ArrayList Uygulamaları

Örnek

Aşağıdaki program önce boş bir *ArrayList* yaratıyor. Sonra sırasıyla şu işleri yapıyor: Listenin boş olup olmadığını yaz. Ambara üç tane öğe ekle. İndisi 2 olan yere bir öğe sokuştur. Listenin uzunluğunu yaz. Listeyi yazdır. Listede bir öğe ara. Listeden bir öğe sil. Listenin uzunluğunu yaz. Listenin ilk öğesini yaz. Listeyi boşalt. Listenin uzunluğunu yaz.

```
import java.util.*;

public class ArrayListDemo {
      public static void main(String[] args) {
            ArrayList list = new ArrayList();
            boolean empty = list.isEmpty();
            System.out.println("Boş mu? " + empty);

            list.add("Yüz");
            list.add("İkiyüz");
            list.add("Dörtyüz");
            list.add(2, "Üçyüz");
            System.out.println("Uzunluk: " + list.size());
            System.out.println(list);

            boolean aranan = list.contains("Dörtyüz");
            System.out.println("Bulunan: " + aranan);

            boolean silinen = list.remove("Dörtyüz");
            System.out.println("Silinen: " + silinen);
            System.out.println("Uzunluk: " + list.size());

            Object element = list.get(0);
            String str = (String) element;
            System.out.println("Index 0 : " + str);

            list.clear();
            System.out.println("Uzunluk: " + list.size());
       }
}
     /*
     Boş mu? true
     Uzunluk: 4
     [Yüz, İkiyüz, Üçyüz, Dörtyüz]
     Bulunan: true
     Silinen: true
     Uzunluk: 3
     Index 0 : Yüz
     Uzunluk: 0
     */
```

Örnek

Aşağıdaki program önce boş bir *ArrayList* ambarı ile boş bir *Vector* ambarı yaratıyor. Sonra `addAll(Collection c)` metodu ile `Vector` ambarındaki öğeleri `ArrayList` ambarına ekliyor. Eklenen öğeler `ArrayList`'in sonuna yerleşiyor. *Vector* ambarının boşalmadığına dikkat ediniz.

```
import java.util.ArrayList;
import java.util.Vector;

public class ArrayListDemo {

      public static void main(String[] args) {

// bir ArrayList nesnesi (ambarı)yarat
            ArrayList arrayList = new ArrayList();

// Arraylist ambarına öğeler ekle
            arrayList.add("a");
            arrayList.add("b");
            arrayList.add("c");
            System.out.println("arrayList : " + arrayList);
// bir Vector nesnesi yarat
            Vector v = new Vector();
            v.add(10);
            v.add(20);
            System.out.println("Vector    : " + v);
/*
 * ArrayList ambarına başka bir ambardaki koleksiyonu eklemek için
 * addAll(Collection c) metodu kullanılır
*/

// Vector ambarındaki bütün öğeleri ArrayList ambarına ekle
            arrayList.addAll(v);

// ArrayList ambarındaki öğeleri yaz
            System.out.println("Eklendikten sonra...");
            System.out.println("arrayList : " + arrayList);

      }
}
     /*
     arrayList : [a, b, c]
     Vector    : [10, 20]
     Eklendikten sonra...
     arrayList : [a, b, c, 10, 20
     */
```

Örnek

Aşağıdaki örnek bir *ArrayList* yapısının öğelerini *Array* yapısına kopyalamak için *toArray()* metodu kullanıyor.

```
import java.util.*;

public class ArrayListDemo {

	public static void main(String[] args) {
// bir ArrayList nesnesi yarat
		ArrayList arrayList = new ArrayList();

// Add elements to ArrayList
		arrayList.add("a");
		arrayList.add("b");
		arrayList.add("c");
		arrayList.add("d");
		arrayList.add("e");
		System.out.println("arrayList'in öğeleri..");
		System.out.print(arrayList);
/*
 * bir Array nesnesi yarat ve arrayList'in öğelerini Array'e kopyala
*/
		Object[] objArray = arrayList.toArray();

		System.out.println("\nArray'in öğeleri ... ");
		for (int index = 0; index < objArray.length; index++)
			System.out.print(objArray[index] + "   ");
		System.out.println("\narrayList'in öğeleri..");
		System.out.print(arrayList);
	}
}
	/*
	arrayList'in öğeleri..
	[a, b, c, d, e]
	Array'in öğeleri ...
	a   b   c   d   e
	arrayList'in öğeleri..
	[a, b, c, d, e]
	*/
```

Örnek

Aşağıdaki program bir ArrayList'in uzunluğunu buluyor ve for döngüsü ile öğelerini yazdırıyor.

```
import java.util.ArrayList;

public class ArrayListDemo {

	public static void main(String[] args) {
		// create an ArrayList object
		ArrayList arrayList = new ArrayList();

		// Arraylist ambarına öğeler ekle
		arrayList.add("Ankara");
		arrayList.add(50);
		arrayList.add("h");

		// size() metodu ArrayList'in uzunluğunu verir
		int totalElements = arrayList.size();

		System.out.println("ArrayList'in öğeleri...");
		// for döngüsü ile öğeleri yaz
```

```
            for (int index = 0; index < totalElements; index++)
                  System.out.println(arrayList.get(index));
      }

}
```

```
/*
ArrayList'in öğeleri...
Ankara
50
h
*/
```

ArrayList'in bir alt listesini oluşturmak için

```
List subList(int ilkIndex, int sonIndex)
```

metodu kullanılır. Bu metot, orijinal listede *ilkIndex* ile *sonIndex* arasındaki bütün öğeleri seçer ve onlardan oluşan yeni bir *List* yaratır. Alt liste orijinal listeye dayalıdır. O nedenle, alt listede yapılacak her değişiklik, listeye aynen yansır.

Uyarı: Bir *lst* referansının (pointer) işaret ettiği bir *List* nesnesinin *i* indisli öğesine erişmek için *lst.get(i)* metodu kullanılır. Bu durumda, aşağıdaki for döngüsü *list* 'in öğeleri sırayla yazar.

```
for(int i=0; i< lst.size() ; i++)
System.out.println(lst.get(i));
```

Eğer, *list'*in öğelerini bir küme olarak yazdırmak istiyorsak,

```
System.out.println(list);
```

deyimini kullanırız.

Örnek

Aşağıdaki örnek bir *ArrayList* yaratıyor. Bir alt listesini oluşturuyor. Alt listeden bir öğe siliyor. Alt listeden silinen öğenin üst listeden de silindiğini gösteriyor.

```
import java.util.ArrayList;
import java.util.List;

public class ArrayListDemo {

      public static void main(String[] args) {

 // bir ArrayList nesnesi (ambarı) yarat
            ArrayList arrayList = new ArrayList();

 // Arraylist ambarına öğeler koy
            arrayList.add("1");
            arrayList.add("2");
            arrayList.add("3");
            arrayList.add("4");
            arrayList.add("5");
```

```
            System.out.println("ArrayList'in öğeleri...");
            System.out.println(arrayList);

 // Alt liste yarat
            List lst = arrayList.subList(1, 3);

            System.out.println("Alt listenin öğeleri ... : ");
            System.out.println(lst);
/*
 * Alt listede yapılacak değişiklik üst listeye de yansır Alt listeden
 * bir öğe sil Onun üst listeden de silinir
*/
            Object obj = lst.remove(0);
            System.out.println(obj + " öğesi alt listeden silindi...");

 // orijinal ArrayList
            System.out.println(obj + " öğesi alt listeden silindikten ");
            System.out.println("  sonra orijinal ArrayList... ");
            System.out.println(arrayList);
      }
     }
     /*
     ArrayList'in öğeleri...
     [1, 2, 3, 4, 5]
     Alt listenin öğeleri ... :
     [2, 3]
     2 öğesi alt listeden silindi...
     2 öğesi alt listeden silindikten sonra orijinal ArrayList...
     [1, 3, 4, 5]
     */
```

Bir listenin ndx numaralı indisinden başlayarak, listeye başka bir c koleksiyonunun öğelerini sokuşturmak (insertion) için

```
boolean addAll(int ndx, Collection c)
```

metodu kullanılır.

Örnek

Aşağıdaki program bir ArrayList yaratıyor. 1 indisli öğesinden başlayarak listeye bir Vector'un öğelerini sokuşturuyor. Bu işlem Vector'un öğelerini taşımaz, yalnızca kopyalar.

```
import java.util.ArrayList;
import java.util.Vector;

public class ArrayListDemo {

      public static void main(String[] args) {

            // bir ArrayList nesnesi yarat
            ArrayList arrayList = new ArrayList();

            // Arraylist ambarına öğeler ekle
            arrayList.add("1");
            arrayList.add("2");
            arrayList.add("3");
            System.out.println(arrayList);
```

```
            // bir Vector nesnesi yarat ve öğeler ekle
            Vector v = new Vector();
            v.add("4");
            v.add("5");
/*
 * Vector'ün öğelerini ArrayList'e 1 nolu indisten başlayarak ekle
*/
            arrayList.addAll(1, v);

            // ArrayList
 System.out.println("ArrayList'in sokuşturmadan sonraki öğeleri ...");
 System.out.println(arrayList);
      }
}

     /*
     [1, 2, 3]
     ArrayList'in sokuşturmadan sonraki öğeleri ...
     [1, 4, 5, 2, 3]
     */
```

iterator() Metodu

lst referansının işaret ettiği bir *ArrayList*'in öğelerini baştan sona taramak (öğelerine erişmek) için, *ArrayList* üzerinde gezinecek bir *iterator* (tekrarlayıcı) nesnesi yaratılır. Bunu yapmak için Iterator sınıfındaki *iterator()* metodu kullanılır. Bu metot, listeyi baştan sona doğru tarayan bir *iterator* (tekrarlayıcı) yaratır. Sözdizimi şöyledir:

```
Iterator itr = arrayList.iterator();
```

deyimi kullanılır.

Örnek

Aşağıdaki program bir ArrayList'in öğelerini baştan sona doğru tarayan bir iterator yaratıyor.

```
import java.util.ArrayList;
import java.util.Iterator;

public class ArrayListDemo {

      public static void main(String[] args) {

            // ArrayList nesnesi yarat
            ArrayList arrayList = new ArrayList();

            // Add elements to Arraylist
            arrayList.add("1");
            arrayList.add("2");
            arrayList.add("3");
            arrayList.add("4");
            arrayList.add("5");
/*
 * iterator() metodu kullanarak, ArrayList üzerinde gezinecek bir
 * Iterator nesnesi yarat.
*/
            Iterator itr = arrayList.iterator();

// hasNext() ve next() metotlarını kullanarak listeyi tara
            System.out.println("ArrayList öğelerini tarıyor...");
```

```
            while (itr.hasNext())
                  System.out.println(itr.next());

      }
}

     /*
     ArrayList öğelerini tarıyor...
     1
     2
     3
     4
     5
     */
```

ListIterator() Metodu

lst referansının işaret ettiği bir *ArrayList*'in öğelerini baştan sona doğru taramak (öğelerine erişmek) için, *ArrayList* üzerinde gezinecek bir *iterator* (tekrarlayıcı) nesnesinin *iterator()* metodu ile nasıl yaratıldığını önceki örnekte görmüştük. Eğer listeyi iki yönlü tarayan bir iterator yaratmak istiyorsak, nesnesinin *iterator()* metodu yerine *ListIterator()* metodunu kullanırız. Bu metot, listeyi baştan sona ya da sondan başa doğru taramamızı sağlar. Baştan sona doğru taramak için, gene hasNext() ve next() metotlarını kullanırız. Sondan başa doğru taramak için ise hasPrevious() ve previous() metotlarını kullanırız. Sözdizimi şöyledir:

```
Iterator itr = arrayList.ListIterator();
```

deyimi kullanılır.

Örnek

Aşağıdaki program bir ArrayList'in öğelerini baştan sona doğru tarayan bir iterator yaratıyor.

```
import java.util.ArrayList;
import java.util.ListIterator;

public class ArrayListDemo {

      public static void main(String[] args) {

            // ArrayList nesnesi yarat
            ArrayList arrayList = new ArrayList();

            // Arraylist ambarına öğe ekle
            arrayList.add("1");
            arrayList.add("2");
            arrayList.add("3");
            arrayList.add("4");
            arrayList.add("5");
/*
 * ListIterator() metodunu kullanarak ArrayList üzerinde gezinen bir
 * ListIterator nesnesi yarat
*/
            ListIterator itr = arrayList.listIterator();
/*
 * Listeyi baştan sona doğru taramak için hasNext() ve next()
 * metotlarını kullan
*/
```

```
            System.out.println("Artan yönde ArrayList'in öğeleri...");
            while (itr.hasNext())
                  System.out.println(itr.next());
/*
 * Listeyi sondan başa doğru taramak için hasPrevious() ve previous()
 * metotlarını kullan
*/
            System.out.println("Azalan yönde ArrayList'in öğeleri...");
            while (itr.hasPrevious())
                  System.out.println(itr.previous());

      }
}
```

```
/*
Artan yönde ArrayList'in öğeleri...
1
2
3
4
5
Azalan yönde ArrayList'in öğeleri...
5
4
3
2
1
*/
```

clear() *metodu*

Örnek

Aşağıdaki program bir *ArrayList* ambarı yaratıyor, içine öğeler koyuyor. Sonra *clear()* metodu ile bütün öğelerini siliyor.

```
import java.util.ArrayList;

public class ArrayListDemo {

      public static void main(String[] args) {
            // bir ArrayList nesnesi yarat
            ArrayList arrayList = new ArrayList();

            // Arraylist ambarına öğeler ekle
            arrayList.add("1");
            arrayList.add("2");
            arrayList.add("3");

System.out.println("ArrayList'in uzunluğu : " + arrayList.size());

System.out.println("ArrayList'in öğelerini sil... ");
            arrayList.clear();
System.out.println("ArrayList'in uzunluğu : " + arrayList.size());

      }
}
```

```
/*
ArrayList'in uzunluğu : 3
```

```
ArrayList'in öğelerini sil...
ArrayList'in uzunluğu : 0
*/
```

Listeden, indisi ndx olan öğeyi silmek için $remove(int\ ndx)$ metodu kullanılır.

Örnek

```
import java.util.ArrayList;

public class ArrayListDemo {

	public static void main(String[] args) {
		// bir ArrayList nesnesi yarat
		ArrayList arrayList = new ArrayList();

		// Arraylist ambarına öğeler eklee
		arrayList.add("1");
		arrayList.add("2");
		arrayList.add("3");
		System.out.println("ArrayList'in öğeleri...");
		System.out.println(arrayList);
/*
 * ArrayList ambarından indisi ndx olan öğeyi atmak için
 * remove(int  ndx) metodu kullanılır
*/
		Object obj = arrayList.remove(1);
		System.out.println(obj + " öğesi atıldı");

		System.out.println("ArrayList'in öğeleri...");
		System.out.println(arrayList);

	}
}
```

```
/*
ArrayList'in öğeleri...
[1, 2, 3]
2 öğesi atıldı
ArrayList'in öğeleri...
[1, 3]
*/
```

$ArrayList$'in bir öğesi yerine başkasını koymak (replace) için

```
Object set(int ndx, Object obj
```

metodu kullanılır. Bu metot, indisi ndx olan öğeyi siler onun yerine obj öğesini koyar.

Örnek

```
import java.util.ArrayList;

public class ArrayListDemo {

	public static void main(String[] args) {
		// bir ArrayList nesnesi yarat
		ArrayList arrayList = new ArrayList();

		// Arraylist ambarına öğeler ekle
		arrayList.add("1");
```

```
            arrayList.add("2");
            arrayList.add("3");
            System.out.println("ArrayList'in öğeleri...");
            System.out.println(arrayList);
System.out.println("indisi 1 olan öğe yerine \"ANKARA\" nesnesini koy");
            arrayList.set(1, "ANKARA");
            System.out.println("ArrayList'in yeni öğeleri...");
            System.out.println(arrayList);
      }
}

     /*
     ArrayList'in öğeleri...
     [1, 2, 3]
     indisi 1 olan öğe yerine "ANKARA" nesnesini koy
     ArrayList'in yeni öğeleri...
     [1, ANKARA, 3]
     */
```

ArrayList ambarında bir nesnenin olup olmadığını anlamak için

```
Boolean contains(Object elem)
```

metodu kullanılır. `elem` nesnesi ambarda ise `contains()` metodu `true`, değilse `false` değer verir. *ArrayList* ambarında bir öğe birden çok kez yer alabilir. Ambarda yer aldığı yerin ilk (en küçük) indisini

```
int indexOf(Object elem)
```

metodu verir. `elem` nesnesi ambarda ise, `indexOf()` metodu `elem` nesnesinin ilk indisini verir. `elem` nesnesi ambarda değilse, `indexOf()` metodu -1 değerini verir.

elem ambarda birden çok kez yer alıyorsa, o yerlerin son (en büyük) indisini

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

metodu verir.

Örnek

```
import java.util.ArrayList;

public class ArrayListExample {

      public static void main(String[] args) {

            // bir ArrayList nesnesi yarat
            ArrayList arrayList = new ArrayList();

            // Arraylist ambarına öğeler ekle
            arrayList.add("1");
            arrayList.add("2");
            arrayList.add("3");
            arrayList.add("4");
```

```
            arrayList.add("5");
            arrayList.add("1");
            arrayList.add("2");

            /*
             * contains(Object elem) metodu ile elem nesnesinin ambarda
olup
             * olmadığını denetle
             */

            boolean bul = arrayList.contains("2");
            System.out.println("arrayList ambarında 2 var mı? " + bul);

            int index = arrayList.indexOf("4");
            if (index == -1)
                  System.out.println("ArrayList ambarında 4  yoktur.");
            else
                  System.out.println("ArrayList ambarındaki 4 nesnesinin
indisi :"
                              + index);
            /*
             * lastIndexOf() metodu ile ambardaki nesnenin son indisini bul
             */
            int lastIndex = arrayList.lastIndexOf("1");
            if (lastIndex == -1)
                  System.out.println("ArrayList ambarında 1  yoktur");
            else
                  System.out.println("Ambarda 1 in yer aldığı son indis  :"
                              + lastIndex);
      }
}
      /*
      arrayList ambarında 2 var mı? true
      ArrayList ambarındaki 4 nesnesinin indisi :3
      Ambarda 1 in yer aldığı son indis  :5
      */
```

Bir *ArrayList* ambarına öğe koymak için

```
boolean add(Object obj)
```

metodu kullanılır. Burada *obj* ambara konulacak nesnedir. Eğer işlem gerçekleşirse, obj listenin sonuna eklenir ve add() metodu true değeri verir. İşlem her hangi bir nedenle gerçekleşmezse add() metodu false değeri verir.

Bir *ArrayList* ambarından indisi ndx olan öğeye erişmek için

```
Object get(int ndx)
```

metodu kullanılır. Burada *ndx* ambarda erişmek istediğimiz öğenin indisidir. Eğer işlem gerçekleşirse, get() metodu, indisi *ndx* olan nesneyi değer olarak verir. Bu öğe ambardan atılmaz; metot yalnızca o öğeye eriimi sağlar. Ambardaki nesnelerden hiç birisine ait olmayan bir indis yazılırsa *get()* metodu `java.lang.IndexOutOfBoundsException` hata iletisini atar.

Örnek

```
import java.util.ArrayList;

public class ArrayListExample {

	public static void main(String[] args) {

		// bir ArrayList nesnesi yarat ve öğeler ekle
		ArrayList arrayList = new ArrayList();

		arrayList.add("Rize");
		arrayList.add("Trabzon");
		arrayList.add("Samsun");

		/*
		 * Object get(int index) metodu ile ndx indisli öğeye eriş
		*/
		System.out.println("ArrayList'in öğeleri");
		System.out.println(arrayList.get(0));
		System.out.println(arrayList.get(1));
		System.out.println(arrayList.get(2));
		// System.out.println(arrayList.get(4)); // Hata uyarısı verir
	}
}
```

```
/*
ArrayList'in öğeleri
Rize
Trabzon
Samsun
*/
```

ArrayList ambarındaki öğeleri sıralamak için

```
Collections.sort(arrayList);
```

metodunu kullanabiliriz. Bu metot ambardaki öğeleri doğal sırasına dizer. Doğal sıralamadan farklı bir sıralama isteniyorsa, comparator kullanılır.

Örnek

```
import java.util.ArrayList;
import java.util.Collections;

public class ArrayListExample {
	public static void main(String[] args) {

		// bir ArrayList nesnesi yarat
		ArrayList arrayList = new ArrayList();

		// Arraylist ambarına öğeler koy
		arrayList.add("1");
		arrayList.add("3");
		arrayList.add("5");
		arrayList.add("2");
		arrayList.add("4");
```

```
            System.out.println("Sırasız liste...");
            System.out.println(arrayList);
            // Ambardaki öğeleri sırala
            Collections.sort(arrayList);

            // ArrayList ambarındaki öğeleri yaz
            System.out.println("Sıralı liste...");
            System.out.println(arrayList);
        }
}
    /*
    Sırasız liste...
    [1, 3, 5, 2, 4]
    Sıralı liste...
    [1, 2, 3, 4, 5]
    */
```