Class Vector

java.util
Class Vector

```
java.lang.Object
  └java.util.AbstractCollection
      └java.util.AbstractList
          └java.util.Vector
```

Kılgıladığı arayüzler:

Cloneable, Collection, List, RandomAccess, Serializable

Altsınıfları

Stack

Bildirimi:

public class **Vector**
extends AbstractList
implements List, RandomAccess, Cloneable, Serializable

Vector sınıfı Java Collections Framework' un bir üyesidir.

Klasik programlama eyleminde *array (dizi)* çok önemli bir rol oynar. Ancak, *array*'in uzunluğu; yani bileşenlerinin sayısı array bildiriminde belirleniyor ve bu uzunluk daha sonra değiştirilemiyordu. Bazı uygulamalarda, bu özelik ciddi bir handikap oluşturur. ***Java 2***, bu sorunu çözmek için ***Vector*** sınıfını ve benzer işi yapan *ArrayList* sınıfını ortaya koydu. Her iki sınıfta, diziye yeni öğeler eklenir ya da varolan öğeler silinirse, dizinin uzunluğu kendiliğinden değişir. Tabii, bu değişimin bellek kullanımı ve zaman açısından bir bedeli (karmaşa – complexity) vardır. Ama, java programcısı bu işleri yapan yordamları kendisi yazmak zorunda değildir. *Vector* ya da *ArrayList* sınıfına ait nesneler bu işleri kendiliğinden yaparlar.

Vector tipinden dizilerin öğeleri, aynen array tipinde olduğu gibi, birer indise sahiptir. Dolayısıyla , vector'un öğelerine indisleri ile doğrudan erişim sağlanır.

ArrayList senkronize olmadığı için *Vector* sınıfına göre daha hızlıdır. Elbette hızlı olmanın bir bedeli vardır. Senkronize olmaması demek, çoklu işlem ortamında güvenirliğin azalması demektir.

Java 2 sürümünde, *Vector* sınıfına *ArrayList* sınıfının bir çok metodu eklendi. Böylece, *Vector* sınıfı bir çok uygulamada *Array* sınıfının yerini aldı.

Vector sınıfı yalnızca nesneler içerebilir; ilkel veri tiplerini içeremez. İlkel veri tipleri kullanılacaksa, onların ilgil nesne içine gömülmesi gerekir. Örneğin `int` tipinden `Vector` sınıfı tanımlanamaz, ama `Integer` sınıfından tanımlanabilir. `int` veri tipi, `Integer` sınıfı tarafından sarmalandığı (wrapping) için, sonunda `int` tipi içeren `Vector` sınıfı kullanılabilir.

İlgili konular:

`Collection`, `List`, `ArrayList`, `LinkedList`, Serialized Form

Vector Sınıfının Nitelemleri	
protected int	**capacityIncrement** `Vector`'un sığası dolduğunda otomatik olarak sığasının ne kadar artacağını belirten sayı.
protected int	**elementCount** `Vector` nesnesinde varolan öğe sayısı.
protected Object[]	**elementData** `Vector`'un öğelerinin konulduğu buffer dizisi.

java.util.AbstractList sınıfından kalıtsal alınan nitelem
modCount

Vector Sınıfının Kurucuları
Vector() Sığası 10 ve sığa artış sayısı 0 olan boş bir vector yaratır.
Vector(Collection c) Verilen koleksiyonu öğeleri olarak kabul eden bir vector yaratır. Öğelerin sırasını koleksonun *iterator*'u belirler.
Vector(int initialCapacity) Sığası verilen sayıya eşit ve sığa artış sayısı 0 olan boş bir vector yaratır.
Vector(int initialCapacity, int capacityIncrement) Sığası ve sığa artış sayısı verilenlere eşit olan boş bir vector yaratır.

Vector Sınıfının Metotları	
void	**add**(int index, Object element) Verilen nesneyi `vector`'de indisi ile belirlenen konuma yerleştirir.
boolean	**add**(Object o) Verilen nesneyi `vector`'ün sonuna koyar.
boolean	**addAll**(Collection c) Verilen koleksiyonun öğelerini bir vector yapısına dönüştürür. Öğelerin sırası, koleksiyonun iterator'u tarafından belirlenir.
boolean	**addAll**(int index, Collection c) Verilen koleksiyonun öğelerini, vector'un belirlenen indisinden başlayarak araya konuşlandırır.
void	**addElement**(Object obj) `Vector`'un sığasını 1 sayı artırarak, verilen nesneyi `vector`'ün sonuna koyar.
int	**capacity**() `Vector`'un sığasını verir; öğe atanmamış indisler dahil.
void	**clear**() `Vector`'un bütün öğelerini siler, onu boş bir vector yapar.

Object	**clone** () `Vector`'un bir kopyasını verir.
boolean	**contains** (Object `elem`) Verilen öğenin `Vector`'ün içinde olup olmadığını belirler.
boolean	**containsAll** (Collection `c`) Verilen koleksiyonun `Vector`'ün içinde olup olmadığını belirler.
void	**copyInto** (Object`[] anArray`) `Vector`'ün öğelerini belirlenen array'e kopyalar..
Object	**elementAt** (`int index`) `Vector`'ün verilen indisteki öğesini verir.
Enumeration	**elements** () `Vector`'ün öğelerini sıralı olarak verir.
void	**ensureCapacity** (`int minCapacity`) `Vector`'ün sığasını, gerekli ise, verilen sayı kadar artırır.
boolean	**equals** (Object `o`) Verilen nesneyi `Vector` ile mukayese eder; eşit olup olmadığını belirler.
Object	**firstElement** () `Vector`'ün ilk öğesini (indisi 0 olan) verir.
Object	**get** (`int index`) `Vector`'ün verilen indisteki öğesini verir.
int	**hashCode** () `Vector`'ün hash kodunu verir.
int	**indexOf** (Object `elem`) Verilen nesnenin vector içinde konuşlandığı ilk indisi verir. Bunu yaparken `equals` metodunu kullanır.
int	**indexOf** (Object `elem, int index`) Verilen indisten başlamak koşuluyla, verilen nesnenin vector içinde konuşlandığı ilk indisi verir. Bunu yaparken `equals` metodunu kullanır.
void	**insertElementAt** (Object `obj, int index`) Verilen öğeyi, verilen indisle belirlenen konuma yerleştirir.
boolean	**isEmpty** () `Vector`'ün boş olup olmadığını belirler.
Object	**lastElement** () `Vector`'ün son öğesini verir.
int	**lastIndexOf** (Object `elem`) Verilen nesnenin, `vector` içinde konuşlandığı son indisi verir.
int	**lastIndexOf** (Object `elem, int index`) Ters sırada aramak koşuluyla, verilen nesnenin, `vector` içinde konuşlandığı indisi verir.
Object	**remove** (`int index`) İndisi verilen öğeyi `Vector`'den siler.
boolean	**remove** (Object `o`)

	Verilen öğeyi, ilk karşılaştığı konumundan siler. Öğe vector içinde değilse, bir değişiklik yapmaz.
boolean	**removeAll** (Collection c) Verilen koleksiyonun bütün öğelerini vector'dan siler.
void	**removeAllElements** () Vector'ün bütün öğelerini siler ve uzunluğunu 0 yapar.
boolean	**removeElement** (Object obj) Verilen öğeyi, ilk karşılaştığı konumundan siler.
void	**removeElementAt** (int index) Verilen indisteki öğeyi siler.
protected void	**removeRange** (int fromIndex, int toIndex) Verilen indisler arasında kalan bütün öğeleri siler; uçlar da silinir.
boolean	**retainAll** (Collection c) Verilen koleksiyona ait olanların dışındaki bütün öğeleri siler.
Object	**set** (int index, Object element) Verilen öğeyi belirtilen indisteki konuma yerleştirir.
void	**setElementAt** (Object obj, int index) Verilen öğeye belirtilen indisi atar.
void	**setSize** (int newSize) Vector'e yeni sığa (uzunluk) atar.
int	**size** () Vector'ün öğe atanmış bileşen sayısını (uzunluğunu) verir.
List	**subList** (int fromIndex, int toIndex) Verilen indisler arasında kalan öğeleri verir; listeye uç noktalar da dahildir.
Object[]	**toArray** () Vector'ü, sırayı koruyarak, bir array yapısına dönüştürür.
Object[]	**toArray** (Object[] a) Vector'ü, sırayı koruyarak, verilen arraye dönüştürür.
String	**toString** () Vector'ü String temsiline dönüştürür.
void	**trimToSize** () Vector'un sığasını mevcut uzunluğuna indirger.

java.util.AbstractList sınıfından kalıtsal alınan metotlar
iterator, listIterator, listIterator

java.lang.Object sınıfından kalıtsal alınan metotlar
finalize, getClass, notify, notifyAll, wait, wait, wait

java.util.List arayüzünden alınan metotlar
iterator, listIterator, listIterator

VECTOR YARATMA

Programda `Vector` nesnesi yaratmak için ya `java.util.Vector` paketi ya da `java.util.*` paketi çağrılmalıdır. Bunu yapmak için, programın başına

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

ya da

```
import java.util.*;.
```

yazılmalıdır. Sonra aşağıdaki kuruculardan istenen birisiyle vektör sınıfı yaratılır.

Vector Kurucuları (constructors)

```
Vector v = new  Vector();
```

deyimi uzunluğu 10 olan boş bir vektör yaratır. Uzunluk belirtilmediği zaman, vektör sınıfının öntanımlı (default) uzunluğu daima 10 bileşendir.

```
Vector v = new  Vector(Collection c);
```

deyimi belirtilen koleksiyonun öğelerini içeren bir vektör yaratır.

```
Vector v = new  Vector(int  başlangıçKapasitesi);
```

deyimi, başlangıç kapasitesi belirlenen boş bir vektör yaratır. Standart kapasite artışı 0 dır.

```
Vector v =
 new  Vector(int  başlangıçKapasite,  int  kapasiteArtışı);
```

deyimi, başlangıç kapasitesi belirlenen boş bir vektör yaratır. Kapasite artışı kapasiteArtışı ile verilen int değerdir.

```
Vector v = new Vector();
```

deyimi kullanılır. Bu deyim 10 bileşenli bir vektör yaratır; yani vektörün öntanımlı uzunluğu 10 dur. Eğer uzunluğu kendimiz belirlemek istiyorsak, örneğin 215 bileşenli bir vektör için

```
Vector v = new Vector(215);
```

deyimini yazarız.

VEKTÖRÜN UZUNLUĞUNU DEĞIŞTIRME

`Vector`, özünde bir arraydir; arrayden öneli farkı uzunluğunundeğiştirilebilir oluşudur. Vector'ün bileşenlerinin hepsine değer atanır ve yeni bileşen ekleme gereksemesi ortaya çıkarsa, teknik olarak yapılan iş şudur:

Mevcut vektörün bileşen sayısından bir fazla bileşeni olan bir Vector yaratılır. Eski Vector'ün bileşenleri, sırasıyla, yeni vektörün bileşenlerine aktarılır. En sona eklenen yeni bileşene yeni değer girilir. Tabii, bu işlem biraz zaman alıcıdır. O nedenle, gerekseme duyulacak bileşen sayısı başlangıçta biliniyorsa, vektör yerine array tercih edilmelidir.

Tabii, vektörün bileşenlerine ekleme yapılabildiği gibi, istenen bileşen silinebilir, bileşenlerin arasına yeni bileşen yerleştirilebilir.

VECTOR'E YENI BILEŞEN EKLEME

Vektöre yeni bileşen eklemek için

```
v.add(s); // v vektörünün sonuna yeni s bileşenini ekler
```

deyimi yazılır.

VECTOR'ÜN ÖĞELERINI LISTELEME

Bir vektörün bütün bileşenlerini listelemek için, elbette for döngüsü kullanılabilir. Ama vektör sınıfı daha kolay bir yöntem sunar:

LISTITERATOR

AŞAĞIDAKİ KOD PARÇASI LİSTİTERATÖRÜN KULLANILIŞINI GÖSTERMEKTEDİR.

```
ListIterator iter = v.listIterator();
while (iter.hasNext()) {
    System.out.println((String)iter.next());
}
```

VECTOR SINIFININ METOTLARINA ÖRNEKLER

Aşağıdaki tabloda `v` bir vektör, `o` bir nesne (object), `i` ise `int` tipinden bir damgadır (index).

Metot	*Açıklama*
`v.add(o)`	v vektörüne o nesnesini ekler
`v.add(i, o)`	i damgalı bileşeni bir sonraya iter, araya `o` nesnesini yerleştirir
`v.clear()`	v vektörünün bütün bileşenlerini siler
`v.contains(o)`	v vektörü `o` nesnesini içeriyorsa true, değilse false değerini alır
`v.firstElement(i)`	v vektörünün ilk bileşenini verir
`v.get(i)`	v vektörünün i-damgalı bileşenini verir
`v.lastElement(i)`	v vektörünün son bileşenini verir
`v.listIterator()`	v vektörünün bütün bileşenleri üzerinde sırayla geçer (döngü yapar)
`v.remove(i)`	v vektörünün i-damgalı bileşenini siler
`v.set(i,o)`	v vektörünün i-damgasını o objesine takar
`v.size()`	v vektörünün mevcut bileşen sayısını (uzunluğunu) verir
`v.toArray(Object[])`	v vektörünü nesne arrayine dönüştürür

Örnek:

Java Collections Framework içindeki her yapıya (ambar) konulan öğeler birer nesne olmak zorundadır. *İlkel veri tipleri* yapı içine konulamaz. Onları önce ait oldukları sınıflara gömdükten sonra yapı içine koyabiliriz. Bir yapıya her hangi bir sınıfa ait nesneler konulabilir; yani aynı ayapıda farklı tipten nesneler yer alabilir. Yapıya belirli sınıftan nesneler koymak istiyorsak, onu ***<E>*** simgesiyle belirtiriz. Burada ***E*** nesnelerin ait olacağı sınıfın adıdır. Aşağıdaki örnekte `Vector<String>` ifadesi, yaratılan vektöre ancak `String` sınısına ait nesnelerin konulabileceğini belirtir. Tabii, `String` yerine istenen başka bir sınıf alınabilir.

Array için olduğu gibi, vector'un bileşenlerinin indisleri ***0*** dan başlar.

Aşağıdaki örnek, boş bir vektör nesnesi yaratıyor, 0-ıncı damgadan başlayıp 4-üncü damgaya kadar bileşenlerine beş tane `String` tipi nesne ekliyor. Sonra 3-damgalı bileşene yeni bir String nesnesi yerleştiriyor (araya giriyor, insertion). Bu işlem 3 ve 4 numaralı bileşenleri birer damga yukarıya kaydırıyor. Dolayısıyla vektörün bileşen sayısı 6 oluyor.

Vector01.java

```
import java.util.Vector;

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

		Vector<String> v = new Vector<String>();

		v.add("Zonguldak");
		v.add("Sinop");
		v.add("Trabzon");
		v.add("Rize");
		v.add("İzmit");
		// indisi 3 olan konuma bir öğe sokuştur (insetion)
		v.add(3, "Bafra");
		// v.size() vektörün bileşen sayısını verir
		System.out.println("Vektörün uzunluğu :" + v.size());
		// v.get(i) vektörün i-inci indisli terimini verir
		for (int i = 0; i < v.size(); i++) {
			System.out.println("Vektör öğesi : " + i + " :" +
v.get(i));
		}
	}
}
	/*
	Çıktı:
	 Vektörün uzunluğu :6
	 Vektör öğesi : 0 :Zonguldak
	 Vektör öğesi : 1 :Sinop
	 Vektör öğesi : 2 :Trabzon
	 Vektör öğesi : 3 :Bafra
	 Vektör öğesi : 4 :Rize
	 Vektör öğesi : 5 :İzmit
	*/
```

Örnek:

Aşağıdaki örnek vektör metotlarının kullanılışını göstermektedir.

```
import java.util.*;

public class VectorDemo {
	public static void main(String[] args) {
		Vector<Object> vector = new Vector<Object>();
		int primitiveType = 100;
		Integer wrapperType = new Integer(200);
		String str = "Başkent Ankara";
		vector.add(primitiveType);
		vector.add(wrapperType);
		vector.add(str);
		vector.add(2, new Integer(300));
		System.out.println("vektörün öğeleri  : " + vector);
		System.out.println("vektörün uzunluğu : " + vector.size());
		System.out.println("vektörün 2 numaralı bileşeni  : "
				+ vector.elementAt(2));
		System.out.println("vektörün ilk öğesi            : "
				+ vector.firstElement());
		System.out.println("vektörün son öğesi            : "
				+ vector.lastElement());
		vector.removeElementAt(2);
		Enumeration e = vector.elements();
```

```
            System.out.println("vektörün öğeleri    : " + vector);

            while (e.hasMoreElements()) {
                  System.out.println("öğe    : " + e.nextElement());

            }
      }
}
      /*
      Çıktı:
       vektörün öğeleri   : [100, 200, 300, Başkent Ankara]
       vektörün uzunluğu : 4
       vektörün 2 numaralı bileşeni   : 300
       vektörün ilk öğesi             : 100
       vektörün son öğesi             : Başkent Ankara
       vektörün öğeleri    : [100, 200, Başkent Ankara]
       öğe    : 100
       öğe    : 200
       öğe    : Başkent Ankara
      */
```

Örnek:

```
import java.util.Vector;

public class QueueDemo {

      public static void main(String[] args) {

            Vector<String> vc = new Vector<String>();

            // <E> Vector'un öğe tipi: String, Integer, Object, vb.

            // add() metodu Vector'a öğe ekler
            vc.add("Burak");
            vc.add("İbrahim");
            vc.add("Deniz");
            vc.add("Mine");
            vc.add("Çağlar");

            // index= 3 olan yere sokuşturma (insertion)
            vc.add(3, "Zeynep");

            // vc.size() metodu Vector'ün sığasını verir
            System.out.println("Vector'ün Sığası :" + vc.size());

            // Vector'ün öğelerini yazdırır
            for (int i = 0; i < vc.size(); i++) {
                  System.out.println("Vector öğesi " + i + " :" +
vc.get(i));
            }
      }
}
      /*
      Çıktı:
```

```
Vector'ün Sığası :6
Vector öğesi 0 :Burak
Vector öğesi 1 :İbrahim
Vector öğesi 2 :Deniz
Vector öğesi 3 :Zeynep
Vector öğesi 4 :Mine
Vector öğesi 5 :Çağlar
*/
```

Örnek:

```
import java.util.*;

public class Vector01 {

	public static void main(String[] args) {
		Vector v = new Vector();
		v.add("doğrusal");
		v.add("cebir");
		System.out.println("Önceki Vector = " + v);
		v.clear();
		System.out.println("Sonraki Vector  = " + v);    }
}
	/*
	Çıktı:
	 Önceki Vector = [doğrusal, cebir]
	 Sonraki Vector  = []
	*/
```