eg-5

Vector :→

Page No.:

1. Resizeable Array / Growable Array. underlined data structure.
2. Insertion order. preserved
3. Duplicates are allowed.
4. Hetrogeniows " "
5. null insertion is possible
6. Implements Serializable, Cloneable, RandomAccess.
7. Thread Safe. because method is synchronized.

Constructors :→

⇒

(1) vector v = new vector();
creates an empty vector objects with default initial capacity 10.

new capacity = currentcapacity * 2.

(2) vector v = new vector(int initial capacity)
creates an empty vector object with specified initial capacity

(3) vector v = new Vector(int initial capacity, int incremental capacity)
Ex - vector v = new Vector(1000, 5)
→ ~~Creates~~

(4) veetor v = new veetor(collection c);
creates an equivalent veetor object for the given collection.
This constructor maint for ~~equival~~ inter conversion b/w collection objects.

⇒ Vector Specific methods :→

(1) To add objects :-
i) add(Object O) → Collection
ii) add(int index, Object O) - --- ~~List~~ list
iii) addElement(Object O) - --- -- vector

(2) To Remove object

remove(Object o) ---- Collection
removeElement(Object o) ---- Vector
remove(int index) ---- L
removeElementAt(int index) ---- V
clear() ---- C
removeAllElements() ---- V

(3) To Get Objects :->

Object get(int index) -> L
Object elementAt(int index) -> V
Object firstElement() -> V
Object LastElement() -> V

(4) other methods :-

int size();
int capacity;
Enumeration elements;

Ex:-

```
import java.util.*;
Public class DemoTest
{
p s v m(String[] args)
{
Vector v = new Vector();
S.O.p(v.capacity()); -> 10
for(int i=0; i<=10; i++)
{
v.addElement(i);
}
S.O.p(v.capacity()); -> 10
v.addElement("A");
S.O.p(v.capacity()); -> 20
S.O.p(v); -> [1,2, ....10, A]
}
}
```