

**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**

10/12/23

**PREPARED BY**

05220000981, Burak Taşkın

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)

# UNESCO Dünya Mirası Listesi

//The platform, version, and programming language used

İnteljj, Java 19, Java

## 1.a UM Alanı Sınıfı Oluşturma

### 1.a.1 Kodlar

***package*** Proje2.data\_proje2;  
  
***public class*** UmAlanı {  
  
 String name;  
 String city;  
 ***int*** year;  
  
 UmAlanı(String name,String city, ***int*** year){  
 ***this***.name = name;  
 ***this***.city = city;  
 ***this***.year = year;  
  
 }  
  
 ***@Override  
 public*** String toString() {  
 ***return*** "{" +  
 "name='" + name + '\'' +  
 ", city='" + city + '\'' +  
 ", year=" + year +  
 '}';  
 }  
}

|  |
| --- |
|  |

### 1.a.2 Açıklama

//Kullanılan veri yapılarını/ veri tiplerinin kısaca anlatımını burada gerçekleştiriniz

Nesne değişkenleri, string tipinde şehir ve isim int tipinde yıl içerir.

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

### 1.b.1 Kodlar

|  |
| --- |
| //İlgili kod parçasını buraya ekleyiniz  List<List<UmAlanı>> genericList = ***new*** ArrayList<>(); List<UmAlanı> innerKaradeniz = ***new*** ArrayList<>(); List<UmAlanı> innerMarmara = ***new*** ArrayList<>(); List<UmAlanı> innerEge = ***new*** ArrayList<>(); List<UmAlanı> innerİcAnadolu = ***new*** ArrayList<>(); List<UmAlanı> innerAkdeniz = ***new*** ArrayList<>(); List<UmAlanı> innerDoguAnadolu = ***new*** ArrayList<>(); List<UmAlanı> innerGuneyDogu = ***new*** ArrayList<>(); |

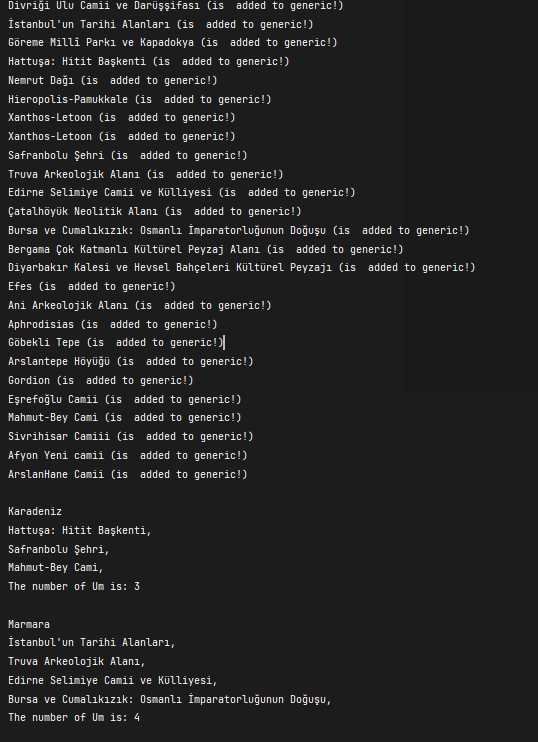
## 1.c Dizideki Her Bir Listedeki Bilgilerin ve Her Bir Bölgedeki UM Alanı Sayısını Yazdırma

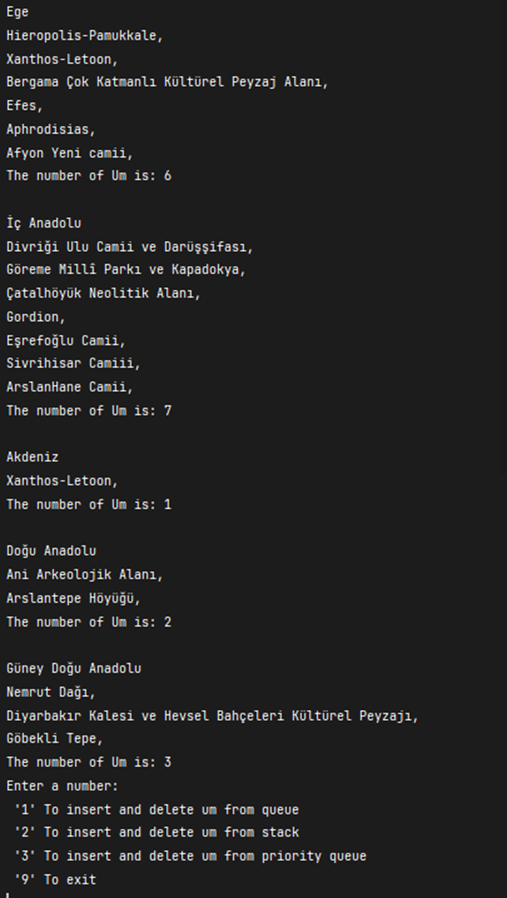
### 1.c.1 Kodlar

|  |
| --- |
| //İlgili kod parçasını buraya ekleyiniz  ***public static void*** printUm(List<List<UmAlanı>> genericList ){  String [] regionsOfStrings = {"Karadeniz","Marmara","Ege","İç Anadolu","Akdeniz","Doğu Anadolu","Güney Doğu Anadolu"};  ***int*** firstIndex = 0;  ***for***(List <UmAlanı> umAlanListesi : genericList ){  ***int*** numberOfUm = 0;  System.out.println();  System.out.println(regionsOfStrings[firstIndex]);  ***for***(UmAlanı birUmAlanı : umAlanListesi ){  System.out.println(birUmAlanı.name + ", ");  numberOfUm++;  }  System.out.println("The number of Um is: " + numberOfUm);  firstIndex++;  } }  ***public static void*** addToGenerics(UmAlanı[] dd, String[][] regions,List<List<UmAlanı>> genericList ){  ***int*** indexOfRegion = 0 ;  ***for***(UmAlanı x1 : dd){  ***for***(String[] region : regions ){  ***for***(String city : region){  ***if*** (x1.city == city ){  genericList.get(indexOfRegion).add(x1);  System.out.println(x1.name + " added to generic");  ***break***;  }  }  indexOfRegion++;  }  indexOfRegion = 0;  } } |

### 1.c.2 Ekran görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyiniz





# 2) YIĞIT ve KUYRUK

//The platform, version, and programming language used

Inteljj, Java19, Java

## 2.a UM Alanı Yığıtı Oluşturma

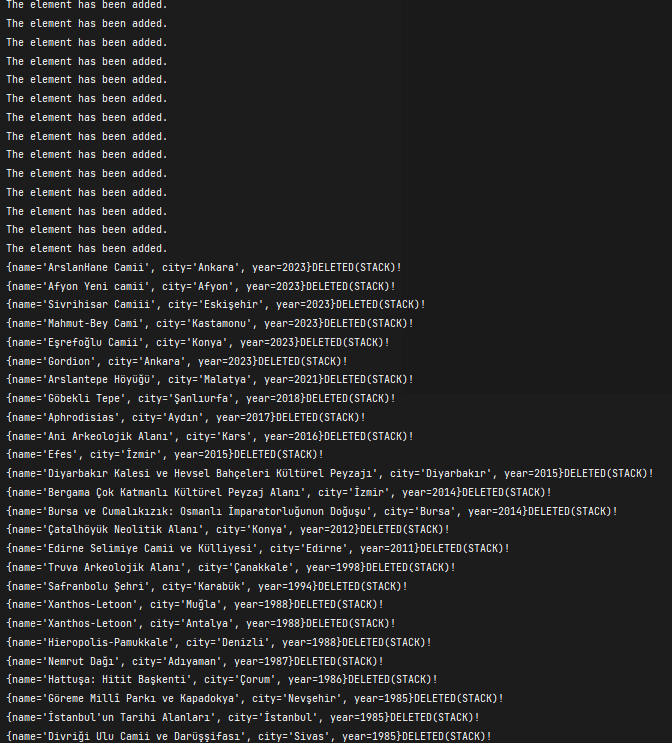
### 2.a.1 Yığıt Sınıfı Kaynak Kodu

|  |
| --- |
| ***package*** Proje2.data\_proje2;  ***public class*** StackArray {  UmAlanı[] umStList;  ***int*** topOfStack;    ***public*** StackArray(***int*** size) {  umStList = ***new*** UmAlanı[size];  topOfStack = -1;  System.out.println("The stack is created!");  } |

### 2.a.2 Yığıt Kullanımı Kaynak Kodu

|  |
| --- |
| ***package*** Proje2.data\_proje2;  ***public class*** StackArray {  UmAlanı[] umStList;  ***int*** topOfStack;    ***public*** StackArray(***int*** size) {  umStList = ***new*** UmAlanı[size];  topOfStack = -1;  System.out.println("The stack is created!");  }   ***public boolean*** isEmpty() {  ***if*** (topOfStack == -1) {  System.out.println("The stack is empty");  ***return true***;  }  *//* ***return false***;  }   ***public boolean*** isFull() {  ***if*** (topOfStack == umStList.length - 1) {  System.out.println("The stack is full!");  ***return true***;  }  ***return false***;  }    ***public void*** push(UmAlanı um) {  ***if*** (isFull()) {  System.out.println("You cannot insert! The stack is full");  } ***else*** {  umStList[topOfStack + 1] = um;  topOfStack++;  System.out.println("The element has been added.");  }  }   ***public*** String pop() {  ***if*** (isEmpty()) {  System.out.println("There is nothing to delete!");  ***return null***;  } ***else*** {  String result = umStList[topOfStack].toString() + "DELETED(STACK)!";  topOfStack--;  ***return*** result;  }  } } |

### 2.a.3 Ekran Görüntüleri



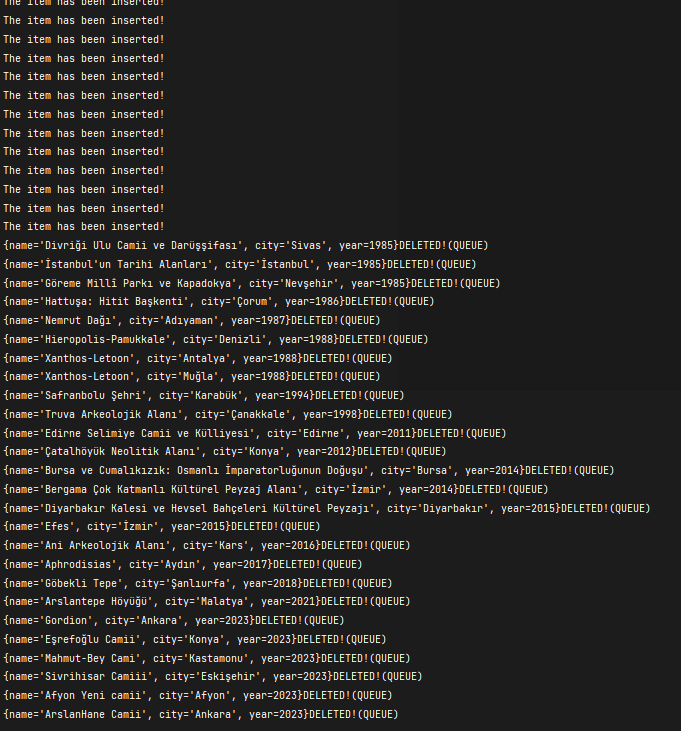
## 2.b UM Alanı Kuyruk Veri Yapısı Oluşturma

### 2.b.1 Kuyruk Sınıfı Kaynak Kodu

|  |
| --- |
| / ***package*** Proje2.data\_proje2; ***public class*** QueueArray {  UmAlanı [] umList;  ***int*** topOfQueue;  ***int*** beginningOfQueue;    QueueArray(***int*** size){  umList = ***new*** UmAlanı[size];  beginningOfQueue = topOfQueue = -1;  System.out.println("Queue is succesfully created.");  }   ***public boolean*** isFull(){  ***if***(topOfQueue == umList.length -1 ){  System.out.println("The queue is full!");  ***return true***;  }  ***return false***;  }   ***public boolean*** isEmtpy(){  ***if***(beginningOfQueue == -1){  System.out.println("The queue is empty!");  ***return true***;  }   ***return false***;  }   ***public void*** enque(UmAlanı um){  ***if***(isFull()){  System.out.println("The queue is full! You cannot insert!");  }  ***else*** {  topOfQueue++;  beginningOfQueue = 0;  umList[topOfQueue] = um;  System.out.println("The item has been inserted!");  }  }   ***public*** String deque(){  ***if***(isEmtpy()){  ***return*** "is no element to delete!";  }  ***else*** {  String result = umList[beginningOfQueue].toString() + "DELETED!(QUEUE)";  beginningOfQueue++;  ***if***(beginningOfQueue > topOfQueue){  beginningOfQueue = topOfQueue = -1;  }  ***return*** result;  }  } } |

|  |
| --- |
|  |

### 2.b.3 Ekran Görüntüleri



# 3) Öncelikli Kuyruk

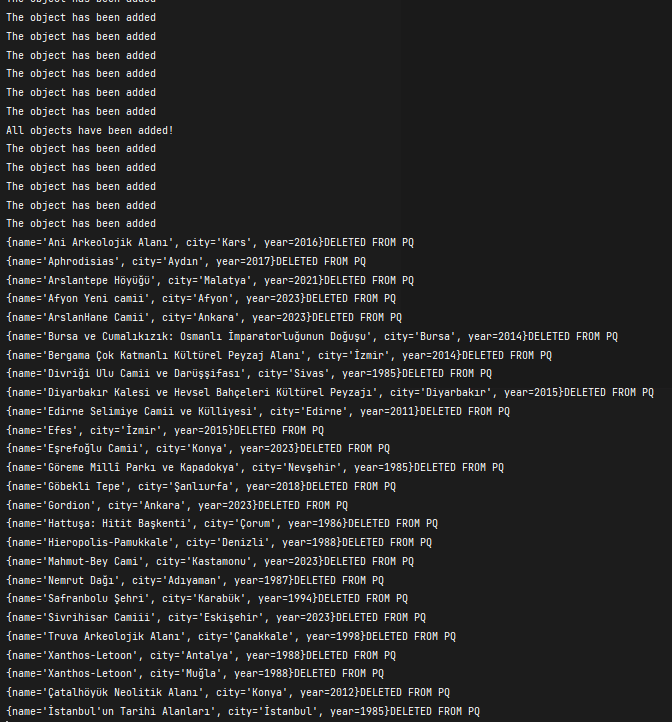
## 3.a Öncelikli Kuyruk (sınıfı) Oluşturma

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

|  |
| --- |
| ***package*** Proje2.data\_proje2;  ***import*** java.util.Arrays;  ***public class*** CustomerPQ {      ***public static void*** main(String[] args) {  CustomerPQ cpq1 = ***new*** CustomerPQ();  cpq1.dequeAndPrintCusts();  }   ***public static final double*** PROCESSING\_TIME = 2.5;  ***int*** beginningOfQueue;  ***int*** topOfQueue;  ***int***[] arr = { 10, 4, 8, 6, 7, 1, 15, 9, 3, 2 };   CustomerPQ(){  beginningOfQueue = -1;  topOfQueue = arr.length - 1 ;  }   ***public boolean*** isFull(){  ***if***(topOfQueue == arr.length -1 ){  System.out.println("The queue is full!");  ***return true***;  }  ***return false***;  }   ***public boolean*** isEmtpy(){  ***if***(beginningOfQueue == -1){  System.out.println("The queue is empty!");  ***return true***;  }   ***return false***;  }   ***public void*** dequeAndPrintCusts(){  ***for***(***int*** i = 0; i < arr.length; i++) {  ***int*** value = arr[i];  ***int*** j = i - 1;  ***while***(j >= 0 && arr[j] > value) {  arr[j + 1] = arr[j];  j = j - 1;  }  arr[j + 1] = value;  }  System.out.println(Arrays.toString(arr));  ***double*** totalSeconds = 0;  ***int*** person = 1;  ***for***(***int*** i = 0 ; i < arr.length; i++) {  ***double*** perCustomerTotal = 0;  ***for*** ( ***int*** j = 0; j <= i; j++){  perCustomerTotal += (***double***) arr[j] \* PROCESSING\_TIME;  }  totalSeconds += perCustomerTotal;  System.out.println(person+". customer's transaction completion time" +  ": " + perCustomerTotal +" sn. and the customer has been deleted!");  person++;  beginningOfQueue++;  }  System.out.println("Average transaction completion time: " +  (***double***) totalSeconds / arr.length);  ***if***(beginningOfQueue == topOfQueue){  System.out.println("All customers have been deleted!");  }  }  } |

### 3.a.3 Ekran görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyiniz



# 4) Kuyruk ve Öncelikli Kuyruk

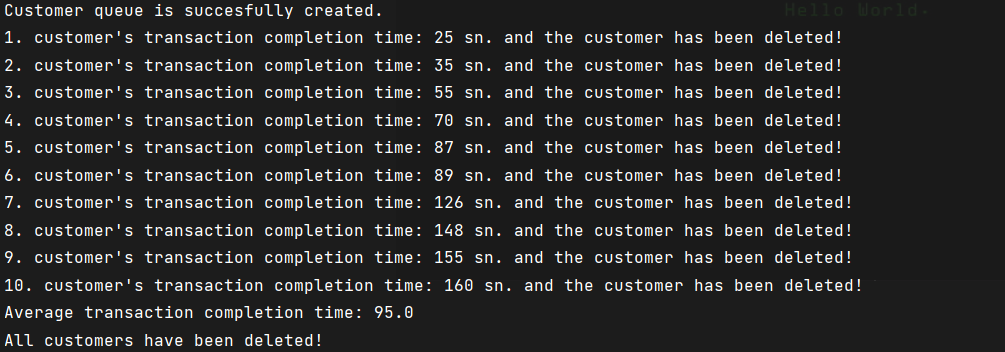
## 4.a Kuyruk kullanarak süre hesaplanması

### 4.a.1 Kaynak Kod

|  |
| --- |
| ***package*** Proje2.data\_proje2;  ***public class*** CustomerQueue {  ***public static final double*** PROCESSING\_TIME = 2.5;  ***int*** topOfQueue;  ***int*** beginningOfQueue;  ***int***[] arr = { 10, 4, 8, 6, 7, 1, 15, 9, 3, 2 };    CustomerQueue(){  beginningOfQueue = -1;  topOfQueue = arr.length - 1 ;  System.out.println("Customer queue is succesfully created.");  }    ***public static void*** main(String[] args) {  CustomerQueue cq = ***new*** CustomerQueue();  cq.calcProcessTime();  }   ***public void*** calcProcessTime(){  ***int*** totalSeconds = 0;     ***int*** person = 1;  ***for***(***int*** i = 0 ; i < arr.length; i++) {  ***int*** perCustomerTotal = 0;  ***for*** ( ***int*** j = 0; j <= i; j++){  perCustomerTotal += arr[j] \* PROCESSING\_TIME;  }  totalSeconds += perCustomerTotal;  System.out.println(person+". customer's transaction completion time" +  ": " + perCustomerTotal +" sn. and the customer has been deleted!");  person++;  beginningOfQueue++;  }  System.out.println("Average transaction completion time: " +  (***double***) totalSeconds / arr.length);  ***if***(beginningOfQueue == topOfQueue){  System.out.println("All customers have been deleted!");  }   }    ***public boolean*** isFull(){  ***if***(topOfQueue == arr.length -1 ){  System.out.println("The queue is full!");  ***return true***;  }  ***return false***;  }   ***public boolean*** isEmtpy(){  ***if***(beginningOfQueue == -1){  System.out.println("The queue is empty!");  ***return true***;  }  ***return false***;  }  } |

### 4.a.2 Ekran görüntüleri

//Konsol çıktısına ait ekran görüntülerini buraya ekleyiniz



## 4.b Öncelikli Kuyruk sınıfı kullanarak süre hesaplanması

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

|  |
| --- |
| ***package*** Proje2.data\_proje2;  ***import*** java.util.Arrays;  ***public class*** CustomerPQ {      ***public static void*** main(String[] args) {  CustomerPQ cpq1 = ***new*** CustomerPQ();  cpq1.dequeAndPrintCusts();  }   ***public static final double*** PROCESSING\_TIME = 2.5;  ***int*** beginningOfQueue;  ***int*** topOfQueue;  ***int***[] arr = { 10, 4, 8, 6, 7, 1, 15, 9, 3, 2 };   CustomerPQ(){  beginningOfQueue = -1;  topOfQueue = arr.length - 1 ;  }   ***public boolean*** isFull(){  ***if***(topOfQueue == arr.length -1 ){  System.out.println("The queue is full!");  ***return true***;  }  ***return false***;  }   ***public boolean*** isEmtpy(){  ***if***(beginningOfQueue == -1){  System.out.println("The queue is empty!");  ***return true***;  }   ***return false***;  }   ***public void*** dequeAndPrintCusts(){  ***for***(***int*** i = 0; i < arr.length; i++) {  ***int*** value = arr[i];  ***int*** j = i - 1;  ***while***(j >= 0 && arr[j] > value) {  arr[j + 1] = arr[j];  j = j - 1;  }  arr[j + 1] = value;  }  System.out.println(Arrays.toString(arr));  ***double*** totalSeconds = 0;  ***int*** person = 1;  ***for***(***int*** i = 0 ; i < arr.length; i++) {  ***double*** perCustomerTotal = 0;  ***for*** ( ***int*** j = 0; j <= i; j++){  perCustomerTotal += (***double***) arr[j] \* PROCESSING\_TIME;  }  totalSeconds += perCustomerTotal;  System.out.println(person+". customer's transaction completion time" +  ": " + perCustomerTotal +" sn. and the customer has been deleted!");  person++;  beginningOfQueue++;  }  System.out.println("Average transaction completion time: " +  (***double***) totalSeconds / arr.length);  ***if***(beginningOfQueue == topOfQueue){  System.out.println("All customers have been deleted!");  }  }   } |

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

|  |
| --- |
| ***for***(***int*** i = 0; i < arr.length; i++) {  ***int*** value = arr[i];  ***int*** j = i - 1;  ***while***(j >= 0 && arr[j] > value) {  arr[j + 1] = arr[j];  j = j - 1;  }  arr[j + 1] = value; } System.out.println(Arrays.toString(arr)); ***double*** totalSeconds = 0; ***int*** person = 1; ***for***(***int*** i = 0 ; i < arr.length; i++) {  ***double*** perCustomerTotal = 0;  ***for*** ( ***int*** j = 0; j <= i; j++){  perCustomerTotal += (***double***) arr[j] \* PROCESSING\_TIME;  }  totalSeconds += perCustomerTotal;  System.out.println(person+". customer's transaction completion time" +  ": " + perCustomerTotal +" sn. and the customer has been deleted!");  person++;  beginningOfQueue++; } System.out.println("Average transaction completion time: " +  (***double***) totalSeconds / arr.length); ***if***(beginningOfQueue == topOfQueue){  System.out.println("All customers have been deleted!"); } |

### 4.b.3 Ekran görüntüleri

## 

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

## 4.c Kuyruk ve Öncelikli Kuyruk Yapılarının Karşılaştırılması

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

1. **Kuyruk (Queue):**
   * Kuyruk, elemanların sırayla eklendiği ve sırayla çıkarıldığı bir veri yapısıdır.
   * Bu yapının çalışma prensibi "ilk gelen, ilk çıkar" (First In, First Out - FIFO) şeklindedir.
   * Örneğin, bir işlem sırası veya bir bekleme sırası gibi durumları modellemek için kullanılabilir.
2. **Öncelikli Kuyruk (Priority Queue):**
   * Öncelikli kuyruk da elemanları depolayan bir veri yapısıdır, ancak her eleman bir öncelik değeri ile ilişkilendirilmiştir.
   * Elemanlar öncelik değerine göre sıralanır, yüksek önceliğe sahip elemanlar daha önce çıkarılır.
   * Örneğin, bir işlemler listesinde bazı işlemlerin diğerlerinden daha yüksek önceliği olabilir ve öncelikli kuyruk bu durumu yönetmek için kullanılabilir.

Öncelikli kuyrukta elemanlar küçükten büyüğe sıralandıkları için toplam süre dolayısıyla bir müşterinin ortalama işlem süresi 95 saniyeden 60 saniyeye düşüyor

### 4.c.2 Önerilen Yöntem

# Müşteriler sırayla hizmet alır, bu da müşterilerin adil bir şekilde sırayla beklemesini sağlar.

# Ancak, PQ ile her müşteriye bir öncelik değeri atanır. Bu öncelik değeri, müşterinin işlem süresini etkileyen faktörleri yansıtabilir, örneğin, ürün sayısı, aciliyet durumu, vb.

# İşlem sırasında, sıradaki müşteri öncelik değerine göre PQ'dan seçilir. Bu sayede, öncelikli müşterilerin daha hızlı hizmet alması sağlanır.

# FIFO ve PQ birleşimi, adil bir sıra düzenine sahip olup, aynı zamanda öncelikli işlemlerin hızlı bir şekilde tamamlanmasını sağlar.

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 2 Maddeleri** | **Puan** | **Tahmini Not** | **Açıklama** |
| **1 a) Sınıf Oluşturma** | **5** | 5 | 7 farklı sınıf oluşturuldu |
| **1 b) Bileşik veri yapısının oluşturulması ve elemanların eklenmesi; Kaynak kod, ekran görüntüsü.** | **15** | 5 | Bileşik veri yapısı yani um alan sınıf oluştuldu addToGeneric metodu ile generic listeye eklendi |
| **1 c) Veri yapısının elemanlarının listelenmesi ve her bölgedeki UM alanı sayısının yazdırılması; Kaynak kodlar, ekran görüntüsü.** | **10** | 10 | Um alanları bir arraye atılır.Ardından for ile taramaya başlanılır. Bununla beraber Türkiyedeki her bölge için elemanları o bölgedeli şehirler olan 7 farklı list oluştrulur. Her taramada eşitlik kontrol edilir. Um alanının bulunduğu şehir Hangi bölgedeki şehire eşitse onun indeksi alınır ve bir arttırılır Daha sonra um sayısı değişkeni Taranan her bölge sonunda 0a eşitlenir |
| **2 a) Yığıt kaynak kod ve ekran görüntüleri** | **10** | 10 | oluşturuldu |
| **2 b) Kuyruk kaynak kod ve ekran görüntüleri** | **10** | 10 | oluşturuldu |
| **3 ) Öncelikli Kuyruk kod ve ekran görüntüleri** | **20** | 20 | Tüm um alanlarının ilk önce baş harfleri aynı ise sonraki harfleri, harfler farklı olana kadar bu eşitliği kontrol eden alfabetik önceliği baz alan temelinde insertion sort algoritması kullanılan bir metotla yapıldı. |
| **4) Kod, sonuçlar tablosu, ekran görüntüleri, karşılaştırma ve soruların cevapları.** | **20** | 20 | yapıldı |
| **5) Özdeğerlendirme Tablosu** | **10** | 10 | Kodlar clean code prensibine uygun, hazır bir yapı kullanılmadan hepsi benim tarafımdan oluşturuldu. |
| **Toplam** | **100** |  | 100 |

**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.**