**VERİ YAPILARI**

**PROJE – 2**

**Proje 2/A:……………………….** Balon Problemi.

**Proje 2/B:……………………….** Banka Problemi.

**05-06-7735 Uğur SUİÇMEZ**

**05-06-7696 Ali Rıza ADIYAHŞİ**

**Proje 2/A (Balon Problemi)**

**Gerçekleştirilen Dil** **:** JAVA

**Gerçekleştirilen Platform**  **:** Eclips Galileo 3.5.0

Problem de istenen beş adet rastgele renklerde balon alabilen ve sayısı kullanıcı tarafından belirlenen sepetler oluşturmak ve bu sepetleri dairesel çift bağlı bir listede tutmak. Yalnız sepetlerde balonlar da yığıt yada kuyruk veri yapısı şeklinde tutulmalıdır. Dairesel bağlaçlı listedeki sepetlerden hangisi kuyruk sepeti hangisi yığıt sepeti olduğu şöyle belirleniyor; en bastan başlanarak sepetler sayılıyor ve kaçıncı sepet olduğuna bakılıyor;

* Sepetin sırası tek sayı ise sepet yığıt sepeti olur.
* Sepetin sırası çift sayı ise sepet kuyruk sepeti olur.

Sepet sayısı ‘m’ ve sepetlerin seçilme aralığı ‘n’ kullanıcıdan alınmalı. Ve sepetlerde balon oldukça dairesel bağlaçlı listede dolaşılıp sepetlerden balon patlatılmalı, eğer sepet boşsa o sepet atlanır. Ve böylece sepetlerde balonlar bitene kadar işlem devam eder.

Balonlar patlatılma işlemine geçmeden önce dairesel tek bağlaçlı listedeki sepetlerdeki renkler ekrana yazdırılmalı.

Her turda ‘n’ değeri, sepetlerde kalanların ve sepetlerden silinenlerin renklerini ekrana yazdırılmalı.

Kullanılan bilgisayarın 3 saniyede beş farklı (m,n) kombinasyonu için kaç adet balon problemi çözebildiği hesaplanmalı.

**Kullanılan veri yapıları, sınıflar ve metotlar**

**Veri yapıları:**

liste: Sepetleri dairesel tek bağlı listede tutan veri yapısı.

sepetY: Bolanları yığıt(Stack) tipinde tutan yığıt sepetini tanımlar.

sepetQ: Balonları kuyruk(Queue) tipinde tutan kuyruk sepetini tanımlar.

**Sınıflar:**

sepetListesi: Dairesel tek bağlı liste oluşturmak için kullanıldı.

Dugum: İçinde veri ve başka bir nesneye referans olan iki bölümden oluşur ve sepetlerin her birini temsil için tanımlanmıştır.

**Metotlar:**

elemanEkle: listeye sepet eklemek için tanımlanmıştır.

**Not:**

*Ayrıca programın içinde hazır sınıf olarak* ***Stack*** *ve* ***Queue*** *sınıfları, hazır* ***Object****(Stack ve Queue yi nesneye çevirmek için),* ***System.nanoTime****(programın 3 saniyede kaç problem çözebildiğini hesaplamak için bilgisayardan zaman almak için),* ***instanceof****(object tipine geçirilen nesnelerin tipine ulaşmak için ) metotlarıda kullanılmıştır.*

**Programın örnek çıktısı:**

listenin 1. elemani Yigit Sepetidir.

Sepetin icindeki balonlarin renkleri:

kirmizi

beyaz

gri

turuncu

kirmizi

listenin 2. elemani Kuyruk Sepetidir.

Sepetin icindeki balonlarin renkleri:

kirmizi

siyah

kirmizi

siyah

turuncu

listenin 3. elemani Yigit Sepetidir.

Sepetin icindeki balonlarin renkleri:

siyah

turuncu

gri

mavi

turuncu

patlatilan balon [kirmizi]

2. sepetten patlatildi!

2. sepette kalanlar:

[siyah, kirmizi, siyah, turuncu]

patlatilan balon [turuncu]

3. sepetten patlatildi!

3. sepette kalanlar=

[siyah, turuncu, gri, mavi]

patlatilan balon [kirmizi]

1. sepetten patlatildi!

1. sepette kalanlar:

[kirmizi, beyaz, gri, turuncu]

patlatilan balon [siyah]

2. sepetten patlatildi!

2. sepette kalanlar:

[kirmizi, siyah, turuncu]

patlatilan balon [mavi]

3. sepetten patlatildi!

3. sepette kalanlar=

[siyah, turuncu, gri]

patlatilan balon [turuncu]

1. sepetten patlatildi!

1. sepette kalanlar:

[kirmizi, beyaz, gri]

patlatilan balon [kirmizi]

2. sepetten patlatildi!

2. sepette kalanlar:

[siyah, turuncu]

patlatilan balon [gri]

3. sepetten patlatildi!

3. sepette kalanlar=

[siyah, turuncu]

patlatilan balon [gri]

1. sepetten patlatildi!

1. sepette kalanlar:

[kirmizi, beyaz]

patlatilan balon [siyah]

2. sepetten patlatildi!

2. sepette kalanlar:

[turuncu]

patlatilan balon [turuncu]

3. sepetten patlatildi!

3. sepette kalanlar=

[siyah]

patlatilan balon [beyaz]

1. sepetten patlatildi!

1. sepette kalanlar:

[kirmizi]

patlatilan balon [turuncu]

2. sepetten patlatildi!

2. sepette kalanlar:

[]

patlatilan balon [siyah]

3. sepetten patlatildi!

3. sepette kalanlar=

[]

patlatilan balon [kirmizi]

1. sepetten patlatildi!

1. sepette kalanlar:

[]

Bu bilgisayar 3 saniyede ortalama bu probleme benzer 142 tane balon problemi cozer

**PROJE 2-B(Banka Problemi)**

**Gerçekleştirilen Dil** **:** JAVA

**Gerçekleştirilen Platform**  **:** Eclips Galileo 3.5.0

**KISA TANITIM**

Bankaya gelen müşterileri bir kuyruğa atıyoruz. Önce kuyruktan normal şekilde çıkarıyoruz. Musterilerin işlem tamamlama sürelerini, numaralarını, işlem sürelerini listeliyoruz. İkinci olarak kuyruktaki müşterileri büyükten küçüğe sıralıyoruz. Muşteri bilg,ilerini listeliyoruz ve ilk kuyruğa göre ortalama işlem tamamlama süresinde ne kadar kazanç sağladığını yazdırıyoruz ve kuyrukta ilk kuyruğ göre daha az bekleyen musterilerin bilgisiyle birlikte ne kadar daha az beklediğini listeliyoruz. Üçüncü olarak kuyruğu küçükten büyüğe sıralıyoruz. Musterielerin bilgilerini listeledikten sonra ortalama işlem tamamlanma süresini yazdırıyoruz. Daha sonra ilk kuyruğa göre daha fazla bekleyen müşterilerin bilgileriyle beraber ne kadar fazla beklediğini listeliyoruz. En sonunda ilk kuyruğa göre ortalama işlem tamamlanma süresinde ne kadar kayıp olduğunu yazdırıyoruz.

**KISA AÇIKLAMALAR**

**CLASSLAR:**

**Musteri:**müşterilerin bilgilerini tutan sınıf.

**Kuyruk:**müşteri tipinde değişkenlerinden oluşan kuyruk oluşturan sınıf. İçinde de aynı tipte bir constracter metod var.

**bKuyruk:** müşteri tipinde değişkenlerinin azalan ve artan sırada kuyruk oluşturan sınıf içinde aynı tipte bir constracter içeriyor.

**KuyrukTest:**işlemlerin yapıldığı sınıf.

**METODLAR:**

**(Kuyruk)ekle:**kuyruk şeklinde müşteri tipindeki değişkenleri ekliyor.

**(Kuyruk)cikar:**kuyruktan eleman çıkartıp bilgilerini tutuyor.

**(Kuyruk)bosMu:**kuyruğun boş olup olmadığına bakıyor.

**(bKuyruk)ekle:**kuyruğa büyükten küçüğe ekleme yapıyor.

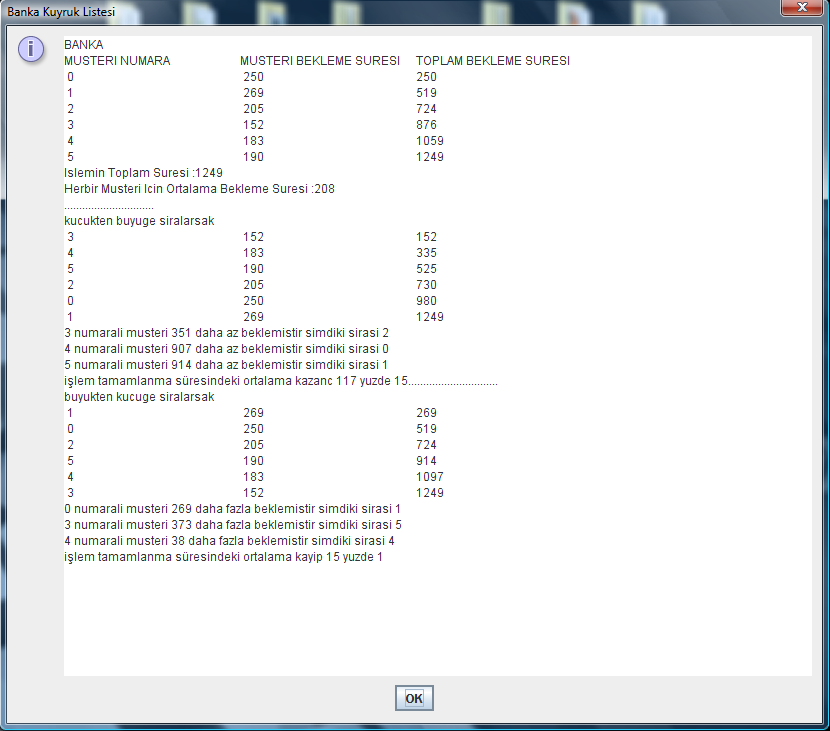
**bKuyruk.cekle:**kuyruğa küçükten büyüğe ekleme yapıyor.

**bKuyruk.sil:**kuyruktan eleman silip bilgilerini tutuyor.

**VERİ YAPILARI:**

numara:müşterinin numarasını tutuyor, sure:müşterinin süresini tutuyor,boyut:kuyruğun boyutunu tutuyor,kuyrukDizi:müşteri tipinde dizi , bas:kuyruğun başını tutuyor,son:kuyruğun sonunu tutuyor,elemanSayisi:kuyruğun ne kadar eleman olacağını tutuyor, bilgisi:müşteri tipinde değişken kuyruktan çıkarılan müşeriyi tutuyor, musteri sayisi:kuyruğun kaç elemanlı olacağını tutuyor, bilgi:eklenecek müşteri, yeni:müşteri tipinde sıralamak için kullanılan geçici değişken, temp:kuyruktan çıkarılacak elemanı tutuyor, işlem:random alınan işlem süresini tutuyor, q:Kuyruk tipinde veri yapısı ilk kuyruk için kullanılıyor, qb:bKuyruk tipinde büyükten küçüğe sıralamak için kullanılıyor, qc:bKuyruk tipinde küçükten büyüğe sıralamak için kullanılıyor, toplam:dizi tipinde ilk kuyruktaki müşterilerin toplam işlem sürelerini tutuyor, must:çıkardığımız müşteriyinin bilgilerini attığımız değişken, toplamb:dizi tipinde ilk önce 2. Kuyruğun müşterilerinin toplam işlem tamamlanma sürelerini daha sonra 3. Kuyruktakilerin işlem tamamlanma sürelerini tutuyor, bul:dizi tipinde belli şekilde sıralanmış müşterilerin numaralarını tutuyor, top:1. İle 2. Kuyruğun ve 1. İle 3. Kuyruğun ortalama tamamlanma sürelerinin farkını hesaplamak için kullanılıyor, yuzde:top un 1. Musteriye oranındaki yüzdeyi tutuyor.

**ÖRNEK ÇIKTI**



Yazılım geliştirme için harcanan süreler:

Uğur SUİÇMEZ 15 saat

Ali Rıza ADIYAHŞİ 30 saat