

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2024–2025 FALL SEMESTER**

**PROJECT-2 REPORT**

**(LIST, STACK, QUEUE AND PRIORITY QUEUE)**

**DELIVERY DATE**

07/12/2024

**PREPARED BY**

05230000334, İrfan Burak Ege

İçindekiler

[1) Ege Denizi Balıkları Listesi 1](#_Toc182335418)

[1.a Sınıf Oluşturma 2](#_Toc182335419)

[1.a.1 EgeDeniziB sınıfı tanımı, açıklaması ve kod bloğu 2](#_Toc182335420)

[1.a.2 Ekran görüntüleri 3](#_Toc182335421)

[1.b Generic List Oluşturma 3](#_Toc182335422)

[1.b.1 Kodlar 3](#_Toc182335423)

[1.b.2 Ekran görüntüleri 5](#_Toc182335424)

[1.c Generic List Dizisi 5](#_Toc182335425)

[1.c.1 Kodlar 5](#_Toc182335426)

[//4 elemanlı Generic List dizisinin kodu. 5](#_Toc182335427)

[1.c.2 Ekran görüntüleri 5](#_Toc182335428)

[1.d Grup Elemanlarının Yazdırılması 5](#_Toc182335429)

[1.d.1 Kodlar 5](#_Toc182335430)

[1.d.2 Ekran görüntüleri 6](#_Toc182335431)

[2. Yığıt ve Kuyruk 10](#_Toc182335432)

[2.a Yığıt Yapısı 10](#_Toc182335433)

[2.a.1 Kodlar 10](#_Toc182335434)

[2.a.2 Ekran görüntüleri 11](#_Toc182335435)

[2.b Kuyruk Yapısı 14](#_Toc182335436)

[2.b.1 Kodlar 14](#_Toc182335437)

[2.b.2 Ekran görüntüleri 15](#_Toc182335438)

[3. Öncelikli Kuyruk 19](#_Toc182335439)

[3.a Öncelikli Kuyruk Sınıfı (Ekle, Sil, bosMu Metotları) 19](#_Toc182335440)

[3.b Ekran görüntüleri 20](#_Toc182335441)

[4. Market Kuyruğu ve Karşılaştırma 24](#_Toc182335442)

[4.a Kuyruk Uygulaması 24](#_Toc182335443)

[4.a.1 Kodlar 24](#_Toc182335444)

[4.a.2 Ekran görüntüleri 25](#_Toc182335445)

[4.b Öncelikli Kuyruk Güncellemesi 25](#_Toc182335446)

[4.b.1 Kodlar 25](#_Toc182335447)

[4.b.2 Ekran görüntüleri 27](#_Toc182335448)

[4.c Queue ve PQ Karşılaştırması 27](#_Toc182335449)

[5. Yorum Sorusu 27](#_Toc182335450)

[Öz değerlendirme Tablosu 28](#_Toc182335451)

# 1) Ege Denizi Balıkları Listesi

//Bu proje, farklı veri yapıları kullanarak balık yönetiminin geliştirilmesinin bir örneğidir. Proje, balıkların isimleri, boyutları ve ortamlarınınve benzeri bilgiler içermektedir. Balıklar farklı veri yapılarına (yığıt, kuyruk, öncelikli kuyruk) eklenir ve bu yapılar kullanılarak balıkların bilgileri kaydedilir.Ayrıca müşteri işlemleri için kuyruk ve öncelikli kuyruk kullanılarak müşterilerin bekleme süreleri bulunmuştur.

//Yığıt (Stack):

Yığıt, LIFO (Last In, First Out) prensibiyle çalışan bir veri yapısıdır.

Projede, balıklar yığıta eklenir ve ardından çıkarılarak sıralı şekilde yazdırılır.

Kuyruk (Queue):

Kuyruk, FIFO (First In, First Out) ile çalışır, yani ilk eklenen öğe ilk çıkar.

Projede, balıklar kuyruk yapısına eklenir ve sırasıyla çıkarılarak yazdırılır.

Öncelikli Kuyruk (Priority Queue):

Öncelikli kuyruk, her öğeye bir öncelik değeri atar ve yüksek önceliğe sahip öğeler önce işleme alınır.

Temel amaç, belirli öğelere öncelik vererek işlemleri sıraya koymaktır. Bu yapı, acil durumlar ve öncelikli işlemler için kullanılır.

Projede, balıklar öncelikli kuyrukta sıraya girer ve önceliğe göre çıkarılır.

Liste (List):

Liste, öğeleri sıralı bir şekilde tutan ve dinamik olarak büyüyebilen bir veri yapısıdır.

Projede, balıklar listede tutulur ve gruplar halinde işlenir.

//Visual Stuido 2022, C#

## 1.a Sınıf Oluşturma

### 1.a.1 EgeDeniziB sınıfı tanımı, açıklaması ve kod bloğu

EgeDeniziB sınıfı, bir balığın adı, diğer adı, boyutu, bilgi ve ortam bilgilerini tutan bir yapıdır. Bu sınıfın özellikleri şunlardır:

BalıkAdı: Balığın adı

DiğerAdı: Balığın diğer adı

Boyut: Balığın boyutu

Bilgi: Balığa ait tüm bilgiler

Ortam: Balığın yaşadığı ortamları içeren bir liste

Sınıfın iki yapıcı metodu vardır: bir boş değerli yapıcı bu yapıcı sil methodunda kullanmak için ve bir parametreli yapıcı. ToString metodu ise sınıfın bilgilerini düzenli bir şekilde yazdırmak için.

class EgeDeniziB

{

public string BalıkAdı, DiğerAdı, Boyut, Bilgi;

public List<string> Ortam { get; set; }//ortam sayısı değişebileceği için liste olarak oluşturduk

public EgeDeniziB()

{

BalıkAdı = "";

DiğerAdı = "";

Boyut = "";

Bilgi = "";

Ortam = new List<string>();

}

public EgeDeniziB(string balıkAdı, string diğerAdı, string boyut, string bilgi, List<string> ortam)//constructor

{

BalıkAdı = balıkAdı;

DiğerAdı = diğerAdı;

Boyut = boyut;

Bilgi = bilgi;

Ortam = ortam;

}

public override string ToString()

{

// ortam listesini string olarak birleştir

string ortamBilgisi = (Ortam != null && Ortam.Count != 0) ? (string.Join(", ", Ortam)) : ("Bilgi yok");

// tüm bilgileri birleştirdik ve döndürdük

return $"BALIK ADI: {this.BalıkAdı}\n" +

$"DİĞER ADI: {this.DiğerAdı}\n" +

$"BOYUT: {this.Boyut}\n" +

$"BİLGİ: {this.Bilgi}\n" +

$"ORTAM: {ortamBilgisi}";

}

}

### 1.a.2 Ekran görüntüleri

//Nesnelerin doğru oluşturulduğuna dair debug ekran görüntüsü.

A screen shot of a computer

Description automatically generated

## 1.b Generic List Oluşturma

### 1.b.1 Kodlar

BalıkStringleriniAyırma methodunda textteki balıkları tek tek bir diziye aktarıyoruz bunun için de başlarındaki sıralama sayılarını kullanıyoruz.

AdAyırma methodunda bize verilen ad satırında eğer “(“ var ise diğer adını ayırıp ikisini birlikte bir diziye aktarıyoruz eğer yok ise dizinin 1. Diğer adı indexini “Diğer Adı yok” şeklinde atıyoruz

BalıkClassOluştur methodunda verilen balık hakkındaki text verisini işleyerek ad diğerAd bilgi boyut ortam verilerini text dosyasından çıkartıp EgeDeniziB objesi döndürüyor.

Main içerisinde de text dosyasını BalıklarAyrılmış[] Dizisine dönüştürüyoruz. Dahasonra bu diziyi kullanarak for döngüsü içinde balıkları tek tek BalıkClassOluştur ile oluşturup yeni yarattığımız balıkListesi listesine ekliyoruz.

static string[] BalıkStringleriniAyırma(string text)// text dosyasını sıralamaya göre ayıran method

{

string[] BalıklarAyrılmış = new string[38];// son array

string[] textSatırAyrılmış = text.Split("\n\n"); satır başlarındaki sıralama sayılarını kontrol etmek için tek tek her harfi dönmemek için oluşturduğumuz array

for (int i = 1; i < 39; i++)

{

int index1 = 0, index2 = 0;

if (i < 10)//s sıralama sayısına eşit ise ("1." den "9." kadar) indexleri alınıyor

{

index1 = text.IndexOf(i + ".") + 2; //2 ekleyerek baştaki sayıyı atıyoruz ve aradığımız sayının başlama indexini buluyoruz

index2 = text.IndexOf((i + 1) + ".") - 4;

}

else if (i < 38)//10,11....37

{

index1 = text.IndexOf(i + ".") + 3; // 4 ekliyoruz çünkü sayılar çift basamaklı ve 1 boşluk da siliyoruz aynı hizada olmaları için

index2 = text.IndexOf((i + 1) + ".") - 4;//sonraki sayının başladığı index \n\n çıkarmak için 4 çıkarıyoruz

}

else //38. eleman için

{

index1 = text.IndexOf(i + ".") + 3;

index2 = text.Length - 1;//39. eleman olmadığı için sona kadar gidyoruz

}

int length = index2 - index1 + 1;//text parçasının uzunluğu

BalıklarAyrılmış[i - 1] = text.Substring(index1, length);

}

return BalıklarAyrılmış;

}

static string[] AdAyırma(string adSatırı)//iki adı varsa diğer adı alınıyor

{

string[] AdSatırı = adSatırı.Split('(');//eğer ( varsa ayırılıyor

string[] AdSatırıAyrılmış = new string[2];

AdSatırıAyrılmış[0] = AdSatırı[0];

AdSatırıAyrılmış[1] = AdSatırı.Length > 1 ? AdSatırı[1].Replace(')', ' ').Trim() : "Diğer Adı Yok";

return AdSatırıAyrılmış;

}

static EgeDeniziB BalıkClassOluştur(string balık)//verilen stringi balık classına dönüştüren method

{

string[] balıkSatırAyrılmış = balık.Split("\n\n");//balığın satırları ayrılıyor

string AdSatırı = balıkSatırAyrılmış[0];//ilk ad satırı

string[] AdSatırıAyrılmış = AdAyırma(AdSatırı);//diğer adı ayrılıyor

string ad = AdSatırıAyrılmış[0];

string DiğerAd = AdSatırıAyrılmış[1];

string boyut = "", bilgi = "";

List<string> bölgeler = new List<string>();

for (int i = 1; i < balıkSatırAyrılmış.Length; i++)//tüm satırları döndüğümüz for döngüsü

{

string satır = balıkSatırAyrılmış[i];

bilgi += satır;//bilgi kısmına satır her türlü ekleniyor

// içinde boyut ile ilgili kelimeler geçen paragraflar boyut paragrafına ekleniyor

if (satır.Contains("boyut")||satır.Contains("cm")||satır.Contains("uzunluk")||satır.Contains("uzunluğa"))

{

boyut += satır;

}

//ortam için bölgelerin paragrafta geçtiğini kontrol ediyoruz

if (satır.Contains("Ege") && !bölgeler.Contains("Ege")) { bölgeler.Add("Ege"); }

else if (satır.Contains("Akdeniz") && !bölgeler.Contains("Akdeniz")) { bölgeler.Add("Akdeniz"); }

else if (satır.Contains("Karadeniz") && !bölgeler.Contains("Karadeniz")){bölgeler.Add("Karadeniz");}

else if (satır.Contains("Marmara") && !bölgeler.Contains("Marmara")) { bölgeler.Add("Marmara"); }

}

return new EgeDeniziB(ad, DiğerAd, boyut, bilgi, bölgeler);

}//balık class oluştur kapanışı

Console.WriteLine();

Console.WriteLine("-----------------------------------------------------------------");

Console.WriteLine();

}

}

static void Main(string[] args)

{

//balıkların bilgisini içeren text

string text = "Burada balık bilgilerini içeren siteden kopyaladığımız yazı oluyor çok uzun olduğu için onu kaldırdım”

string[] BalıklarAyrılmış = BalıkStringleriniAyırma(text);//balıkları 1. 2. olarak stringlere ayıran methodu çalıştırma

List<EgeDeniziB> balıkListesi = new List<EgeDeniziB>();// Balık generic list oluşturma

for (int i = 0; i < BalıklarAyrılmış.Length; i++)//balıkları EgeDeniziB objesi haline getirip listeye atma döngüsü

{

EgeDeniziB balık = BalıkClassOluştur(BalıklarAyrılmış[i]);// balık nesnesi oluşturma

balıkListesi.Add(balık);

}

### 1.b.2 Ekran görüntüleri

//Konsol çıktısı istenmemekte

## 1.c Generic List Dizisi

## 1.c.1 Kodlar

### //4 elemanlı Generic List dizisinin kodu.

10 lu gruplamak için iç içe iki for döngüsü kullanıldı. İlk for döngüsünde 4 kez dönüğ her döngüde o gruba aktarılacak balıklar için liste oluğturuluyor. İçerdeki for döngüsünde ise 10 kez dönüp eğer balıkListesinde sona gelinmediyse gruba ekleniyor içteki for döngüsü bitince grup, bileşik listeye ekleniyor.

Main içerisinde:

List<List<EgeDeniziB>> BileşikBalıkListesi = new List<List<EgeDeniziB>>(4);//balık listesini 10 10 10 8 şeklinde atacağımız bileşik liste

for (int i = 0; i < 4; i++)//4 grup haline getirmek için döngü

{

List<EgeDeniziB> grup = new List<EgeDeniziB>(10);//Grup listesini oluşturma

for (int j = 0; j < 10; j++)//10 eleman ekleme döngüsü

{

int index = i \* 10 + j;

if (index < balıkListesi.Count) // boyut kontrolü

{ grup.Add(balıkListesi[index]); }//gruba ekleme

}

BileşikBalıkListesi.Add(grup);// grubu bileşik listeye ekleme

}

### 1.c.2 Ekran görüntüleri

A black background with white lines

Description automatically generated with medium confidence

## 1.d Grup Elemanlarının Yazdırılması

### 1.d.1 Kodlar

// Grup dizisindeki balıkların bilgilerinin listelenmesi kodu.

BileşikListeyiYazdır methodunda foreach döngüsü ile parametre olarak verilen bağlı liste içindeki her bir listeyi dönerken içindeki balık elemanını bir değişkene atayıp bu değişkenin bilgilerini yazdırıyoruz eğer diğer adı var ise kontrol edilip listenin döngüsünde oluşturduğumuz DiğerAdSayısı artırılıyor listeyi dönme bittiğinde ise bu grup için toplam DiğerAdSayısı yazdırılıyor.

static void BileşikListeyiYazdır(List<List<EgeDeniziB>> BListe)

{

int count = 0, DiğerAdSayısı;

foreach (List<EgeDeniziB> Liste in BListe)

{

int sayac = 0;

DiğerAdSayısı = 0;

Console.WriteLine("GRUP " + count + " İÇİN BİLGİLER: \n");

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

{

EgeDeniziB balık = Liste[i];

Console.WriteLine("BALIK " + (i + 1));

if (balık.DiğerAdı != "Diğer Adı Yok") { DiğerAdSayısı++; }//diğer adı varsa sayaç artıyor

Console.WriteLine(balık.ToString());

Console.WriteLine();

if(balık.Boyut.Equals(null)){ sayac++; }

}

count++;

Console.WriteLine("Bu gruptaki Diğer Ad sayısı: " + DiğerAdSayısı);

Console.WriteLine();

Console.WriteLine("-----------------------------------------------------------------");

Console.WriteLine();

}

}

### 1.d.2 Ekran görüntüleri

// Konsol çıktısına ait ekran görüntüleri

A screen shot of a computer code

Description automatically generated

A screen shot of a computer screen

Description automatically generated

## A black background with blue dots

## A screenshot of a computer screen

## 2. Yığıt ve Kuyruk

### 2.a Yığıt Yapısı

### 2.a.1 Kodlar

// Yığıt yapısının güncellenmesi ve EgeDeniziB sınıfı nesnelerinin yığıta eklenmesi kodu.

class BalıkYığıtı

{

private int maxSize; // arrayin boyutu

private EgeDeniziB[] stackArray;

private int top; // en üst eleman

public BalıkYığıtı(int size) // constructor

{

maxSize = size; //Arrayin boyutunu ayarlıyoruz

stackArray = new EgeDeniziB[maxSize]; // array oluştur

top = -1; // item yok

}

public void push(EgeDeniziB b) // en üste eleman ekleme

{

stackArray[++top] = b; // top 1 artırıp elemanı ekle

}

public EgeDeniziB pop() // en üstten eleman silme

{

return stackArray[top--]; // elemanı döndürüp en üstü 1 azaltıyoruz

}

public EgeDeniziB peek() // en üstteki elemanı görme

{

return stackArray[top];

}

public bool isEmpty()

{

return (top == -1); // en üstü 1 azaltıyoruz

}

public bool isFull()

{

return (top == maxSize - 1); // top sınıra ulaşmış mı kontrolü

}

public int count()

{

return top + 1;

}

}// BalıkYığıtı Sonu

//main içerisinde:

BalıkYığıtı balıkYığıtı = new BalıkYığıtı(38);//Balık yığıtı oluşturma

foreach (EgeDeniziB balık in balıkListesi)//balıklistesindeki elemanları balıkyığıtına aktarma döngüsü

{

balıkYığıtı.push(balık);//en üste ekleme

}

// Yığıttan eleman çıkarma ve ekran çıktısı kodu.(Main içerisinde)

//Balık yığıtını yazdırma döngüsü

Console.WriteLine("BALIK YIĞITI: ");

Console.WriteLine();

while (!balıkYığıtı.isEmpty())

{

Console.WriteLine(balıkYığıtı.pop());//sondaki elemanı silip yazdırıyoruz

Console.WriteLine();

}

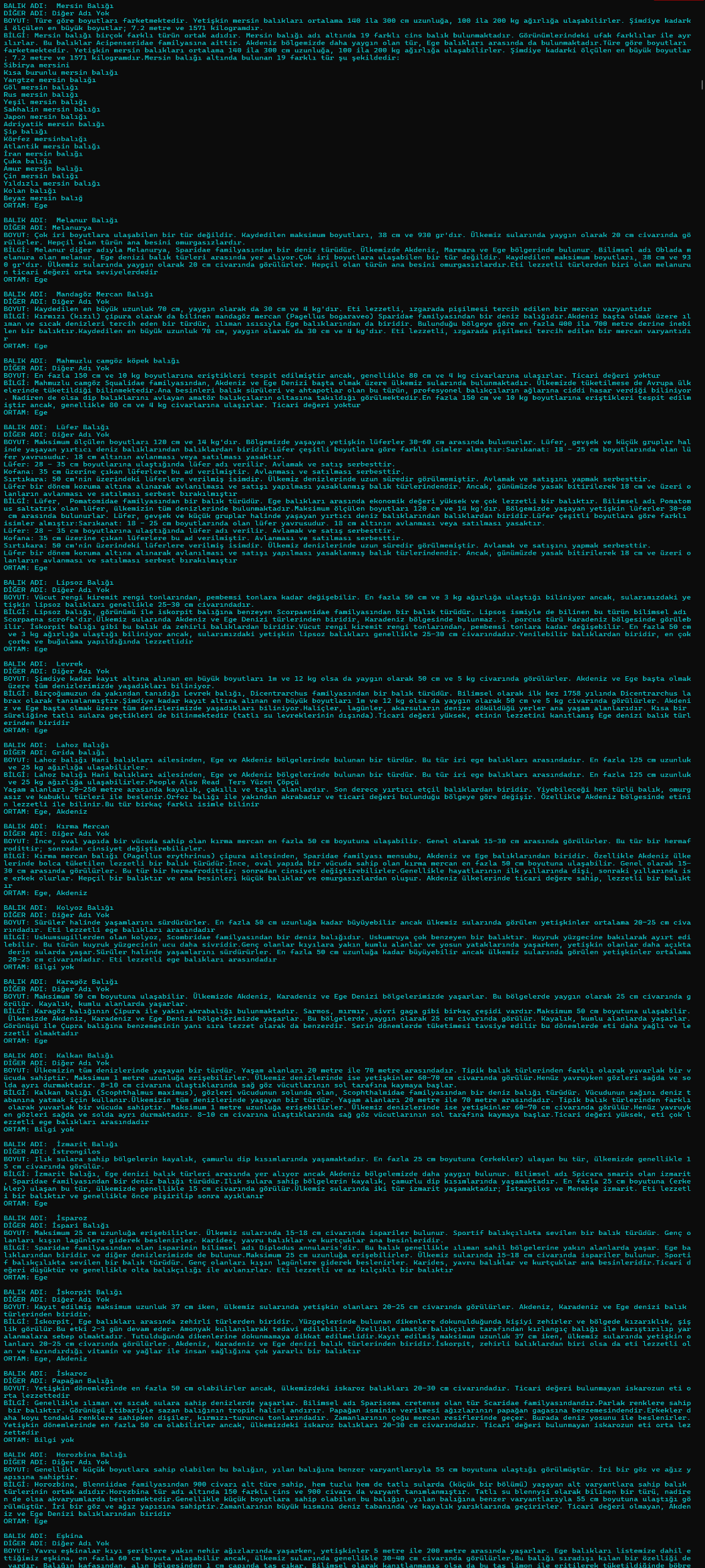
Console.WriteLine("------------------------------------------------------------------------------------------------");//çıktı daha düzgün görünmesi için araya çizgi ekleme

Console.WriteLine();

### 2.a.2 Ekran görüntüleri

// Yığıt işlemlerinin ekran görüntüsü.

A black background with blue squares

Description automatically generated

### 

### 2.b Kuyruk Yapısı

### 2.b.1 Kodlar

//Kuyruk yapısının oluşturulması ve 2a işlemine benzer şekilde EgeDeniziB nesnelerinin kuyruğa eklenmesi kodu

class BalıkKuyruğu

{

private int maxSize;

private EgeDeniziB[] kuyrukArray;

private int front,rear;

private int nItems;

public BalıkKuyruğu(int s) // constructor

{

maxSize = s;

kuyrukArray = new EgeDeniziB[maxSize];

front = 0;

rear = -1;

nItems = 0;

}

public void insert(EgeDeniziB j) // en arkaya eleman ekleme

{

if (rear == maxSize - 1) // en sona ulaştıysa başa dön

rear = -1;

kuyrukArray[++rear] = j; // önü 1 artırıp ekle

nItems++; // item sayısını artır

}

public EgeDeniziB remove() // en önden eleman silme

{

EgeDeniziB temp = kuyrukArray[front++]; // elemanı alıp en önü 1 artır

if (front == maxSize) // en sona ulaştıysa başa dön

front = 0;

nItems--; // item sayısını azalt

return temp;//silinen elemanı döndür

}

public EgeDeniziB peekFront() // en öndeki elemanı döndürme

{

return kuyrukArray[front];

}

public bool isEmpty() // boş mu

{

return (nItems == 0);

}

public bool isFull() // dolu mu

{

return (nItems == maxSize);

}

public int size() //kuyruk boyutu

{

return nItems;

}

}//BalıkKuyruğu bitiş

//(main içerisinde):

//Balık Kuyruğu

BalıkKuyruğu balıkKuyruğu = new BalıkKuyruğu(38);

foreach (EgeDeniziB balık in balıkListesi)//balıklistesindeki elemanları balıkKuyruğuna aktarma döngüsü

{

balıkKuyruğu.insert(balık);//en üste ekleme

}

//Kuyruktan eleman çıkarma ve ekran çıktısı kodu.

//yine main içerisinde aynı kodun devamı

//Balık Kuyruğunu yazdırma döngüsü

Console.WriteLine("BALIK KUYRUĞU: ");

Console.WriteLine();

while (!balıkKuyruğu.isEmpty())

{

Console.WriteLine(balıkKuyruğu.remove());//baştaki elemanı silip yazdırıyoruz

Console.WriteLine();

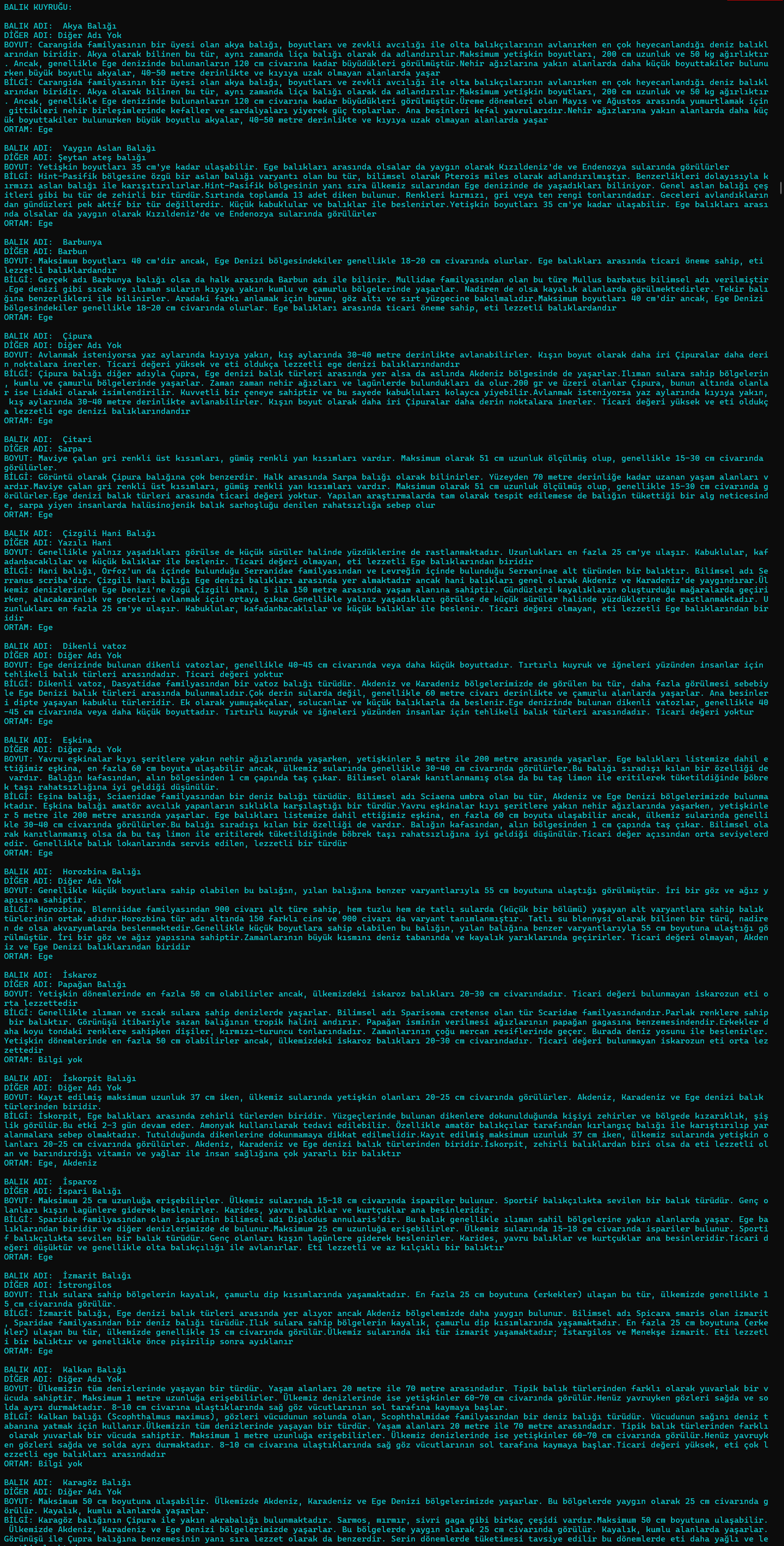
}

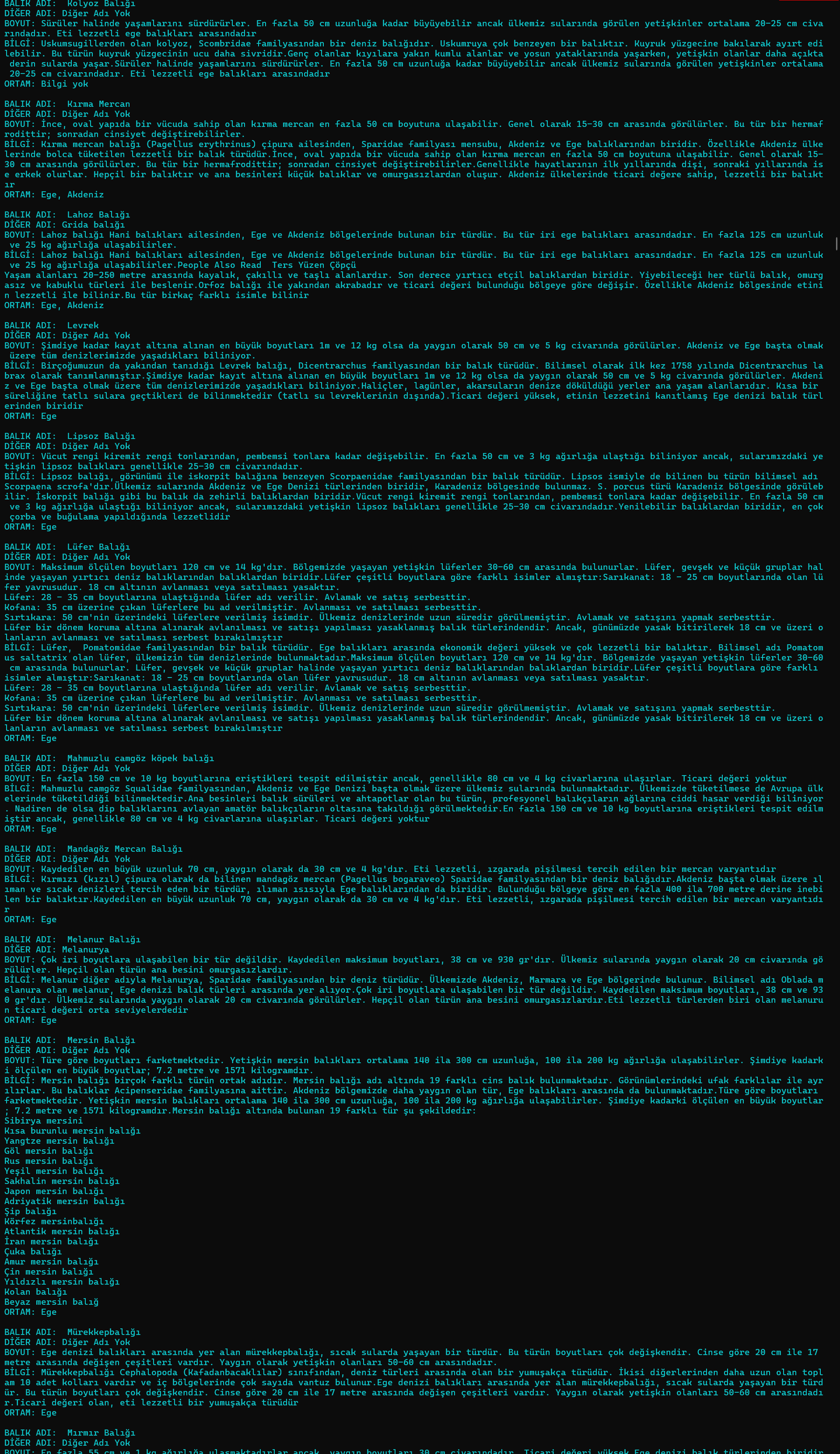
Console.WriteLine("----------------------------------------------------------------------------------------------");

Console.WriteLine();

### 2.b.2 Ekran görüntüleri

// Kuyruk işlemlerinin ekran görüntüsü.







### 3. Öncelikli Kuyruk

Sınıfın tanımı ve metotların açıklaması.

Öncelikli kuyruk sınıfı gönderilen verileri belirlenen öncelik sırasına göre örneğin tamsayılar için artan sırada, verilerin tutulmasını sağlayan sınıftır.

Yapıcı metotta belirlenen boyutta Öncelikli Kuyruk objesi oluşturulur.

Ekle metodunda objenin içindeki liste veya dizi içerisindeki elemanlar dolaşılarak verilen parametre önceliğinin olduğu yere eklenir.

Sil metodunda en küçük eleman silinir.

BosMu methodunda içerisinde veri olup olmadığı kontrol edilir.

### 3.a Öncelikli Kuyruk Sınıfı (Ekle, Sil, bosMu Metotları)

Sınıfın tanımı ve metotların açıklaması.

Bu sınıf balık nesenelerini bir kuyruk yapısı içinde adlarına göre sıralı şekilde çıkarmak için tasarlanmıştır.

Yapılandırıcı metodda kuyruğun kapasitesini ayarlayarak uygun liste oluşturulur.

ekle() metodunda verilen EgeDeniziB parametresini listeye ekler.

sil() metodunda listeden alfabetik olarak adı en önce gelen balığı bulur ve çıkarır.

Bunun için enÖncelikliAd ve enÖncelikliBalık nesneleri oluşturulup liste dolaşılırken en öncelikli ada sahip balık bulunur ve bu değerler atanır daha sonra bu balık listeden çıkartılır ve ekrana yazdırılır.

bosMu() metodunda listede eleman olup olmadığı kontrol edilir boş ise true değil ise false döndürülür.

Verinin alfabetik sıraya göre silinmesi için gerekli kod.

class ÖncelikliBalıkKuyruk

{

private int maxSize;

private List<EgeDeniziB> liste;

public ÖncelikliBalıkKuyruk(int s)//yapılandırıcı method

{

maxSize = s;

liste = new List<EgeDeniziB>(s);

}

public void ekle(EgeDeniziB b)//listeye eleman ekleme

{

liste.Add(b);

}

public EgeDeniziB sil()//en öncelikli elemanı silme

{

string enÖncekiAd="";// karşılaştırma için en öncelikli ad

EgeDeniziB enÖncekiBalık=new EgeDeniziB();//silinecek en öncelikli ada sahip balık

foreach(EgeDeniziB b in liste)

{

if (enÖncekiAd == "")

{

enÖncekiAd = b.BalıkAdı;

enÖncekiBalık = b;

}

else if (b.BalıkAdı.CompareTo(enÖncekiAd) == -1)

{

enÖncekiAd = b.BalıkAdı;

enÖncekiBalık=b;

}

}

liste.Remove(enÖncekiBalık);

return enÖncekiBalık;

}

public bool bosMu(){ return liste.Count == 0; }

}//öncelikli kuyruk bitiş

//(main içinde):

//Öncelikli Balık Kuyruğu

ÖncelikliBalıkKuyruk öncelikliBalıkKuyruğu = new ÖncelikliBalıkKuyruk(38);

foreach (EgeDeniziB balık in balıkListesi)//balıklistesindeki elemanları öncelikliKuyruğa aktarma döngüsü

{

öncelikliBalıkKuyruğu.ekle(balık);//en üste ekleme

}

//Öncelikli Kuyruğu yazdırma döngüsü

Console.WriteLine("BALIK ÖNCELİKLİ KUYRUĞU: ");

Console.WriteLine();

while (!öncelikliBalıkKuyruğu.bosMu())

{

Console.WriteLine(öncelikliBalıkKuyruğu.sil());//en öncelikli elemanı silip yazdırıyoruz

Console.WriteLine();

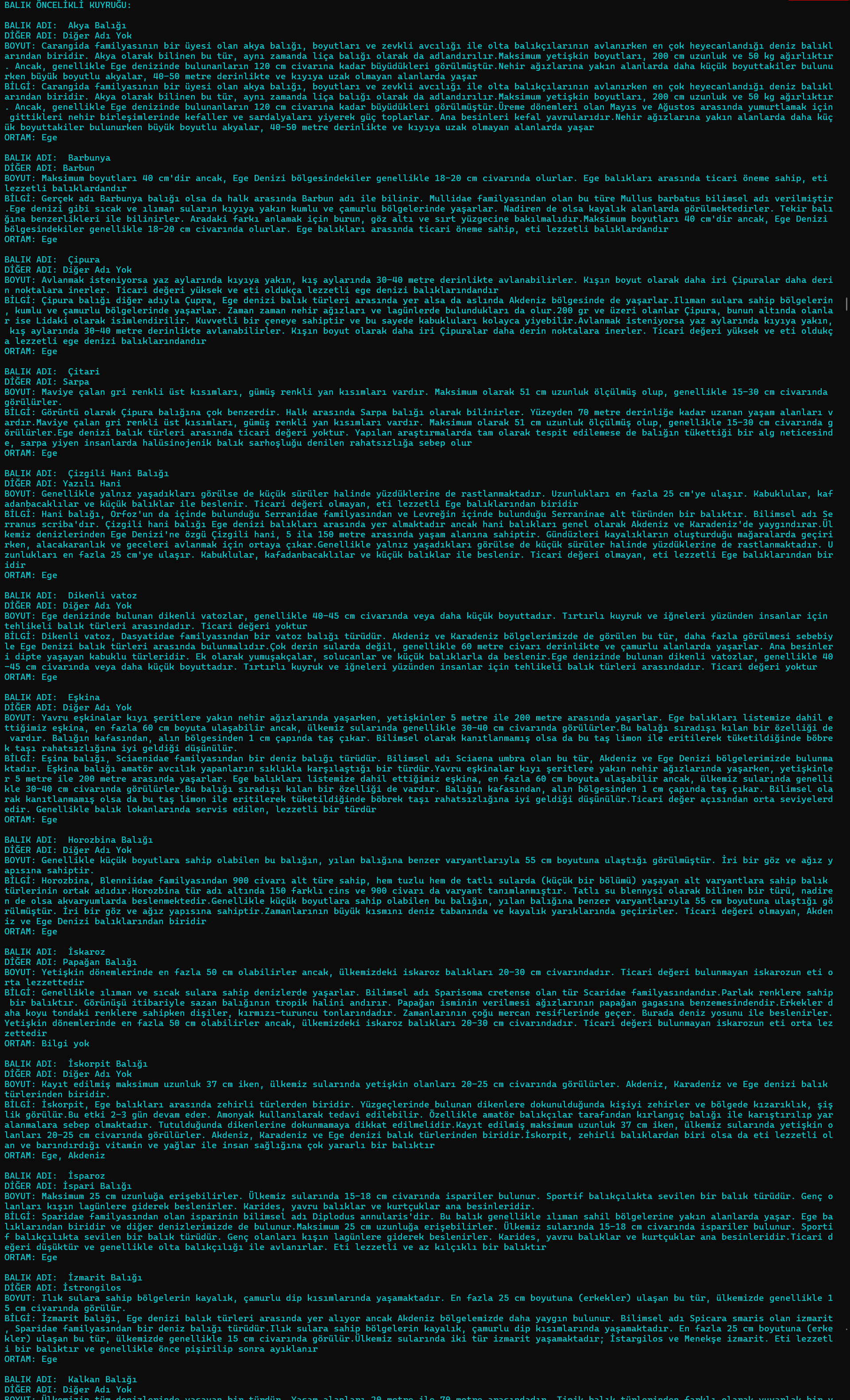
}

Console.WriteLine("----------------------------------------------------------------------------------------------");

Console.WriteLine();

### 3.b Ekran görüntüleri

// Öncelikli Kuyruk işlemlerine ait ekran görüntüsü.

A screen shot of a computer code

Description automatically generated

## 4. Market Kuyruğu ve Karşılaştırma

### 4.a Kuyruk Uygulaması

### 4.a.1 Kodlar

// Müşterilerin işlem süresi hesaplama kodu.

class TamsayıKuyruğu

{

private Queue<int> kuyruk;

private int toplamMüşteriSayısı;

private double toplamSüre = 0,ortalama = 0;

private double aktifSüre;

private double işlemSüresi = 3.3;

public TamsayıKuyruğu(int s)// oluşturucu

{

kuyruk = new Queue<int>(s);

toplamMüşteriSayısı = 0;

aktifSüre = toplamSüre = 0;

}

public void ekle(int j) // en arkaya eleman ekleme

{

kuyruk.Enqueue(j);

}

public int sil() // en önden eleman silme ve süre hesaplama

{

int sepet=0;

if (!this.bosMu())//kuyruk boş mu kontrolü

{

sepet = kuyruk.Dequeue();//öndeki elemanı silip değere ata

}

else { Console.WriteLine("Kuyruk Boş");return 0; }//boşsa 0 döndür

aktifSüre = sepet \* işlemSüresi;//aktif müşterinin işlem süresi

toplamSüre += aktifSüre;//toplam süreyi artır

toplamMüşteriSayısı++;//toplam item sayısını artır (ortalama hesaplamak için)

ortalama = toplamSüre / toplamMüşteriSayısı;

Console.WriteLine($"{sepet} ürünü olan öncelikli müşterinin; İşlem tamamlanma süresi: {toplamSüre:F2}\t Ortalama işlem süresi: {ortalama:F2}");

Console.WriteLine();

return sepet;//silinen elemanı döndür

}

public bool bosMu() // boş mu

{

return (kuyruk.Count == 0);

}

public int uzunluk() { return kuyruk.Count; }

public double getToplamSüre()

{

return toplamSüre;

}

public double getOrtalama()

{

return ortalama;

}

}

//main içi:

int[] müşteriSepetleri = { 15, 1, 12, 8, 7, 4, 21, 3, 2, 6, 5, 9, 11};//müşterilerin sepetlerindeki elemanları tutan array

TamsayıKuyruğu marketSırası = new TamsayıKuyruğu(müşteriSepetleri.Length);// marketSırası kuyruğumuzu oluşturuyoruz

List<double[]> müşteriİşlemSüreleri = new List<double[]>();//müşterilerin normal kuyruğa göre sürelerini tutan bağlı liste

foreach (int s in müşteriSepetleri) { marketSırası.ekle(s); } //elemanları oluşturduğumuz kuyruğa ekliyoruz

Console.WriteLine("NORMAL KUYRUĞA GÖRE İŞLEM SÜRELERİ:");

while (!marketSırası.bosMu())

{

int silinen = marketSırası.sil(); //sırayla sepetleri siliyoruz ve ürün sayısını değere atıyoruz (sil metodu zaten değerleri yazdırıyor)

müşteriİşlemSüreleri.Add(new double[] { silinen, marketSırası.getToplamSüre(), marketSırası.getOrtalama() });

}

Console.WriteLine("Normal kuyruk için müşteri işlem süreleri; ");

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

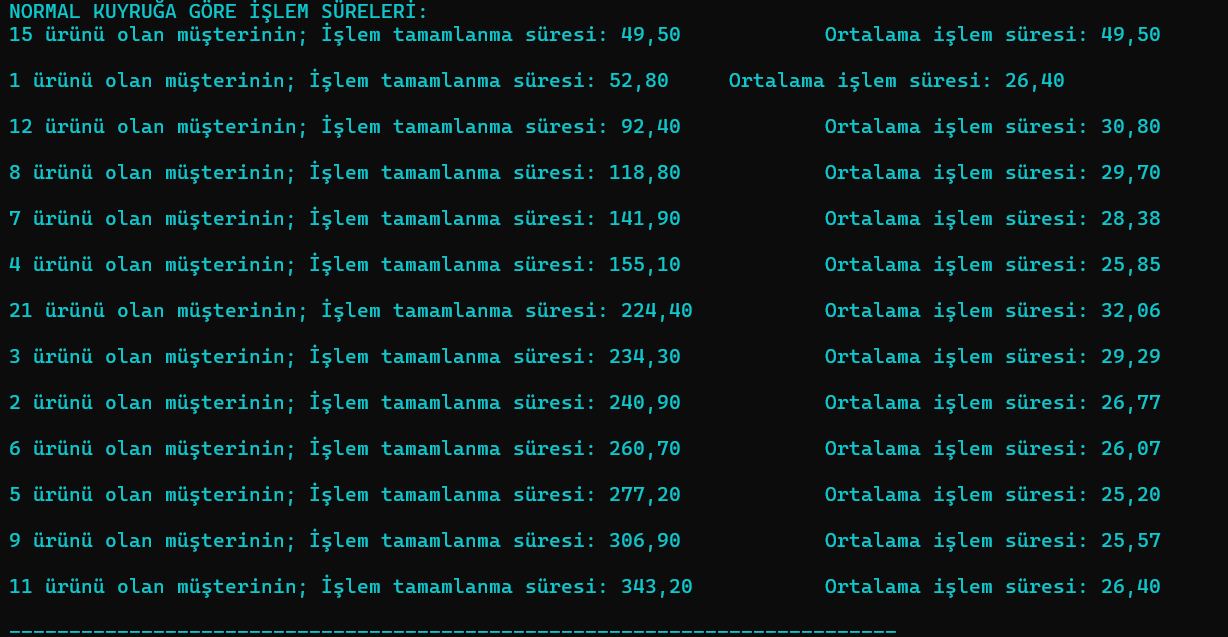
{

Console.WriteLine($"{müşteriİşlemSüreleri[i][0]} ürünü olan müşteri; \t İşlem süresi: {müşteriİşlemSüreleri[i][1]:F2} sn,\t Ortalama süre: {müşteriİşlemSüreleri[i][2]:F2} sn");

}

### 4.a.2 Ekran görüntüleri

// Her müşterinin işlem süresi çıktısına dair ekran görüntüsü



### 4.b Öncelikli Kuyruk Güncellemesi

### 4.b.1 Kodlar

//Öncelikli Kuyruk ile işlem süresi hesaplama kodu.

class ÖncelikliTamSayıKuyruğu

{

private List<int> liste;

private int toplamMüşteriSayısı;

private double toplamSüre = 0,ortalama = 0;

private double aktifSüre;

private double işlemSüresi = 3.3;

public ÖncelikliTamSayıKuyruğu(int s)//yapılandırıcı method

{

toplamMüşteriSayısı = 0;

liste = new List<int>(s);

}

public void ekle(int b)//listeye eleman ekleme

{

liste.Add(b);

}

public int sil()//en öncelikli elemanı silme

{

int enKüçükSayı = int.MaxValue;// karşılaştırma için en büyük sayı atıyoruz

if (bosMu())//kuyruk boş ise 0 döndürüyoruz

{

Console.WriteLine("Kuyruk Boş");

return 0;

}

else

{

enKüçükSayı = liste.Min();//listedeki en küçük sayıyı bulma methodu

liste.Remove(enKüçükSayı);//en küçük sayıyı çıkarma

aktifSüre = enKüçükSayı \* işlemSüresi;//şuanki müşteri için toplam işlem süresi

toplamSüre += aktifSüre;//şuanki süreyi toplam süreye ekleme

toplamMüşteriSayısı++;//toplam item sayısını artır (ortalama hesaplamak için)

ortalama = toplamSüre / toplamMüşteriSayısı;

Console.WriteLine($"{enKüçükSayı} ürünü olan öncelikli müşterinin; İşlem tamamlanma süresi: {toplamSüre:F2}\t Ortalama işlem süresi: {ortalama:F2}");

Console.WriteLine();

return enKüçükSayı;

}

}

public bool bosMu()

{

return liste.Count == 0;

}

public double getToplamSüre()

{

return toplamSüre;

}

public double getOrtalama(){return ortalama;}

//main içi:

ÖncelikliTamSayıKuyruğu marketSırasıÖncelikli = new ÖncelikliTamSayıKuyruğu(müşteriSepetleri.Length);// marketSırası kuyruğumuzu oluşturuyoruz

List<double[]> müşteriÖncelikliİşlemSüreleri = new List<double[]>();//müşterilerin öncelikli kuyruğa göre sürelerini tutan bağlı liste

foreach (int s in müşteriSepetleri) { marketSırasıÖncelikli.ekle(s); } //elemanları oluşturduğumuz kuyruğa ekliyoruz

Console.WriteLine("--------------------------------------------------------------------------");

Console.WriteLine("ÖNCELİKLİ KUYRUĞA GÖRE İŞLEM SÜRELERİ:");

while (!marketSırasıÖncelikli.bosMu())

{

int silinen = marketSırasıÖncelikli.sil();//sırayla sepetleri siliyoruz ve ürün sayısını değere atıyoruz

//süreleri ürün sayısı ile birlikte bağlı listemize ekliyoruz

müşteriÖncelikliİşlemSüreleri.Add(new double[] { silinen, marketSırasıÖncelikli.getToplamSüre(), marketSırasıÖncelikli.getOrtalama() });

}

}//öncelikli tamsayı kuyruk bitiş

### 4.b.2 Ekran görüntüleri

// İşlem süreleri ekran çıktısı.

A screenshot of a computer

Description automatically generated

### 4.c Queue ve PQ Karşılaştırması

//Queue ve PQ’nun karşılaştırması; avantaj ve dezavantajların açıklanması.

Queue daha basittir ilk giren ilk çıkar ve herkes eşittir sırasında işlem görür. Ancak bekleme süreleri uzun olabilir önceki müşterilerin ürün sayısı fazla ise az ürünü olan müşteriler fazla bekler.

PQ da az ürünü olan müşteriler daha kısa süre bekler ve verimlilik daha fazla olmuş olur ancak eşitlik yoktur ilk geldiği halde fazla ürünü olduğu için fazla bekleyen müşteriler olabilir.

Console.WriteLine("MÜŞTERİ İŞLEM SÜRELERİ KARŞILAŞTIRMASI\t\t İŞLEM SÜRESİ ORTALAMA SÜRE");

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

{

double ürünSayısı = müşteriÖncelikliİşlemSüreleri[i][0];

double[] normal = sıralıSüreler[i];

double[] öncelikli = müşteriÖncelikliİşlemSüreleri[i];

Console.WriteLine($"{ürünSayısı} ürünü olan müşteri; \t Normal Kuyruk :\t {normal[1]:F2} sn,\t {normal[2]:F2} sn" +

$"\n\t\t\t Öncelikli Kuyruk :\t {öncelikli[1]:F2} sn,\t {öncelikli[2]:F2} sn");

Console.WriteLine();

}

### 5. Yorum Sorusu

Avantajı: Chatbot kullanmak internet üzerinden bulmaktan daha hızlıdır istediğimiz bilgiye direkt ulaşabiliriz ve verilen bilgiyi istediğimiz şekilde düzenleyebilir,kişiselleştirebiliriz kullanacağımız şekle getirtebiliriz

Dezavantajı: Bilginin doğruluğundan tam olarak emin olamayız bilgileri bize aktarırken veya düzenlerken eksik, yanlış söylemiş olabilir.

Örneğin tüm balıkların sadece boyut ile ilgili paragraflarını vermesini istersek istediğimize çok kolay şekilde ulaşırız ve bunları aktarmamız daha kolay olur ancak bu paragraflardan bazıları boyut paragrafı olmadığı halde bize aktarmış olabilir ya da atladığı paragraflar olursa aldığımız veri hatalı olur.

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Puan** | **Tahmini Not** | **Açıklama** |
| **1 a) Sınıf Oluşturma** | **5** | **5** | **Yapıldı,Projede istenilenlere göre bir sınıf oluşturuldu sınıfın içindeki metodlar daha önce kurulmuş classlardan yardım alınarak yapıldı.** |
| **1 b) Generic List veri yapısının oluşturulması ve elemanların eklenmesi; Kaynak kod, ekran görüntüsü.** | **15** | **15** | **Yapıldı, metin kod içerisinde bir değişkene kopyalanıp balıklar listeye ayrıldı.** |
| **1 c) Generic List Dizisinin oluşturulması** |  |  | **Yapıldı,her grup için liste oluşturarak yapıldı.** |
| **1 d) Veri yapısının elemanlarının listelenmesi ve gruplardaki diğer adı olan balık sayılarının yazdırılması; Kaynak kodlar, ekran görüntüsü.** | **10** | **10** | **Yapıldı, ayrı bir bileşik listeyi yazdırma metodu yazılarak o method içinde 2 for döngüsü kullanarak yapıldı.** |
| **2 a) Yığıt kaynak kod ve ekran görüntüleri** | **10** | **10** | **Yapıldı, ders kitabındaki program incelenerek gereken değişiklikler yapıldı.** |
| **2 b) Kuyruk kaynak kod ve ekran görüntüleri** | **10** | **10** | **Yapıldı, ders kitabındaki program incelenerek gereken değişiklikler yapıldı.** |
| **3) Öncelikli Kuyruk kod ve ekran görüntüleri** | **20** | **20** | **Yapıldı,başka öncelikil kuyruklara bakarak temel kısımları yapıldı sil metodunda for döngüsü ile öncelikli eleman bulundu.** |
| **4) Market Kuyruğu Kod, sonuçlar tablosu, ekran görüntüleri, karşılaştırma ve soruların cevapları.** | **15** | **15** | **Yapıldı,varsayılan queue kullanılarak oluşturulan tamsayıKuyruğu ile sil metodu içinde süreler hesaplanıp yazdırıldı.** |
| **5) Yorum Sorusu** | **5** | **5** | **Yapıldı** |
| **6) Özdeğerlendirme Tablosu** | **10** | **10** | **Yapıldı** |
| **Toplam** | **100** |  |  |

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır.**

**Not: Raporu teslim edilmeyen projeler değerlendirmeye alınmayacaktır.**