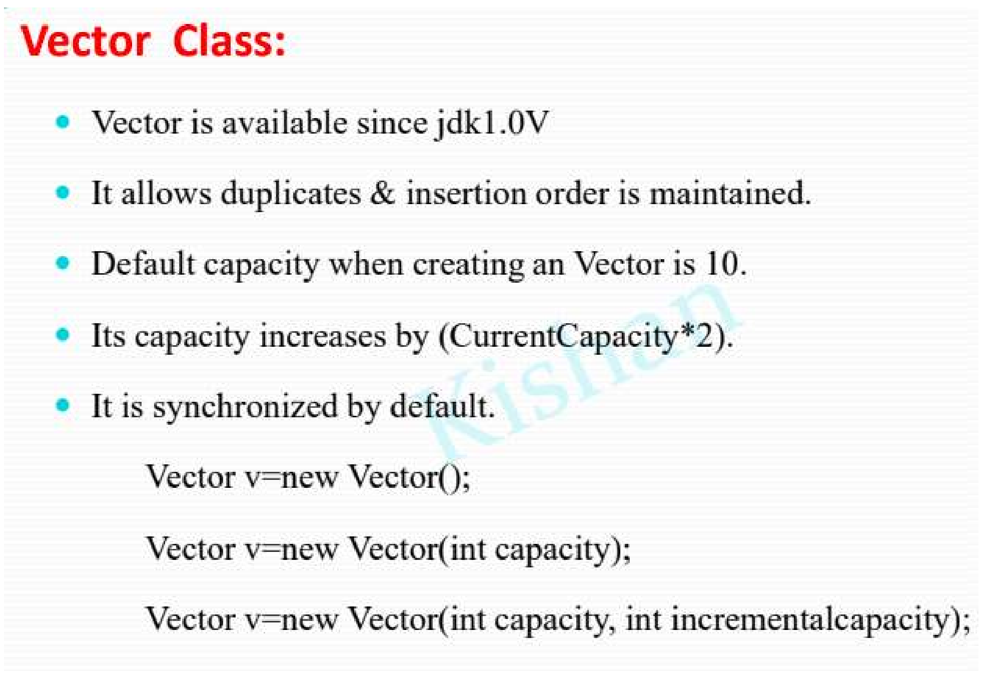
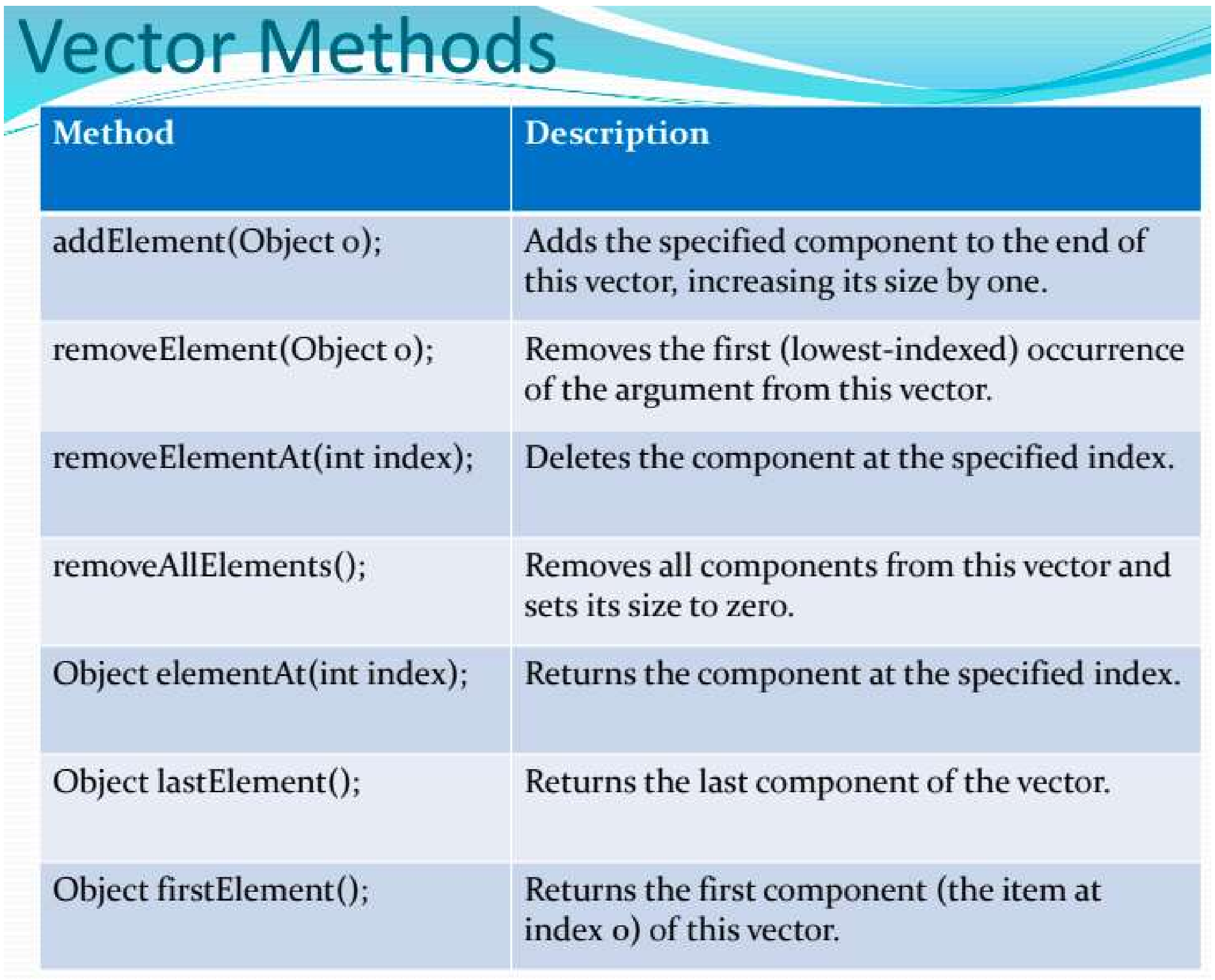


Vector Class:

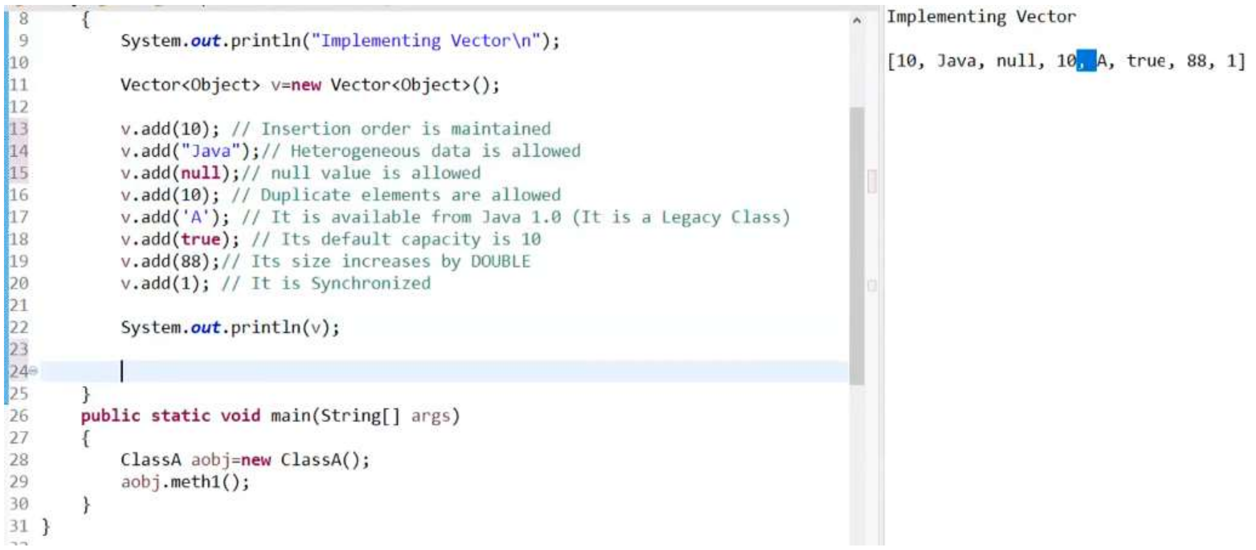
- Vector is available since jdk1.0V
- It allows duplicates & insertion order is maintained.
- Default capacity when creating an Vector is 10.
- Its capacity increases by (CurrentCapacity*2).
- It is synchronized by default.

```
Vector v=new Vector();

Vector v=new Vector(int capacity);

Vector v=new Vector(int capacity, int incrementalcapacity);
```



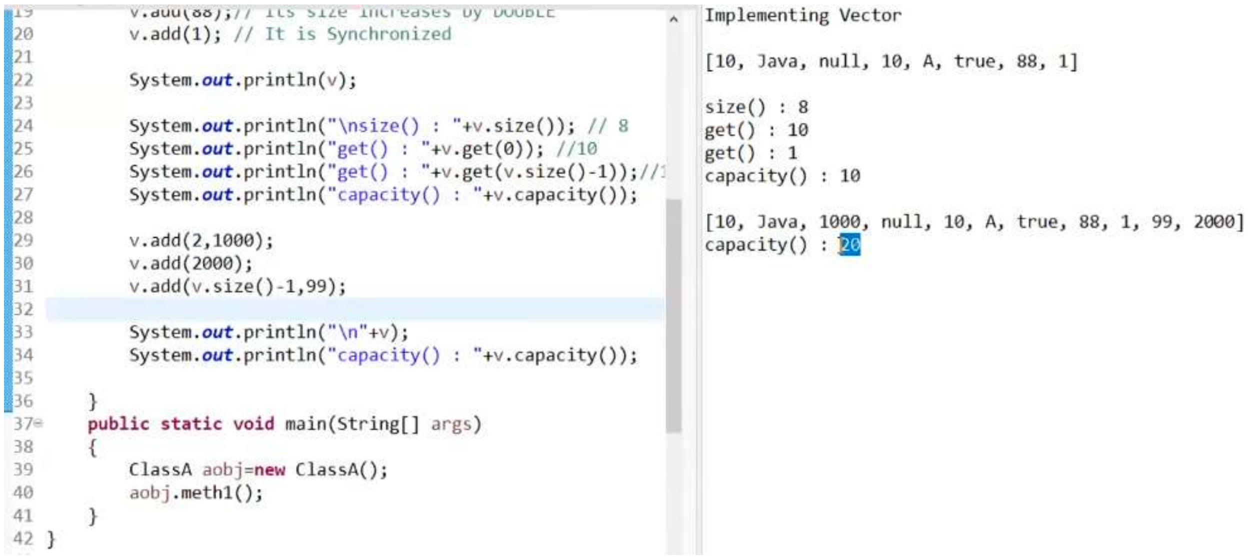
Vector Methods

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.



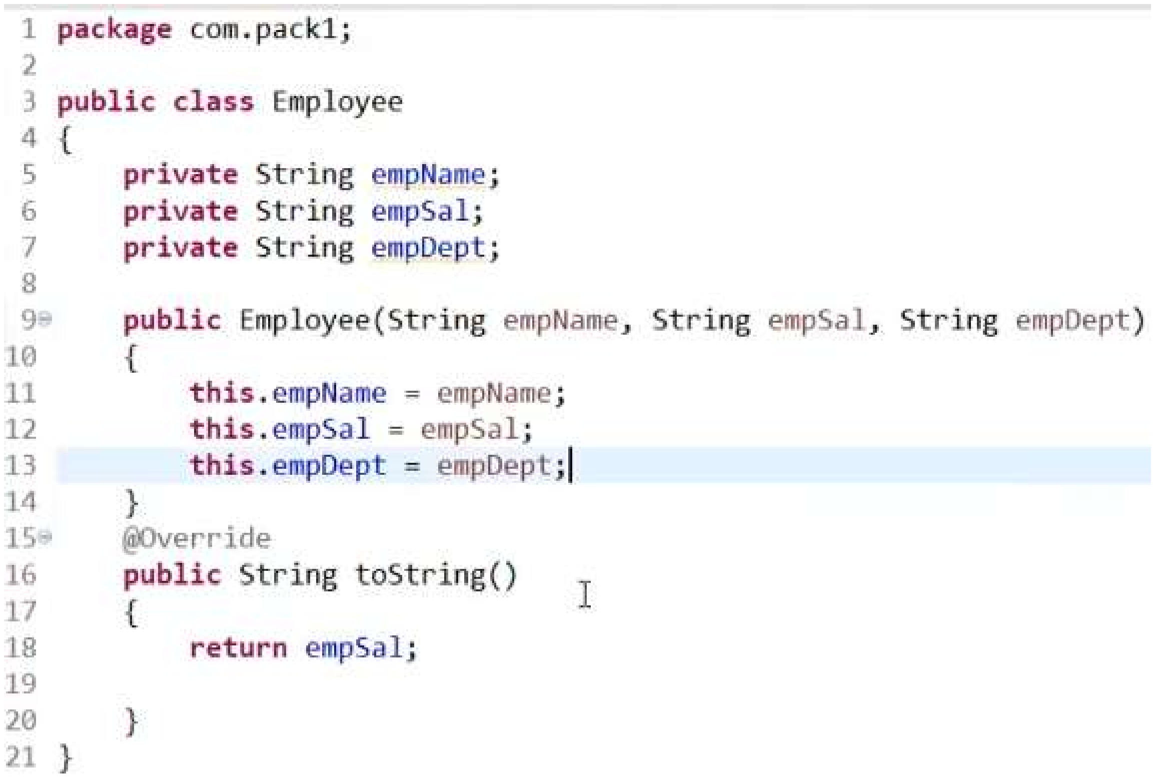
```java
    {
        System.out.println("Implementing Vector\n");

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

        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)
        v.add(true); // Its default capacity is 10
        v.add(88);// Its size increases by DOUBLE
        v.add(1); // It is Synchronized

        System.out.println(v);

        
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
Implementing Vector

[10, Java, null, 10, A, true, 88, 1]
```

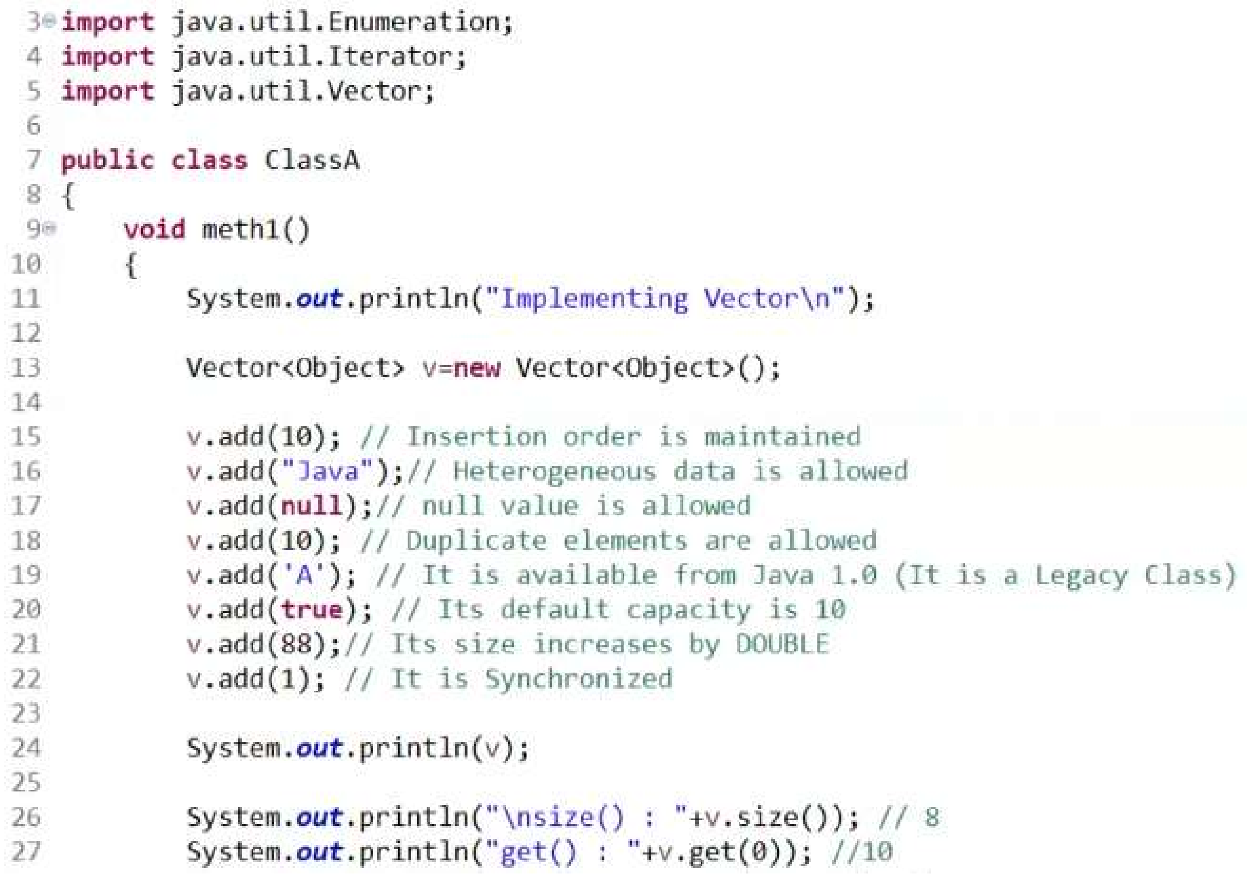


```java
        v.add(88);// Its size increases by DOUBLE
        v.add(1); // It is Synchronized

        System.out.println(v);

        System.out.println("\nsize() : "+v.size()); // 8
        System.out.println("get() : "+v.get(0)); //10
        System.out.println("get() : "+v.get(v.size()-1));//
        System.out.println("capacity() : "+v.capacity());

        v.add(2,1000);
        v.add(2000);
        v.add(v.size()-1,99);

        System.out.println("\n"+v);
        System.out.println("capacity() : "+v.capacity());

    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

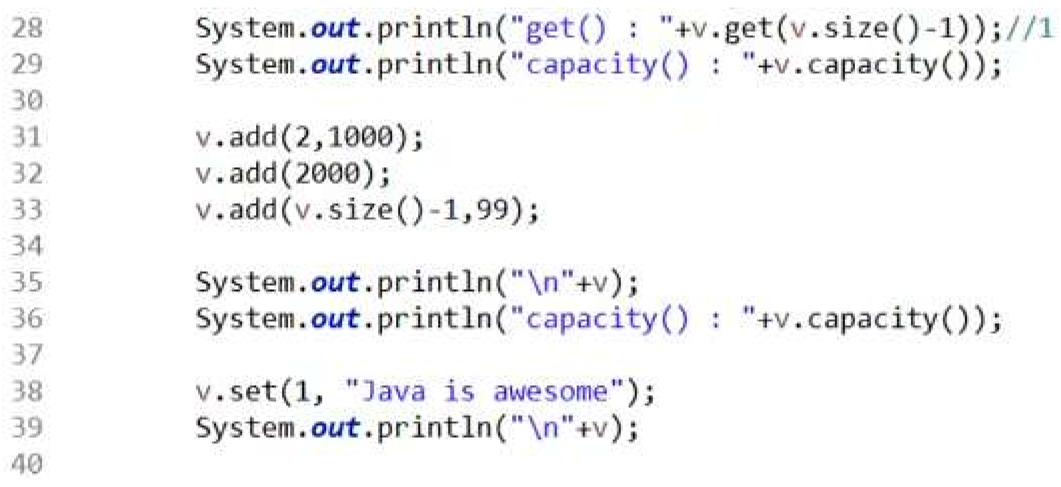
```
Implementing Vector

[10, Java, null, 10, A, true, 88, 1]

size() : 8
get() : 10
get() : 1
capacity() : 10

[10, Java, 1000, null, 10, A, true, 88, 1, 99, 2000]
capacity() : 20
```



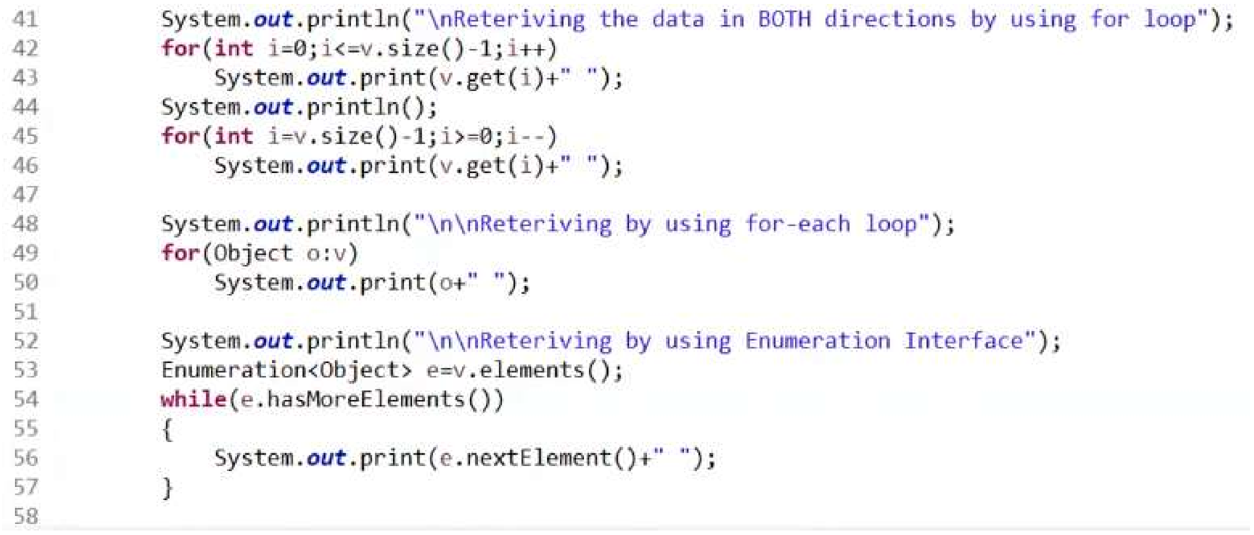
```java
package com.pack1;

public class Employee
{
    private String empName;
    private String empSal;
    private String empDept;

    public Employee(String empName, String empSal, String empDept)
    {
        this.empName = empName;
        this.empSal = empSal;
        this.empDept = empDept;
    }
    @Override
    public String toString()
    {
        return empSal;

    }
}
```



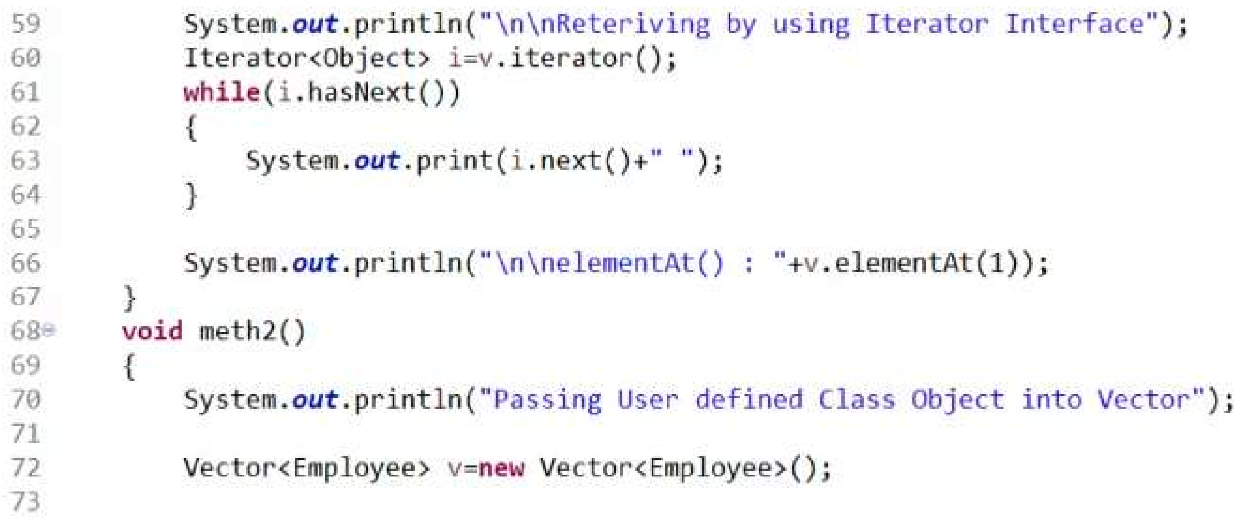
```java
import java.util.Enumeration;
import java.util.Iterator;
import java.util.Vector;

public class ClassA
{
    void meth1()
    {
        System.out.println("Implementing Vector\n");

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

        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)
        v.add(true); // Its default capacity is 10
        v.add(88);// Its size increases by DOUBLE
        v.add(1); // It is Synchronized

        System.out.println(v);

        System.out.println("\nsize() : "+v.size()); // 8
        System.out.println("get() : "+v.get(0)); //10
```



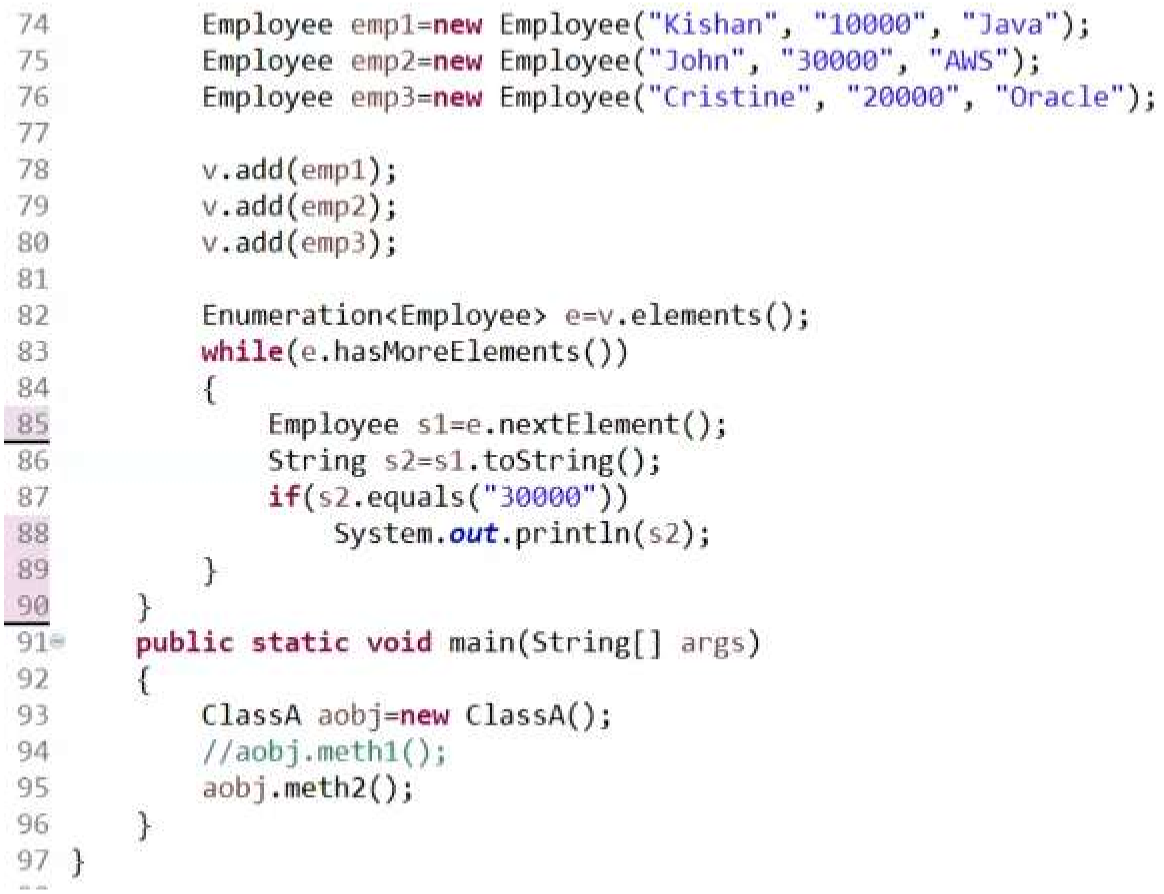
```java
        System.out.println("get() : "+v.get(v.size()-1));//1
        System.out.println("capacity() : "+v.capacity());

        v.add(2,1000);
        v.add(2000);
        v.add(v.size()-1,99);

        System.out.println("\n"+v);
        System.out.println("capacity() : "+v.capacity());

        v.set(1, "Java is awesome");
        System.out.println("\n"+v);

```



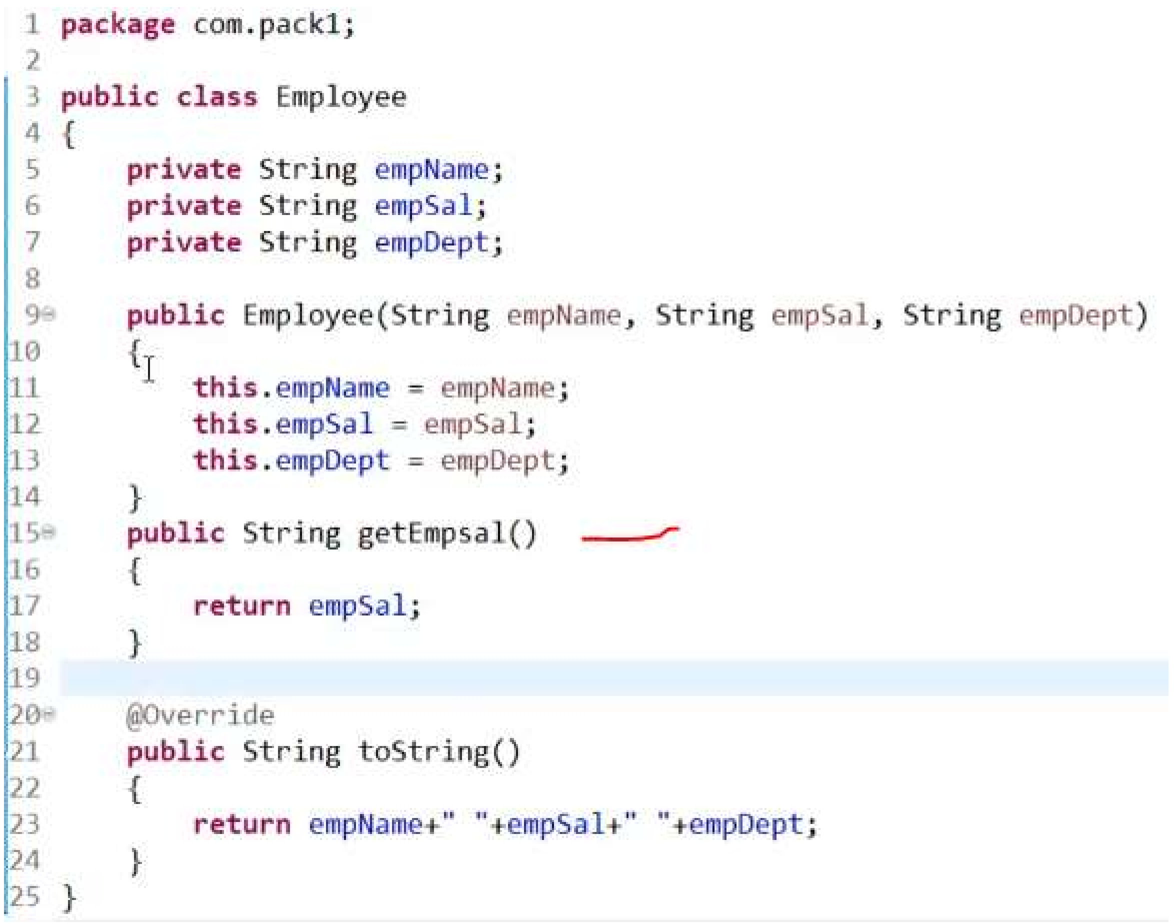
```java
        System.out.println("\nReteriving the data in BOTH directions by using for loop");
        for(int i=0;i<=v.size()-1;i++)
            System.out.print(v.get(i)+" ");
        System.out.println();
        for(int i=v.size()-1;i>=0;i--)
            System.out.print(v.get(i)+" ");

        System.out.println("\n\nReteriving by using for-each loop");
        for(Object o:v)
            System.out.print(o+" ");

        System.out.println("\n\nReteriving by using Enumeration Interface");
        Enumeration<Object> e=v.elements();
        while(e.hasMoreElements())
        {
            System.out.print(e.nextElement()+" ");
        }

```



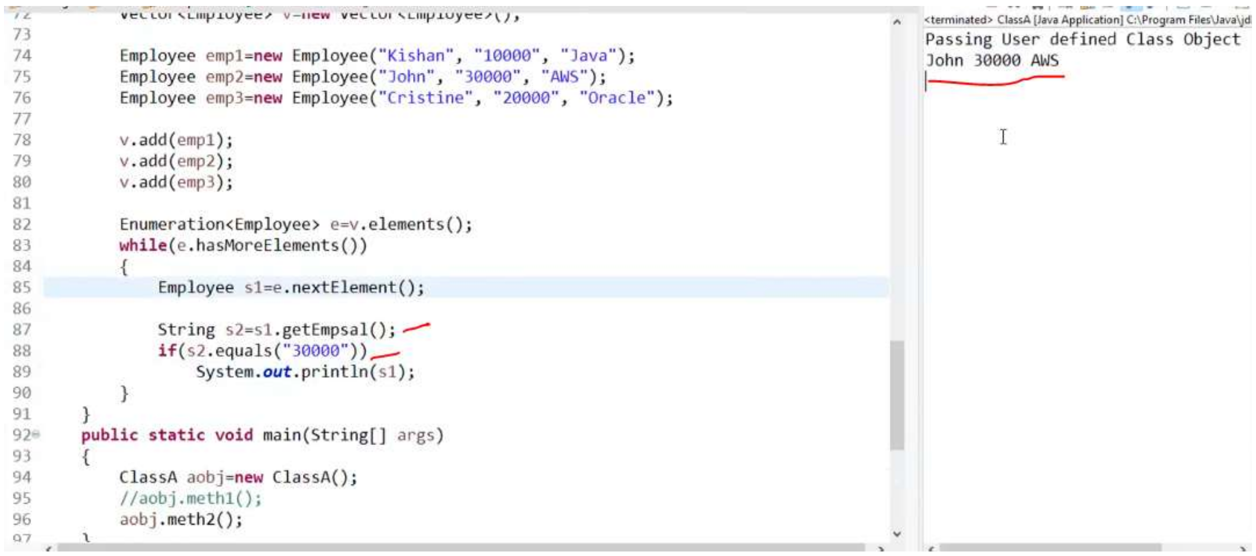
```java
        System.out.println("\n\nReteriving by using Iterator Interface");
        Iterator<Object> i=v.iterator();
        while(i.hasNext())
        {
            System.out.print(i.next()+" ");
        }

        System.out.println("\n\nelementAt() : "+v.elementAt(1));
    }
    void meth2()
    {
        System.out.println("Passing User defined Class Object into Vector");

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

```


```
        Employee emp1=new Employee("Kishan", "10000", "Java");
        Employee emp2=new Employee("John", "30000", "AWS");
        Employee emp3=new Employee("Cristine", "20000", "Oracle");

        v.add(emp1);
        v.add(emp2);
        v.add(emp3);

        Enumeration<Employee> e=v.elements();
        while(e.hasMoreElements())
        {
            Employee s1=e.nextElement();
            String s2=s1.toString();
            if(s2.equals("30000"))
                System.out.println(s2);
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();
        aobj.meth2();
    }
}
```

I need entire data if my salary is 30000 in meth2

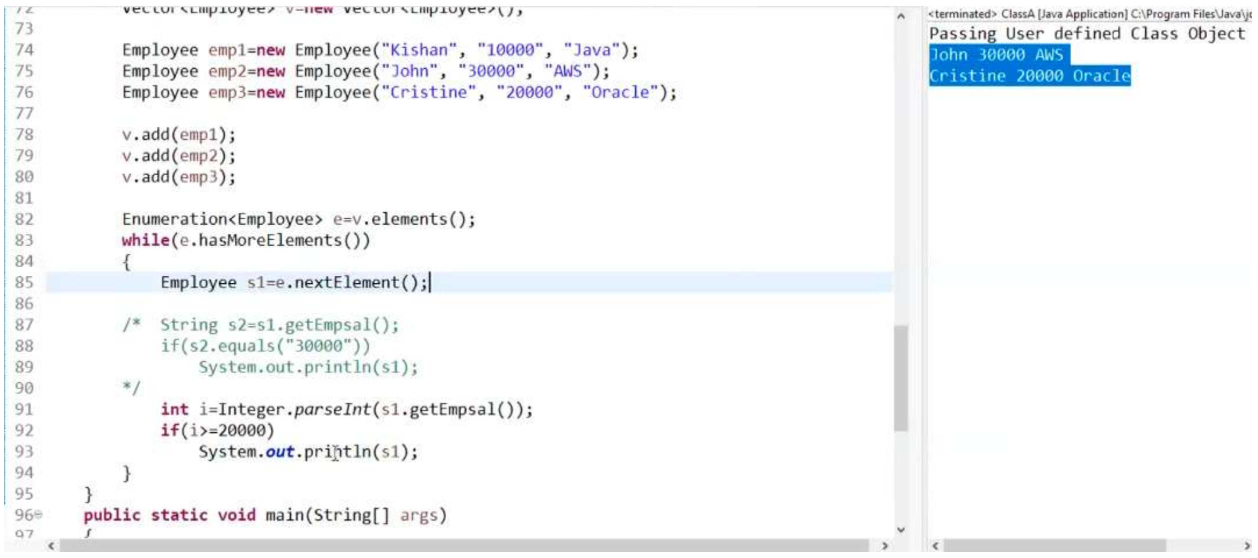


```java
package com.pack1;

public class Employee
{
    private String empName;
    private String empSal;
    private String empDept;

    public Employee(String empName, String empSal, String empDept)
    {
        this.empName = empName;
        this.empSal = empSal;
        this.empDept = empDept;
    }
    public String getEmpsal()
    {
        return empSal;
    }

    @Override
    public String toString()
    {
        return empName+" "+empSal+" "+empDept;
    }
}
```



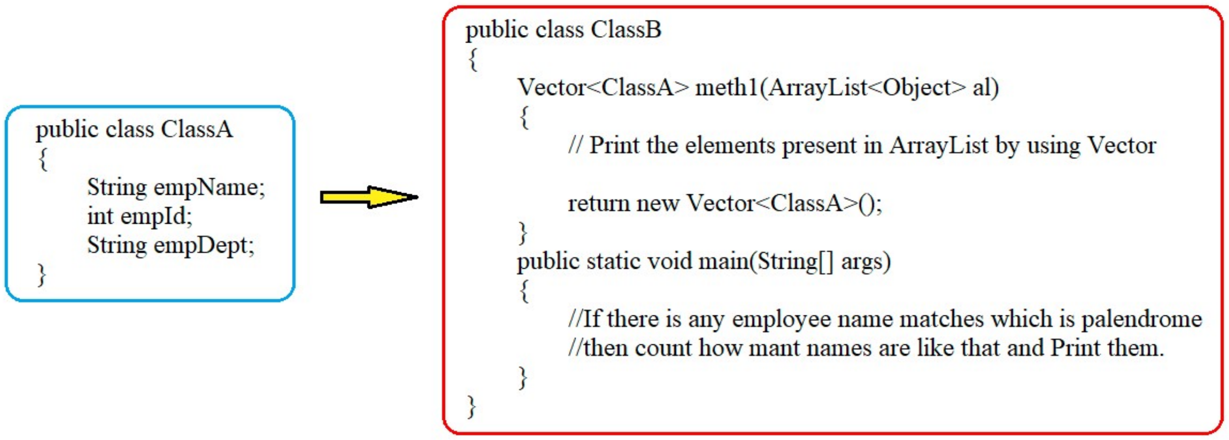
```java
        Vector<Employee> v=new Vector<Employee>();

        Employee emp1=new Employee("Kishan", "10000", "Java");
        Employee emp2=new Employee("John", "30000", "AWS");
        Employee emp3=new Employee("Cristine", "20000", "Oracle");

        v.add(emp1);
        v.add(emp2);
        v.add(emp3);

        Enumeration<Employee> e=v.elements();
        while(e.hasMoreElements())
        {
            Employee s1=e.nextElement();

            String s2=s1.getEmpsal();
            if(s2.equals("30000"))
                System.out.println(s1);
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //aobj.meth1();
        aobj.meth2();
```

```
<terminated> ClassA [Java Application] C:\Program Files\Java\jd
Passing User defined Class Object
John 30000 AWS
```



```
        Vector<Employee> v=new Vector<Employee>();

        Employee emp1=new Employee("Kishan", "10000", "Java");
        Employee emp2=new Employee("John", "30000", "AWS");
        Employee emp3=new Employee("Cristine", "20000", "Oracle");

        v.add(emp1);
        v.add(emp2);
        v.add(emp3);

        Enumeration<Employee> e=v.elements();
        while(e.hasMoreElements())
        {
            Employee s1=e.nextElement();

        /*  String s2=s1.getEmpsal();
            if(s2.equals("30000"))
                System.out.println(s1);
        */
            int i=Integer.parseInt(s1.getEmpsal());
            if(i>=20000)
                System.out.println(s1);
        }
    }
    public static void main(String[] args)
```

```
<terminated> ClassA [Java Application] C:\Program Files\Java\j
Passing User defined Class Object
John 30000 AWS
Cristine 20000 Oracle
```



```
public class ClassA
{
        String empName;
        int empId;
        String empDept;
}
```

```
public class ClassB
{
        Vector<ClassA> meth1(ArrayList<Object> al)
        {
                // 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