

**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**

05190000114, MAHMUT ÇELİK

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)

[1.b.2 Ekran görüntüleri 5](#_Toc91168781)

[1.b.3 Veri Yapıları ve Açıklama 5](#_Toc91168782)

[1.c Bileşik Veri Yapısı Bilgi Çıkarma 5](#_Toc91168783)

[1.c.1 Kaynak Kod 5](#_Toc91168784)

[1.c.2 Ekran görüntüleri 6](#_Toc91168785)

[2.a Yığıt 6](#_Toc91168786)

[2.a.1 Kaynak Kod 6](#_Toc91168787)

[2.a.2 Ekran görüntüleri 8](#_Toc91168788)

[2.b Kuyruk 8](#_Toc91168789)

[2.b.1 Kaynak Kod 8](#_Toc91168790)

[2.b.2 Ekran görüntüleri 10](#_Toc91168791)

[3.a Öncelikli Kuyruk Oluşturma 10](#_Toc91168792)

[3.a.1 Kaynak Kod 10](#_Toc91168793)

[3.a.2 Ekran görüntüleri 12](#_Toc91168794)

[3.b ArrayList ve Dizi altyapılarının karşılaştırılması 12](#_Toc91168795)

[4.a Öncelikli Kuyruk Güncelleme 12](#_Toc91168796)

[4.b Ortalama İşlem Tamamlama Süresi 14](#_Toc91168797)

[4.b.1 Kaynak Kod 14](#_Toc91168798)

[4.b.2 Ekran görüntüleri 16](#_Toc91168799)

[4.c Öncelikli Kuyruk Tartışma 17](#_Toc91168800)

[5. Özdeğerlendirme Tablosu 17](#_Toc91168801)

LİSTE, YIĞIT, KUYRUK ve ÖNCELİKLİ KUYRUK VERİ YAPILARI

NETBEANS, 12.2, and JAVA used

## 1.a Bileşik Veri Yapısı için Ön Çalışma

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 1.b Bileşik Veri Yapısı Kodlama ve Çalıştırma

### 1.b.1 Kaynak Kod

public class DENEME {

static final String[] MahalleAdi = {"Özkanlar", "Evka 3", "Atatürk", "Erzene", "Kazımdirik", "Mevlana","Doğanlar", "Ergene" }; //VERILEN ORNEK MAHALLELER

static final int[] TeslimatSayisi = {5,2,7,2,7,3,0,1}; //VERILEN ORNEK TESLIMAT SAYILARI

public static void main(String[] args) { //MAIN METODU

//GENEL KULLANIM ICIN ARRAYLISTI OLUSTURDUM

ArrayList<Mahalle> MotoKurye = new ArrayList<>(); //ARRAYLIST OLUSTURULMASI

for(int a = 0 ; a<MahalleAdi.length;a++){ //MAHALLENIN UZUNLUGU KADAR DONECEK BIR FOR DONGUSU HER BIR DONMEDE SIRAYLA MAHALLE ADI VE TESLIMAT SAYISI DIZIDEN ALINIR VE ARRAYLISTE YENI MAHALLE NESNESI OLUSTURULARAK EKLENIR

MotoKurye.add(new Mahalle(MahalleAdi[a],TeslimatSayisi[a]));

}

System.out.println("\*\*\*\*\*\*\*\*\*SORU 1.B VE 1.C İÇİN ÇÖZÜMLER\*\*\*\*\*\*\*\*\*\*");

bilesikVeriYapisindakiElemanlarinYazdirilmasi(MotoKurye);

}

}

public static void bilesikVeriYapisindakiElemanlarinYazdirilmasi(ArrayList<Mahalle> MotoKurye){ //MOTOKURYE ARRAYLISTI FOREACH DONGUSUYLE DOLASILIR VE HER ELEMAN MAHALLE CLASSINDAKI TO STRING METODU KULLANILARAK YAZDIRILIR

for(Mahalle kuryeMahallesi: MotoKurye){

System.out.println(kuryeMahallesi.toString());

}

}

TESLIMAT CLASSI

import java.util.Random;

public class Teslimat {

int maxSiparisAdedi = 100; //MAX SIPARIS ADEDI 100 OLARAK ALDIM

String[] yemekler = {"Pilav","Suşi","Türlü","Pizza","Tavuk Döner", "Patates Kızartması","Sosis","Baklava","Sütlaç","Hamsi Tava","Salata","Sucuklu Yumurta"};

private String yemekAdi; //TESLIMAT CLASSININ DEGISKENLERI

private int siparisAdedi;

public Teslimat(){

Random r = new Random();

this.yemekAdi = yemekler[r.nextInt(yemekler.length)];

this.siparisAdedi = r.nextInt(maxSiparisAdedi)+1;

}

public String getYemekAdi() { //GETTER VE SETTER

return yemekAdi;

}

public void setYemekAdi(String yemekAdi) {

this.yemekAdi = yemekAdi;

}

public int getSiparisAdedi() {

return siparisAdedi;

}

public void setSiparisAdedi(int siparisAdedi) {

this.siparisAdedi = siparisAdedi;

}

@Override

public String toString(){ //TO STRING METODU

return "Yemek Adı: " + yemekAdi + " Sipariş Adedi: " + siparisAdedi;

}

}

MAHALLE CLASSI

import java.util.ArrayList;

import java.util.List;

public class Mahalle {

private String mahalleAdi; //MAHALLE CLASSINA AIT DEGISKENLER

private ArrayList<Teslimat> genericList = new ArrayList<>();

public Mahalle(String mahalleAdi , int teslimatSayisi){

this.mahalleAdi = mahalleAdi;

for(int a = 0 ; a<teslimatSayisi;a++){

genericList.add(new Teslimat());

}

}

public String getMahalleAdi() { //GETTER VE SETTER

return mahalleAdi;

}

public void setMahalleAdi(String mahalleAdi) {

this.mahalleAdi = mahalleAdi;

}

public List<Teslimat> getGenericList() {

return genericList;

}

public void setGenericList(ArrayList<Teslimat> genericList) {

this.genericList = genericList;

}

public String teslimatYazdirilmasi(){ //HER MAHALLENIN TESLIMAT NESNELERININ YAZDIRILMASI TO STRING METODUNDA KULLANILIYOR

String metin= "";

for(Teslimat : genericList){

metin +=teslimat.toString()+" ";

}

return metin;

}

@Override

public String toString(){ // TO STRING METODU

return "Mahalle Adi: " + mahalleAdi + "\nYemekler;\n" +teslimatYazdirilmasi();

}

}

### 1.b.2 Ekran görüntüleri

SADECE 1.B NIN ÇIKTILARI VAR

metin, açık hava, gece göğü içeren bir resim

Açıklama otomatik olarak oluşturuldu

### 1.b.3 Veri Yapıları ve Açıklama

**ArrayList :** Mahalle classından olusturulan nesneleri tutmak için ArrayList kullanımı gerçekleştirdim. Her gelen yeni mahalleyi add metodu kullanarak sonuna ekleme yapıyor

**String Dizisi:** Örnekte verilen ya da başka örnek mahallelerin saklanması için gerekli dizi.

**Integer Dizisi**: Örnekte verilen ya da başka örnek teslim sayılarının saklanması için gerekli dizi.

**Generic List Oluşturmak İçin ArrayList:** Mahalle classından olusturulan nesnelerin içinde bulunacak teslimat classından olusturulan nesneleri saklamak için generic ArrayList kullandım.

## 1.c Bileşik Veri Yapısı Bilgi Çıkarma

### 1.c.1 Kaynak Kod

public static void main(String[] args) { //MAIN METODU

elemanSayisi(MotoKurye);

toplamTeslimatSayisi(MotoKurye); //1B DEKI KODLARA EK OLARAK MAIN METODU ICERISINDE BU KISIM VE METODLARA EK OLARAK ASAGIDAKI METODLAR EKLENIYOR.

public static void elemanSayisi(ArrayList<Mahalle> MotoKurye){

System.out.println("\nDinamik Dizide Bulunan Eleman Sayisi: " +MotoKurye.size());

}

public static void toplamTeslimatSayisi(ArrayList<Mahalle> MotoKurye){

int toplamTeslimatSayisi =0;

for (Mahalle kuryeMahallesi : MotoKurye){

for(Teslimat siparisİcerik : kuryeMahallesi.getGenericList()){

toplamTeslimatSayisi++;

}

}

System.out.println("Toplam Teslimat Sayısı: "+toplamTeslimatSayisi);

}

### 1.c.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 2.a Yığıt

### 2.a.1 Kaynak Kod

import java.util.ArrayList;

public class DENEME {

static final String[] MahalleAdi = {"Özkanlar", "Evka 3", "Atatürk", "Erzene", "Kazımdirik", "Mevlana","Doğanlar", "Ergene" }; //VERILEN ORNEK MAHALLELER

static final int[] TeslimatSayisi = {5,2,7,2,7,3,0,1}; //VERILEN ORNEK TESLIMAT SAYILARI

public static void main(String[] args) { //MAIN METODU

ArrayList<Mahalle> MotoKurye = new ArrayList<>(); //ARRAYLIST OLUSTURULMASI

for(int a = 0 ; a<MahalleAdi.length;a++){

MotoKurye.add(new Mahalle(MahalleAdi[a],TeslimatSayisi[a]));

}

Stack stack = new Stack(MahalleAdi.length); //STACK CLASSINDAN OLUSTURULAN stack adlı NESNE MAHALLE SAYISI KADAR UZUNLUKTADIR

for(Mahalle mahalle : MotoKurye){ //ARRAYLISTTEKI ELEMANLARIN FOR EACH DONGUSUYLE STACK E PUSHLANMASI

stack.push(mahalle);

}

while(!stack.isEmpty()){ //STACKTEKI ELEMANLARIN ISEMPTY KULLANARAK TUM YIGITIN POP KULLANILARAK CIKARTILIP YAZDIRILMASI

System.out.println(stack.pop().toString());

}

}

STACK CLASS I

public class Stack {

private int maxSize; //STACK CLASSI ICIN GEREKLI OZELLIKLER

private Mahalle[] stackList;

private int top;

public Stack(int maxSize){ //STACK CONSTRUCTOR U DIZI KULLANARAK YAPTIGIM ICIN (DIZI KULLANMAMA 3. SORU ICIN ANLADIM) MAX SIZE GELIR VE ATAMA YAPILIR

this.maxSize = maxSize;

stackList = new Mahalle[maxSize];

top = -1; //TOP -1 OLARAK BASLAR

}

public void push(Mahalle eklenecekMahalle){ //OLUSTURULAN STACK IN EN USTUNE ELEMAN EKLENIR ONCESINDE TOP 1 ARTTIRILIR

if(top == maxSize-1){

return;

}

stackList[++top] = eklenecekMahalle;

}

public Mahalle pop(){ // OLUSTURULAN STACKIN EN USTUNDEKI ELEMAN CIKARTILIR VE TOP 1 EKSILTILIR

if(top>=0){

return stackList[top--];

}

return null;

}

public Mahalle peek(){ //ODEV ICIN GEREKLI DEGIL FAKAT YAZDIM PEEK METODU ILE STACKIN EN USTUNDEKI ELEMANI CIKARMADAN GOREBILIRIZ

return stackList[top];

}

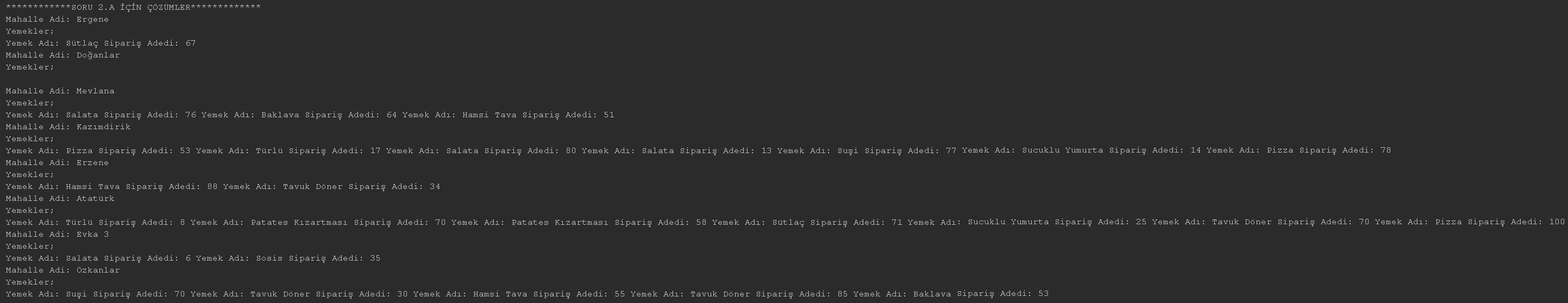
public boolean isEmpty(){ //IS EMPTY KULLANILARAK STACKIN BOS OLUP OLMADIGINI ANLAYABILIRIZ. TOP -1 OLUYOR ISE RETURN TRUE GONDERIR

return (top ==-1);

}

}

### 2.a.2 Ekran görüntüleri



## 2.b Kuyruk

### 2.b.1 Kaynak Kod

import java.util.ArrayList;

public class DENEME {

static final String[] MahalleAdi = {"Özkanlar", "Evka 3", "Atatürk", "Erzene", "Kazımdirik", "Mevlana","Doğanlar", "Ergene" }; //VERILEN ORNEK MAHALLELER

static final int[] TeslimatSayisi = {5,2,7,2,7,3,0,1}; //VERILEN ORNEK TESLIMAT SAYILARI

public static void main(String[] args) { //MAIN METODU

ArrayList<Mahalle> MotoKurye = new ArrayList<>(); //ARRAYLIST OLUSTURULMASI

for(int a = 0 ; a<MahalleAdi.length;a++){

MotoKurye.add(new Mahalle(MahalleAdi[a],TeslimatSayisi[a]));

}

Queue queue = new Queue(MahalleAdi.length);

for(Mahalle mahalle : MotoKurye){ //FOREACH DONGUUSYLE MOTOKURYE ARRAYLISTINDEKI ELEMANLAR DOLASILARAK KUYRUGA ENQUEUE METODU SAYESINDE EKLENIRLER

queue.enQueue(mahalle);

}

while(!queue.isEmpty()){ // KUYRUKTA ELEMAN KALMADIGINI ISEMPTY ILE KONTROL EDEREK ELEMANLAR CIKARILIR VE YAZDIRILIR

System.out.println(queue.deQueue().toString());

}

}

}

QUEUE CLASS I

public class Queue {

private int maxSize; //QUEUE CLASSI ICIN GEREKLI OZELLIKLER

private int head;

private int tail;

private Mahalle[] queueList;

private int elementCount;

public Queue(int maxSize){ //DIZI KULLANARAK OLUSTURDUGUM ICIN MAXSIZE GELIR VE GEREKLI ATAMALAR YAPILIR

this.maxSize = maxSize;

head = 0;

tail = -1;

elementCount = 0;

queueList = new Mahalle[maxSize];

}

public void enQueue(Mahalle eklenecekMahalle){ // ENQUEUE KULLANARAK KUYRUGUN SONUNA ELEMAN EKLEMESI YAPILIR DAIRESEL KUYRUK SEKLINDE YAZDIM FAKAT DIZI ELEMAN UZUNLUGUNDA OLUSTURULDUGU ICIN OLMASA DA CALISABILIR

if(tail == maxSize-1){

tail = -1;

}

queueList[++tail] = eklenecekMahalle; //KUYRUK 1 ARTTILIR VE EKLEME YAPILIR

elementCount++; //ELEMAN SAYISI 1 ARTTIRILIR

}

public Mahalle deQueue(){

Mahalle temp = queueList[head++]; //ELEMAN ALINIR VE HEAD 1 ARTTIRILIR

if(head == maxSize){ //DAIRESEL OLMAYI SAGLAYAN IF KISMI

head =0;

}

elementCount--; //ELEMAN SAYISI 1 AZALTILIR VE ALINAN ELEMAN RETURNLE GERI GONDERILIR

return temp;

}

public boolean isEmpty(){ //ISEMPTY KULLANARAK KUYRUKTA ELEMAN OLUP OLMADIGI OGRENILIR ELEMAN SAYISI 0 SA KUYRUK BOSTUR RETURN TRUE OLUR

return (elementCount ==0);

}

public boolean isFull(){

return (elementCount ==maxSize);

}

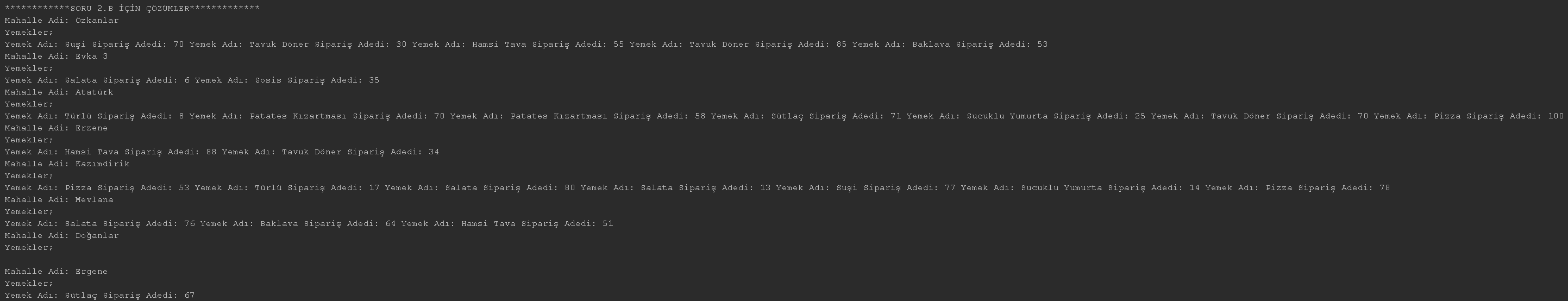
public int size(){ //SIZE IZE ELEMAN SAYISINI DONDURUR

return elementCount;

}

}

### 2.b.2 Ekran görüntüleri



## 3.a Öncelikli Kuyruk Oluşturma

### 3.a.1 Kaynak Kod

import java.util.ArrayList;

public class DENEME {

static final String[] MahalleAdi = {"Özkanlar", "Evka 3", "Atatürk", "Erzene", "Kazımdirik", "Mevlana","Doğanlar", "Ergene" }; static final int[] TeslimatSayisi = {5,2,7,2,7,3,0,1}; //VERILEN ORNEK TESLIMAT SAYILARI

public static void main(String[] args) { //MAIN METODU

ArrayList<Mahalle> MotoKurye = new ArrayList<>(); //ARRAYLIST OLUSTURULMASI

for(int a = 0 ; a<MahalleAdi.length;a++){

MotoKurye.add(new Mahalle(MahalleAdi[a],TeslimatSayisi[a]));

}

PriorityQUEUE priorityQueue = new PriorityQUEUE(); //PRIORITYQUEUE CLASSINDAN NESNE OLUSTURULUYOR

for(Mahalle mahalle : MotoKurye){

priorityQueue.enQueue(mahalle);

}

while(!priorityQueue.isEmpty()){

System.out.println(priorityQueue.deQueue().toString());

}

}

}

PRIORITYQUEUE CLASS I

import java.util.ArrayList;

public class PriorityQUEUE {

ArrayList<Mahalle> priorityQUEUEList ;

private int elementNumber;

public PriorityQUEUE(){

this.priorityQUEUEList = new ArrayList<>();

this.elementNumber =0;

}

public void enQueue(Mahalle eklenecekMahalle){ //ENQUEUE METODU ILE OLUSTURULAN ARRAYLISTE ELEMAN EKLENECEK GELEN ELEMAN OLUSTURULAN ARRAYLISTIN SONUNA EKLENIR

priorityQUEUEList.add(eklenecekMahalle);

elementNumber++;

}

public Mahalle deQueue(){

Mahalle silinecekMahalle = priorityQUEUEList.get(0);

if(priorityQUEUEList.size()>1){

int maxTeslimatSayi =0;

for(Mahalle mahalleler : priorityQUEUEList){

if(mahalleler.getGenericList().size()>maxTeslimatSayi){

silinecekMahalle = mahalleler;

maxTeslimatSayi = mahalleler.getGenericList().size();

}

}

}

priorityQUEUEList.remove(silinecekMahalle); // EN COK TESLIMAT YAPILAN MAHALLE ARRAYLISTTEN SILINIR

//System.out.println("SİLİNEN MAHALLEYE YAPILAN TESLIMAT ADEDI: "+silinecekMahalle.getGenericList().size()); BENIM ICIN GEREKLIYDI

return silinecekMahalle; //EN COK TESLIMAT YAPILAN MAHALLE GERI GONDERILIR

}

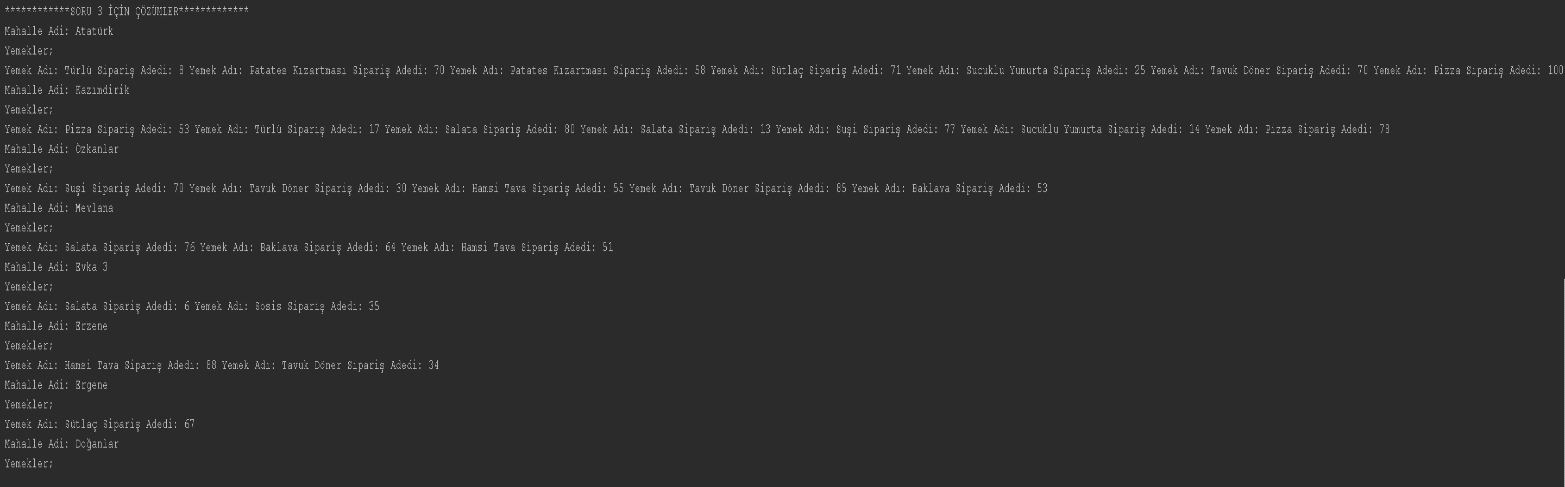
public boolean isEmpty(){ //ARRAYLISTIN BOS OLUP OLMADIGINI OGRENMEMIZI SAGLAYAN METOD

return (priorityQUEUEList.size()==0);

}

}

### 3.a.2 Ekran görüntüleri



## 3.b ArrayList ve Dizi altyapılarının karşılaştırılması

Priority Queue gerçekleştiriminde ArrayList yerine Dizi kullansaydık;

**1-**Belirli bir index sayısında dizi oluşturmamız gerekeceği için taşma gerçekleşebilirdi.

**2-**Eleman sildikten sonra kalan elemanlar için kaydırma yapmamız gerekirdi.Bu da oldukça masraflı bir durum

## 4.a Öncelikli Kuyruk Güncelleme

ODEV DOKUMANINDAKI 4B ICIN OLAN KODLAR

package queuevspriorityqueue;

import java.util.ArrayList;

public class FOURTHQUESTIONwithPRIORITYQUEUE { //4.SORUNUN B SIKKI ICIN GEREKLI CLASS.

static final int[] musteriSepetleri = {6,7,2,1,12,5,3,7,4,2}; //ORNEK VERILEN MUSTERI SEPETLERININ DIZISI

static int urunOkumaSuresi = 3; //VERILEN URUN OKUMA SURE SABITI

public static void main(String[] args){

PRIORITYQUEUE priorityQueue = new PRIORITYQUEUE(); //CLASSTAN NESNE OLUSTURULMASI

for(int eklenecekSayi : musteriSepetleri){

priorityQueue.enQueue(eklenecekSayi);

}

urunlerinOkutulmasi(priorityQueue); //URUNLERIN OKUTULMASI METODUYLA HER MUSTERININ BEKLEME SURESI VE TOPLAM BEKLEME SURESI ORTALAMASI HESAPLANILARAK YAZDIRILIR

}

public static void urunlerinOkutulmasi(PRIORITYQUEUE priorityQueue){ //BU METODDA TUM ELEMANLAR ARTAN SIRAYLA OLACAK SEKILDE YAPIDAN CIKARILIR VE HESAPLAMALAR YAPILIR

int toplamMusteri =0;

int herMusteriBeklemeSuresi =0; //GEREKLI DEGISKENLER

int toplamIslemTamamlanmaSuresi =0;

System.out.println("PRIORITY QUEUE DIZILIMINE GORE SONUCLAR");

while(!priorityQueue.isEmpty()){ //PRIORITYQUEUE NESNESI BOS OLASIYA KADAR CALISACAK WHILE DONGUSU ICINDE DEQUEUE METODU ILE ELEMANLAR CIKARILIR

toplamMusteri++;

int silinecekMusteridekiToplamUrunSayisi = priorityQueue.deQueue();

herMusteriBeklemeSuresi += (silinecekMusteridekiToplamUrunSayisi\*urunOkumaSuresi );

toplamIslemTamamlanmaSuresi+= herMusteriBeklemeSuresi; //MUSTERILERIN BEKLEME SURELERI EKLENIR

System.out.println(toplamMusteri+".Müşterinin işlem tamamlanma süresi: " + (herMusteriBeklemeSuresi));

}

ortalamaIslemTamamlanmaYazdirilmasi(toplamIslemTamamlanmaSuresi, toplamMusteri);

}

public static void ortalamaIslemTamamlanmaYazdirilmasi(int toplamIslemTamamlanmaSuresi, int toplamMusteri){

double ortalamaIslemTamamlanmaSuresi = (toplamIslemTamamlanmaSuresi/(double)toplamMusteri);

System.out.println("BU KASADAKI MUSTERILERIN ORTALAMA ISLEM TAMAMLANMA SURESI: " + ortalamaIslemTamamlanmaSuresi);

}

}

PRIORITYQUEUE CLASSI

class PRIORITYQUEUE{

ArrayList<Integer> queueList; //ICINDE ARRAYLIST KULLANARAK YAZDIM HEAD VE TAIL ASLINDA GEREKLI DEGIL

private int elementNumber;

public PRIORITYQUEUE(){ //CLASSIN CONSTRUCTORU GEREKLI ATAMALAR YAPILDI

this.elementNumber= 0;

this.queueList = new ArrayList<>();

}

public void enQueue(int eklenecekSayi){ //ENQUEUE METODU ILE YAPININ SONUNA ELEMAN EKLENIR VE ARTTIRMALAR YAPILIR

queueList.add(eklenecekSayi);

elementNumber++;

}

public int deQueue(){

int silinecekSayi = queueList.get(0);

if(queueList.size()>1){

int enKucukSayi = silinecekSayi;

for(int kuyruktakiSayilar: queueList){ //ARRAYLISTIN DOLASILMASI

if(kuyruktakiSayilar<enKucukSayi){

silinecekSayi = kuyruktakiSayilar;

enKucukSayi = kuyruktakiSayilar;

}

}

}

queueList.remove(queueList.indexOf(silinecekSayi)); //EN KUCUK SAYIYA SAHIP MUSTERI ARRAYLISTTEN INDEX KULLANILARAK SILINIR

elementNumber--;

return silinecekSayi; //SILINECEK MUSTERININ URUN SAYISI GERI GONDERILIR

}

public boolean isEmpty(){ //ELEMAN SAYISI 0 OLDUGUNDA TRUE GONDERECEK ISEMPTY METODU

return (elementNumber==0);

}

}

## 4.b Ortalama İşlem Tamamlama Süresi

### 4.b.1 Kaynak Kod

package queuevspriorityqueue;

public class FOURTHQUESTIONwithQUEUE { //4.sorunun QUEUE ILE GERCEKLESTIRILMESI YANI A SIKKI

static final int[] musteriSepetleri = {6,7,2,1,12,5,3,7,4,2}; //ORNEK VERILEN MUSTERILERIN SEPETLERI DIZISI

static int urunOkumaSuresi = 3; //VERILEN URUN OKUMA SURESI 3 SN

public static void main(String[] args) { //MAIN METODU

QUEUE queue = new QUEUE(musteriSepetleri.length); //QUEUE CLASSINDAN NESNE OLUSTURULUYOR OLUSTURULAN QUEUE DIZIDEN OLUSTUGU ICIN MUSTERI SAYISI KADAR UZUNLUKTA OLUSTURDUM

for(int sayi: musteriSepetleri){ //MUSTERILERIN SEPETLERI FOR EACH DONGUSUYLE DOLASILARAK KUYRUK YAPISINA EKLENIR

queue.enQueue(sayi);

}

urunlerınOkutulmasi(queue); //URUNLERIN OKUTULMASI METODU ILE HER MUSTERININ NE KADAR BEKLEDIGI VE ORTALAMA BEKLEME SURESI HESAPLANIR

}

public static void urunlerınOkutulmasi(QUEUE queue){ //KAC TANE MUSTERI OLDUGU HER BIRININ BEKLEME SURESI VE TOPLAM BEKLEME SURELERI HESAPLANIR

int kacinciMusteri =0; //GEREKLI DEGISKENLER

int herMusteriBeklemeSuresi =0;

int toplamIslemTamamlanmaSuresi =0;

System.out.println("QUEUE DIZILIMINE GORE SONUCLAR");

while(!queue.isEmpty()){ //KUYRUKTAKI TUM ELEMANLAR CIKARTILASIYA KADAR CALISACAK WHILE DONGUSU

kacinciMusteri++; // HER CIKARTILAN ELEMANDA MUSTERI SAYISI 1 ARTAR

int musteridekiUrunSayisi = queue.deQueue(); //KUYRUKTAKI ELEMAN CIKARILIR

herMusteriBeklemeSuresi+= (musteridekiUrunSayisi\*urunOkumaSuresi );

toplamIslemTamamlanmaSuresi+= herMusteriBeklemeSuresi;

System.out.println(kacinciMusteri+".Müşterinin işlem tamamlanma süresi: " + (herMusteriBeklemeSuresi));

}

ortalamaIslemTamamlanmaYazdirilmasi(toplamIslemTamamlanmaSuresi, kacinciMusteri);

}

public static void ortalamaIslemTamamlanmaYazdirilmasi(int toplamIslemTamamlanmaSuresi, int kacinciMusteri){

double ortalamaIslemTamamlanmaSuresi = (toplamIslemTamamlanmaSuresi/(double)kacinciMusteri); //HER MUSTERININ BEKLEME SURELERININ TOPLANDIGI DEGISKENI MUSTERI SAYISINA BOLEREK HESAPLANIR

System.out.println("BU KASADAKI MUSTERILERIN ORTALAMA ISLEM TAMAMLANMA SURESI: " + ortalamaIslemTamamlanmaSuresi);

}

}

QUEUE CLASSI

class QUEUE{

private int head; //GEREKLI OZELLIKLER DIZI KULLANDIM AKSI BELIRTILMEMIS

private int tail;

private int elementNumber;

private int[] queueList;

private int maxSize;

public QUEUE(int maxSize){ //CLASS A AIT CONSTRUCTOR. DIZI KULLANDIGIM ICIN ICINDE MAXSIZE GELECEK VE BU UZUNLUGA GORE KUYRUK DIZISI OLUSTURULACAK

this.maxSize = maxSize;

this.head = 0 ;

this.tail = -1;

this.elementNumber = 0;

this.queueList = new int[maxSize];

}

public void enQueue(int eklenecekSayi){

if(tail == maxSize-1){

tail = -1;

}

queueList[++tail] = eklenecekSayi; //TAIL 1 ARTTIRILIR VE EKLENECEK SAYI EKLENIR ELEMAN SAYISI 1 ARTTIRILIR

elementNumber++;

}

public int deQueue(){

int silinecekSayi = queueList[head++]; //SILINECEK SAYI ICIN HEADDEN ELEMAN ALINIR VE HEAD 1 ARTTIRILIR

if(head == maxSize){

head =0;

}

elementNumber--;

return silinecekSayi;

}

public boolean isEmpty(){ //ELEMAN SAYISI 0 OLDUGUNDA ISEMPTY METODU TRUE DONDURUR

return (elementNumber==0);

}

}

### 4.b.2 Ekran görüntüleri

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu  
4.b.3 Sözel olarak karşılaştırma

Priority Queue ile yapılan gerçekleştirimde, Queue ile yapılan gerçekleştirime göre zamandan kazanç bulunmaktadır çünkü küçük işlemleri olan müşteriler daha az bekleyeceği için ortalama düşürülür. Örnek verilecek olursak 6 müşterimiz olsun. İlk 5 müşterinin işi uzun sürsün son müşterinin kısa sürsün. Queue ile gerçekleştirilirse son müşteri ilk 5 müşterinin işi kadar bekleyecektir yani ortalama daha da artacaktır. Priority Queue ile gerçekleştirilirse Queue deki son müşteri burada ilk müşteri olacağı için ve geri kalan müşteri kendi içinde aynı ek olarak da ilk müşterinin kısa işlemi kadar bekleyeceği için ortalama azalmış olur.

## 4.c Öncelikli Kuyruk Tartışma

Dezavantaj olarak; Queue sıralamasında ilk başta olması gereken bir kişi, Priority Queue sıralamasında son kişi olabilir. Bunun tersi olarak Queue sıralamasında sonda olması gereken bir kişi, Priority Queue sıralamasında ilk kişi olabilir.

Tüm elemanların aynı önceliğe sahip olduğu durumlarda Priority Queue kullanılamaz.

# 5. Özdeğerlendirme Tablosu

**Özdeğerlendirme Tablosu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 2 Maddeleri** | **Puan** | **Tahmini Not** | **Açıklama** |
| **1 a) A4 Ön çalışma** | **20** | **18** | YAPILDI(KAĞIDA KURŞUN KALEMLE YAZILARAK) |
| **1 b) Kaynak kod, ekran görüntüsü, veri yapısının elemanlarının listelenmesi** | **20** | **20** | **YAPILDI** |
| **1 c) Kaynak kodlar, Liste sayısı, veri yapısındaki toplam teslimat sayısı** | **5** | **5** | YAPILDI VE VERİLEN ÖRNEKLE AYNI ÇIKTIĞI İÇİN DOĞRU |
| **2 a) Yığıt kaynak kod ve ekran görüntüleri** | **5** | **5** | **YAPILDI** |
| **2 b) Kuyruk kaynak kod ve ekran görüntüleri** | **5** | **5** | **YAPILDI** |
| **3 a) Öncelikli Kuyruk kod ve ekran görüntüleri** | **10** | **10** | **YAPILDI** |
| **3 b) ArrayList ve Dizi altyapılarının karşılaştırılması** | **5** | **3** | YAPILDI FAKAT AÇIKLAMANIN DOĞRULUĞUNDAN TAM OLARAK EMİN DEĞİLİM |
| **4) Kod, sonuçlar tablosu, ekran görüntüleri, karşılaştırma ve soruların cevapları.** | **20** | **18** | YAPILDI VE SONUÇLAR ELLE HESAPLAMAYLA AYNI ÇIKIYOR YANİ DOĞRU |
| **5) Özdeğerlendirme Tablosu** | **10** | **10** | **YAPILDI** |
| **Toplam** | **100** | **94** |  |

**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.**