

**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

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

**2023–2024 FALL SEMESTER**

**PROJECT-2 REPORT**

**LİSTE, YIĞIT, KUYRUK ve ÖNCELİKLİ KUYRUK VERİ YAPILARI**

**DELIVERY DATE**

23/11/2023

**PREPARED BY**

Bahrihan Torpil- 05210000261

İçindekiler

[1) UNESCO Dünya Mirası Listesi 3](#_Toc150853391)

[1.a UM Alanı Sınıfı Oluşturma 3](#_Toc150853392)

[1.a.1 Kodlar 3](#_Toc150853393)

[1.a.2 Açıklama 3](#_Toc150853394)

[1.b Yedi Elemanlı Bir Generic List Dizisi Oluşturma 3](#_Toc150853395)

[1.b.1 Kodlar 3](#_Toc150853396)

[1.c Dizideki Her Bir Listedeki Bilgilerin ve Her Bir Bölgedeki UM Alanı Sayısını Yazdırma 3](#_Toc150853397)

[1.c.1 Kodlar 3](#_Toc150853398)

[1.c.2 Ekran görüntüleri 3](#_Toc150853399)

[2) YIĞIT ve KUYRUK 3](#_Toc150853400)

[2.a UM Alanı Yığıtı Oluşturma 3](#_Toc150853401)

[2.a.1 Yığıt Sınıfı Kaynak Kodu 3](#_Toc150853402)

[2.a.2 Yığıt Kullanımı Kaynak Kodu 3](#_Toc150853403)

[2.a.3 Ekran Görüntüleri 3](#_Toc150853404)

[2.b UM Alanı Kuyruk Veri Yapısı Oluşturma 3](#_Toc150853405)

[2.b.1 Kuyruk Sınıfı Kaynak Kodu 3](#_Toc150853406)

[2.b.2 Kuyruk Kullanımı Kaynak Kodu 4](#_Toc150853407)

[2.b.3 Ekran Görüntüleri 4](#_Toc150853408)

[3) Öncelikli Kuyruk 4](#_Toc150853409)

[3.a Öncelikli Kuyruk (sınıfı) Oluşturma 4](#_Toc150853410)

[3.a.1 Öncelikli Kuyruk Sınıfı Kaynak Kod 4](#_Toc150853411)

[3.a.2 Öncelikli Kuyruk Kullanımı Kaynak Kodu 4](#_Toc150853412)

[3.a.3 Ekran görüntüleri 4](#_Toc150853413)

[4) Kuyruk ve Öncelikli Kuyruk 4](#_Toc150853414)

[4.a Kuyruk kullanarak süre hesaplanması 4](#_Toc150853415)

[4.a.1 Kaynak Kod 4](#_Toc150853416)

[4.a.2 Ekran görüntüleri 4](#_Toc150853417)

[4.b Öncelikli Kuyruk sınıfı kullanarak süre hesaplanması 4](#_Toc150853418)

[4.b.1 Güncellenmiş Öncelikli Kuyruk Kaynak Kod 4](#_Toc150853419)

[4.b.2 Öncelikli Kuyruk Süre Hesaplaması Kaynak Kod 4](#_Toc150853420)

[4.b.3 Ekran görüntüleri 4](#_Toc150853421)

[4.c Kuyruk ve Öncelikli Kuyruk Yapılarının Karşılaştırılması 4](#_Toc150853422)

[4.c.1 Kuyruk ve Öncelikli Kuyruk Karşılaştırması 4](#_Toc150853423)

[4.c.2 Önerilen Yöntem 4](#_Toc150853424)

[Öz değerlendirme Tablosu 5](#_Toc150853425)

# 1) UNESCO Dünya Mirası Listesi

VISUAL STUDIO, .NET FRAMEWORK 4.8, C#

## 1.a UM Alanı Sınıfı Oluşturma

### 1.a.1 Kodlar

|  |
| --- |
| public class UMAlani  {  public string AlanAdi { get; set; }  public List<string> IlAdlari { get; set; }  public int IlanYili { get; set; }  public UMAlani(string AlanAdi, List<String> IlAdlari, int IlanYili)  {  this.AlanAdi = AlanAdi;  this.IlAdlari = IlAdlari;  this.IlanYili = IlanYili;  }  } |

### 1.a.2 Açıklama

String AlanAdi : UM alan ismini tutan nesne.

List<string> IlAdlari: generic list olarak tutulan bu nesne UM alanı birden çok il için geçerli olması durumunu da ele almak amacıyla bu veri türünde tutulmuştur.

İnt IlanYili : UM alanının ilan yılını tutar.(String olarak da tutulabilir çünkü üzerinde bir işlem yapılmasına gerek yoktur.)

## 1.b Yedi Elemanlı Bir Generic List Dizisi Oluşturma

### 1.b.1 Kodlar

|  |
| --- |
| List<UMAlani>[] genericList = new List<UMAlani>[7];  for (int i = 0; i < genericList.Length; i++)  {  genericList[i] = new List<UMAlani>();  } |

## 1.c Dizideki Her Bir Listedeki Bilgilerin ve Her Bir Bölgedeki UM Alanı Sayısını Yazdırma

### 1.c.1 Kodlar

|  |
| --- |
| private static void CreateUMField(List<UMAlani>[] genericList, string[] regions)  {  string UMText = "C:\\Users\\LENOVO\\bahrihan\\Masaüstü\\C# VS\\Proje3\\Proje3\\UnescoMirasListesi.txt";  try  {  using (StreamReader sr = new StreamReader(UMText, Encoding.GetEncoding("iso-8859-9")))  {  while (!sr.EndOfStream)  {  string row = sr.ReadLine();  string[] elements = row.Split(',');  List<string> cityNamesList = new List<string>();  string updatedCityName = RemoveParenthesis(elements[1]);  string[] citiesName = updatedCityName.Split('-');  foreach (string city in citiesName)  {  cityNamesList.Add(city);  }  int acceptedYear = int.Parse(elements[2]);  UMAlani umAlan = new UMAlani(elements[0], cityNamesList, acceptedYear);  InsertElements(umAlan, genericList);  }  /\*  string lastRow = sr.ReadLine();// 21. satır için  string[] lastRowElements = lastRow.Split(',');  List<string> lastCityNamesList = new List<string>();  string lastUpdatedCitiesName = RemoveParenthesis(lastRowElements[1]);  string[] lastCitiesName = lastUpdatedCitiesName.Split('/');  int lastAcceptedYear = int.Parse(lastRowElements[2]);  foreach (string city in lastCitiesName)  {  string[] tempRegionNameList = city.Split('-');  lastCityNamesList.Add(tempRegionNameList[0]);  }  UMAlani lastUMAlan = new UMAlani(lastRowElements[0], lastCityNamesList, lastAcceptedYear);  InsertElements(lastUMAlan, genericList);  \*/  PrintUMAlanlari(genericList, regions);  }  }  catch (Exception e)  {  Console.WriteLine("Hata: " + e.Message);  }  }  private static void PrintUMAlanlari(List<UMAlani>[] genericList, string[] regions)  {  int j = 0;  foreach (var item in genericList)  {  Console.WriteLine($"Bölge: {regions[j]} - Bölgedeki toplam UM sayısı: {genericList[j].Count}");  foreach (var item1 in item)  {  Console.Write($"Alan adı: {item1.AlanAdi}| UM'nin bulunduğu il adı/adları:");  foreach (var city in item1.IlAdlari)  {  Console.Write($"{city} ");  }  Console.WriteLine($"| İlan yılı:{item1.IlanYili}");  }  Console.WriteLine();  j++;  }  } |

### 1.c.2 Ekran görüntüleri

Menü

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Seçenek 1 ->

# metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim Açıklama otomatik olarak oluşturuldu2) YIĞIT ve KUYRUK

VISUAL STUDIO, .NET FRAMEWORK 4.8, C#

## 2.a UM Alanı Yığıtı Oluşturma

### 2.a.1 Yığıt Sınıfı Kaynak Kodu

|  |
| --- |
| class Stack<T>  {  private const int DefaultCapacity = 4;  private T[] items;  private int size;  public Stack()  {  items = new T[DefaultCapacity];  size = 0;  }  public int Count  {  get { return size; }  }  public void Push(T item)  {  if (size == items.Length)  {  ResizeArray();  }  items[size++] = item;  }  public T Pop()  {  if (size == 0)  {  throw new InvalidOperationException("Stack is empty");  }  T poppedItem = items[--size];  return poppedItem;  }  public T Peek()  {  if (size == 0)  {  throw new InvalidOperationException("Stack is empty");  }  return items[size - 1];  }  public bool isEmpty()  {  return size == 0;  }  private void ResizeArray()  {  int newCapacity = items.Length \* 2;  Array.Resize(ref items, newCapacity);  }  } |

### 2.a.2 Yığıt Kullanımı Kaynak Kodu

|  |
| --- |
| private static void CreateWithStack(List<UMAlani>[] genericList, string[] regions)  {  string UMText = "C:\\Users\\LENOVO\\bahrihan\\Masaüstü\\C# VS\\Proje3\\Proje3\\UnescoMirasListesi.txt";  try  {  Stack<UMAlani> stack = new Stack<UMAlani>();  using (StreamReader sr = new StreamReader(UMText, Encoding.GetEncoding("iso-8859-9")))  {  while (!sr.EndOfStream)  {  string row = sr.ReadLine();  string[] elements = row.Split(',');  List<string> cityNamesList = new List<string>();  string updatedCityName = RemoveParenthesis(elements[1]);  string[] citiesName = updatedCityName.Split('-');  foreach (string city in citiesName)  {  cityNamesList.Add(city);  }  int acceptedYear = int.Parse(elements[2]);  UMAlani umAlan = new UMAlani(elements[0], cityNamesList, acceptedYear);  stack.Push(umAlan);  }  while (!stack.isEmpty())  {  UMAlani uMAlani = stack.Pop();  Console.Write($"Alan adı: {uMAlani.AlanAdi},");  foreach (var item in uMAlani.IlAdlari)  {  Console.Write($"il adı/adları: {item}, ");  }  Console.Write($"ilan yılı: {uMAlani.IlanYili}");  Console.WriteLine();  }  }  }  catch (Exception e)  {  Console.WriteLine("Hata: " + e.Message);  }  } |

### 2.a.3 Ekran Görüntüleri

## metin, ekran görüntüsü, siyah beyaz, yazı tipi içeren bir resim Açıklama otomatik olarak oluşturuldu2.b UM Alanı Kuyruk Veri Yapısı Oluşturma

### 2.b.1 Kuyruk Sınıfı Kaynak Kodu

|  |
| --- |
| public class Queue<T>  {  private const int DefaultCapacity = 4;  private T[] items;  private int size;  private int front;  private int rear;  public Queue()  {  items = new T[DefaultCapacity];  size = 0;  front = 0;  rear = -1;  }  public int Count  {  get { return size; }  }  public void Enqueue(T item)  {  if (size == items.Length)  {  ResizeArray();  }  rear = (rear + 1) % items.Length;  items[rear] = item;  size++;  }  public T Dequeue()  {  if (size == 0)  {  throw new InvalidOperationException("Queue is empty");  }  T dequeuedItem = items[front];  front = (front + 1) % items.Length;  size--;  return dequeuedItem;  }  public T Peek()  {  if (size == 0)  {  throw new InvalidOperationException("Queue is empty");  }  return items[front];  }  public bool isEmpty()  {  return size == 0;  }  private void ResizeArray()  {  int newCapacity = items.Length \* 2;  T[] newItems = new T[newCapacity];  for (int i = 0; i < size; i++)  {  newItems[i] = items[(front + i) % items.Length];  }  items = newItems;  front = 0;  rear = size - 1;  }  } |

### 2.b.2 Kuyruk Kullanımı Kaynak Kodu

|  |
| --- |
| private static void CreateWithQueue(List<UMAlani>[] genericList, string[] regions)  {  string UMText = "C:\\Users\\LENOVO\\bahrihan\\Masaüstü\\C# VS\\Proje3\\Proje3\\UnescoMirasListesi.txt";  try  {  Queue<UMAlani> queue = new Queue<UMAlani>();  using (StreamReader sr = new StreamReader(UMText, Encoding.GetEncoding("iso-8859-9")))  {  while (!sr.EndOfStream)  {  string row = sr.ReadLine();  string[] elements = row.Split(',');  List<string> cityNamesList = new List<string>();  string updatedCityName = RemoveParenthesis(elements[1]);  string[] citiesName = updatedCityName.Split('-');  foreach (string city in citiesName)  {  cityNamesList.Add(city);  }  int acceptedYear = int.Parse(elements[2]);  UMAlani umAlan = new UMAlani(elements[0], cityNamesList, acceptedYear);  queue.Enqueue(umAlan);  }  while (!queue.isEmpty())  {  UMAlani uMAlani = queue.Dequeue();  Console.Write($"Alan adı: {uMAlani.AlanAdi},");  foreach (var item in uMAlani.IlAdlari)  {  Console.Write($"il adı/adları: {item}, ");  }  Console.Write($"ilan yılı: {uMAlani.IlanYili}");  Console.WriteLine();  }  }  }  catch (Exception e)  {  Console.WriteLine("Hata: " + e.Message);  }  } |

### 2.b.3 Ekran Görüntüleri

metin, ekran görüntüsü, siyah beyaz, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 3) Öncelikli Kuyruk

## 3.a Öncelikli Kuyruk (sınıfı) Oluşturma

### 3.a.1 Öncelikli Kuyruk Sınıfı Kaynak Kod

|  |
| --- |
| public class PriorityQueue  {  private List<UMAlani> umAlanlari; // Generic List  public PriorityQueue()  {  umAlanlari = new List<UMAlani>();  }  public void Add(UMAlani umAlan)  {  umAlanlari.Add(umAlan); // liste sonuna ekleme  }  public void Delete()  {  if (umAlanlari.Count == 0)  {  Console.WriteLine("PQ is empty!");  return;  }  //alfabetik sıraya göre silme  UMAlani mostPriority = umAlanlari[0];  for (int i = 1; i < umAlanlari.Count; i++)  {  if (umAlanlari[i].AlanAdi.CompareTo(mostPriority.AlanAdi) < 0)  {  mostPriority = umAlanlari[i];  }  }  umAlanlari.Remove(mostPriority);  Console.WriteLine($"'{mostPriority.AlanAdi}' adlı UM Alanı Silindi.");  }  public bool isEmpty()  {  return umAlanlari.Count == 0;  }  } |

### 3.a.2 Öncelikli Kuyruk Kullanımı Kaynak Kodu

|  |
| --- |
| private static void CreateWithPQ(List<UMAlani>[] genericList, string[] regions)  {  string UMText = "C:\\Users\\LENOVO\\bahrihan\\Masaüstü\\C# VS\\Proje3\\Proje3\\UnescoMirasListesi.txt";  try  {  PriorityQueue pq = new PriorityQueue();  using (StreamReader sr = new StreamReader(UMText, Encoding.GetEncoding("iso-8859-9")))  {  while (!sr.EndOfStream)  {  string row = sr.ReadLine();  string[] elements = row.Split(',');  List<string> cityNamesList = new List<string>();  string updatedCityName = RemoveParenthesis(elements[1]);  string[] citiesName = updatedCityName.Split('-');  foreach (string city in citiesName)  {  cityNamesList.Add(city);  }  int acceptedYear = int.Parse(elements[2]);  UMAlani umAlan = new UMAlani(elements[0], cityNamesList, acceptedYear);  pq.Add(umAlan);  }  while (!pq.isEmpty())  {  pq.Delete();  }  }  }  catch (Exception e)  {  Console.WriteLine("Hata: " + e.Message);  }  } |

### 3.a.3 Ekran görüntüleri

# metin, ekran görüntüsü, yazı tipi, siyah beyaz içeren bir resim Açıklama otomatik olarak oluşturuldu4) Kuyruk ve Öncelikli Kuyruk

## 4.a Kuyruk kullanarak süre hesaplanması

### 4.a.1 Kaynak Kod

|  |
| --- |
| class Queue  {  public int[] items;  public int rear;  public int front;  public Queue(int capacity)  {  items = new int[capacity];  front = 0;  rear = -1;  }  public void Enqueue(int item)  {  if (rear == items.Length - 1)  {  Console.WriteLine("Sıra dolu, eleman eklenemiyor.");  return;  }  items[++rear] = item;  }  public int Dequeue()  {  if (IsEmpty())  {  throw new InvalidOperationException("Sıra boş.");  }  int item = items[front++];  return item;  }  public int Peek()  {  if (IsEmpty())  {  throw new InvalidOperationException("Sıra boş.");  }  return items[front];  }  public bool IsEmpty()  {  return front > rear;  }  public int Count()  {  return rear - front + 1;  }  }  private static void ReadTimeWithQ(int[] customerList)  {  Queue queue = new Queue(customerList.Length);  for (int i = 0; i < customerList.Length; i++)  {  queue.Enqueue(customerList[i]);  }  double sumOfTime = 0;  double totalSum = 0;  int j = 0;  while (!queue.IsEmpty())  {  double customerTime = queue.Dequeue() \* READ\_TIME;  sumOfTime += customerTime;  totalSum += sumOfTime;  j++;  Console.WriteLine($"{j}. müşterinin işlem tamamlama süresi: {sumOfTime}");  }  Console.WriteLine($"Ortalama işlem süresi: {totalSum / customerList.Length}");  } |

### 4.a.2 Ekran görüntüleri

## metin, ekran görüntüsü, yazı tipi, siyah beyaz içeren bir resim Açıklama otomatik olarak oluşturuldu

## 4.b Öncelikli Kuyruk sınıfı kullanarak süre hesaplanması

### 4.b.1 Güncellenmiş Öncelikli Kuyruk Kaynak Kod

|  |
| --- |
| public class PriorityQueue<T> where T : IComparable<T>  {  //@INFO  // IComparable interface'i T nesnesinin karşılaştırılabilir olmasını sağlar.  // 2 <T> nesneyi karşılaştırır.  private T[] array;  private int count;  public PriorityQueue()  {  array = new T[10];  count = 0;  }  public int Count  {  get { return count; }  }  public void Enqueue(T item)  {  if (count == array.Length)  {  Array.Resize(ref array, array.Length \* 2);  }  int index = count;  // item.CompareTo(array[index - 1]) < 0 eklenen yeni nesneyi doğru yere alana kadar nesneyi kaydırır.  while (index > 0 && item.CompareTo(array[index - 1]) < 0)  {  array[index] = array[index - 1];  index--;  }  array[index] = item;  count++;  }  public T Dequeue()  {  if (count == 0)  {  throw new InvalidOperationException("Priority Queue is empty.");  }  T frontItem = array[0];  for (int i = 1; i < count; i++)  {  array[i - 1] = array[i];  }  count--;  return frontItem;  }  } |

### 4.b.2 Öncelikli Kuyruk Süre Hesaplaması Kaynak Kod

|  |
| --- |
| private static void ReadTimeWithPQ(int[] customerList)  {  PriorityQueue<int> pq = new PriorityQueue<int>();  for (int i = 0; i < customerList.Length; i++)  {  pq.Enqueue(customerList[i]);  }  int j = 0;  double sumOfTime = 0;  double totalSum = 0;  while (pq.Count > 0)  {  double customerTime = pq.Dequeue() \* READ\_TIME;  sumOfTime += customerTime;  totalSum += sumOfTime;  j++;  Console.WriteLine($"{j}. öncelikli müşterinin işlem süresi: {sumOfTime}");  }  Console.WriteLine($"Ortalama işlem süresi: {totalSum / customerList.Length}");  } |

### 4.b.3 Ekran görüntüleri

## metin, ekran görüntüsü, yazı tipi, siyah beyaz içeren bir resim Açıklama otomatik olarak oluşturuldu

## 4.c Kuyruk ve Öncelikli Kuyruk Yapılarının Karşılaştırılması

### 4.c.1 Kuyruk ve Öncelikli Kuyruk Karşılaştırması

Kuyrukta müşteriler ürün sayısı dikkate alınmaksızın sırayla okuma işlemleri gerçekleştirilir. Bu yüzden müşteri memnuniyeti ürünü az olanlar veya çoğunluk için düşüktür. (Ortalama işlem süresi)

Öncelikli kuyrukta ürün sayısı dikkate alınarak sıralandığında ürün sayısı az olan müşterilere öncelik tanınır. Bu öncelik çoğunluk için daha kısa süre sırada kalmaya neden olur. Bu yüzden müşteri memnuniyeti çoğunluk için yüksektir. (Ortalama işlem süresi)

### 4.c.2 Önerilen Yöntem

/\*

\* FIFO VE PQ yöntemlerini birleştirmeye öneri:

\* Yine az ürün öncelikli olarak sırayla her müşterinin sırayla birer ürününü okutarak devam etmek olmalıdır.

\* bkz. birer birer okuttuğumuzda her müşteri için ortalama memnuniyet artmalıdır.

\*/

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 2 Maddeleri** | **Puan** | **Tahmini Not** | **Açıklama** |
| **1 a) Sınıf Oluşturma** | **5** | 5 | Yapıldı. İstenilen veri türlerinde oluşturuldu. |
| **1 b) Bileşik veri yapısının oluşturulması ve elemanların eklenmesi; Kaynak kod, ekran görüntüsü.** | **15** | 15 | Yapıldı. İstenilen veri yapıları oluşturuldu. Dosyadan çekim yapıldı. |
| **1 c) Veri yapısının elemanlarının listelenmesi ve her bölgedeki UM alanı sayısının yazdırılması; Kaynak kodlar, ekran görüntüsü.** | **10** | 10 | Yapıldı. Dosyadan çekilen veriler UM alanı türünde bölgelere göre generic listeye eklendi. |
| **2 a) Yığıt kaynak kod ve ekran görüntüleri** | **10** | 10 | Yapıldı. LIFO şeklinde yazdırıldı. |
| **2 b) Kuyruk kaynak kod ve ekran görüntüleri** | **10** | 10 | Yapıldı. FIFO şeklinde yazdırıldı. |
| **3 ) Öncelikli Kuyruk kod ve ekran görüntüleri** | **20** | 20 | Yapıldı. Alfabeye göre UM alanları sıralandı. |
| **4) Kod, sonuçlar tablosu, ekran görüntüleri, karşılaştırma ve soruların cevapları.** | **20** | 19 | Yapıldı. Sonuçlar tutarlı. PQ ve Q veri yapıları karşılaştırıldı. Sonuçlar ekrana basıldı.Önerilen yöntem işe yaramayabilir. |
| **5) Özdeğerlendirme Tablosu** | **10** | 10 | Yapıldı. |
| **Toplam** | **100** | 99 |  |

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır. Tahmini not kısmına da ilgili maddeden kaç almayı beklediğinizi yazmalısınız.**

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