NESNE YÖNELİMLİ PROGRAMLAMA(Object Oriented Programming/OOP)

Öğr. Gör. Celil ÖZTÜRK
Marmara Üniversitesi
Teknik Bilimler Meslek Yüksekokulu

İçerik

- ✓ ArrayListler
- ✓ LinkedListler
- ✓ Sıralama Algoritmaları
- ✓ Arama Algoritmaları

ArrayList

- ArrayListler Java Collections Framework içinde yer alan bir sınıftır.
- Java.util paketinin altında bulunurlar.
- Dinamik diziler yada sıralanmış diziler olarak tanımlanabilirler.
- Veri eklenip silindikçe dizi uzunluğu değişkenlik gösterir.(Dinamik boyutlu)
- Arraylist ekleme,silme ve arama işlemleri için metotlara sahiptir.

ArrayList

- Add():Listeye eleman ekler.
- Add(int index,E e):Bir elemanı listede istenen indekse ekler.
- Get(int index):Listenin belirli indeksindeki elamanı verir.
- Size():Listedeki eleman sayısını verir.
- Set(int index, E e):Belirlir bir indexteki elemanı değiştirmek için kullanılır.
- Remove(E e): Listedeki bir elemanı siler.
- Remove(int i): Bir listede belirli indeksteki elemanı siler.
- Clear():Tüm listeyi temizler.

ArrayList

```
import java.util.ArrayList;


public class Main {
   public static void yazdir(ArrayList<String> a) {

      for (int i = 0; i < a.size() ; i++) {

         System.out.println("Element " + (i+1) + ": " + a.get(i));

      }
   }
```

ArrayList

```
public static void main(String[] args) {
      ArrayList<String> arraylist = new ArrayList<String>();
      arraylist.add("Andrea Bocelli");
      arraylist.add("Luciano Pavarotti");
      arraylist.add("José Carreras");
      arraylist.add("Murat Karahan");
      arraylist.add("Hakan Aysev");
      arraylist.add("Bülent Bezdüz");
     System.out.println(arraylist.get(0));
     System.out.println(arraylist.get(3));
     System.out.println(arraylist.get(5));
     System.out.println(arraylist.size());
     //arraylist.remove(2);
     //arraylist.remove("Luciano Pavarotti");
     System.out.println(arraylist.indexOf("Murat Karahan"));
     System.out.println(arraylist.lastIndexOf("Murat Karahan"));
     System.out.println(arraylist.indexOf("Arda Doğan"));
     arraylist.set(3," Murat Can Güvem ");
       yazdir(arraylist);
  }}
```

ArrayList

```
import java.util.ArrayList;
public class İlceler {
   public static void main(String[] args) {
            ArrayList<String> ilceler = new ArrayList<String>();
            ilceler.add("Kadıköy");
            ilceler.add("Beşiktaş");
            ilceler.add("Adalar");
            ilceler.add("Beykoz");
            ilceler.add("Fatih");
            /*for (int i = 0; i < ilceler.size();i++) {
                  System.out.println(ilceler.get(i));
                              }*/
            System.out.println();
            for (String s: ilceler) {
            System.out.println(s);
            }
            System.out.println("----------------------------");
            ilceler.add(1,"Üsküdar");
            for (String s: ilceler) {
                  System.out.println(s);
            }
         }
   }
```

ArrayList

```java
package arraylistl;

import java.util.ArrayList;

public class ArrayList1 {

    public static void main(String[] args) {

        ArrayList<String> arr = new ArrayList<String>();

        Ogrenci ogrenci = new Ogrenci("Eylül","20201122");
        arr.add(ogrenci.ad);
        arr.add(ogrenci.ogrenciNo);

        System.out.println("Liste uzunluk: "+arr.size());
        for(String x:arr)
        {
            System.out.println(x);
        }
    }
}
```

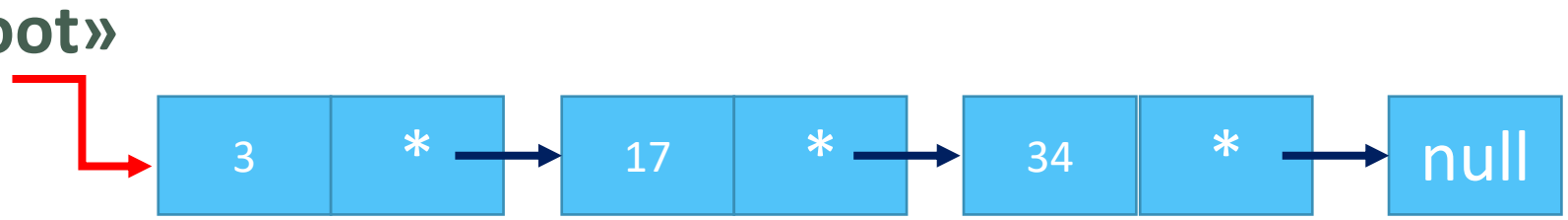
```java
package arraylistl;

/**
 *
 * @author C3lll
 */
public class Ogrenci {
    public String ad;
    public String ogrenciNo;

    public Ogrenci(String Ad, String OgrenciNo)
    {
        this.ad=Ad;
        this.ogrenciNo=OgrenciNo;
        System.out.println("Ogrenci sınıfından yeni nesne yaratıldı!! Constructor çalıştı");
    }
}
```

LinkedList(Bağlı Liste)

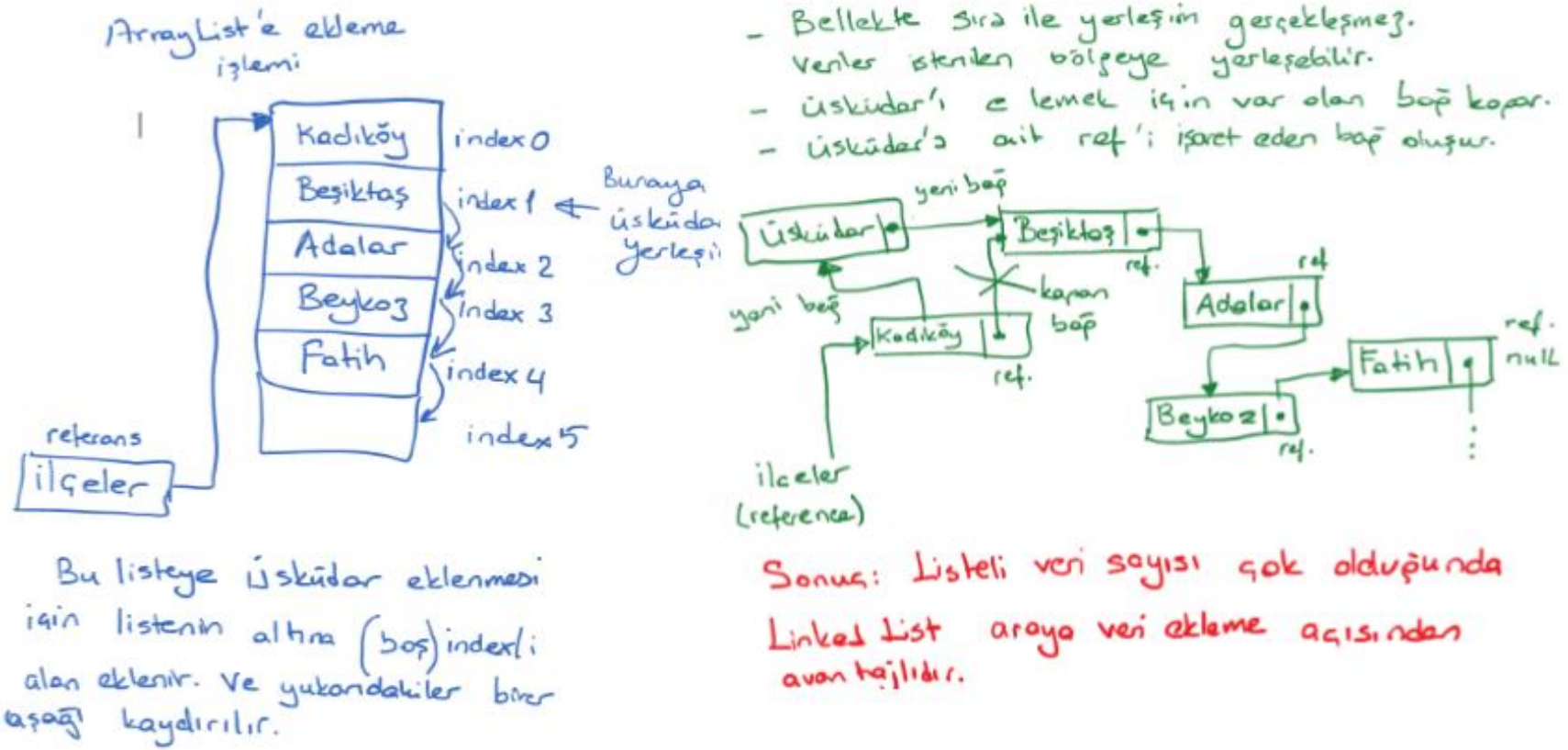
- LinkedList bellekte dağınık olarak bulunabilir.
- Linkedlistlerde bir pointer bulunur ve bir sonraki elemanı işaret eder.
- LinkendListlerde root(head) ilk elemanı temsil eder.
- Root ilk eleman olarak bir sonraki elemanı, daha sonraki her bir eleman bir sonraki elemanı temsil eder.

LinkedList(Bağlı Liste)

- Tekil Bağlı Liste



Arraylist ve LinkedList Arasındaki Farklar



LinkedList(Bağlı Liste)

```
package linkedlistornek;

import java.util.LinkedList ;
import java.util.Iterator;
public class Linkedlistornek {

    public static void main(String[] args) {
        LinkedList<String> bagliListe = new LinkedList<String>();
        bagliListe.add("Test1");    //Listeye eleman eklemek için kullanılır..
        bagliListe.add("Test2");
        bagliListe.add("Test3");
        System.out.println(bagliListe);
        bagliListe.add(1,"Araya eklenen."); //Listede bir aralığa eleman eklemek için ....
        System.out.println(bagliListe);
        System.out.println("İlk eleman:"+bagliListe.getFirst()); //Listenin ilk elemanını okumak için...
        System.out.println("son eleman:"+bagliListe.getLast()); //Listenin son elemanını okumak için...
```

LinkedList(Bağlı Liste)

```
        System.out.println("silinen ilk eleman ="+bagliListe.removeFirst()); //Listeden ilk elemanı silmek için....
        System.out.println(bagliListe);
        System.out.println("silinen son eleman ="+bagliListe.removeLast()); //Listeden son elemanı silmek için...
        System.out.println(bagliListe);

        bagliListe.addFirst("Test1-A");
        bagliListe.addLast("Test5-B");
        System.out.println(bagliListe);

        System.out.println("Eleman arama : "+ bagliListe.contains("Test2")); //Dizide elemanın olduğunu bulmak için kullanılır,
                                                                              //ve boolean döndürür.
    }
}
```

LinkedList(Bağlı Liste)

```
run:
[Test1, Test2, Test3]
[Test1, Araya eklenen., Test2, Test3]
İlk eleman:Test1
son eleman:Test3
silinen ilk eleman =Test1
[Araya eklenen., Test2, Test3]
silinen son eleman =Test3
[Araya eklenen., Test2]
[Test1-A, Araya eklenen., Test2, Test5-B]
Eleman arama : true
BUILD SUCCESSFUL (total time: 0 seconds)
```

LinkedList(Bağlı Liste)

```
import java.util.LinkedList;
import java.util.ListIterator;

public class Main {

public static void listeyi_bastir(LinkedList<String> gidilecek_yerler) {

/*for (String s: gidilecek_yerler) {
System.out.println(s);
}*/
//LinkedList üzerinde daha etkili olmak için iterator oluşturulur.
ListIterator<String> iterator = gidilecek_yerler.listIterator();
while(iterator.hasNext()) {
System.out.println(iterator.next());
}
}
```

LinkedList(Bağlı Liste)

```
public static void sirali_ekle(LinkedList<String> gidilecek_yerler,String yeni){

ListIterator<String> iterator = gidilecek_yerler.listIterator();

while (iterator.hasNext()) {

int karsilastir = iterator.next().compareTo(yeni);

if (karsilastir == 0) {

// İki değer eşit

System.out.println("Listeniz bu eleman zaten var....");

return;

}
```

LinkedList(Bağlı Liste)

```
else if (karsilastir < 0) {

// Yeni değer iterator.next ten daha büyük.

}

else if (karsilastir > 0 ) {

iterator.previous();

iterator.add(yeni);

return;

}
}

iterator.add(yeni);
}
```

LinkedList(Bağlı Liste)

```
public static void main(String[] args) {
LinkedList<String> gidilecek_yerler = new LinkedList<String>();
sirali_ekle(gidilecek_yerler, "Postane");
sirali_ekle(gidilecek_yerler, "Market");
sirali_ekle(gidilecek_yerler, "Ev");
/*gidilecek_yerler.add("Postane");
gidilecek_yerler.add("Market");
gidilecek_yerler.add("Okul");
gidilecek_yerler.add("Kütüphane");
gidilecek_yerler.add("Spor Salonu");
gidilecek_yerler.add("Ev");
listeyi_bastir(gidilecek_yerler);
System.out.println("----------------------------------------");
//gidilecek_yerler.add(4,"Alışveriş Merkezi");
//gidilecek_yerler.remove(3);
//listeyi_bastir(gidilecek_yerler); */
listeyi_bastir(gidilecek_yerler);
}
}
```

Main Metot

Sıralama Algoritmaları

Sıralama, bir grup veriyi artan ya da azalan bir şekilde art arda yerleştirme işlemidir. Sıralama işlemi hem sayısal hem de string türdeki veriler için artan yani 0'dan 9'a veya A'dan Z'ye doğru ya da azalan yani 9'dan 0'a veya Z'den A'ya doğru yapılabilir.

NOT: String ifadelerin sıralanmasında Türkçe karakterler dikkate alınmaz.

Sıralama işlemi için birçok algoritma geliştirilmiştir. Bu algoritmaların her biri farklı performansta olmakla birlikte seçilen veri modeline (dizi, bağlantılı liste gibi) göre de farklılık göstermektedir.

Sıralama Algoritmaları/Kabarcık Sıralama (Bubble Sort)

Yer değiştirmeli sıralama (Exchange Sort) olarak da bilinir. Dizide yer alan her eleman, sırasıyla kendisinden sonra gelen elemanla karşılaştırılır ve gerekiyorsa yer değiştirilir. Sıralama işlemi yer değiştirme olduğu sürece devam eder.

n elemanlı bir dizi, maksimum n-1 karşılaştırmada sıralanacaktır.

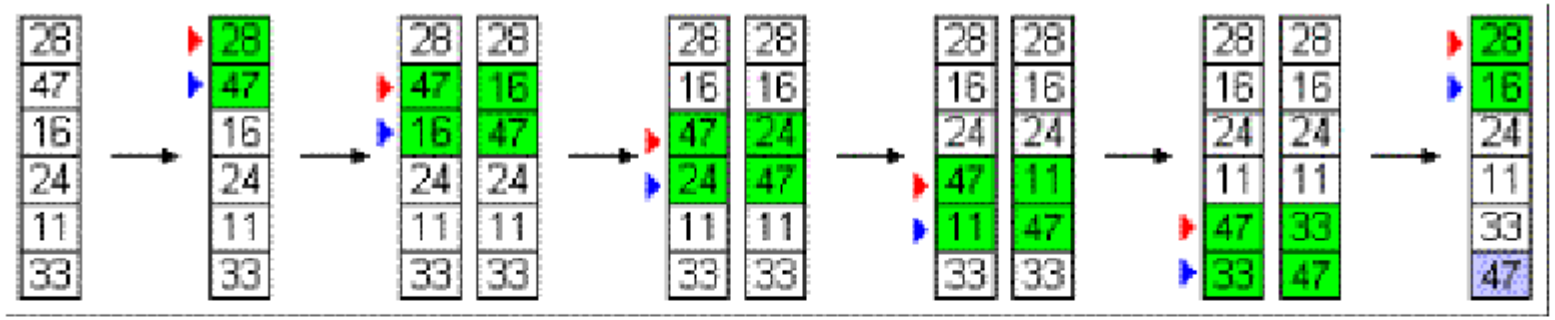
Kabarcık sıralama algoritması kullanılarak dizi elemanlarının artan şekilde sıralanması için gerekli işlem basamakları:

Sıraya konulmamış elemanların her birinin değeri, bir sonraki değerle yeni komşusu ile karşılaştırılır.

Eğer karşılaştırılan değer komşu elemandan daha büyükse, komşusu ile yer değiştirlir. Böylece sıralı olmayan elemanlar her tarandığında, sadece yan yana bulunan iki eleman arasında sıralama yapılmış olur. Dizinin başından sonuna kadar tüm elemanlar bir kez işleme tabi tutulduğunda dizinin son elemanı en büyük eleman haline gelecektir.(Yani ilk geçişte, dizi içerisindeki en büyük eleman en sona gider.)

Tüm elemanlar sıralana kadar, sıralanmamış elemanların taranma işlemi tekrarlanır.

Sıralama Algoritmaları/Kabarcık Sıralama (Bubble Sort)



Java Programlama Dilinde Kodlanması

```
for(i=0; i<n-1; i++){
        for(k=0; k<n-1-i; k++)
            if(a[k+1]<a[k]){
                Bos=a[k];
                a[k]=a[k+1];
                a[k+1]=Bos;
            }
}
```

Sıralama Algoritmaları/Kabarcık Sıralama (Bubble Sort)

- Örnek: Bilgisayarın 0 ile 100 arasında rastgele ürettiği n adet sayıyı küçükten büyüğe doğru kabarcık sıralama (bubble sort) algoritmasını kullanarak sıralayan programı yazınız.

Sıralama Algoritmaları/Kabarcık Sıralama (Bubble Sort)

Ekran Çıktısı

```
Dizi boyutunu giriniz
5
a dizisinin sıralanmamış elemanları
58      86      52      18      26
a dizisinin sıralı elemanları
18      26      52      58      86
```

```
import java.util.Scanner;
public class Kabarcik {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int i,k,Bos,n;
System.out.println("Dizi boyutunu giriniz");
n=tara.nextInt();
int a[]=new int [n];
System.out.println("a dizisinin sıralanmamış elemanları");
for (i=0;i<n;i++){
        a[i]=(1+(int)(Math.random()*100));
        System.out.print(a[i]+"\t");
}
for(i=0; i<n-1; i++){
        for(k=0; k<n-1-i; k++)
                if(a[k+1]<a[k]){
                        Bos=a[k];
                        a[k]=a[k+1];
                        a[k+1]=Bos;
                }
}
System.out.println();
System.out.println("a dizisinin sıralı elemanları");
for (i=0;i<n;i++){
        System.out.print(a[i]+"\t");
}
}}
```

Sıralama Algoritmaları/ Araya Yerleştirerek Sıralama (Insertion Sort) Algoritmas

Araya yerleştirerek/sokarak sıralama algoritmasında sıralanacak dizisinin ilk elemanı dışındaki tüm elemanlar sırasıyla incelenir. Sıralanacak dizi iki parça gibi düşünülür.

Sıralı olan kısım ve

Henüz sıralanmış olan kısım.

Sıralanmamış kısımdan elemanlar alınarak sıralı kısımda uygunyerde uygun yerde araya sokulur. Şöyle ki, dizinin ikinci elemanı alındığında, dizi 2 elemanlı gibi kendi içinde sıralanır; 3. eleman alındığında dizi 3 elemanlı gibi kendi içinde sıralanır ve benzer şekilde dizinin henüz sıralanmamış tarafından yeni bir eleman alınıp, sıralı olan tarafta araya yerleştirilir. Tüm elemanlar sıralanana kadar bu işlem devam eder.

Sıralama Algoritmaları/ Araya Yerleştirerek Sıralama (Insertion Sort) Algoritmas

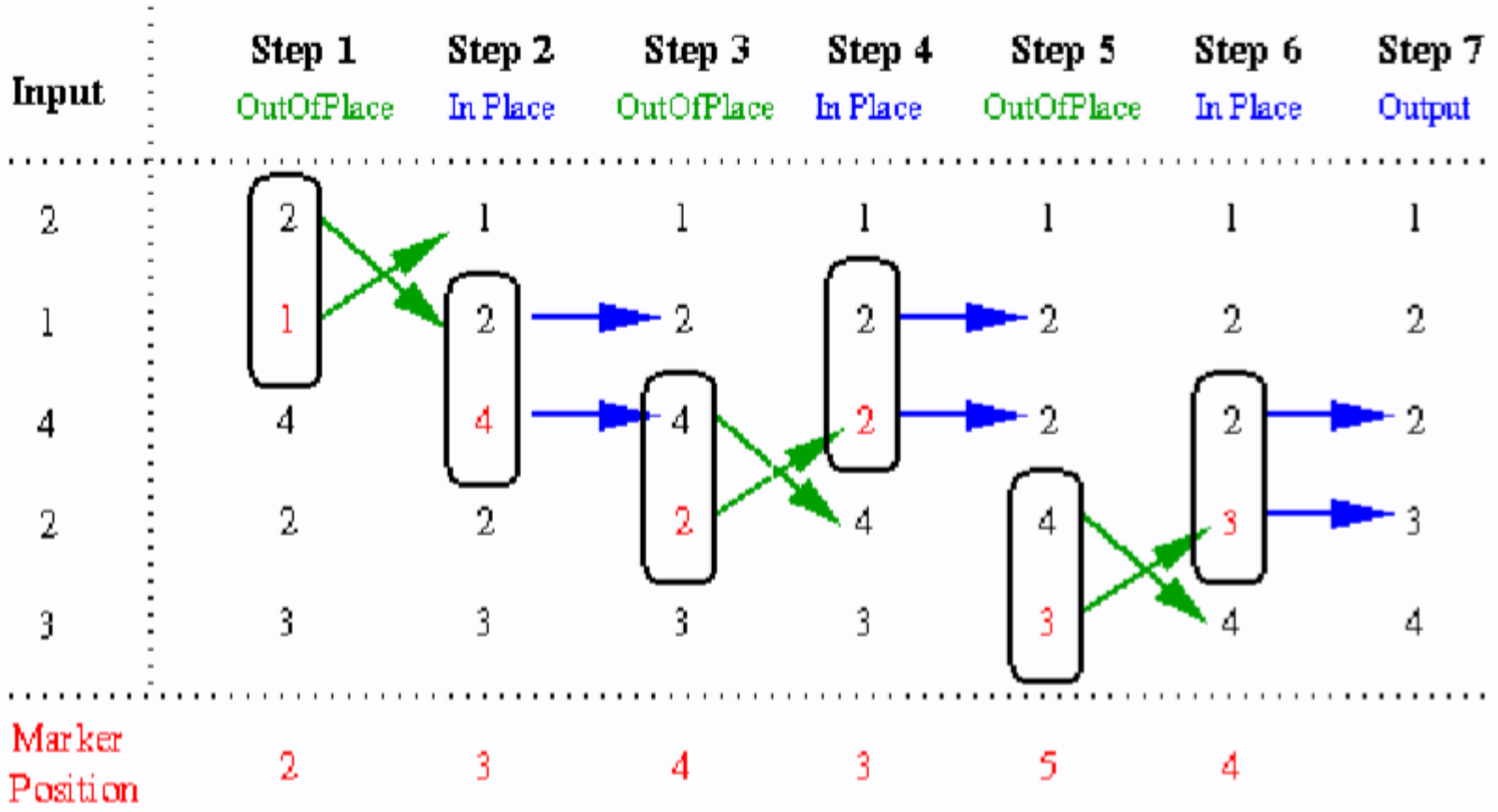
Araya yerleştirerek sıralama algoritmasını kullanarak dizi elemanlarını artan şekilde sıralamak için gerekli işlem basamakları;

Sıraya konulmamış elemanların ilkini al.

Bu elemanı, ilk olarak kendinden önceki eleman (sıraya konulmuş son eleman) ile karşılaştır. Eğer kendinden önceki elemanın değeri daha büyükse onunla yer değiştir. Bu işleme, kendisinden önceki elemanının değerini daha küçük bulana kadar devam et.

Tüm elemanlar sıraya konulana dek, yukarıdaki iki adımı tekrarla.

Sıralama Algoritmaları/ Araya Yerleştirerek Sıralama (Insertion Sort) Algoritmas



Sıralama Algoritmaları/ Araya Yerleştirerek Sıralama (Insertion Sort) Algoritmas

Java Programlama Dilinde Kodlanması

```
for (i=1; i<n; i++){
        Bos=a[i];
        while(i>0 &&a[i-1]>Bos){
                a[i]=a[i-1];
                --i;
        }
        a[i]=Bos;
}
```

NOT: while(i>0 &&a[a-1]>Bos) satırı *while(a[a-1]>Bos && i>0)* şeklinde yazılırsa program dizisinin sınırları aşıldığı gerekçesi ile duracaktır. Bunun nedenlerinden biri **&&** işaretinin simetrik bir operatör olmamasıdır.

Sıralama Algoritmaları/ Araya Yerleştirerek Sıralama (Insertion Sort) Algoritması

Ekran Çıktısı

```
Dizi boyutunu giriniz
5
a dizisinin sıralanmamış elemanları
83    52    19    68    31
a dizisinin sıralı elemanları
19    31    52    68    83
```

CEVAP:

```
import java.util.Scanner;
public class Insertion {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int i,Bos,n;
System.out.println("Dizi boyutunu giriniz");
n=tara.nextInt();
int a[]=new int [n];
System.out.println("a dizisinin sıralanmamış elemanları");
for (i=0;i<n;i++){
        a[i]=(1+(int)(Math.random()*100));
        System.out.print(a[i]+"\t");
}
for(i=1; i<n; i++){
        Bos=a[i];
        while(i>0 && a[i-1]>Bos){
                a[i]=a[i-1];
                --i;
        }
        a[i]=Bos;
}
System.out.println();
System.out.println("a dizisinin sıralı elemanları");
for (i=0; i<n; i++){
        System.out.print(a[i]+"\t");
}
}}
```

Sıralama Algoritmaları/ Seçmeli Sıralama (Selection Sort) Algoritması

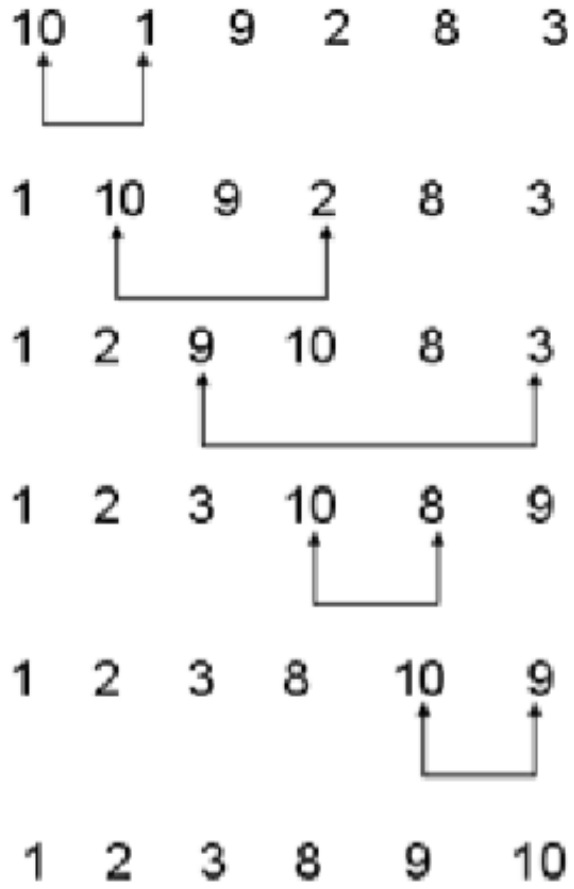
Seçmeli sıralamasını kullanarak dizi elemanlarını artan şekilde sıralamak için gerekli işlem basamakları:

Sıraya konulmamış elemanlar içindeki en küçük değerdeki elemanı bul.

Bulunan bu elemanı, sıraya konulmamış ilk eleman ile yer değiştir.

Sıraya konulmamış ilk elemanın yeri dizinin sonu olana kadar bu işlemi tekrarla.

Sıralama Algoritmaları/ Seçmeli Sıralama (Selection Sort) Algoritması



Java Programlama Dilinde Kodlanması

```
for (i=0; i<n-1; i++){
        enk=i;
        for(k=i+1; k<n; k++){
                if(a[enk]>a[k]){
                        enk=k;
                }
        }
        Bos=a[i];
        a[i]=a[enk];
        a[enk]=Bos;
}
```

Sıralama Algoritmaları/ Seçmeli Sıralama (Selection Sort) Algoritması

Örnek: Bilgisayarın 0 ile 100 arasında rastgele ürettiği n adet sayıyı küçükten büyüğe doğru seçmeli sıralama (selection sort) algoritmasını kullanarak sıralayan programı yazınız.

Sıralama Algoritmaları/ Seçmeli Sıralama (Selection Sort) Algoritması

Ekran Çıktısı

```
Dizi boyutunu giriniz
5
a dizisinin sıralanmamış elemanları
83    52    19    68    31
a dizisinin sıralı elemanları
19    31    52    68    83
```

```
import java.util.Scanner;
public class Secmeli {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int i,k,Bos,n,enk;
System.out.println("Dizi boyutunu giriniz");
n=tara.nextInt();
int a[]=new int [n];
System.out.println("a dizisinin sıralanmamış elemanları");
for (i=0;i<n;i++){
        a[i]=(1+(int)(Math.random()*100));
        System.out.print(a[i]+"\t");}
for (i=0; i<n-1; i++){
        enk=i;
        for(k=i+1; k<n; k++){
                if(a[enk]>a[k]){
                        enk=k;}
        }
        Bos=a[i];
        a[i]=a[enk];
        a[enk]=Bos;}
System.out.println();
System.out.println("a dizisinin sıralı elemanları");
for (i=0; i<n; i++){
        System.out.print(a[i]+"\t");}
}}
```

Sıralama Algoritmaları/ Hızlı Sıralama (Quick Sort) Algoritması

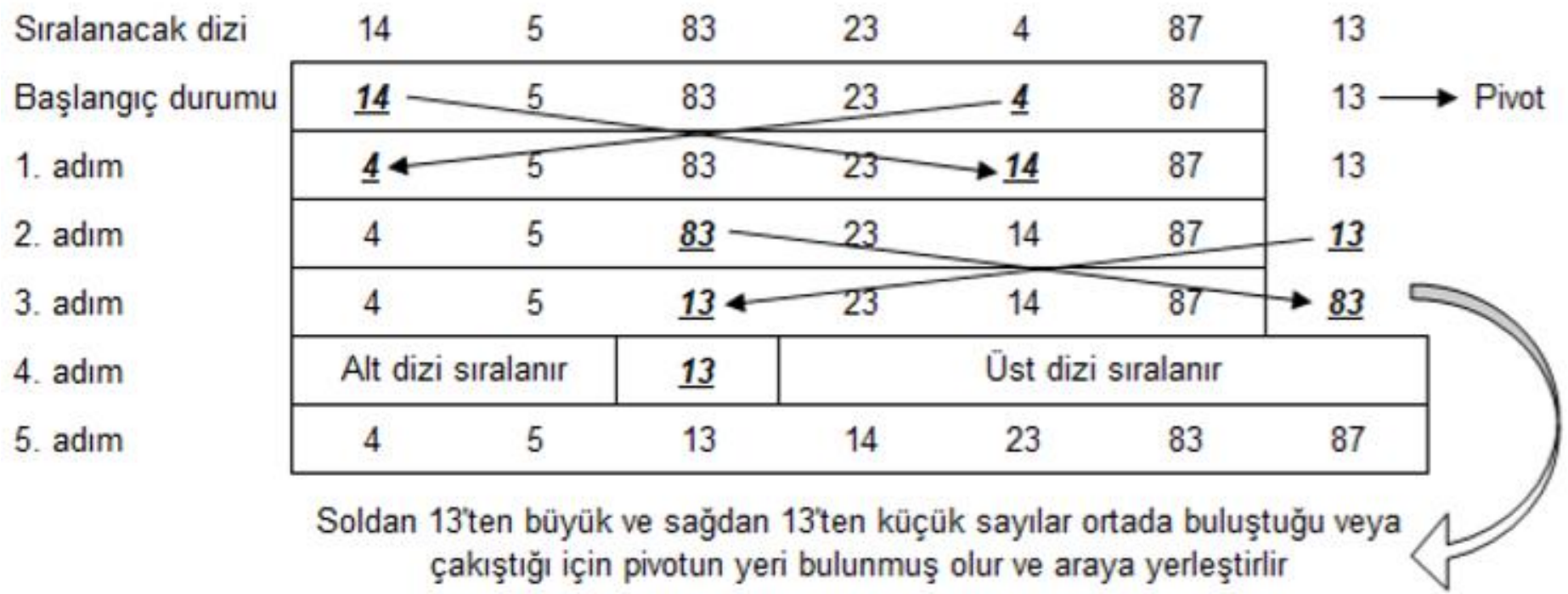
Hızlı sıralama (Quick Sort) algoritmasını kullanarak dizi elemanlarını artan şekilde sıralamak için gerekli işlem basamakları:

İlk olarak sıralanacak diziyi ikiye bölmek için Pivot (karşılaştırma değeri) seçilir. Pivot genellikle verilen dizinin ilk elemanı ya da son elemanı olabilir. Dizide pivottan büyük elemanlar pivotun sağına(üst), pivottan küçük elemanlar ise pivotun soluna (alt) konur. Pivot ise oluşan bu iki kümenin ortasına (orta) yerleşir. Böylece verilen dizi birbirinden bağımsız olarak iki alt diziye ayrılmış olur.

Hızlı sıralama algoritması bağımsız bu iki alt dizi (ust ve alt) içerisinde de recursive (özyineleme) olarak çağrılır ve bu diziler kendi içerisinde 1. adım tekrarlanarak ikiye ayrılır. Bu işlemler diziler parçalanamayacak duruma gelinceye kadar tekrarlanır.

Alt	Orta	Ust
Pivot elemanından küçük elemanlar	Pivot elemanı	Pivot elemanından büyük elemanlar

Sıralama Algoritmaları/ Hızlı Sıralama (Quick Sort) Algoritması



Sıralama Algoritmaları/ Hızlı Sıralama (Quick Sort) Algoritması

Sıralanacak dizinin son sayısını (13) pivot (karşılaştırma) elemanı olarak seçtik bu eleman daha sonraki arama ve yer değiştirme işlemlerine tabi olmaz.

Sol başta pivot elemanından örneğimizde 13'den büyük olan ilk sayı bulunur ve sağ baştan 13'den küçük ilk sayı bulunur ve bu iki sayı yer değiştirilir.

Soldan pivot elemanından büyük ve sağdan pivot elemanından küçük sayılar ortada buluşana kadar 2. Adım tekrarlanır.

3.adımda soldan 13'den büyük ve sağdan 13'den küçük sayılar ortada buluştuğu için 13 ile 83 yer değiştirir ve pivotun solundaki alt dizi ve sağındaki üst dizi kendi içinde aynı teknikle sıralanır.

Sıralama Algoritmaları/ Hızlı Sıralama (Quick Sort) Algoritması

```
import java.util.Scanner;
import java.util.*;
public class quicksort{
        static Random rnd=new Random();
        static Scanner scn=new Scanner(System.in);
        public static void quickSort(int[] a, int altindis, int üstindis) {
//bir metot oluşturuldu
        int i = altindis, j = üstindis, h;
//i değişkenine altindis yani dizinin sıfırıncı indeksi
/*j değişkenine üstindis yani dizinin uzunluğunun bir eksiği indeksi atanmış.*/
        int x = a[(altindis + üstindis) / 2];
/*pivotu dizinin ortasındaki değer olarak kabul edildi.
*Ortadaki değeri bulmak için dizinin başlangıç ve bitiş indislerini toplayıp 2 ye
bölündü.
*Pivot değeri "x" değişkenine atandı.
*/
```

Sıralama Algoritmaları/ Hızlı Sıralama (Quick Sort) Algoritması

```
do {
//sağ taraf yani üst indis altindise yetişene kadar çalışacak kodlar
            while (a[i] < x)
//pivottan küçük olan değerler varsa i yi sağa kaydırıyor
            i++;
            while (a[j] > x)
//pivottan büyük olan değerler varsa j yi sola kaydırıyor
            j--;
            if (i <= j) {
                        h = a[i];
/*pivottan büyük değerler pivotun sağına pivottan küçük değerler pivotun
soluna yerleştiriliyor , indisler yine kaydırılıyor.*/
                        a[i] = a[j];
                        a[j] = h;
                        i++;
                        j--;
                        }
}
```

Sıralama Algoritmaları/ Hızlı Sıralama (Quick Sort) Algoritması

```
while (i <= j);
//bu kısım recursive sorgu yinelemeli yani bu metodun alacağı parametre
değerleri
/*şartlara bağlı olarak atıyor*/
                    if (altindis < j)
                    quickSort(a, altindis, j);
                    if (i < üstindis)
                    quickSort(a, i, üstindis);
}
```

Sıralama Algoritmaları/ Hızlı Sıralama (Quick Sort) Algoritması

```
public static void main(String[] args) {
                        int u,i=0;
                        System.out.println("Dizinizin uzunluğunu giriniz");
                        u=scn.nextInt();
                        int[] sayi=new int[u];
                        for(i=0;i<u;i++)
                        sayi[i]=rnd.nextInt(100);
                        System.out.println("Sıralamadan önce: ");
                        System.out.println(Arrays.toString(sayi));
                        quickSort(sayi, 0, u-1);
//dizinin sıralanması için metoda değer gönderiyor
                        System.out.println("Sıralamadan sonra:");
                        System.out.println(Arrays.toString(sayi));
                                }
}
```

Sıralama Algoritmaları/ Arama Algoritmaları (Searching Algorithm)

Sıralı ya da sırasız listedeki bir elemanın yerinin bulunması işlemine arama denir.

Bir listede ya da dizi içerisinde aranan ifadeye anahtar denir.

En temel arama algoritmaları doğrusal arama (linear search) ve ikili arama (binary search) algoritmalarıdır. Genelde küçük boyutlu ve sırasız verilerin aranmasında doğrusal arama, büyük boyutlu ve sıralı verilerin aranmasında ikili arama algoritmaları tercih edilir.

Sıralama Algoritmaları/ Doğrusal Arama (Linear Search)

Doğrusal arama sıralı ya da ardışık arama olarak da isimlendirilir ve bilinen en basit arama algoritmasıdır. Kayıt sayısının az olduğu veri gruplarında arama yaparken doğrusal arama algoritması kullanılabilir.

Arama işlemine genelde dizinin başındaki elemanla başlanır, aranan bulununcaya kadar ya da listede eleman kalmayıncaya kadar devam edilir.

Sıralama Algoritmaları/ Doğrusal Arama (Linear Search)

Java Dilinde Kodlanması:

```
int x;
do{
        if (x==a[i]){
            System.out.println(x + " dizisinin " + i + ".eleman olarak bulundu");
            System.exit(0);
        }
        else{
            i=i+1;
        }
}while (i<n);
System.out.println(x +  "sayısı listede yoktur");
```

Sıralama Algoritmaları/ Doğrusal Arama (Linear Search)

Ornek: Bir grup sayı içerisinden aranan sayının olup olmadığını bulan programı yazınız.

Çözüm:

```
import java.util.Scanner;
public class Search {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int a[]={6,8,3,7,5,6,1,4};
System.out.println("Aranan sayıyı giriniz");
int aranan = tara.nextInt();
boolean durum=false;
for (int x:a)
        {if (x==aranan)
                {durum=true; break;}
}
if(durum)
        System.out.println("Aranan bulundu!");
else
        System.out.println("Aranan bulunamadı!");
}}
```

Sıralama Algoritmaları/ Doğrusal Arama (Linear

Örnek: Bilgisayarın rastgele ürettiği 1 ile 100 arasındaki sayılardan oluşan n elemanlı bir A dizisi içerisinde, aranan sayının, dizinin kaçıncı sırasında olduğunu bulan programı doğrusal arama algoritmasını kullanarak yazınız.

Çözüm:

```
import java.util.Scanner;
public class Dogrusalarama {
public static void main(String[] args) {
Scanner tara=new Scanner(System.in);
int i,n,aranan,sayac=0;
System.out.println("Dizi boyutunu giriniz");
n=tara.nextInt();
int a[]=new int [n];
System.out.println("a dizisinin elemanları");
for (i=0;i<n;i++){
            a[i]=(1+(int)(Math.random()*100));
            System.out.print(a[i]+"\t");}
System.out.println();
System.out.println("Aradığınız sayıyı giriniz");
aranan=tara.nextInt();
do
            {if (aranan==a[sayac])
                        {System.out.println(aranan + " dizisinin " + sayac + ".elemanı olarak bulundu");
                        System.exit(0);}
            else
            sayac=sayac+1;
}while (sayac<n);
System.out.println(aranan + " sayısı listede yoktur");
```

Sıralama Algoritmaları/ İkili Arama (Binary Search)

İkili arama algoritması, sayısal ya da string sıralı haldeki veriler üzerinde böl ve yönet tekniği ile çalışan bir algoritmadır.

İkili algoritmasının gerçekleştirilebilmesi için verilerin önceden, artan ya da azalan bir şekilde sıralanması gerekir.

İkili arama algoritmasında, sıralı dizi ortadan ikiye ayrılarak aranan veri bu alt dizilerde aranır. Arama işlemi alt dizilerde tekrarlanarak aranan veri bulunmaya çalışılır.

Sıralama Algoritmaları/ İkili Arama (Binary Search)

Java Dilinde Kodlanması:

```
public int ikiliarama (int dizi[], int aranan){
int alt=0;
int ust=dizi.length-1;
while (alt<=ust){
        int orta=(alt+ust)/2;
        if (dizi [orta]==aranan)
                return orta;
        else if (dizi[orta]<aranan)
                alt=orta+1;
        else
                ust=orta-1;
}
return -1;
}
```

Sıralama Algoritmaları/ İkili Arama (Binary Search)

Örnek: 0'dan 30'a kadar olan çift sayıların tutulduğu bir A dizi içerisinde, aranan sayının dizinin kaçıncı sırasında olduğunu bulan programı ikili arama algoritmasını kullanarak yazınız.

Çözüm:

```
class Ikiliara{
static int ara(int dizi[ ], int aranan){
int alt=0;
int ust=dizi.length-1;
while (alt<=ust){
        int orta=(alt+ust)/2;
        if (dizi [orta]==aranan)
                return orta;
        else if (dizi[orta]<aranan)
                alt=orta+1;
        else
                ust=orta-1;
}
return -1;}

public static void main(String[] args) {
int a[]= {0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30};
Scanner tara=new Scanner (System.in);
int i,x;
System.out.println("Aradığınız sayıyı giriniz");
x=tara.nextInt();
i=Ikiliara.ara(a,x);
if(i==-1)
        System.out.println("Aradığınız sayı dizide yoktur");
else
        System.out.println(x + " dizisinin " + i + " .elemanıdır.");
}}
```

Kaynaklar

- Dr Öğr. Üyesi Aysun Zehra ALTIKARDEŞ- Nesne Yönelimli Programlama Notları
- https://medium.com/@denizf.b/java-collections-arraylist-nedir-7519bc1b7654
- https://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/Collections/ClassArrayList01.pdf
- https://gelecegiyazanlar.turkcell.com.tr/konu/android/egitim/android-101/list
- https://merttopuz.com/yazilim/java/java-linked-lists
- https://www.muslu.net/2017/01/linkedlist-yapisi-javada-uygulanisi.html
- http://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/Collections/ClassLinkedList01.pdf