Veri Yapıları

**Proje-2**

**Ali Yeşilkanat** – 05120000223  
**İzel Çavuşoğlu –** 05120000203  
**Gökçe Yardımcı -** 05110000049

**İçindekiler**

İçindekiler

1)Program 2.1**3**

1) Gerçekleştirilen Platform3

2) Proje Tanımı3  
3) Kullanılan Metotlar ve Bileşik Veri Tipleri ............................................................................. 3  
 3.1) Kullanılan Metotlar.................................................................................................3  
 3.2) Kullanılan Bileşik Veri Tipleri...................................................................................3  
3.3) Kullanılan Sınıflar....................................................................................................4  
4) Örnek Sonuçlar.......................................................................................................................5

2)Program 2.2**8**

1) Gerçekleştirilen Platform8  
2) Proje Tanımı8  
3) Kullanılan Metotlar ve Bileşik Veri Tipleri.............................................................................. 8  
 3.1) Kullanılan Metotlar.................................................................................................8  
 3.2) Kullanılan Bileşik Veri Tipleri...................................................................................9  
3.3) Kullanılan Sınıflar....................................................................................................9  
4) Örnek Sonuçlar.....................................................................................................................10

Ek: Yazılım Geliştirme İçin Harcanan Süre.............................................................................................12

# Program 2.1

1. **Gerçekleştirilen Platform:** Visual Studio 2010 , .Net Framework 4
2. **Proje Tanımı:** Bir fabrikadan üretim hatası ile çıkan arabalar, daha sonra kaza testlerinde kullanılmak için bir otoparka yerleştirilmektedir. Otomobiller, otoparkın ilk yani 1. katına bir kuyruk, 2. katına bir yığıt ve 3. katına da bir dairesel bağlaçlı liste düzeninde yerleştirilip alınmaktadır.

Arabalar sadece 1. kattan çıkabilmektedir ve 1 adet araba kaza testi için otoparktan alınmaktadır. Ardından 2. kattaki yığıttan bir araba 1. kata, 3. kattaki bağlaçlı listeden de bir araba 2. kata inmektedir. Sadece arabaların bağlaçlı liste düzeninde yerleştirildiği katlarda, alınacak araba balon problemine benzer şekilde n adet araba atlanarak seçilmekte ve bağlaçlı listelerde kalınan yer tutulmaktadır.

1. **Kullanılan Metotlar, Sınıflar ve Bileşik Veri Tipleri:**

**3.1) Kullanılan Metotlar**

**private** **static** **int** **SayiAl**()  
**private** **static** **int** **SayiAl**(**int** a,**int** u)  
Sayıların kullanıcıdan alınmasını sağlar. Eğer kullanıcı sayıdan başka bir girdi girerse kullanıcıya hata mesajı verip tekrar girmesini ister. Argüman isteyen aşırı yüklenmiş versiyonu ise, kullanıcının üst ve alt sınır değerleri arasında bir sayı girmesini sağlar.

**private** **static** **int** **IslemSay**(Random r, ArrayList otopark, **int** n  
3 saniyede ortalama kaç adet otopark işlemi çözebildiğini hesaplar.

**private** **static** **void** **TumArabalariCikar**(ArrayList otopark)  
Tüm arabalar bitene kadar bir arabanın çıkarılması işlemi tekrarlanır.

**private** **static** **string** **BirArabayiCikar**(ArrayList otopark)  
Arabalar sadece 1. kattan çıkabilmektedir ve 1 adet araba kaza testi için otoparktan alınmaktadır.

**private** **static** **void** **KatlariYazdir**(ArrayList otopark)  
Katlara yerleştirilen arabaları ekrana yazdırır.

**private** **static** **void** **KatlariDoldur**(ArrayList otopark, Random r, **int** n)  
Tüm veri yapılarına 9'ar adet araba yerleştirir.

**3.2) Kullanılan Bileşik Veri Tipleri**

ArrayList otopark = **new** ArrayList();   
Otoparktaki katları tutar.

CircularList ustKat = **new** CircularList(n);  
Queue<**string**> zeminKat = **new** Queue<**string**>();  
Stack<**string**> ortaKat = **new** Stack<**string**>();  
Otoparktaki katları temsil ederler. Zemin kat kuyruk, orta kat yığın, üst kat ise dairesel listedir.

**string**[] renkler = { "Siyah", "Gümüş", "Gri", "Beyaz", "Kırmızı", "Bordo", "Mor", "Yeşil", "Sarı", "Turuncu", "Mavi", "Lacivert" };  
Rastgele üretilecek renkleri tutar

**3.3) Kullanılan Sınıflar**

**class** **Node**Düğüm sınıfıdır. Arabanın rengini ve bağlı listede sonraki arabayı işaret eden düğüm referansı tutar.  
 **class** **CircularList**Bağlı liste sınıfıdır.   
Metotlar:

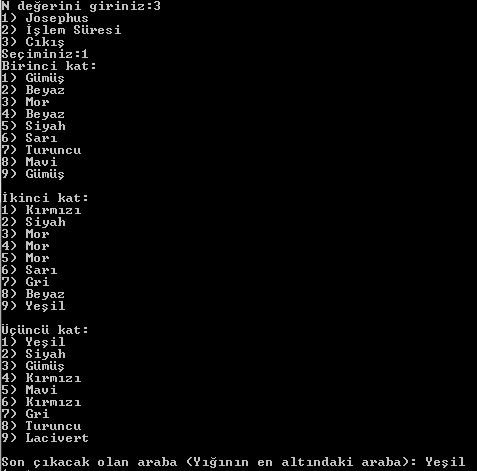
**public** **int** **GetBoyut**()  
Bağlı listenini boyutunu döndürür.

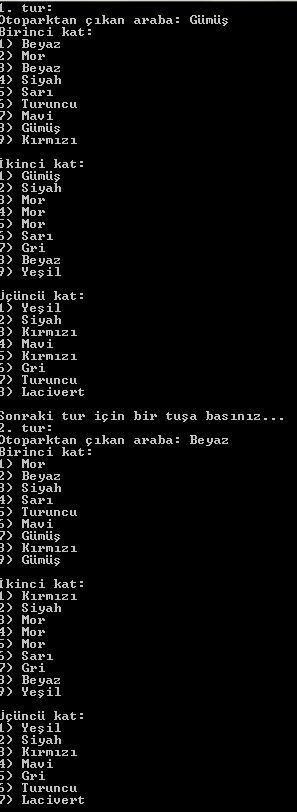
**public** Boolean **isEmpty**()  
Bağlı listenin boş olup olmadığına göre bool döndürür.

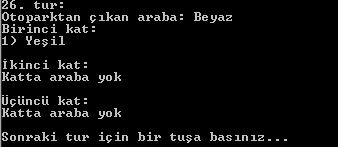
**public** **void** **Ekle**(String re)  
Bağlı listeye yeni eleman ekler.

**public** **string** **Cikar**()  
Balon problemine benzer şekilde n adet araba atlanarak çıkarma işlemi yapılır.i  
  
**private** **void** **Atla**(**int** n)   
Kullanıcıdan alınan n değerine göre balon problemine benzer şekilde n adet araba atlanır  
  
**public** **void** **yazdir**()  
Dairesel bağlaçlı listedeki arabaları yazdırır.

1. **Örnek Sonuçlar**

****





C:\Users\smsng\Desktop\Yeni klasör\4.JPG  
**İşlem Sayısı = 373154**

# Program 2.2

1. **Gerçekleştirilen Platform:** Visual Studio 2010 , .Net Framework 4
2. **Proje Tanıtımı:** Bir otoparkta sürücüleri içlerinde olan N adet arabanın otoparktan çıkarken bir kuyruk oluşturdukları düşünülmekte ve her birinin çıkış işleminin 15 saniye ile 250 saniye arasında rastgele bir zaman aldığı varsayılmaktadır. Bir araba çıkmadan diğer araba çıkış işlemine başlayamamaktadır. Her bir arabanın işi biterek kuyruktan çıkarıldığında işlem tamamlanma süresini listelenen ve N araba için ortalama işlem tamamlanma süresini hesaplatan program geliştirmemiz istenmektedir.

Ayrıca aynı işlemi aynı değerleri kullanarak, işlem süresi en kısa olan araba ilk(öncelikli) çıkacak şekilde tasarlayarak FIFO kuyruğa göre daha az bekleyen arabalar da listelenmektedir.

Aynı işlemleri 3 çıkışlı otopark için de gerçekleştiren yazılımı geliştirmemiz istenmektedir.

1. **Kullanılan Sınıflar, Metotlar ve Birleşik Veri Tipleri**

**3.1) Kullanılan Metotlar**

**public** **static** **void** **beklemeSuresiHesapla**(Kuyruk kuyruk)

Otoparkın 3 katında bulunan arabaların bekleme sürelerini hesaplar.

**public** **static** **int**[] **beklemeSuresiBul**(Kuyruk kuyruk)   
 Bu metod sadece arabaları sıraladıktan sonra arabaların bekleme süresini değiştirirken kullanılır.

Arguman olarak sadece sıralanmısListe verilir. Return edilen dizinin elemanları, sıralanmış listenin bekleme sürelerine atanır.

public static void kuyrukYazdir(Kuyruk kuyruk)

Kuyrukta bulunan arabaları yazdırır.

**public** **static** **int** **mini**(**int**[] dizi)

3’lü gişede işlem süresi en az olan gişeyi bulmak için kullanılır.

**public** **static** **int** **maxi**(**int**[] dizi)

3’lü gişede işlem süresi en fazla olan gişeyi bulmak için kullanılır.

**public** **static** **double** **ortalamaBul**(Kuyruk kuyruk)

Kuyruktaki arabaların bekleme sürelerinin ortalamasını bulur.

**public** **static** **int** **Menu**()

Kullanıcının karşısına çıkacak menüyü listeler ve kullanıcının seçimini alır.

**private** **static** **int** **SayiAl**()

Try catch metodu ile hata yakalayarak sayısal metod için string girişine izin vermez.

**private** **static** **int** **SayiAl**(**int** a, **int** u)

Try catch metodu ile hata yakalayarak sayısal metod için string girişine izin vermez ve kullanıcıdan belli sayı aralığında değer alır.

**3.2) Kullanılan Birleşik Veri Tipleri:**

**Program.cs’deki bileşik veri tipleri**

Kuyruk kuyruk = **new** Kuyruk(); : Sıralanmamış şekilde gelen ilk kuyruğu tutar.

OncelikKuyrugu on = **new** OncelikKuyrugu(); : İşlem sırası düşük olanın önceliği olan kuyruktur.

Kuyruk sl = **new** Kuyruk(); : Öncelik kuyruğuyla oluşturduğumuz sıralanmış kuyruğu atadığımız kuyruktur.

**int**[] ilkBeklemeDegerleri: “kuyruk” için her arabanın bekleme süresini tutar

**int**[] sonBeklemeDegerleri: “sl” için her arabanın bekleme sürelerini tutar.

Kuyruk[] kuyrukdizisi = **new** Kuyruk[3]; : 3 gişeli otoparkta her bir gişe için oluşan kuyruğu tutar.

**int**[] toplam = **new** **int**[3]; : “kuyrukdizisi” arrayindeki her kuyruk için, arabaların bekleme sürelerinin toplamını tutar.

**int**[] bs = **new** **int**[kuyruk.Count];  **public** **static** **int**[] **beklemeSuresiBul**(Kuyruk kuyruk) metodunda, bekleme sürelerinin listelesidir.

**OncelikKuyrugu.cs’deki bileşik veri tipleri**

**public** List<Araba> alist; : arabaları tutmak için kullanılan generic.  
 **Kuyruk.cs’deki bileşik veri tipleri**

**public** List<Araba> arabaKuyrugu; : arabaları tutmak için kullanılan generic.

**3.3) Kullanılan Sınıflar**

**class** **Araba**

Araba sınıfıdır. Arabaların işlem ve bekleme süreleri ile sıralarını tutar.

**public** **int** **CompareTo**(Araba ar)

İşlem sürelerini kıyaslar. Kuyruğu sıralamak amacıyla yazılmıştır.

**public** **override** String **ToString**()

Araba bilgilerini ekrana yazdırır.

**class** **Kuyruk**

Kuyruk sınıfıdır.

**public** **void** **Ekle**(Araba araba)

Kuyruğa araba ekler.

**public** Araba **Cikar**()

Kuyruktan araba çıkarır.

**public** Araba **ElementAt**(**int** n)

Kuyruğun belirli indeksindeki arabayı döndürür.

**public** **int** Count

Kuyruktaki araba sayısını döndürür.

**class** **OncelikKuyrugu**

Öncelik kuyruğu sınıfıdır.

**public** **void** **ekle**(Araba ar)

Öncelik kuyruğuna araba ekler.

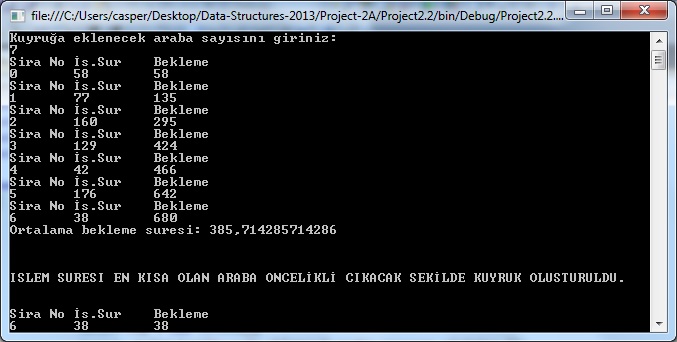
**public** Araba **cikar**()

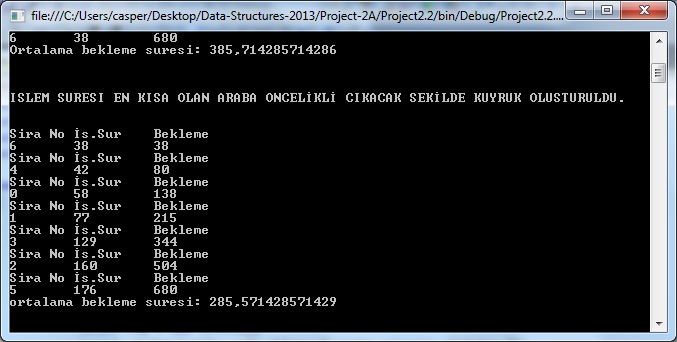
Öncelik kuyruğundan araba çıkarır.

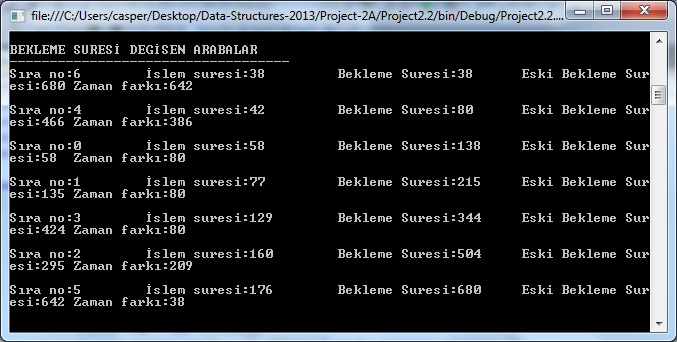
**public** Araba **ElementAt**(**int** n)

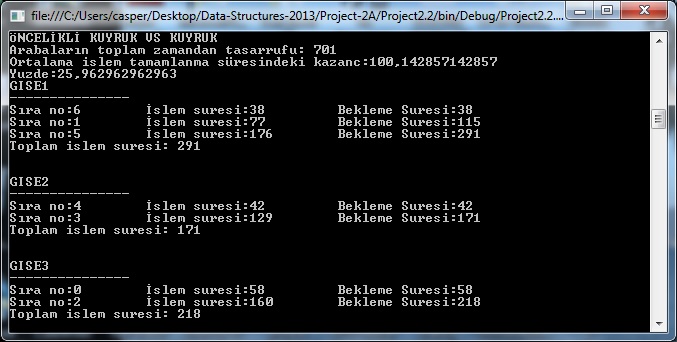
Öncelik kuyruğunun belirli indeksindeki arabayı döndürür.

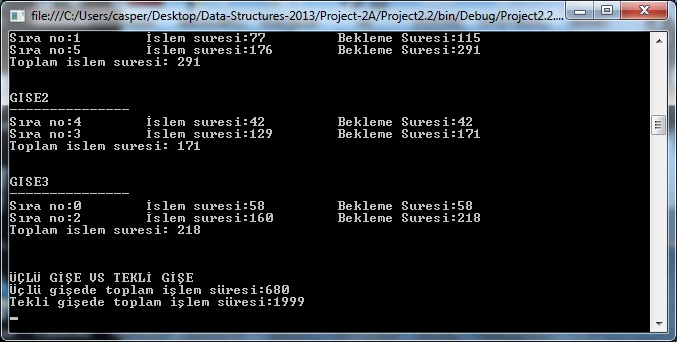
1. **Örnek Sonuçlar**

****

****

****

****

****

**Yazılım Geliştirme İçin Harcanan Süreler:**

Ali Yeşilkanat: 13 saat  
İzel Çavuşoğlu: 9 saat  
Gökçe Yardımcı: 7 saat