ÇİFT YÖNLÜ BAĞLI LİSTE

Çift yönlü bağlı listede her düğümün kendinden sonraki düğümü gösteren ileri işaretçisi, kendinden önceki düğümü gösteren geri işaretçisi ve içerik bilgisi (int, string vb.) bulunmaktadır. Böylece liste üzerinde iki yönlü hareket ederek, araya düğüm ekleme veya silme işlemleri tek yönlü bağlı listeye göre daha kolay hale gelebilmektedir. Bu çalışmada çift yönlü bağlı liste yapısı kullanılarak ekleme, silme, arama listeleme vb. fonksiyonları içeren Java uygulamasına yer verilmiştir. Öncelikle çift yönlü bağlı liste düğüm veri yapısı tanımlaması yapılmıştır.

```
package Cift_Yonlu_Bagli_Liste;

class Link {
        public int veri;
        public Link sonraki;
        public Link onceki;

        public Link(int d)
        {
            veri = d;
        }
```

Çift yönlü bağlı liste üzerinde gezinme veya düzenleme işlemi yapabilmek için listenin ilk ve son düğümünün bilinmesi gerekir. Yapılandırıcı içerisinde ilk ve son veriyi tutan referanslar oluşturulduktan sonra yapılandırıcı içerisinde ilk değerleri null olarak atanır. Listeye ekleme veya silme işlemleri yapılırken listenin boş olup olmadığının kontrolü de yapılmalıdır.

```
class IkiYonluListe {
        Link ilk;
        Link son;

        public IkiYonluListe()
        {
            ilk = null;
            son = null;
        }

        public boolean bosMu()
        {
            return ilk == null;
        }
```

Çift yönlü bağlı listede liste başına eleman eklerken öncelikle listenin boş olup olmadığına bakılır. Listede hiç düğüm yoksa eklenen yeni düğüm son düğüm olarak düzenlendikten sonra listenin ilk düğümü yapılır. Liste boş değilse listenin ilk elemanı artık yeni düğüm olacağından ilk elemanın önceki işaretçisi yeni düğümü göstermelidir. Yeni düğümün sonraki işaretçisi de ilk düğümü gösterdikten sonra yeni düğüm listenin ilk elemanı olarak güncellenir.

```
public void basaEkle(int yeni)
{
        Link yeniDugum = new Link(yeni);

        if (bosMu())
                son = yeniDugum;
        else
                ilk.onceki = yeniDugum;
        yeniDugum.sonraki = ilk;
        ilk = yeniDugum;
}
```

Liste sonuna eklerken de benzer adımlar uygulanır. Liste boşsa yeni düğüm listenin ilk düğümü yapıldıktan sonra son düğüm olarak da düzenlenir. Liste boş değilse listenin son düğümünün sonraki işaretçisi yeni düğümü, yeni düğümün önceki işaretçisi de son düğümü gösterdikten sonra yeni düğüm listenin son düğümü olarak güncellenir.

```
public void sonaEkle(int yeni)
{
        Link yeniDugum = new Link(yeni);
        if (bosMu())
                ilk = yeniDugum;
        else {
                son.sonraki = yeniDugum;
                yeniDugum.onceki = son;
        }
        son = yeniDugum;
}
```

Çift yönlü bağlı listede araya ekleme işlemi eklenecek yeni düğüm anahtar bilgisine sahip düğümün sonrasına gelir. Anahtar içeriğine sahip düğümü bulabilmek için liste üzerinde dolaşmak gerekir. Temp listenin ilk elemanından başlayarak anahtar içeriğini görene kadar ilerlerken null olursa yani listenin sonuna gelirse ekleme yapmaz, anahtar içeriğini bulduğunda ise fonksiyona gelen yeni değeri oluşturulan yeni düğümün içeriğine eşitleyip anahtar düğümün işaretçisi ile yeni düğümü gösterir. Eğer anahtar düğüm listenin son elemanı ise yeni düğüm eklendikten sonra yeni düğüm listenin son elemanı olarak güncellenir. Değilse ilk olarak yeni düğümün sonraki işaretçisi temp'in sonraki işaretçisinin gösterdiği düğümü gösterir, yeni düğümün önceki işaretçisi temp'i gösterdikten sonra da temp yeni düğümü işaret eder.

```
public boolean arkasinaEkle(int anahtar, int yeni) {
        Link temp = ilk;
        while (temp.veri != anahtar)
        {
                temp = temp.sonraki;
                if (temp == null)
                        return false;
        }
        Link yeniDugum = new Link(yeni);

        if (temp == son)
        {
                yeniDugum.sonraki = null;
                son = yeniDugum;
        } else
        {
                yeniDugum.sonraki = temp.sonraki;

                temp.sonraki.onceki = yeniDugum;
        }
        yeniDugum.onceki = temp;
        temp.sonraki = yeniDugum;
        return true;
}
```

Anahtar değeri ile çift yönlü bağlı liste üzerinde arama yaparken yine temp tanımlanarak liste üzerinde dolaşılabilir. Temp liste sonuna geldiğinde aranan değeri bulduğunda true bulamadığında false değerini döndürecektir.

```
public boolean Ara(int anahtar) {
        Link temp = ilk;
        while (temp.veri != anahtar)
        {
                temp = temp.sonraki;
                if (temp == null)
                        return false;
        }
        return true;
}
```

Çift yönlü bağlı listede liste başı yani ilk eleman silineceği zaman listede tek eleman varsa düğüm silindiği koşulda liste boş olur ve güncellenmesi gerekir. Yani ilk düğümün sonraki işaretçisi null ise listenin sonu null yapıldıktan sonra listenin ilk işaretçisi de null yapılır. Listede birden fazla eleman varsa listenin ikinci elemanı listenin ilk elemanı olacağından ikinci elemanın önceki işaretçisi null yapılıp listenin ilk işaretçisi de ikinci elemanı gösterecek şekilde güncellenir.

```
public Link bastanSil()
{
        Link gecici = ilk;
        if (ilk.sonraki == null)
                son = null;
        else
                ilk.sonraki.onceki = null;
        ilk = ilk.sonraki;
        return gecici;
}
```

Listenin sonundan silerken de benzer adımlar uygulanır. Listede tek eleman varsa listenin ilk işaretçisi null yapılır. Listede birden fazla eleman varsa listenin sondan ikinci düğümü artık son düğüm olacağından bu düğümün sonraki işaretçisi null olarak güncellenir. Silinen düğümler de geçici bir düğümde tutulabilir.

```
public Link sondanSil()
{
        Link gecici = son;
        if (ilk.sonraki == null)
                ilk = null;
        else
                son.onceki.sonraki = null;
        son = son.onceki;
        return gecici;
}
```

Liste ortasından düğüm silineceği zaman yine anahtar değerine sahip düğümü bulabilmek için liste üzerinde dolaşılır. Temp listenin ilk düğümünden başlayarak anahtar içeriğini görene dek ilerlerken null olursa yani yine listenin sonuna gelirse silme işlemi yapılmaz, anahtar içeriğini bulduğunda ise eğer temp yani anahtar içeriği listenin ilk düğümü ise liste başından düğüm silme işlemi yapılır, listenin son düğümü ise listenin sonundan silme işlemi yapılır. Değilse temp'in sonraki işaretçisinin gösterdiği düğümün gerisi tempten önceki düğümü gösterecek şekilde güncellenir.

```
public Link seciliSil(int anahtar)
{
        Link temp = ilk;
        while (temp.veri != anahtar)
        {
                temp = temp.sonraki;
                if (temp == null)
                        return null;
        }
        if (temp == ilk)
                ilk = temp.sonraki;
        else

                temp.onceki.sonraki = temp.sonraki;

        if (temp == son)
                son = temp.onceki;
        else

                temp.sonraki.onceki = temp.onceki;
        return temp;
}
```

Çift yönlü bağlı listede maksimum değeri bulmak için yine temp ile liste üzerinde dolaşılır.

```
void enyuksek(){
        Link temp=ilk;
        Link max=null;
        while(temp.sonraki!=null ){
                if(temp.veri>temp.sonraki.veri){
                        max=temp;}
                        temp=temp.sonraki; }
                System.out.println(max.veri);
}
```

Çift yönlü bağlı listede elemanları listelemek için yine temp ile liste üzerinde dolaşmaya başlanarak listenin son düğümüne gelene dek her düğümün içeriği yazdırılabilir.

```
public void yazdir() {
        System.out.print("Liste : ");
        Link temp = ilk;
        while (temp != null)
        {
                temp.listele();
                temp = temp.sonraki;
        }
        System.out.println("");
}
```

Çift yönlü bağlı liste üzerinde işlem yapan fonksiyonlar main metodunda yazılarak aşağıda sunulmuştur.

```
        public static void main(String[] args) {
                IkiYonluListe liste = new IkiYonluListe();
                liste.basaEkle(4);
                liste.basaEkle(3);
                liste.sonaEkle(6);
                liste.sonaEkle(7);
                liste.arkasinaEkle(4, 5);
                liste.yazdir();

                if(liste.bosMu()){
                    System.out.println("liste bos");
                }
                else{
                    System.out.println("liste bos değil");
                }

                liste.bastanSil();
                liste.yazdir();
                liste.sondanSil();
                liste.yazdir();
                liste.seciliSil(4);
                liste.yazdir();

                if(liste.Ara(4)){
                    System.out.println("bulundu");
                }
                else{
                    System.out.println("bulunamadı");
                }
        }
}
```