ArrayList Sınıfı

```
java.util
Class ArrayList
java.lang.Object
  └java.util.AbstractCollection
      └java.util.AbstractList
          └java.util.ArrayList
```

Java Collections Framework içinde yer alan bu sınıfın üç tane kurucusu vardır:

ArrayList()

Başlangıç olarak sığası 10 terim olan boş bir dizi oluşturur.

ArrayList(Collection c)

Parametrede belirtilen koleksiyona ait öğeler içeren bir liste oluşturur. Öğeler, *iterator*'un belirlediği sırayla dizilidirler.

ArrayList(int initialCapacity)

Sığası (capacity) parametrenin belirlediği sayıda olan bir dizi oluşturur.

ArrayList<E>

java.util.ArrayList<E> sınıfı kendiliğinden büyüyebilen dizi (*array*) kurar ve temelde *Collections Vector* sınıfı ile aynı sayılır. *ArrayList* sınıfının şu özelikleri vardır:

- Veri eklenip silindikçe ArrayList kendi uzunluğunu otomatik olarak ayarlar.
- ArrayList listesine erişim işlemi O(1), sokuşturma (insertion) işlemi O(n) ve silme (diletion) işlemi O(n) zaman karmaşasına sahiptir.
- ArrayList sokuşturma, silme ve arama eylemlerini yapan metotlara sahiptir.
- ArrayList üzerinde foreach döngüsü, iteratörler ve indexler yardımıyla gezinilebilir.

ArrayList ve array Arasında Seçim

Programcı, ne zaman *ArrayList* ve ne zaman *array* kullanması gerektiği konusunda ikileme düşebilir. Eğer, depoya konulacak öğe sayısı belirli ve o sayı sık sık değişmeyecekse array seçimi uygun olur. Ama öğe sayısı baştan bilinemiyor ya da o sayı sık sık değişiyorsa ArrayList doğru bir seçimdir. Tabii, buna ek olarak şunu söylemeliyiz: *ArrayList<E>* nesnelerin depolanması içindir. İlkel veri tipleri depolamak için *array* seçilmesi uygun olur. Bütün bunların ötesinde ArrayList sınıfı List arayüzünün metotlarını kullanma yeteneğine sahiptir; dolayısıyla array yapısına oranla programcıya daha çokkolaylık sağlar.

Başlıca ArrayList Metotları

Sun, generic tipler için *<E>* simgesinin kullanılmasını öneriyor. *E* simgesi koleksiyon içindeki öğelerin veri tipi yerine geçer. Aşağıdaki metotlarda şu bildirimlerin yapıldığını varsayacağız:

```
int i;
ArrayList<E> a;
E e;
Iterator<E> iter;
ListIterator<E> liter;
E[] earray;
Object[]
```

`void` **add**`(int index,` Object `element)`

Listede indisi belirtilen yere öğe sokuşturur (insert). O indisten sonraki öğelerin konumları birer geriye kayar.

`boolean` **add**`(`Object `o)`

Parametrede verilen nesneyi listenin sonuna ekler.

`boolean` **addAll**`(`Collection `c)`

Parametrede verilen koleksiyonun bütün öğelerini listenin sonuna ekler. Ekleme sırası koleksiyonun iteratörü'nün belirlediği sıradadır.

`boolean` **addAll**`(int index,` Collection `c)`

Belirtilen index'ten başlayarak verilen koleksiyonu listeye yerleştirir.

`void` **clear**`()`

Listedeki bütün öğeleri siler; boş liste haline getirir.

Object **clone**`()`

`ArrayList` kılgısının (instance) bir kopyasını yapar.

`boolean` **contains**`(`Object `o)`

Parametrede belirtilen nesne listede varsa `true` değerini alır.

`void` **ensureCapacity**`(int minCapacity)`

Gerekiyorsa `ArrayList` nesnesinin sığasını artırarak, parametrenin belirlediği minimum sığa kadar öğeyi depo edebilmesini sağlar.

Object **get**`(int index)`

İndeksi belirtilen öğeyi verir.

`int` **indexOf**`(`Object `o)`

Parametrede verilen nesnenin listedeki indeksini verir. Nesne listede yoksa -1 değerini verir.

`boolean` **isEmpty**`()`

Listenin boş olup olmadığını söyler.

`int` **lastIndexOf**`(`Object `elem)`

Belirtilen öğenin listedeki son indeksini söyler.

`Object` **remove**`(int index)`
İndeksi verilen öğeyi listeden siler.

`protected void` **removeRange**`(int fromIndex, int toIndex)`
Belirtilen indeksler arasında kalan öğeleri listeden siler.

`Object` **set**`(int index,` `Object` `element)`
Verilen nesneyi istenen indisli öğenin yerine koyar.

`int` **size**`()`
Listedeki öğe sayısını verir.

`Object[]` **toArray**`()`
Listedeki öğeleri, aynı sırayla bir array haline getirir.

`Object[]` **toArray**`(Object[] a)`
Listedeki öğeleri, aynı sırayla bir array haline getirir; koşma anındaki tipi belirtilen tip olur.

`void` **trimToSize**`()`
`ArrayList` nesnesinin sığasını, listedeki öğe sayısına indirir; yani boş terimleri siler.

Aşağıdaki örnekler *`ArrayList`* kullanımını gösteren örneklerdir.

Örnek 1:

Aşağıdaki program önce listeyi bir *`ArrayList`* olarak yaratıyor, listenin öğelerine veri atıyor ve listenin öğelerini yazdırıyor.

```
import java.util.*;

public class ArrayList01 {

 public static void main(String[] args) {
       List list = new ArrayList();
       list.add("İZMİR");
       list.add("ERZURUM");
       list.add("GİRESUN");
       list.add("KONYA");
       list.add("ANTALYA");
       System.out.println(list);
       System.out.println("3: " + list.get(3));
       System.out.println("0: " + list.get(0));
 }
}
     /*
     Çıktı:
     [İZMİR, ERZURUM, GİRESUN, KONYA, ANTALYA]
     3: KONYA
     0: İZMİR
     */
```

Listeye veri eklemek için

```
boolean  add( Obj )
```

metodu kullanılmaktadır. Bu metot bir koleksiyona *Obj* nesnesini (öğe) ekler ve `boolean` değer alır. Ekleme başarılı ise `true`, değilse `false` değerini verir.

Listenin öğelerine erişmek için

```
Object get(int index)
```

metodu kullanılmaktadır. Bu metot, listedeki indeksi `index` parametresi ile belirtilen terimin değerini verir.

Örnek 2:

```
import java.util.*;

public class ArrayList02 {

	public static void main(String[] args) {
		List a = new ArrayList();
		a.add("Bir");
		a.add("İki");
		a.add("Üç");
		a.add("Dört");
		a.add("Beş");
		a.add("Altı");
		System.out.println("ArrayList = " + a);
		}
	}
	    /*
	    Çıktı:
	      ArrayList = [Bir, İki, Üç, Dört, Beş, Altı]
	    */
```

Burada `System.out.println(a)` deyiminin listenin öğelerini sırayla yazdığına dikkat ediniz. Ayrı bir yazdırma metodu ya da döngü kullanılmadı.

Örnek 3:

```
import java.util.*;

public class ArrayList03 {
	public static void main(String[] args) {
		List a = new ArrayList();
		a.add("Bir");
		a.add("İki");
		a.add("Üç");
		a.add("Dört");
		a.add("Beş");
		a.add("Altı");
		System.out.println("ArrayList = " + a);

		List b = new ArrayList();
		b.addAll(a);
		System.out.println("Yeni dizi = " + b);
	}
}
	    /*
	    Çıktı:
	     a ArrayListi = [Bir, İki, Üç, Dört, Beş, Altı]
	     b ArrayListi = [Bir, İki, Üç, Dört, Beş, Altı]
	    */
```

Bu program `a` **adlı bir** *ArrayList* **yaratıyor,** `add()` **metodu ile onun terimlerine** `Bir, İki, Üç, Dört, Beş, Altı` öğelerini atıyor. Sonra `b` adlı bir *ArrayList* yaratıyor ve `addAll()` metodu ile `a` listesinin bütün öğelerini `b` ye atıyor. Görüldüğü gibi, `addAll()` metodu bir koleksiyondaki bütün öğeleri başka bir koleksiyona yerleştiriyor.

boolean addAll(Coll)

metodu bir koleksiyon içindeki bütün öğeleri başka bir koleksiyona ekler ve `boolean` değer alır. Ekleme başarılı ise `true`, değilse `false` değerini verir.

Örnek 4:

```
import java.util.*;

public class Koleksiyon {
      public static void main(String[] args) {
            List a = new ArrayList();
            a.add("Bir");
            a.add("İki");
            a.add("Üç");
            a.add("Dört");
            a.add("Beş");
            a.add("Altı");
            System.out.println(" a ArrayListi = " + a);

            System.out.println(a.isEmpty());
            System.out.println(a.contains("Altı"));

            a.remove(3);
            System.out.println(" a ArrayList = " + a);
            System.out.println(" a ArrayList = " + a.remove(1));

            a.clear();
            System.out.println(" a ArrayList = " + a);
      }
}
          /*
          Çıktı:
           a ArrayListi = [Bir, İki, Üç, Dört, Beş, Altı]
           false
           true
           a ArrayList = [Bir, İki, Üç, Beş, Altı]
           a ArrayList = İki
           a ArrayList = []
          */
```

Bu program `a` adlı bir *ArrayList* yaratıyor, `add()` metodu ile onun terimlerine `Bir, İki, Üç, Dört, Beş, Altı` öğelerini atıyor ve o terimleri yazdırıyor.

`a.isEmpty()` metodu, *false* değerini vererek listenin boş olmadığını söylüyor.

`a.contains("Altı")` metodu, *true* değerini vererek listede aranan "Altı" öğesinin listede olduğunu söylüyor.

`a.remove(3)` metodu, indeksi 3 olan "Dört" değerini listeden siliyor.

`a.remove(1)` metodunun değeri "İki" dir.

`a.clear()` metodu listenin bütün öğelerini silerek boş bir liste haline getiriyor.

Örnek 5:

```
package koleksiyon;

import java.util.*;

public class Koleksiyon {
	public static void main(String[] args) {
		List a = new ArrayList();
		a.add("Çiğdem");
		a.add("Papatya");
		a.add("Kardelen");
		a.add("Lale");
		a.add("Sümbül");
		a.add("Gül");
		System.out.println(" a ArrayListi = " + a);

		a.add(3, "Diken");
		System.out.println(" a ArrayListi = " + a);
		System.out.println(a.indexOf("Diken"));
		a.set(3, "Gelincik");
		System.out.println(" a ArrayListi = " + a);
		System.out.println(a.lastIndexOf("Kamelya"));
	}
}
		/*
		Çıktı:
		 a ArrayListi = [Çiğdem, Papatya, Kardelen, Lale, Sümbül, Gül]
		 a ArrayListi = [Çiğdem, Papatya, Kardelen, Diken, Lale, Sümbül, Gül]
		 3
		 a ArrayListi = [Çiğdem, Papatya, Kardelen, Gelincik, Lale, Sümbül, Gül]
		 -1
		*/
```

Bu program `a` adlı bir *ArrayList* yaratıyor, `add()` metodu ile onun terimlerine `Çiğdem, Papatya, Kardelen, Lale, Sümbül, Gül` öğelerini atıyor ve o terimleri yazdırıyor.

`a.add(3, "Diken")` metodu, indeksi 3 olan *"Lale"* öğesi yerine aynı index ile *"Diken"* öğesini sokuşturuyor (insertion). *"Lale"* ve sonrakiler birer geriye itiliyor.

`a.indexOf("Diken")` metodu, "Diken" nesnesinin indeksi olan 3 sayısını veriyor

`a.set(3, "Gelincik")` metodu, "Gelincik" nesnesini, indeksi 3 olan terimin yerine koyuyor. "Diken" siliniyor. Öteki öğeler yerlerinde kalıyor.

`a.lastIndexOf("Kamelya")` metodu, "Kamelya" nesnesinin (son) indeksini arıyor. Bulamadığı için -1 veriyor.

.

Örnek 6:

```
package koleksiyon;

import java.util.*;

public class Koleksiyon {
	public static void main(String[] args) {
		List a = new ArrayList();
		a.add("On");
		a.add("Yirmi");
		a.add("Otuz");
		a.add("Kırk");
		a.add("Elli");
		a.add("Altmış");
		System.out.println(" a ArrayListi = " + a);

		System.out.println(a.size());

		a.set(3, "Beşyüz");
		System.out.println(" a ArrayListi = " + a);
		System.out.println(a.size());

		a.add(3, "Altıyüz");
		System.out.println(" a ArrayListi = " + a);
		System.out.println(a.size());
	}
}
		/*
		Çıktı:
		 a ArrayListi = [On, Yirmi, Otuz, Kırk, Elli, Altmış]
		 6
		 a ArrayListi = [On, Yirmi, Otuz, Beşyüz, Elli, Altmış]
		 6
		 a ArrayListi = [On, Yirmi, Otuz, Altıyüz, Beşyüz, Elli, Altmış]
		 7
		*/
```

Bu program `a` adlı bir *ArrayList* yaratıyor, `add()` metodu ile onun terimlerine `On, Yirmi, Otuz, Kırk, Elli, Altmış` öğelerini atıyor ve o terimleri yazdırıyor.

`a.size()` metodu, listede 6 öğe olduğunu bildiriyor.

`a.set(3,"Beşyüz")` metodu, indeksi 3 olan "Kırk" öğesi yerine aynı index ile "Beşyüz" öğesini koyuyor. "Kırk" siliniyor.

Bu eylemde, listede indeksi 3 olan öğe değişiyor, ama liste uzunluğu değişmiyor; 6 olarak kalıyor.

`a.add(3, "Altıyüz")` metodu, "Altıyüz" nesnesini, indeksi 3 olan terimin yerine sokuşturuyor (insertion), öteki terimleri birer geriye itiyor. Listede hiçbir terim silinmiyor.

Bu eylemde, "Altıyüz" nesnesi listede indeksi 3 olan öğe oluyor; öteki terimler bire geriye itiliyor (insertion). Liste uzunluğu 1 artarak 7 oluyor.