear == maxSize - 1) // deal with wraparound

### rear = -1;

### queArray[++rear] = j; // increment rear and insert

### nItems++; // one more item

### }

### //--------------------------------------------------------------

### public Müşteri remove() // take item from front of queue

### {

### if (isEmpty())

### {

### Console.WriteLine("Queue is empty.");

### return new Müşteri("Null", 0);

### }

### Müşteri temp = queArray[front++]; // get value and incr front

### if (front == maxSize) // deal with wraparound

### front = 0;

### nItems--; // one less item

### return temp;

### }

### //--------------------------------------------------------------

### public Müşteri peekFront() // peek at front of queue

### {

### return queArray[front];

### }

### //--------------------------------------------------------------

### public bool isEmpty() // true if queue is empty

### {

### bool a = (nItems == 0);

### return a;

### }

### //--------------------------------------------------------------

### public bool isFull() // true if queue is full

### {

### return (nItems == maxSize);

### }

### //--------------------------------------------------------------

### public int size() // number of items in queue

### {

### return nItems;

### }

//Main metodu gerçekleştirimi egederse yüklenen kaynak kodlar için bulunmaktadır…}

### 2.b.2 Ekran görüntüleri

## 3.a Öncelikli Kuyruk Oluşturma

### 3.a.1 Kaynak Kod

### class ÖncelikliKuyruk

### {

### private List<Müşteri> queArray;

### private int front;

### private int rear;

### private int nItems;

### //--------------------------------------------------------------

### public ÖncelikliKuyruk() // constructor

### {

### queArray = new List<Müşteri>();

### front = 0;

### rear = -1;

### nItems = 0;

### }

### //--------------------------------------------------------------

### public void insert(Müşteri j) // put item at rear of queue

### {

### queArray.Add(j);

### rear++; // increment rear and insert

### nItems++; // one more item

### }

### //--------------------------------------------------------------

### public Müşteri dpq() // take item from front of queue

### {

### int max = 0;

### if (isEmpty())

### {

### Console.WriteLine("Queue is empty.");

### return new Müşteri("Null", 0);

### }

### Müşteri maxitem = queArray[0];

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

### {

### if (max < queArray[i].getÜrünSayısı())

### {

### max = queArray[i].getÜrünSayısı();

### maxitem = queArray[i];

### }

### }

### nItems--;

### queArray.Remove(maxitem);

### return maxitem;

### }

### public Müşteri apq() // take item from front of queue

### {

### int min = queArray[0].getÜrünSayısı();

### if (isEmpty())

### {

### Console.WriteLine("Queue is empty.");

### return new Müşteri("Null", 0);

### }

### Müşteri minitem = queArray[0];

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

### {

### if (min > queArray[i].getÜrünSayısı())

### {

### min = queArray[i].getÜrünSayısı();

### minitem = queArray[i];

### }

### }

### nItems--;

### queArray.Remove(minitem);

### return minitem;

### }

### //--------------------------------------------------------------

### public Müşteri peekFront() // peek at front of queue

### {

### return queArray[front];

### }

### //--------------------------------------------------------------

### public bool isEmpty() // true if queue is empty

### {

### bool a = (nItems == 0);

### return a;

### }

### //--------------------------------------------------------------

### public int size() // number of items in queue

### {

### return nItems;

### }

//Main metodu gerçekleştirimi egederse yüklenen kaynak kodlar için bulunmaktadır…

### }

### 3.a.2 Ekran görüntüleri

## 3.b ArrayList ve Dizi altyapılarının karşılaştırılması

Dizinin boyutu önceden belirlenmesi gerektiği için sonradan eleman eklemek sorun yaratacaktır ve diziden eleman çıkarmak için bütün diziyi dolaşmak gerekeceği için maliyeti arttıracaktır bu yüzden liste kullanılması avantajlıdır.

## 4.a Öncelikli Kuyruk Güncelleme

## public Müşteri apq() // take item from front of queue

## {

## int min = queArray[0].getÜrünSayısı();

## if (isEmpty())

## {

## Console.WriteLine("Queue is empty.");

## return new Müşteri("Null", 0);

## }

## Müşteri minitem = queArray[0];

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

## {

## if (min > queArray[i].getÜrünSayısı())

## {

## min = queArray[i].getÜrünSayısı();

## minitem = queArray[i];

## }

## }

## nItems--;

## queArray.Remove(minitem);

## return minitem;

## }

## 4.b Ortalama İşlem Tamamlama Süresi

### 4.b.1 Kaynak Kod

### int islemcountapq = 0;

### for (int i = 0; i < lenght; i++)

### {

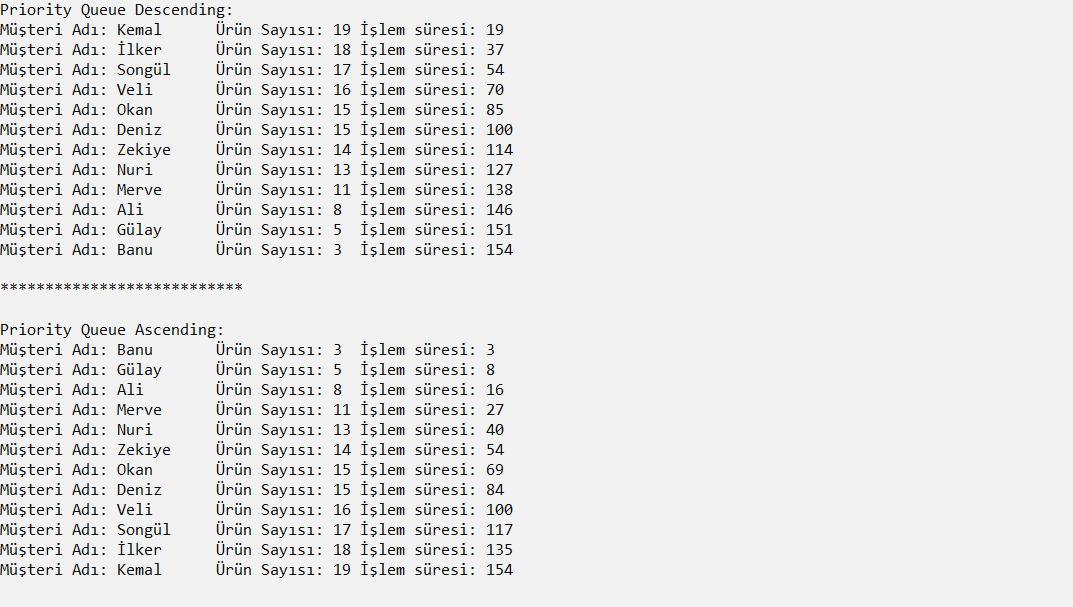
### Müşteri removeapq = aspq.apq();

### islemcountapq += removeapq.getÜrünSayısı();

### Console.WriteLine(removeapq + "\tİşlem süresi: " + islemcountapq);

### }

### 4.b.2 Ekran görüntüleri

  
4.b.3 Sözel olarak karşılaştırma

En avantajlı olan en kısa işi olan önce hallettiği için artan sırada öncelikli kuyruktur.Daha sonra Kuyruk ve en dezavantajlı ise en uzun işi olan önce halletiği için azalan sırada öncelikli kuyruktur.

## 4.c Öncelikli Kuyruk Tartışma

Eğer azalan sırada öncelikli kuyruk kullanılırsa işlem süresi ve maliyet maksimum olacaktır.Bu durumda öncelikli kuyruk kullanılmamalıdır.

## 4.d Öncelikli Kuyruk Öneri

Müşterilerin geliş sırasına göre gruplandırılıp (3’erli ,5’erli vb…) daha sonra gruplar içinde öncelikli kuyruk kullanılabilir.

# 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** | **8** | **Yapıldı.** |
| **Toplam** | **100** | **98** | **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.**