- *Constructor Summary*

Constructors

Constructor and Description
`Vector()` Constructs an empty vector so that its internal data array has size `10` and its standard capacity increme
`Vector(Collection<? extends E> c)` Constructs a vector containing the elements of the specified collection, in the order they are returned by
`Vector(int initialCapacity)` Constructs an empty vector with the specified initial capacity and with its capacity increment equal to ze
`Vector(int initialCapacity, int capacityIncrement)` Constructs an empty vector with the specified initial capacity and capacity increment.

- *Method Summary*

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`boolean`	`add(E e)` Appends the specified element to the end of this Vector.
`void`	`add(int index, E element)` Inserts the specified element at the specified position in this Vector.
`boolean`	`addAll(Collection<? extends E> c)` Appends all of the elements in the specified Collection to the end of t returned by the specified Collection's Iterator.
`boolean`	`addAll(int index, Collection<? extends E> c)` Inserts all of the elements in the specified Collection into this Vector
`void`	`addElement(E obj)` Adds the specified component to the end of this vector, increasing its

int	**capacity**() Returns the current capacity of this vector.
void	**clear**() Removes all of the elements from this Vector.
Object	**clone**() Returns a clone of this vector.
boolean	**contains**(**Object** o) Returns `true` if this vector contains the specified element.
boolean	**containsAll**(**Collection**<?> c) Returns true if this Vector contains all of the elements in the specifie
void	**copyInto**(**Object**[] anArray) Copies the components of this vector into the specified array.
E	**elementAt**(int index) Returns the component at the specified index.
Enumeration<**E**>	**elements**() Returns an enumeration of the components of this vector.
void	**ensureCapacity**(int minCapacity) Increases the capacity of this vector, if necessary, to ensure that it ca components specified by the minimum capacity argument.
boolean	**equals**(**Object** o) Compares the specified Object with this Vector for equality.
E	**firstElement**() Returns the first component (the item at index `0`) of this vector.
void	**forEach**(**Consumer**<? super **E**> action) Performs the given action for each element of the `Iterable` until al action throws an exception.
E	**get**(int index) Returns the element at the specified position in this Vector.

int	**hashCode**() Returns the hash code value for this Vector.
int	**indexOf**(**Object** o) Returns the index of the first occurrence of the specified element in t contain the element.
int	**indexOf**(**Object** o, int index) Returns the index of the first occurrence of the specified element in t from index, or returns -1 if the element is not found.
void	**insertElementAt**(**E** obj, int index) Inserts the specified object as a component in this vector at the speci
boolean	**isEmpty**() Tests if this vector has no components.
Iterator<**E**>	**iterator**() Returns an iterator over the elements in this list in proper sequence.
E	**lastElement**() Returns the last component of the vector.
int	**lastIndexOf**(**Object** o) Returns the index of the last occurrence of the specified element in th contain the element.
int	**lastIndexOf**(**Object** o, int index) Returns the index of the last occurrence of the specified element in th from index, or returns -1 if the element is not found.
ListIterator<**E**>	**listIterator**() Returns a list iterator over the elements in this list (in proper sequen
ListIterator<**E**>	**listIterator**(int index) Returns a list iterator over the elements in this list (in proper sequen the list.
E	**remove**(int index) Removes the element at the specified position in this Vector.
boolean	**remove**(**Object** o)

	Removes the first occurrence of the specified element in this Vector I element, it is unchanged.
boolean	**removeAll**(**Collection**<?> c) Removes from this Vector all of its elements that are contained in the
void	**removeAllElements**() Removes all components from this vector and sets its size to zero.
boolean	**removeElement**(**Object** obj) Removes the first (lowest-indexed) occurrence of the argument from
void	**removeElementAt**(int index) Deletes the component at the specified index.
boolean	**removeIf**(**Predicate**<? super **E**> filter) Removes all of the elements of this collection that satisfy the given pr
protected void	**removeRange**(int fromIndex, int toIndex) Removes from this list all of the elements whose index is between fr exclusive.
void	**replaceAll**(**UnaryOperator**<**E**> operator) Replaces each element of this list with the result of applying the oper
boolean	**retainAll**(**Collection**<?> c) Retains only the elements in this Vector that are contained in the spe
E	**set**(int index, **E** element) Replaces the element at the specified position in this Vector with the
void	**setElementAt**(**E** obj, int index) Sets the component at the specified index of this vector to be the sp
void	**setSize**(int newSize) Sets the size of this vector.
int	**size**() Returns the number of components in this vector.
void	**sort**(**Comparator**<? super **E**> c)

	Sorts this list according to the order induced by the specified **Compar**
Spliterator<**E**>	**spliterator**() Creates a ***late-binding*** and *fail-fast* **Spliterator** over the elemer
List<**E**>	**subList**(int fromIndex, int toIndex) Returns a view of the portion of this List between fromIndex, inclusiv
Object[]	**toArray**() Returns an array containing all of the elements in this Vector in the c
<T> T[]	**toArray**(T[] a) Returns an array containing all of the elements in this Vector in the c returned array is that of the specified array.
String	**toString**() Returns a string representation of this Vector, containing the String r
void	**trimToSize**() Trims the capacity of this vector to be the vector's current size.

- Methods inherited from class java.lang.Object

  finalize, getClass, notify, notifyAll, wait, wait, wait

- Methods inherited from interface java.util.Collection

  parallelStream, stream