**EGE UNIVERSITY**

**FACULTY of ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**DATA STRUCTURES**

**2012-2013 FALL SEMESTER**

**PROJECT 2**

**Proje Başlıkları:**

1. **Katlı Otopark Problemi**
2. **Banka Sıra İşlemleri Problemleri**

**Hazırlayanlar :**

**05100000068 ONUR İLHAN**

**05100000076 M. MUSTAFA TEKDEMİR**

**05110000079 ABDULLAH PEKMEZ**

1. **Gerçekleştirilen Platform, Sürüm Adı ve Programlama Dili**

Programımız C# programlama diliyle yazılmış olup, Microsoft Visual Studio platformunun 2010 Ultimate sürümünde başarıyla sonuçlandırılmıştır.

1. **Program – 1 Hakkında**

Bir fabrikadan üretim hatası ile çıkan arabalar, daha sonra kaza testlerinde kullanılmak için bir otoparka yerleştirilmesi istenmektedir. Otomobiller, otoparkın ilk yani 1. katına bir kuyruk, 2. katına bir yığıt ve 3. katına da bir dairesel bağlaçlı liste düzeninde yerleştirilip alınacaktır. Otopark 9 katlı olup bu yapı üst katlarda da tekrarlanması gerekmektedir.

Random bir kat numarası seçilmekte ve ardından 1 adet araba kaza testi için otoparktan alınmalıdır. Sadece arabaların bağlaçlı liste düzeninde yerleştirildiği katlarda, alınacak araba balon problemine benzer şekilde n adet araba atlanarak seçilmeli ve bağlaçlı listelerde kalınan yer tutulmalıdır.

**Balon problemi ise aşağıda tanımlanmıştır.**

m sayıda kişinin bulunduğu bir dairesel liste oluşturulur. Dairenin dışında bulunan bir kişi ise dairedeki kişileri elemek için hazır bulunmaktadır. Eleme işlemini ise dairenin çevresinde dolanırken n kişi atlayarak gerçekleştirir. Dairenin sonuna geldiğinde ise aynı işlemi gerçekleştirir. En son kalan kişi listeden çıkarılmış olunur.

1. **Program – 1 İçin Kullanılan Bileşik Veri Tipleri ve Metodlar**

public static void prg(int n, int secim) : Ana işlemi gerçekleştiren metod

public static void çizgi() : ekran görüntülerini estetikleştirmek amacıyla

oluşturulmuştur.

public static int menu() : menüyü listeler ve ardından kullanıcının seçimini döndürür.

public static void kuyruk\_listele(Queue kuyruk) : kuyruktaki elemanların bilgilerini

listeler

public static void yigin\_listele(Stack yigin) : kuyruktaki elemanların bilgilerini

listeler

class dugum // bağlı listenin yapısının oluşturulduğu class yapısıdır

{

public object veri;

public dugum sonraki;

public dugum(object o)

public void yazdir() // Düğümün verisini yazdırır

}

class Bliste // Bağlı liste yapısının gerçekleştirildiği class yapısıdır.

{

public dugum bas = new dugum(null);

public dugum son = new dugum(null);

public int n = -1;

public void ekleme(object o)

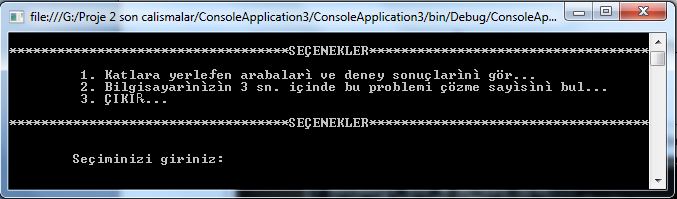
public bool bosmu()

public string çıkar()

public void listele()// Bağlı Listeyi Baştan Sona Listeler

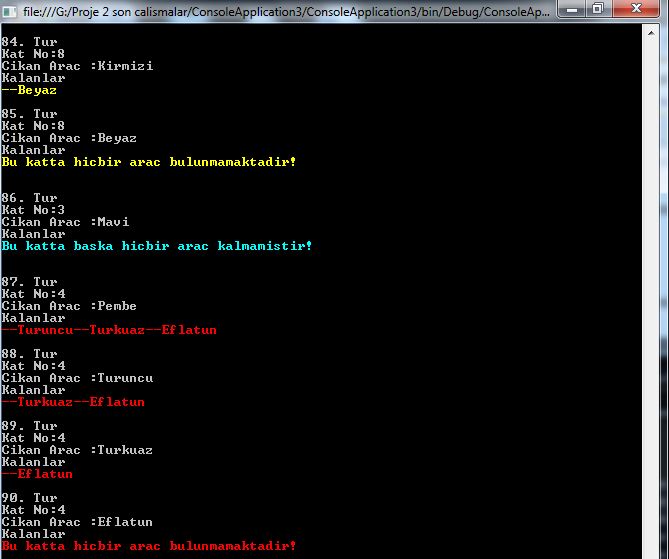
}

1. **Program – 1 ( Örnek Ekran Çıktıları)**

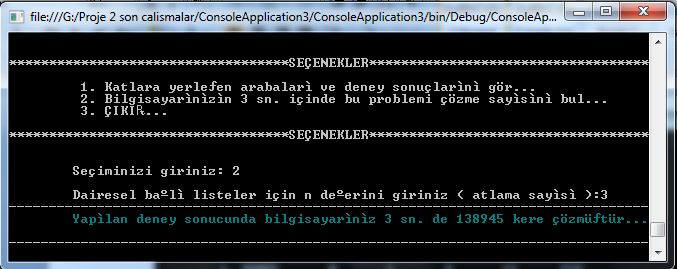
****

Programımız çalışmaya başladığında yukarıdaki menü ekranda listelenmiş olur.

Ardından menünün 1. Seçeneğinin seçilmesi ile 90 adet araç sırayla her bir turda birisi çıkacak şekilde otoparktan çıkarılır.



Menünün 2. Seçeneğinin seçilmesi ile de bilgisayarımızın 3 sn içerisinde kaç kere çözdüğü görüntülenir.



Menünün 3. ve son seçeneğinde programdan çıkış sağlanır.

1. **Program – 2 Hakkında**
2. Bir bankaya sabahleyin gelmiş olan N kişinin, geliş sıralarına göre kuyruk oluşturduklarını düşünülmektedir. Her birinin işleminin **10 saniye** ile **600 sn.** arasında random bir zaman aldığını varsayılmalı.Bunun için öncelikle, C#/Java’da FIFO kuyruk yapısı (sınıf) tasarlanması istenmektedir. Her bir banka müşterisine numara verilecek ve her birine random birer işlem süresi atayarak N elemanlı bir kuyruk oluşturulacaktır. Her bir müşterinin işi biterek kuyruktan çıkarıldığında işlem tamamlanma süresini (işlem süresi dahil ne kadar süre kuyrukta kaldığını) liste halinde yazdırılması istenmektedir. Ayrıca, N müşteri için ortalama işlem tamamlanma süresinin hesaplanması istenmektedir.
3. Aynı işlemi aynı değerleri kullanarak öncelikli kuyruk için tekrarlayıp, kuyruğu, işlem süresi en kısa olan müşteri ilk (öncelikli) olarak hizmet alacak şekilde tasarlanması istenmektedir (kuyruk, elemanları küçükten büyüğe sıralı tutmalıdır). Bu yapıda, hangi müşterilerin FIFO kuyruğa göre, **daha az** beklediğinin listelenmesi istenmektedir(işlem süreleri ve sıra numaraları ile). Ardından ortalama işlem tamamlanma süresindeki kazancın (a şıkkına göre) (fark ve %) hesaplanması istenmektedir.
4. En son olarak da aynı işlemleri (a ve b) 3 adet gişe için sıradaki müşterinin en az sayıda kişinin beklediği ilk kuyruğa girecek şekilde tekrarlanması istenmiştir.
5. **Program – 2 İçin Kullanılan Bileşik Veri Tipleri ve Metodlar**

public static int oncelik\_bilgi\_goster(OncelikKuyruk k, int k\_no) :

Oluşturulan öncelik kuyruğunun elemanlarının bilgilerini listeler ve ardından listede bulunan bütün kişilerin toplam işlem süresini geri döndürür.

public static int fifo\_bilgi\_goster(Kuyruk k, int k\_no) :

Oluşturulan öncelik kuyruğunun elemanlarının bilgilerini listeler ve ardından listede bulunan bütün kişilerin toplam işlem süresini geri döndürür.

public static int menu() : menüyü listeler ve ardından kullanıcının seçimini döndürür.

public static void çizgi() : ekran görüntülerini estetikleştirmek amacıyla

oluşturulmuştur.

class Kuyruk

{

private int kapasite;

private Musteri[] m;

private int ön;

private int arka;

private int kişi;

public Kuyruk(int s)

public void ekle(Musteri j)

public Musteri sil()

public int sayı()

}

class Musteri

{

public int no;

public int sure;

}

class OncelikKuyruk

{

public ArrayList liste;

public OncelikKuyruk()

public void ekle(Musteri eklenen) // ekle metodu

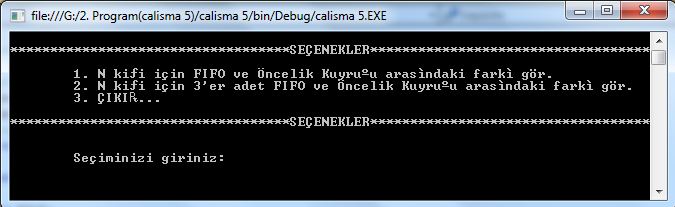
public Musteri sil() // sil metodu

public bool Bosmu()

public int eleman\_say()

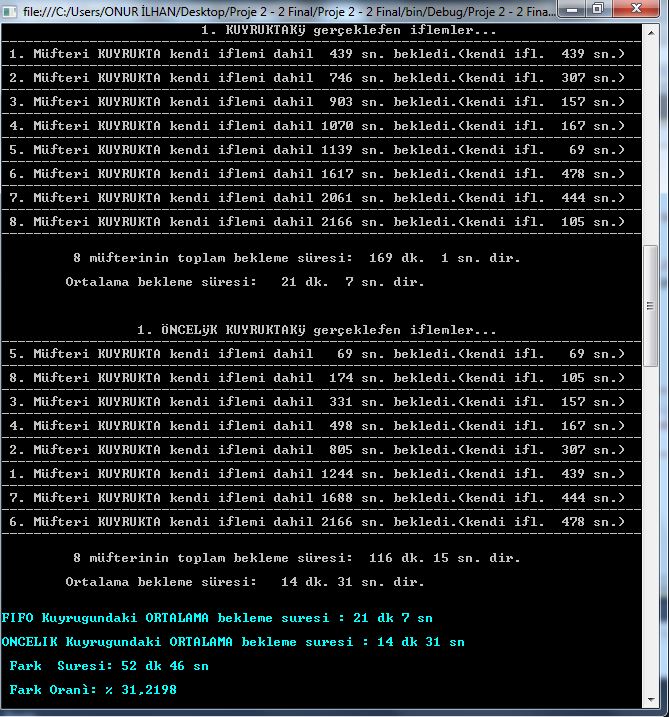
}

1. **Program – 2 ( Örnek Ekran Çıktıları)**

****

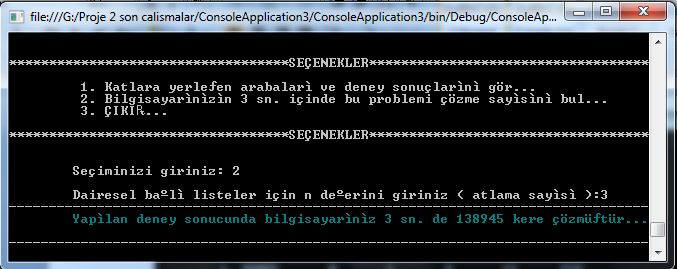
Programımız çalışmaya başladığında yukarıdaki ekran görüntüsüyle karşılaşırız. Ardından kullanıcının seçimini girmesi beklenir.

Ve 1. Seçeneğin seçilmesinin ardından fifo ve öncelik kuyruğuna yerleşen müşterilerin bilgileri ekranda listelenir. Ardından istenen genel bilgiler ekranda görüntülenir.



Menünün 2. Seçeneğinde ise bu işlemler 3 farklı fifo ve 3 farklı öncelik kuyruğu için gerçekleştirlir.

Aşağıda bunun bir örneği mevcuttur.



1. **Yazılımın Tamamlanması İçin Harcanan Süreler**

**Harcanan Süre**

1. **Program**

Tasarım : 3 Saat

Kod : 7 Saat

Rapor : 1 Saat

1. **Program**

Tasarım : 3 Saat

Kod : 8 Saat

Rapor : 1 Saat