# TEK YÖNLÜ BAĞLI LİSTE

**Örnek:** Öğrencilere ait no, ad, soyad, vize ve genel sınav notlarının tutulduğu tek yönlü bir bağlı liste tanımlayınız. Bu liste üzerinde arama, silme, ekleme, listeleme vb. işlemlerini gerçekleştiren metodları yazınız.

Tek yönlü bağlı listede her elemanın bir içeriği ve sonraki elemanı gösteren bir işaretçisi vardır. Bu örnek için ilk olarak **Ogrenci** isimli bir eleman, öğrenciye ait bilgiler ve bir sonraki öğrenciyi gösteren işaretçi tanımlanmıştır.

```
class Ogrenci {
    int no;
    String ad;
    String soyad;
    int vize;
    int genel;
    Ogrenci sonraki;

    Ogrenci(int no, String ad, String soyad, int vize, int genel) {
        this.no = no;
        this.ad = ad;
        this.soyad = soyad;
        this.vize = vize;
        this.genel = genel;
        sonraki = null;
    }
}
```

Dizilere ulaşmak için nasıl başlangıç adresleri kullanılıyorsa bağlı listeler için de listenin başını ve sonunu gösteren işaretçiler kullanılır. Bağlı liste tanımında listenin ilk elemanını gösteren **ilk** işaretçisi ve son elemanını gösteren **son** işaretçisi tanımlanmış ve ilk değerleri **null** değerine eşitlenmiştir.

```
public class Liste {

    Ogrenci ilk, son = null;
```

Listenin başına eleman ekleneceği zaman ilk olarak listenin boş olup olmadığı kontrol edilip boş ise **ilk** ve **son** eleman yeni eklenen eleman yapılmıştır. Eğer boş değilse listenin başına yeni eleman eklenmiş ve **ilk** değişkeni yeni eleman yapılmıştır. Ayrıca listenin eski **ilk** elemanı yeni elemanın sonrasına yerleştirilmiştir.

```
    public void BasaEkle(int no, String ad, String soyad, int vize, int genel) {
        Ogrenci ogr = new Ogrenci(no, ad, soyad, vize, genel);
        if (son == null) {
            ilk = ogr;
            son = ogr;
        } else {
            ogr.sonraki = ilk;
            ilk = ogr;
        }
    }
```

Listenin sonuna eleman ekleneceği zaman ilk olarak listenin boş olup olmadığı kontrol edilip boş ise **ilk** ve **son** eleman yeni eklenen eleman yapılmıştır. Eğer boş değilse listenin sonuna yeni eleman eklenmiş, listenin eski **son** elemanının işaretçisi yeni elemanı gösterecek şekilde değiştirilmiştir. **son** değişkeni yeni eleman yapılmıştır.

```
public void SonaEkle(int no, String ad, String soyad, int vize, int genel) {
    Ogrenci ogr = new Ogrenci(no, ad, soyad, vize, genel);
    if (ilk == null) {
        ilk = ogr;
        son = ogr;
    } else {
        son.sonraki = ogr;
        son = ogr;
    }
}
```

Araya ekleme metodunda kullanıcıdan bir sıra değeri alınmış ve yeni öğrenci bu sıra değerindeki elemanın sonrasına eklenmiştir. Kullanıcıdan alınan sıra değerinin listedeki öğrenci sayısından fazla olmaması gerekir. Bu yüzden bir metod ile listedeki öğrenci sayısı bulunmuş ve eğer girilen sıra değeri öğrenci sayısından büyük değilse araya ekleme işlemi yapılmıştır. **j** değeri gezilen elemanın sırasını göstermektedir. Listenin ilk elemanından başlanarak son elemana gelene kadar bir gezinti yapılır. Bu gezintide sıra numarası girilen **sira** değerine eşit olan eleman bulunduğunda bu elemanın sonrasına ekleme gerçekleştirilir.

```
public void ArayaEkle(int no, String ad, String soyad, int vize, int genel, int sira) {
    Ogrenci ogr = new Ogrenci(no, ad, soyad, vize, genel);
    int ogr_sayisi = OgrenciSayisi();
    int j = 1;
    Ogrenci gecici = ilk;
    if (sira <= ogr_sayisi) {
        while (gecici != null) {
            if (j == sira) {
                ogr.sonraki = gecici.sonraki;
                gecici.sonraki = ogr;
            }
            gecici = gecici.sonraki;
            j++;
        }
    } else {
        System.out.println("Araya ekleme yapılamıyor...");
    }
}
```

Listedeki öğrenci sayısını bulmak için listenin ilk elemanından başlanarak listenin son elemanına kadar bir gezinti yapılır. Her geçilen elemanda tutulan **ogrenci_sayisi** değişkeni bir artırılır. Listenin sonuna gelindiğinde toplam öğrenci sayısı bulunmuş olur.

```
    public int OgrenciSayisi() {
        int ogrenci_sayisi = 0;
        Ogrenci gecici = ilk;
        while (gecici != null) {
            gecici = gecici.sonraki;
            ogrenci_sayisi++;
        }
        return ogrenci_sayisi;
    }
}
```

Tek yönlü bağlı listede bulunan öğrencileri listelemek için listenin ilk elemanından başlanarak listenin son elemanına kadar bir gezinti yapılır. Her geçilen öğrencinin bilgileri ekrana yazdırılır.

```
public void Listele() {
    Ogrenci gecici = ilk;
    while (gecici != null) {
        System.out.println("Öğrenci No: " + gecici.no + " Adı: " + gecici.ad + " Soyadı: " + gecici.soyad
                + " Vize: " + gecici.vize + " Genel: " + gecici.genel);
        gecici = gecici.sonraki;
    }
}
```

Listenin ilk elemanı silindiğinde yeni **ilk** eleman eski **ilk** elemandan sonra gelen eleman olur. Yani **ilk** değişkeni listenin ikinci elemanı yapılır. Eğer silme sonucunda liste boşalırsa listenin **son** elemanı da **null** yapılır.

```
    public void BastanSil() {
        ilk = ilk.sonraki;
        if (ilk == null) {
            son = null;
        }
    }
```

Listenin son elemanı silindiğinde yeni **son** elemanının eski **son** elemandan önce gelen eleman yapılması gerekir. Ancak tek yönlü bağlı listede sadece ileri doğru gidebildiğimiz için sondan bir önceki elemana doğrudan ulaşamayız. Sondan bir önceki elemanı bulmak için listenin başından son elemanına kadar bir gezinti yapılır. Bu gezinti sırasında bulunulan eleman **gecici** değişkeninde bir önceki eleman ise **once** değişkeninde tutulur. Böylece **gecici** değişkeni son eleman olduğunda **once** değişkeninde sondan bir önceki eleman bulunur. Sondan bir önceki eleman bulunduktan sonra listede tek eleman olup olmadığının kontrolü yapılır. Eğer tek eleman varsa **once** değişkeni **null** olacaktır. Tek elemanlı listede silme yapıldığında liste boşalacağı için listenin **ilk** ve **son** değişkenleri **null** yapılır. Eğer birden fazla eleman varsa listenin **son** elemanı **once** değişkeninde bulunan eleman olur. Listenin yeni **son** elemanının işaretçisi de **null** yapılır.

```
public void SondanSil() {
    Ogrenci gecici, once;
    gecici = ilk;
    once = null;
    while (gecici != son) {
        once = gecici;
        gecici = gecici.sonraki;
    }
    if (once == null) {
        ilk = null;
        son = null;
    } else {
        once.sonraki = null;
        son = once;
    }
}
```

Numarası girilen öğrenci silinmek istendiğinde ilk olarak bu numaraya sahip öğrencinin listede bulunması gerekir. O yüzden listenin ilk elemanından son elemanına kadar bir gezinti yapılır. Gezinti sırasında bulunulan elemanın **no** değeri girilen **no** değerine eşit olduğunda silinecek eleman bulunmuş olur ve listeden çıkarılır. Aradan silme yapılacağı için silinen elemanın öncesinde ve sonrasında bulunan elemanların bağlanması gerekir. Silinecek elemandan önce gelen eleman da gezinti sırasında **once** değişkeninde tutulur. Silme için **once** değişkeninin **sonraki** işaretçisi silinecek elemandan bir sonraki elemanı gösterecek şekilde değiştirilir.

```
public void NumaradanSil(int no) {
    Ogrenci gecici, once;
    gecici = ilk;
    once = null;
    while (gecici.no != no) {
        once = gecici;
        gecici = gecici.sonraki;
    }
    once.sonraki = gecici.sonraki;
}
```

Numarası girilen öğrencinin bilgilerini listelemek için listenin başından sonuna kadar bir gezinti yapılır. Gezilen elemanların **no** değerleri kullanıcının girdiği **no** değeri ile karşılaştırılır. Girilen **no** değerine sahip eleman bulunduğunda o öğrenciye ait bilgiler yazdırılır.

```
public void OgrenciBilgileri(int no) {
    Ogrenci gecici = ilk;
    while (gecici != null) {
        if (gecici.no == no) {
            System.out.println("Öğrenci No: " + gecici.no + " Adı: " + gecici.ad + " Soyadı: " + gecici.soyad
                    + " Vize: " + gecici.vize + " Genel: " + gecici.genel);
            break;
        } else {
            gecici = gecici.sonraki;
        }
    }
}
```

Tek yönlü bağlı listede bulunan öğrencileri numaralarının tek ve çift olmalarına göre ayrı listeler oluşturmak için ilk olarak **Tek** ve **Cift** adında iki yeni liste oluşturulur. Listenin başından sonuna kadar bir gezinti yapılır. Gezilen elemanların **no** değerlerinin tek mi çift mi oldukları kontrol edilir. Tek ise **Tek** listesine çift ise **Cift** listesine sondan ekleme metodu kullanılarak eklenir. En sonda ise **Tek** ve **Cift** listeleri yazdırılır.

```
public void TekCift() {
    Liste Tek = new Liste();
    Liste Cift = new Liste();
    Ogrenci gecici = ilk;
    while (gecici != null) {
        if (gecici.no % 2 != 0) {
            Tek.SonaEkle(gecici.no, gecici.ad, gecici.soyad, gecici.vize, gecici.genel);
        } else {
            Cift.SonaEkle(gecici.no, gecici.ad, gecici.soyad, gecici.vize, gecici.genel);
        }
        gecici = gecici.sonraki;
    }
    System.out.println("Tek numaralı öğrencilerin listesi:");
    Tek.Listele();
    System.out.println("Çift numaralı öğrencilerin listesi:");
    Cift.Listele();
}
```

Listedeki öğrencilerin ortalamalarını bulmak için listenin başından sonuna kadar bir gezinti yapılır. Her öğrencinin vize ve genel sınav notları kullanılarak ortalama hesaplanır. Öğrenci numarası ve ortalaması ekrana yazdırılır.

```
public void OrtalamaBul() {
    double ort = 0;
    Ogrenci gecici = ilk;
    while (gecici != null) {
        ort = gecici.vize * 0.4 + gecici.genel * 0.6;
        System.out.println("Öğrenci No: " + gecici.no + " Ortalama Notu: " + ort);
        gecici = gecici.sonraki;
    }
}
```

Listede en düşük genel sınav notuna sahip olan öğrenciyi bulmak için listenin başından sonuna kadar bir gezinti yapılır. İlk başta **not** değişkenine ilk öğrencinin genel sınav notu atanır. Gezinti sırasında her öğrencinin genel sınav notu not değişkenindeki değer ile karşılaştırılır. Eğer öğrenci bu değerden daha düşük nota sahip ise o öğrenci **tmp** değişkenine atanır. Bu şekilde gezinti sonunda **tmp** değişkeninde en düşük genel sınav notuna sahip öğrenci yer almış olur.

```
public void DusukGenel() {
    int not = ilk.genel;
    Ogrenci gecici = ilk;
    Ogrenci tmp = ilk;
    while (gecici != null) {
        if (gecici.genel < not) {
            not = gecici.genel;
            tmp = gecici;
        }
        gecici = gecici.sonraki;
    }
    System.out.println("En düşük genel sınav notuna sahip öğrenci:");
    System.out.println("Öğrenci No: " + tmp.no + " Adı: " + tmp.ad + " Soyadı: " + tmp.soyad + " Genel: " + tmp.genel);
}
```

Main metodunda tek yönlü bağlı listede tanımlanan metodları kullanarak bazı işlemler gerçekleştirilmiştir. Bu işlemler bir menü şeklinde tanımlanmıştır. Menüden kullanıcının seçimine göre tanımlanan metodlar çağrılarak sonuçlar döndürülmüştür. Hazırlanan menü yapısına ve seçimlere ait kod parçası aşağıda yer almaktadır.

```
int secim;
Scanner klavye = new Scanner(System.in);
Liste ogrenciler = new Liste();
do {
    System.out.println("1- Öğrenciyi başa ekle");
    System.out.println("2- Öğrenciyi sona ekle");
    System.out.println("3- Öğrenciyi araya ekle");
    System.out.println("4- Öğrencileri listele");
    System.out.println("5- Baştaki öğrenciyi sil");
    System.out.println("6- Sondaki öğrenciyi sil");
    System.out.println("7- Numarası girilen öğrenciyi sil");
    System.out.println("8- Numarası girilen öğrencinin bilgilerini getir");
    System.out.println("9- Listedeki öğrenci sayısını getir");
    System.out.println("10- Tek ve çift numaralı öğrencileri listele");
    System.out.println("11- Öğrencilerin sınav ortalamalarını bul");
    System.out.println("12- Genel sınav notu en düşük olan öğrenciyi getir");
    System.out.println("0- Çıkış");
    secim = klavye.nextInt();
```

```
switch (secim) {
    case 1:
        System.out.println("Öğrencinin numarasını giriniz...");
        int bno = klavye.nextInt();
        System.out.println("Öğrencinin adını giriniz...");
        String bad = klavye.next();
        System.out.println("Öğrencinin soyadını giriniz...");
        String bsoyad = klavye.next();
        System.out.println("Öğrencinin vize notunu giriniz...");
        int bvize = klavye.nextInt();
        System.out.println("Öğrencinin genel sınav notunu giriniz...");
        int bgenel = klavye.nextInt();
        ogrenciler.BasaEkle(bno, bad, bsoyad, bvize, bgenel);
        break;
```

```
case 2:
    System.out.println("Öğrencinin numarasını giriniz...");
    int sno = klavye.nextInt();
    System.out.println("Öğrencinin adını giriniz...");
    String sad = klavye.next();
    System.out.println("Öğrencinin soyadını giriniz...");
    String ssoyad = klavye.next();
    System.out.println("Öğrencinin vize notunu giriniz...");
    int svize = klavye.nextInt();
    System.out.println("Öğrencinin genel sınav notunu giriniz...");
    int sgenel = klavye.nextInt();
    ogrenciler.SonaEkle(sno, sad, ssoyad, svize, sgenel);
    break;
```

```
case 3:
    System.out.println("Öğrencinin numarasını giriniz...");
    int ano = klavye.nextInt();
    System.out.println("Öğrencinin adını giriniz...");
    String aad = klavye.next();
    System.out.println("Öğrencinin soyadını giriniz...");
    String asoyad = klavye.next();
    System.out.println("Öğrencinin vize notunu giriniz...");
    int avize = klavye.nextInt();
    System.out.println("Öğrencinin genel sınav notunu giriniz...");
    int agenel = klavye.nextInt();
    System.out.println("Öğrenciyi eklemek istediğiniz sırayı giriniz...");
    int sira = klavye.nextInt();
    ogrenciler.ArayaEkle(ano, aad, asoyad, avize, agenel, sira);
    break;

case 4:
    ogrenciler.Listele();
    break;

case 5:
    ogrenciler.BastanSil();
    break;

case 6:
    ogrenciler.SondanSil();
    break;

case 7:
    System.out.println("Silmek istediğiniz öğrencinin numarasını giriniz...");
    int sil_no = klavye.nextInt();
    ogrenciler.NumaradanSil(sil_no);
    break;

case 8:
    System.out.println("Bilgilerini istediğiniz öğrencinin numarasını giriniz...");
    int no = klavye.nextInt();
    ogrenciler.OgrenciBilgileri(no);
    break;

case 9:
    int sayi = ogrenciler.OgrenciSayisi();
    System.out.println("Listedeki öğrenci sayısı: " + sayi);
    break;

case 10:
    ogrenciler.TekCift();
    break;

case 11:
    ogrenciler.OrtalamaBul();
    break;

case 12:
    ogrenciler.DusukGenel();
    break;
```