

**T.C.**

**EGE UNIVERSITESİ**

## MÜHENDISLIK FAKÜLTESİ

**BILGISAYAR MÜHENDISLIK BÖLÜMÜ**

Proje-2.2

Banka Kuyrugu

# AD SOYADİ: Tsogtgerel CHIMEDTSEREN, Erdene JARGAL

# OGRENCİ NO: 05080010088, 05060008421

**İZMİR.16/11/2009**

**1.Gerçekleştiren Platform- Dil ve Sürüm Adi:**

Java, Netbeans IDE 6.7.1

**2.Problemin kısa tanımı:**

Bir bankaya sabahleyin gelmiş olan N kişi geliş sıralarına göre kuyruk oluşturmaktadır. Her birinin işlem süresi 20 saniye ile 500 saniye arasinda random bir zamandir. Bunun için:

1. Kuyruk oluşturulup müşterilerin işlemlerini gerçekleştirip kuyruktan çıkarıldıgında işlem tamamlanma süresini listelememizi ve N müşteri için ortalama işlem tamamlanma süresini de hesaplayıp yazdırmamızı istedi.
2. Ayni islemi ayni degerler uzerinde oncelikli kuyruk için tekrarlarken :

Önceliği şöyle: işlem süresi en az olan en önde olacak şekilde kuyruk oluşturulacak.

Ayrıca, N müşteri için ortalama işlem tamamlanma süresini de hesaplayıp yazdırınız. Bu yapıda, hangi müşterilerin FIFO kuyruğa göre, **daha az** beklediğini listeleyiniz (işlem süreleri ve sıra numaraları ile). Ortalama işlem tamamlanma süresindeki kazancı (a şıkkına göre) (fark ve %) hesaplayınız.

1. Aynı işlemi öncelik kuyruğu elemanlarını büyükten küçüğe sıralı tutarak tekrarlayınız.

**3.Kullanılan Veri Yapılarının, Sınıflarının ve Metotlarının tanımı:**

**Veri yapıları:**

**Vektör veri yapısı:**

Dizi veri yapısına benzer ama bellek kullanım açısından daha etkin bir çözüm.

**Sınıflar:**

**Müşteri sınıfı:**

Kuyruktakı müşteriler nesne olarak bu sınıftan türetilir. Sınıf elemanları olarak müşteri no, müşterinin işlem süresi ve degişik kuyruklarda geçen bekleme süreleri(FIFO,KtenB,BdenK) tutulacak.

**Metototlar:**

Metotları olarak random saniye üreten islem\_suresi\_hesapla() metodu ve getter setter metotları var.

**Banka sınıfı:**

Banka sınıfı içinde müşteri kuyruğu oluşturulup problemin a,b,c kısımları için ayrı ayrı işlemler yapılır. Işlemi tamalanan müşterileri kuyruktan çıkarıldığını varsayarak yanı işlem süresi bitince çıktıları elde ediyoruz ama müşteri kuyrukta kalıyor. Çünkü aynı degerlerin tekrar üretilip kuyruğun oluşturulması random sayı üreten fonksıyona bağlıdır. O yüzden problemin b ve c kısımları için aynı vektörü(kuyruğu) ele alıp sıralama işlemi yaptık.

**Metotlar:**

Banka ()-yapıcı metodu:

initKuyruk() metodunu kullanarak ilk deger atama yapıyor.yanı kuyruk kullanıcıdan verilen degere göre yaratılyor.

Run() metodu:

Main metotda sınıf nesnesi oluşturulduğunda Programın çalışmasını sağlayaıp gerekli metotları çağırır.

runAccordA() metodu:

problemin a kısmını gerçekleştirir. Müşterilerin işlem süresini ele alıp for dönğüsü içinde kuyrukta bekleme süresini belirleyip ekrana liste halinde yazdırır. Ayrıca, N müşteri için ortalama işlem tamamlama süresini de yazdırır.

kucuktenBuyugeSirala() metodu:

Bubble sorting algotımasını kullanarak kuyruğu küçükten büyüğe sıralar.

runAccordB()metodu:

problemin b kısmını gerçekleştirir. kucuktenBuyugeSirala() metodunu kullanarak kuyruğu sıralı haline getirip for dönğü içinde müşterilerin işlem süresini hesaplayıp bekleme sürelerini listeler. Sonradan toplamWait değişken yardımıyla ortalama bekleme süreyi ve

listKtenB\_FIFO metoduyla kazanclı müşterileri listeler. En sonunda Ortalama işlem tamamlanma süresindeki kazancı (a şıkkına göre) (fark ve %) hesaplayıp ekrana yazdırır.

runAccordC() metodu:

problemin c kısmını gerçekleştirir. buyuktenKucugeSirala() metot yardımıyla kuyruğu sıralayıp sonradan runAccordB()metodun aynısnı işler. Ama listKtenB\_FIFO metot yerine listBtenK\_FIFO metodunu kullanır çünkü farklı kuyruklar üzerine işlem yapıyor.

buyuktenKucugeSirala() metodu:

kucuktenBuyugeSirala() metodunu kullanarak kuyruğu büyükten kücüğe sıralayıp tersCevir() ile çevirıp büyükten küçüğe sıralı kuyruk oluşturur.

tersCevir() metodu:

Yeni kuyruk yaratıp geçici kuyruk olarak kullanacak. Yeni kuyruğa sira kuyruğu ters ekleyecek. Sonradan sira’ya yeni’yi atar. Böylece ters kuyruk yaratılmış olur.

initKuyruk() metodu:

müşteriden değer alıp değere göre kuyruğu oluşturur.

listKtenB\_FIFO metodu:

For dönğü içinde müşterilerin KtenB ve FIFO değerlerinin farkını bulup 0’dan fazlaysa yanı kazançlıysa ekrana yazar.

listBtenK\_FIFO metodu:

For dönğü içinde müşterilerin BtenK ve FIFO değerlerinin farkını bulup 0’dan fazlaysa yanı kazançlıysa ekrana yazar.

**Müşteri sayısını gir: 5 girilidiğinde :**

FIFO'ya gore musteriler islendi !\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 Nolu musteri bekleme suresi :256

1 Nolu musteri bekleme suresi :640

2 Nolu musteri bekleme suresi :724

3 Nolu musteri bekleme suresi :1135

4 Nolu musteri bekleme suresi :1612

Musterilerin ortalama bekleme suresi : 873\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

kucukten buyuge sirali musteriler islendi !\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 Nolu musteri bekleme suresi :84

0 Nolu musteri bekleme suresi :340

1 Nolu musteri bekleme suresi :724

3 Nolu musteri bekleme suresi :1135

4 Nolu musteri bekleme suresi :1612

Musterilerin ortalama bekleme suresi : 779\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FIFO ' ya gore daha az bekleyen musteriler

2 nolu musteri islem suresi : 84

Ortalama islem tamamlanma suresindeki kazanc(FIFO'ya gore)

fark = 94.0

yuzde = 0.10767468499427263

buyukten kucuge sirali musteriler islendi !\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4 Nolu musteri bekleme suresi :477

3 Nolu musteri bekleme suresi :888

1 Nolu musteri bekleme suresi :1272

0 Nolu musteri bekleme suresi :1528

2 Nolu musteri bekleme suresi :1612

Musterilerin ortalama bekleme suresi : 1155\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FIFO ' ya gore daha az bekleyen musteriler

4 nolu musteri islem suresi : 477

3 nolu musteri islem suresi : 411

Ortalama islem tamamlanma suresindeki kazanc(FIFO'ya gore)

fark = -282.0

yuzde = -0.3230240549828179

**Müşteri sayısını gir: 10 girilidiğinde :**

FIFO'ya gore musteriler islendi !\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 Nolu musteri bekleme suresi :354

1 Nolu musteri bekleme suresi :662

2 Nolu musteri bekleme suresi :933

3 Nolu musteri bekleme suresi :1399

4 Nolu musteri bekleme suresi :1593

5 Nolu musteri bekleme suresi :1789

6 Nolu musteri bekleme suresi :2123

7 Nolu musteri bekleme suresi :2478

8 Nolu musteri bekleme suresi :2755

9 Nolu musteri bekleme suresi :3035

Musterilerin ortalama bekleme suresi : 1712\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

kucukten buyuge sirali musteriler islendi !\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4 Nolu musteri bekleme suresi :194

5 Nolu musteri bekleme suresi :390

2 Nolu musteri bekleme suresi :661

8 Nolu musteri bekleme suresi :938

9 Nolu musteri bekleme suresi :1218

1 Nolu musteri bekleme suresi :1526

6 Nolu musteri bekleme suresi :1860

0 Nolu musteri bekleme suresi :2214

7 Nolu musteri bekleme suresi :2569

3 Nolu musteri bekleme suresi :3035

Musterilerin ortalama bekleme suresi : 1460\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FIFO ' ya gore daha az bekleyen musteriler

4 nolu musteri islem suresi : 194

5 nolu musteri islem suresi : 196

2 nolu musteri islem suresi : 271

8 nolu musteri islem suresi : 277

9 nolu musteri islem suresi : 280

6 nolu musteri islem suresi : 334

Ortalama islem tamamlanma suresindeki kazanc(FIFO'ya gore)

fark = 252.0

yuzde = 0.14719626168224298

buyukten kucuge sirali musteriler islendi !\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3 Nolu musteri bekleme suresi :466

7 Nolu musteri bekleme suresi :821

0 Nolu musteri bekleme suresi :1175

6 Nolu musteri bekleme suresi :1509

1 Nolu musteri bekleme suresi :1817

9 Nolu musteri bekleme suresi :2097

8 Nolu musteri bekleme suresi :2374

2 Nolu musteri bekleme suresi :2645

5 Nolu musteri bekleme suresi :2841

4 Nolu musteri bekleme suresi :3035

Musterilerin ortalama bekleme suresi : 1878\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FIFO ' ya gore daha az bekleyen musteriler

3 nolu musteri islem suresi : 466

7 nolu musteri islem suresi : 355

6 nolu musteri islem suresi : 334

9 nolu musteri islem suresi : 280

8 nolu musteri islem suresi : 277

Ortalama islem tamamlanma suresindeki kazanc(FIFO'ya gore)

fark = -166.0

yuzde = -0.0969626168224299