Vector Class:

- Vector is available since jdk1.0V
- It allows duplicates & insertion order is maintained.
- Default capacity when creating an Vector is 10.
- Its capacity increases by (CurrentCapacity*2).
- It is synchronized by default.

Vector v=new Vector();

Vector v=new Vector(int capacity);

Vector v=new Vector(int capacity, int incremental capacity);

Vector Methods

Method	Description
addElement(Object o);	Adds the specified component to the end of this vector, increasing its size by one.
removeElement(Object o);	Removes the first (lowest-indexed) occurrence of the argument from this vector.
removeElementAt(int index);	Deletes the component at the specified index.
removeAllElements();	Removes all components from this vector and sets its size to zero.
Object elementAt(int index);	Returns the component at the specified index.
Object lastElement();	Returns the last component of the vector.
Object firstElement();	Returns the first component (the item at index o) of this vector.

```
Implementing Vector
 9
             System.out.println("Implementing Vector\n");
10
                                                                                                     [10, Java, null, 10, A, true, 88, 1]
11
12
             Vector<Object> v=new Vector<Object>();
13
14
             v.add(10); // Insertion order is maintained
             v.add("Java");// Heterogeneous data is allowed
             v.add(null);// null value is allowed
             v.add(10); // Duplicate elements are allowed
v.add('A'); // It is available from Java 1.0 (It is a Legacy Class)
16
17
             v.add(true); // Its default capacity is 10
v.add(88);// Its size increases by DOUBLE
18
19
20
             v.add(1); // It is Synchronized
21
22
23
24
             System.out.println(v);
26
        public static void main(String[] args)
27
             ClassA aobj=new ClassA();
28
29
             aobj.meth1();
30
31 }
              V.auu(88);// ILS SIZE INCREASES DY DOUBLE
                                                                               Implementing Vector
20
              v.add(1); // It is Synchronized
21
                                                                                [10, Java, null, 10, A, true, 88, 1]
              System.out.println(v);
                                                                                size() : 8
             System.out.println("\nsize() : "+v.size()); // 8
System.out.println("get() : "+v.get(0)); //10
System.out.println("get() : "+v.get(v.size()-1));//1
                                                                                get(): 10
25
                                                                                get(): 1
26
                                                                                capacity(): 10
27
              System.out.println("capacity() : "+v.capacity());
                                                                                [10, Java, 1000, null, 10, A, true, 88, 1, 99, 2000]
              v.add(2,1000);
                                                                                capacity(): 20
30
              v.add(2000);
31
              v.add(v.size()-1,99);
32
33
34
              System.out.println("\n"+v);
              System.out.println("capacity() : "+v.capacity());
35
36
37€
         public static void main(String[] args)
38
39
              ClassA aobj=new ClassA();
40
              aobj.meth1();
41
42 }
```

```
1 package com.pack1;
 2
 3 public class Employee
 4 {
 5
       private String empName;
 6
       private String empSal;
 7
       private String empDept;
 8
 99
       public Employee(String empName, String empSal, String empDept)
10
11
           this.empName = empName;
12
           this.empSal = empSal;
13
           this.empDept = empDept;
14
159
       @Override
16
       public String toString()
                                    T
17
       {
18
           return empSal;
19
20
       }
21 }
 3@import java.util.Enumeration;
 4 import java.util.Iterator:
 5 import java.util.Vector;
 7 public class ClassA
 8 {
 98
       void meth1()
10
       {
11
           System.out.println("Implementing Vector\n");
12
13
           Vector<Object> v=new Vector<Object>();
14
15
           v.add(10); // Insertion order is maintained
16
           v.add("Java");// Heterogeneous data is allowed
17
           v.add(null);// null value is allowed
18
           v.add(10); // Duplicate elements are allowed
19
           v.add('A'); // It is available from Java 1.0 (It is a Legacy Class)
20
           v.add(true); // Its default capacity is 10
           v.add(88);// Its size increases by DOUBLE
21
22
           v.add(1); // It is Synchronized
23
24
           System.out.println(v);
25
           System.out.println("\nsize() : "+v.size()); // 8
26
27
           System.out.println("get() : "+v.get(0)); //10
```

```
System.out.println("get() : "+v.get(v.size()-1));//1
28
29
             System.out.println("capacity() : "+v.capacity());
30.
31
             v.add(2,1000);
32
             v.add(2000);
33
             v.add(v.size()-1,99);
34
35
             System.out.println("\n"+v);
36
             System.out.println("capacity() : "+v.capacity());
37
             v.set(1, "Java is awesome");
38
39
             System.out.println("\n"+v);
40
41
          System.out.println("\nReteriving the data in BOTH directions by using for loop");
42
          for(int i=0;i<=v.size()-1;i++)</pre>
              System.out.print(v.get(i)+" ");
43
44
          System.out.println();
45
          for(int i=v.size()-1;i>=0;i--)
              System.out.print(v.get(i)+" ");
47
48
          System.out.println("\n\nReteriving by using for-each loop");
49
          for(Object o:v)
50
              System.out.print(o+" ");
51
          System.out.println("\n\nReteriving by using Enumeration Interface");
52
53
          Enumeration<Object> e=v.elements();
54
          while(e.hasMoreElements())
55
          {
              System.out.print(e.nextElement()+" ");
56
57
          }
59
            System.out.println("\n\nReteriving by using Iterator Interface");
            Iterator<Object> i=v.iterator();
60
61
            while(i.hasNext())
62
            {
63
                System.out.print(i.next()+" ");
64
65
            System.out.println("\n\nelementAt() : "+v.elementAt(1));
66
67
        }
68e
       void meth2()
69
            System.out.println("Passing User defined Class Object into Vector");
70
71
            Vector<Employee> v=new Vector<Employee>();
72
73
```

```
Employee emp1=new Employee("Kishan", "10000", "Java");
74
75
           Employee emp2=new Employee("John", "30000", "AWS");
           Employee emp3=new Employee("Cristine", "20000", "Oracle");
76
77
78
           v.add(emp1);
           v.add(emp2);
79
80
           v.add(emp3);
81
82
           Enumeration<Employee> e=v.elements();
83
           while(e.hasMoreElements())
84
           {
85
                Employee s1=e.nextElement();
86
                String s2=s1.toString();
87
               if(s2.equals("30000"))
88
                    System.out.println(s2);
           }
89
90
91=
       public static void main(String[] args)
92
           ClassA aobj=new ClassA();
93
94
           //aobj.meth1();
95
           aobj.meth2();
96
       }
97 }
```

I need entire data if my salary is 30000 in meth2

```
1 package com.pack1;
 2
 3
    public class Employee
 4 {
 5
          private String empName;
 6
          private String empSal;
 7
          private String empDept;
 8
 98
          public Employee(String empName, String empSal, String empDept)
10
11
                this.empName = empName;
12
                this.empSal = empSal;
13
                this.empDept = empDept;
14
158
          public String getEmpsal()
16
17
                return empSal;
18
19
20e
          @Override
          public String toString()
21
22
23
                return empName+" "+empSal+" "+empDept;
24
25 }
16
          vector comproyees v-new vector comproyees(),
                                                                                   <terminated> ClassA [Java Application] C:\Program Files\Java\jd
73
                                                                                   Passing User defined Class Object
         Employee empl=new Employee("Kishan", "10000", "Java");
Employee emp2=new Employee("John", "30000", "AWS");
74
                                                                                   John 30000 AWS
75
         Employee emp3=new Employee("Cristine", "20000", "Oracle");
76
77
78
         v.add(emp1);
                                                                                         Ι
79
         v.add(emp2);
80
         v.add(emp3);
81
         Enumeration<Employee> e=v.elements();
82
83
         while(e.hasMoreElements())
84
85
             Employee s1=e.nextElement();
86
87
             String s2=s1.getEmpsal(); -
88
             if(s2.equals("30000"))
                 System.out.println(s1);
89
90
         }
91
920
      public static void main(String[] args)
93
94
         ClassA aobj=new ClassA();
95
         //aobj.meth1();
96
         aobj.meth2();
```

```
vector comproyees v-new vector comproyees(),
                                                                                                                                     <terminated> ClassA [Java Application] C:\Program Files\Java\jo
73
74
                                                                                                                                     Passing User defined Class Object
               Employee emp1=new Employee("Kishan", "10000", "Java");
Employee emp2=new Employee("John", "30000", "AWS");
Employee emp3=new Employee("Cristine", "20000", "Oracle");
75
76
77
               v.add(emp1);
78
               v.add(emp1);
v.add(emp2);
v.add(emp3);
79
80
81
82
               Enumeration<Employee> e=v.elements();
83
               while(e.hasMoreElements())
84
85
                     Employee s1=e.nextElement();
86
87
                     String s2=s1.getEmpsal();
if(s2.equals("30000"))
88
                          System.out.println(s1);
89
90
91
                     int i=Integer.parseInt(s1.getEmpsal());
                     if(i>=20000)
92
93
                          System.out.println(s1);
94
95
         public static void main(String[] args)
96€
```

```
public class ClassA {
    String empName; int empId; String empDept; }
    Public class ClassA {
        // Print the elements present in ArrayList by using Vector
        return new Vector<ClassA>();
    }
    public static void main(String[] args) {
        //If there is any employee name matches which is palendrome //then count how mant names are like that and Print them.
    }
}
```

Feature	ArrayList	Vector
Thread Safety	Not synchronized (Not thread-safe)	Synchronized (Thread-safe)
Performance	Faster (no overhead of synchronization)	Slower (due to synchronization)
Legacy	Part of Java 1.2 (modern)	Older, from Java 1.0 (legacy)
Growth Policy	Increases size by 50% when full	Doubles its size when full
Use Case	Used in single- threaded environments	Used in multi- threaded environments
Enumeration	Doesn't support Enumeration directly	Supports both Enumeration and Iterator