Queue (Kuyruk)

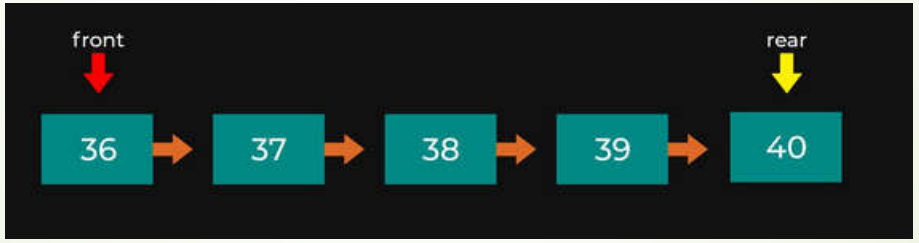
Bilginin geliş sırasına göre, ilk önce gelen elemana ilk erişilen liste yapısına **kuyruk (queue)** denir.

Bu erişimde ***First-In-First-Out (FIFO)*** prensibi vardır. Yani ilk giren eleman, ilk çıkar. Örneğin sinema bileti almak için sıraya girmiş kişileri düşünebiliriz. İlk önce gelen kişi bileti daha önce alacaktır.

Queue veri yapısında, verilere ***iki uçtan erişim*** vardır. Bir uçtan eleman ekleme ***(enqueue)***, diğer uçtan eleman çıkarma ***(dequeue)*** işlemleri yapılır.

Kuyruk işlemleri;

- ✓ Eleman Ekle **(enqueue)**
- ✓ Eleman Çıkar **(dequeue)**
- ✓ Sıradaki Elemanı Getir **(peek)**



Kuyruk Yapısı

front (ön): kuyruğun önündeki elemanı gösteren işaretçi.

rear (arka): kuyruğun arkasındaki elemanı gösteren işaretçi

Bağlı Liste ile Queue (Kuyruk)

Düğüm ve Kuyruk Sınıfı

Kuyruk yapısını bağlı liste ile gerçeklemek için, düğüm ve kuyruk sınıfları tanımlanmalı.

```
class Dugum
    {
      public  int deger;
      public  Dugum sonraki;
       public Dugum(int deger)
        {
            this.deger = deger;
            this.sonraki = null;
        }
    }

 class Kuyruk
     {
         Dugum on, arka;
         public Kuyruk()
         {
             this.on = this.arka = null;
         }
 }
```

Queue (Kuyruk)

1. Eleman Ekle (enqueue)

Öncelikle kuyruğun boş olduğu durumunu kontrol etmeliyiz. Eğer kuyruk boş ise (arka(rear) elemanı NULL ise) boş bir kuyruğa eleman ekleme işlemi yapılır.

```
if(arka == NULL)
{
on = arka = new dugum();
arka.data = deger;
}
```

Eğer kuyruk boş değilse: yeni bir düğüm oluşturmalı ve bu düğümü kuyruğun en arkasına eklemeliyiz.

```
else
{
dugum yeni= new dugum();
yeni.data = deger;
arka.sonraki = yeni;
arka=yeni;
}
```

Queue (Kuyruk)

1. Eleman Ekle (enqueue)

Öncelikle kuyruğun boş olduğu durumunu kontrol etmeliyiz. Eğer kuyruk boş ise (arka(rear) elemanı NULL ise) boş bir kuyruğa eleman ekleme işlemi yapılır.

```
//kuyruğa ekleme
        public void enqueue(int d)
        {
            //kuyruğa eklenecek düğüm
            Dugum yeni = new Dugum(d);


            //eğer kuyruk boş ise ön ve arka yeni düğüm olur
            if(this.arka==null)
            {
                this.on = this.arka = yeni;
               // return;
            }
            else
            {   // ön         arka  yeni
                // 34- 56- 77- 88    76
                //kuyruk dolu, yeni düğüm ekle. arka->sonraki=yeni
                this.arka.sonraki = yeni;
                this.arka = yeni;
            }

        }//enqueue sonu
```

Queue (Kuyruk)

2. Eleman Çıkarma (deQueue)

Öncelikle kuyruğun boş olduğu durumunu kontrol etmeliyiz. Eğer kuyruk boş ise (arka(rear) elemanı NULL ise) boş bir kuyruğa eleman ekleme işlemi yapılır.

```
public Dugum deQueue()
        {
            //eğer kuyruk boş ise Null döndür
            if (this.on == null)
                return null;

            //önceki ön saklanacak
            //ön'ü ön düğümün sonrakine ata
            Dugum silinecek = this.on;
            this.on = this.on.sonraki;

            //eğer ön null olursa arka'da null olur
            if (this.on == null)
                this.arka = null;
            return silinecek;
        }//deQUeue sonu
```

Queue (Kuyruk)

Display ve Peek()
Tüm liste ve en öndeki sıradaki eleman

```
public void goster()
        {
            Dugum t;
            t = this.on;
            while(t.sonraki!=null)
            {
                Console.Write(t.deger+"-");
                t = t.sonraki;
            }
        } //goster sonu

        public int peek()
        {
            return on.deger;
        } //peek sonu
```

Queue (Kuyruk)

Driver Kod

```
static void Main(string[] args)
        {
            Kuyruk k1 = new Kuyruk();
            k1.enqueue(20);
            k1.enqueue(44);
            k1.enqueue(88);
            k1.goster();
            k1.deQueue();
            k1.deQueue();
           Console.WriteLine("\n Son eleman:"+ k1.peek());
            k1.goster();
        }
```