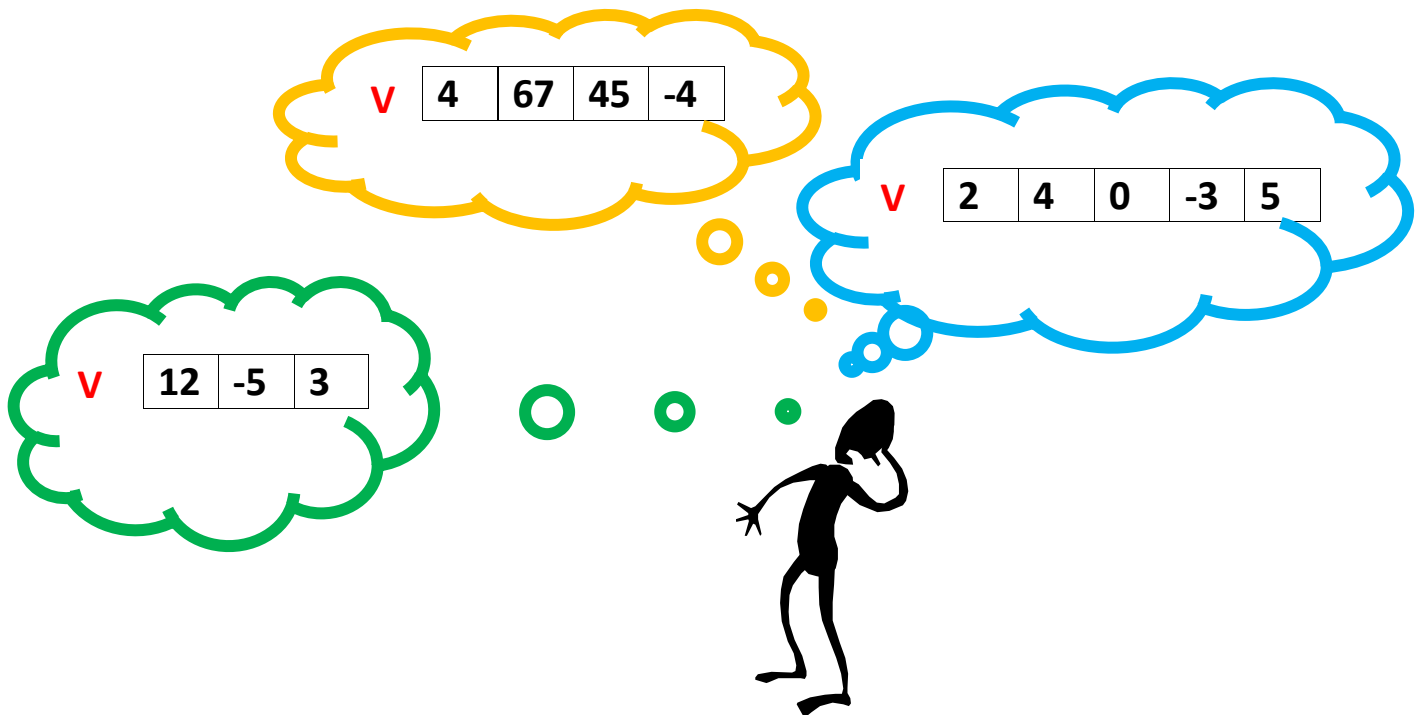


CLASS VECTOR Y SUS METODOS



CVector

V

0	1	2	3	4	5	6	7	...	12

dim = -1

dim → -1 **V**

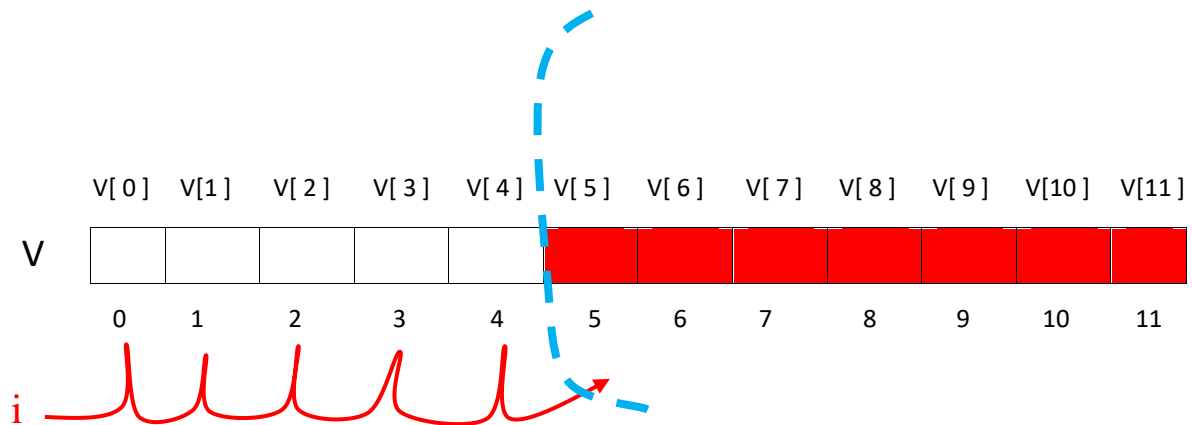
0		V[0]
1		V[1]
2		V[2]
3		V[3]
4		V[4]
5		V[5]
6		V[6]
7		V[7]
8		V[8]
9		V[9]
10		V[10]
11		V[11]

```
public class CVector {  
  
    private int V[] = new int[12];  
    private int dim;  
  
    public CVector() {  
        dim = -1;  
    }  
}
```

El vector puede tener 12 espacios para colocar elementos, esto quiere decir que pueden ocuparse los 12 espacios o solo unos cuantos, los espacios que se ocuparan los indicara el usuario final y el control se realizara con la variable **dim**

Al crearse la estructura esta contiene **basura**

NOTA:

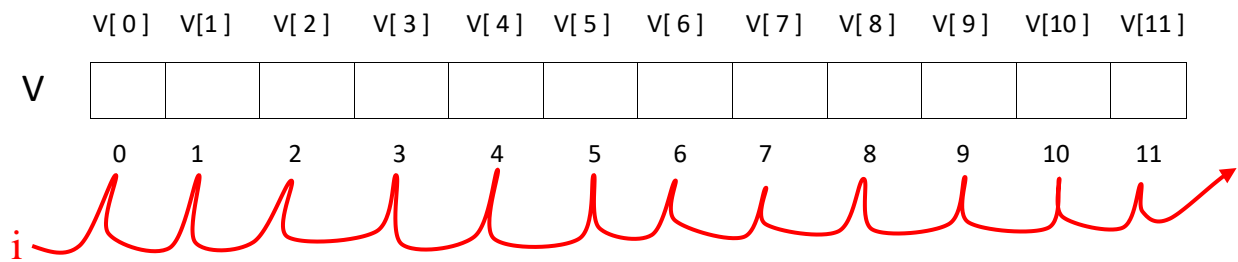


$\text{dim} = 4$

Elementos en el vector desde la posición 0 hasta la 4, en total 5 elementos.

Del total de 12 casillas sólo se están ocupando hasta donde indica dim .

La estructura V la podemos ver en forma de columna o fila

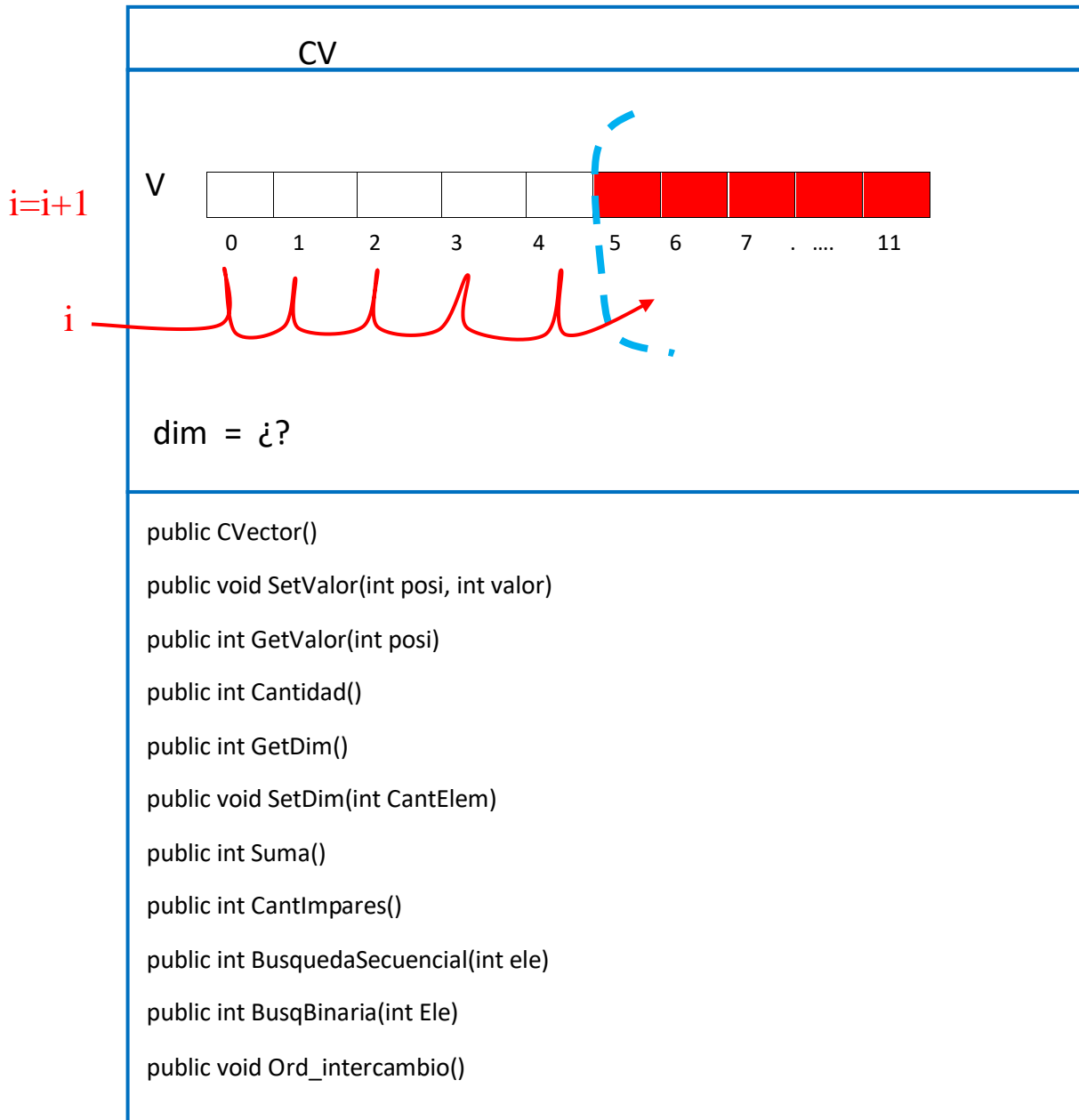


Para recorrer el Vector se necesita de un ciclo FOR y con la variable i para que incremente de uno en uno, este incremento de i es de la siguiente forma: $i=i+1$

TRABAJANDO CON EL OBJETO CV

En la práctica se está trabajando con la variable CV que es un objeto de tipo CVector.

OBJETO



CONSTRUCTOR

Cuando se define el objeto :

```
...ava\JFrameNumero.java X CNumero.java X CString.java X JString.java X JVector.java X CVector.java X
Source Design History
9
10 /**
11  *
12  * @author DellLaptop
13  */
14 public class JFVector extends javax.swing.JFrame {
15
16     /**
17      * Creates new form JFVector
18      */
19     CVector cv;
```

La estructura V se carga con basura, entonces se hace pasar por el constructor:

`cv=new CVector();`

Se tendrá:

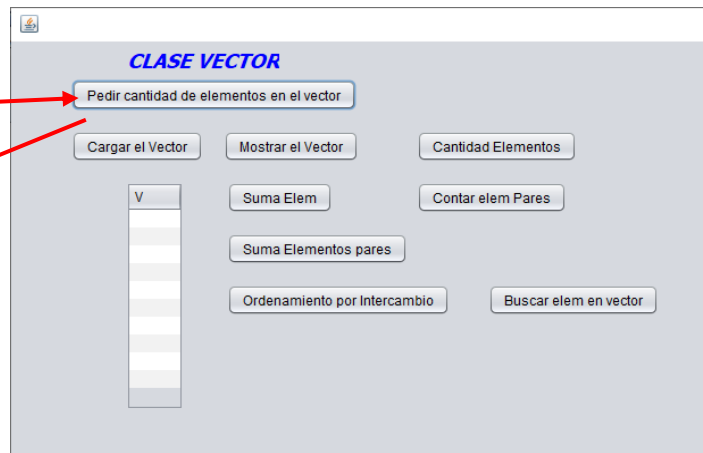
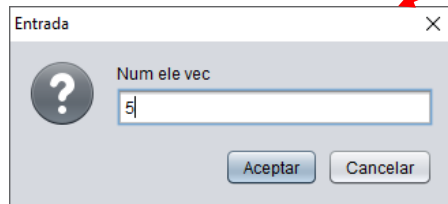
```
...ava\JFrameNumero.java X CNumero.java X CString.java X JString.java X JVector.java X CVector.java X
Source Design History
9
10 /**
11  *
12  * @author DellLaptop
13  */
14 public class JFVector extends javax.swing.JFrame {
15
16     /**
17      * Creates new form JFVector
18      */
19     CVector cv;
20     public JFVector() {
21         cv=new CVector();
22         initComponents();
23     }
24 }
```

CV											
V	89	5	-7	27	4	52	-3	1	...	2	
	0	1	2	3	4	5	6	7	8	...	11
dim = -1 //Lo que significa que no se tienen elementos válidos en el vector											
public CVector()											

COLOCAR CANTIDAD DE ELEMEN PARA EL VECTOR

Si el usuario indica que quiere se introduzcan **5 elementos**, entonces la variable **dim** que inicialmente estaba en **-1**, cambiara a la **cantidad-1**, esto porque en el vector se tiene la posición **0**

En el formulario JFrameVector:

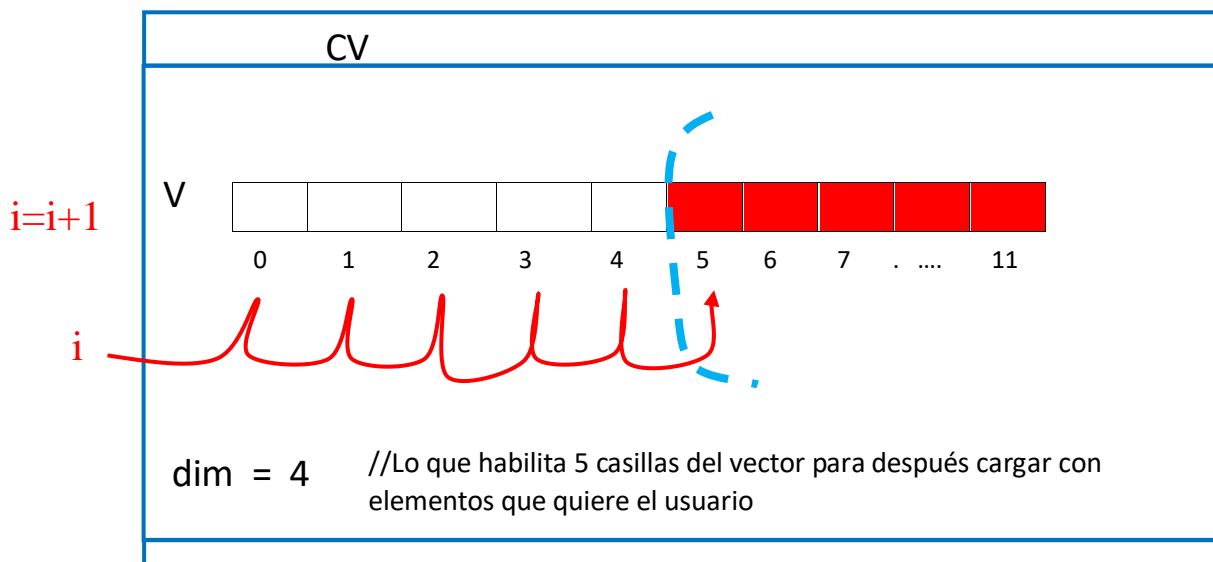


En el Source del JFrameVector:

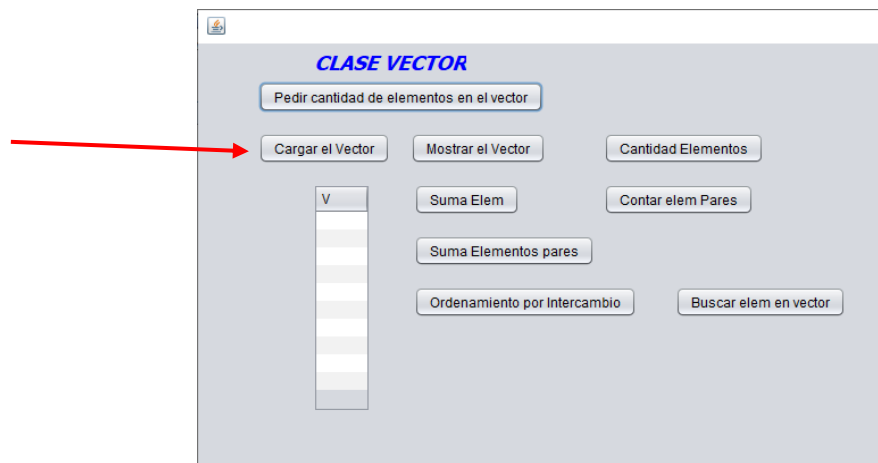
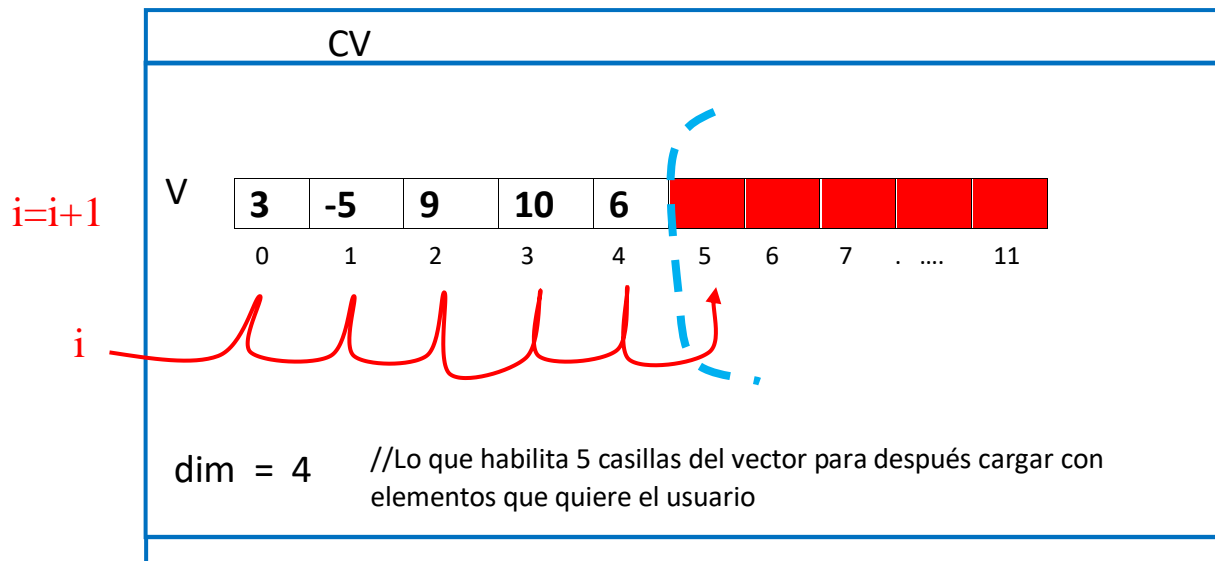
```
264 private void JBPedirDim2019ActionPerformed(java.awt.event.ActionEvent evt) {  
265     // TODO add your handling code here:  
266     int val;  
267     String cad=JOptionPane.showInputDialog("Num ele vec");  
268     val = Integer.parseInt(cad);  
269     cv.SetDim(val);  
270 }
```

En la class vector: `public void SetDim(int CantElem) {
 dim = CantElem - 1;
}`

Con esta asignación el valor de **dim** es CantElem -1



CARGAR ELEMENTOS PARA EL VECTOR



Entrada

V[0]=

3

Aceptar Cancelar

Entrada

V[1]=

-5

Aceptar Cancelar

Entrada

V[2]=

9

Aceptar Cancelar

Entrada

V[4]=

6

Aceptar Cancelar

Entrada

V[3]=

10

Aceptar Cancelar

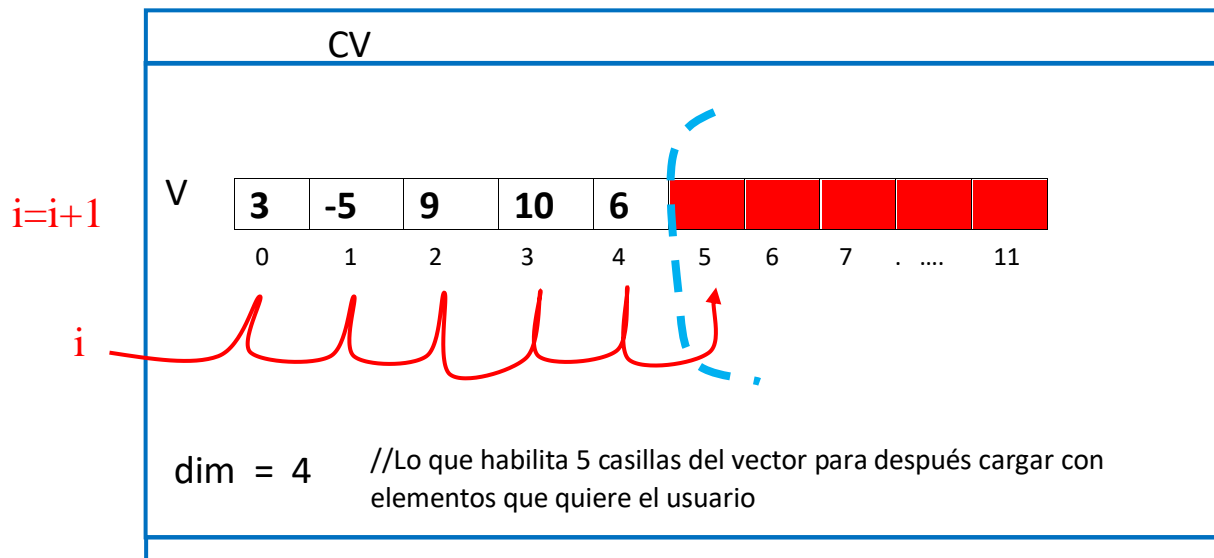
Cada elemento introducido se carga en la variable **cad** pero como es de tipo **String**, luego convertimos en **integer** y lo guardamos en la variable **valor**. En el SOURCE de JFRAME

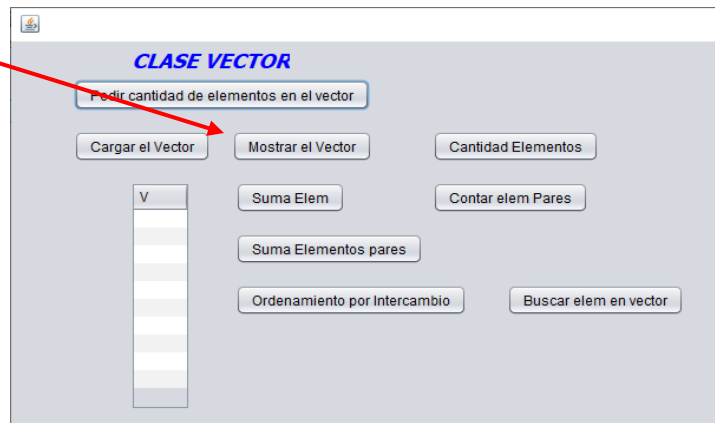
```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    cv.CargarVector();  
}
```

En la Clase Vector se implementa:

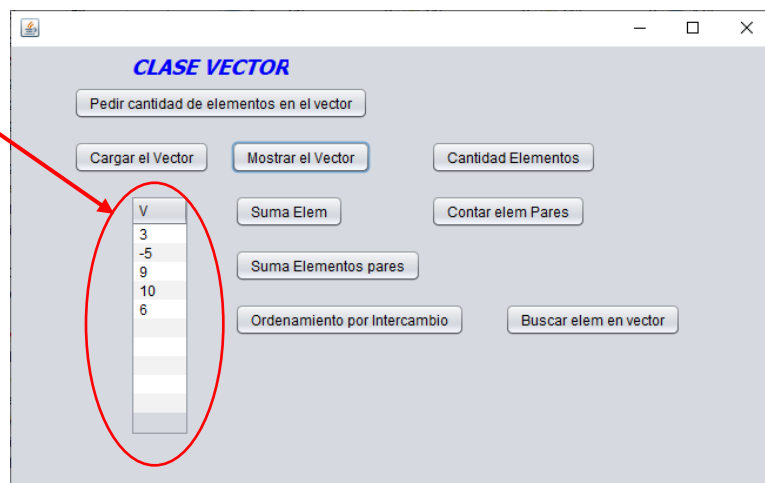
```
public void CargarVector()  
{  
    String Cad="";  
    int valor;  
    for(int i=0;i<=dim;i++)  
    {  
        Cad= JOptionPane.showInputDialog("valor para posicion "+i );  
        valor=Integer.parseInt(Cad);  
        v[i]=valor;  
  
        //this.SetValor(i,valor);  
    }  
}
```

MOSTRAR ELEMEN PARA EL VECTOR





Muestra en el objeto jTable1



El proceso de mostrar, saca los elementos de la estructura **V** definida como atributo en el objeto **CV** de tipo CVector, esto lo hace a través del código escrito para el botón **Mostrar el vector**.

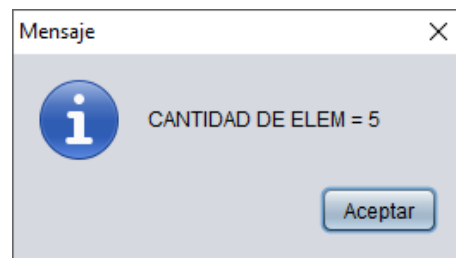
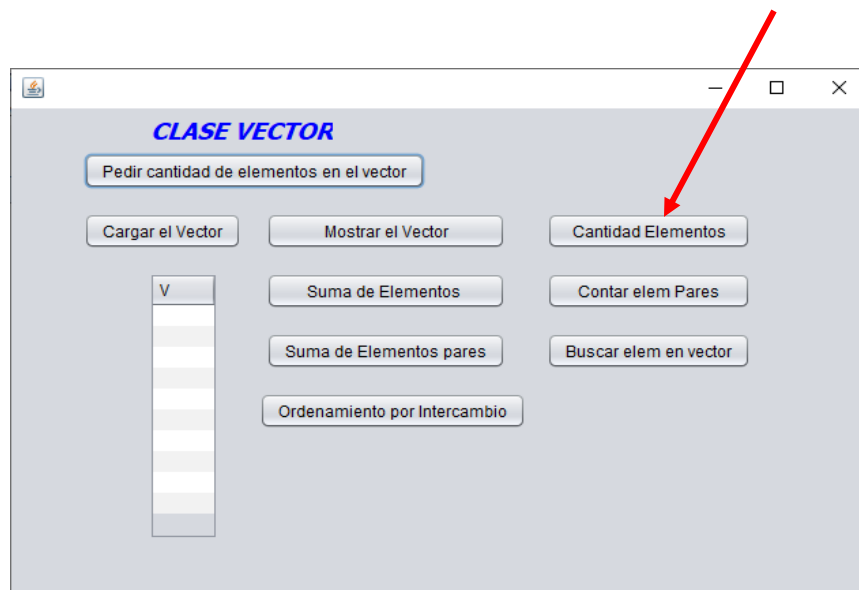
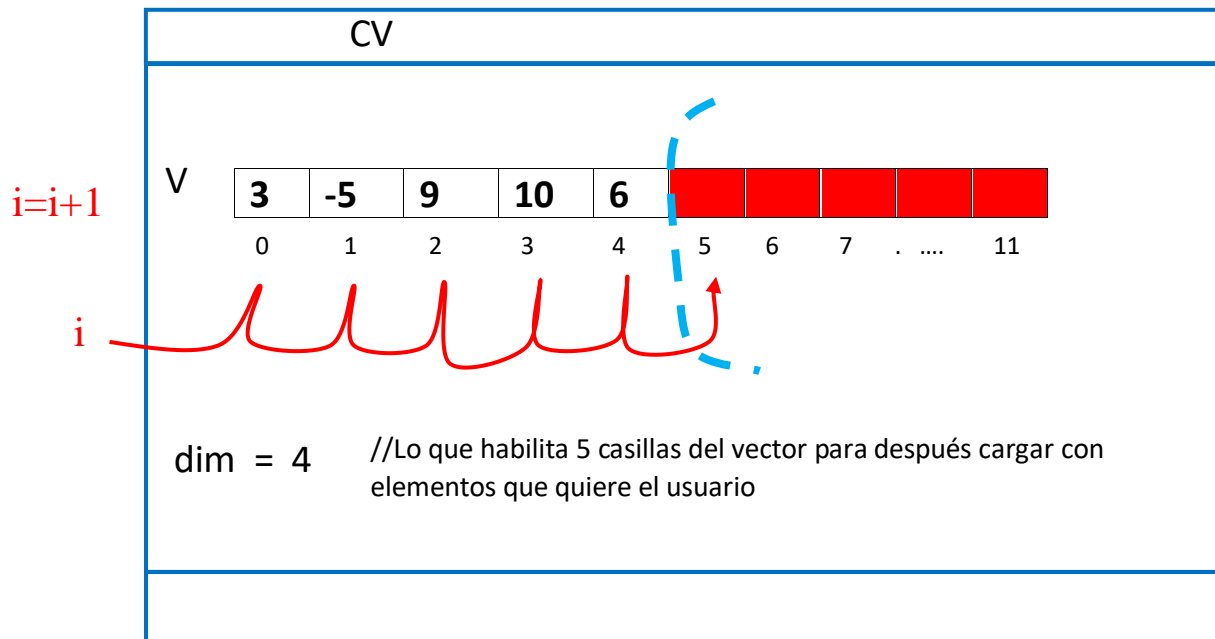
En el SOURCE de JFrame

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    cv.MostrarVector(JTVector);
}
```

En Clase Vector:

```
public void MostrarVector(JTable JTVector)
{
    //int valor;
    for(int i=0;i<=dim;i++)
    {
        //valor = V[i];
        JTVector.setValueAt(V[i], i, 0);
    }
}
```

CANTIDAD DE ELEMENTOS DEL VECTOR



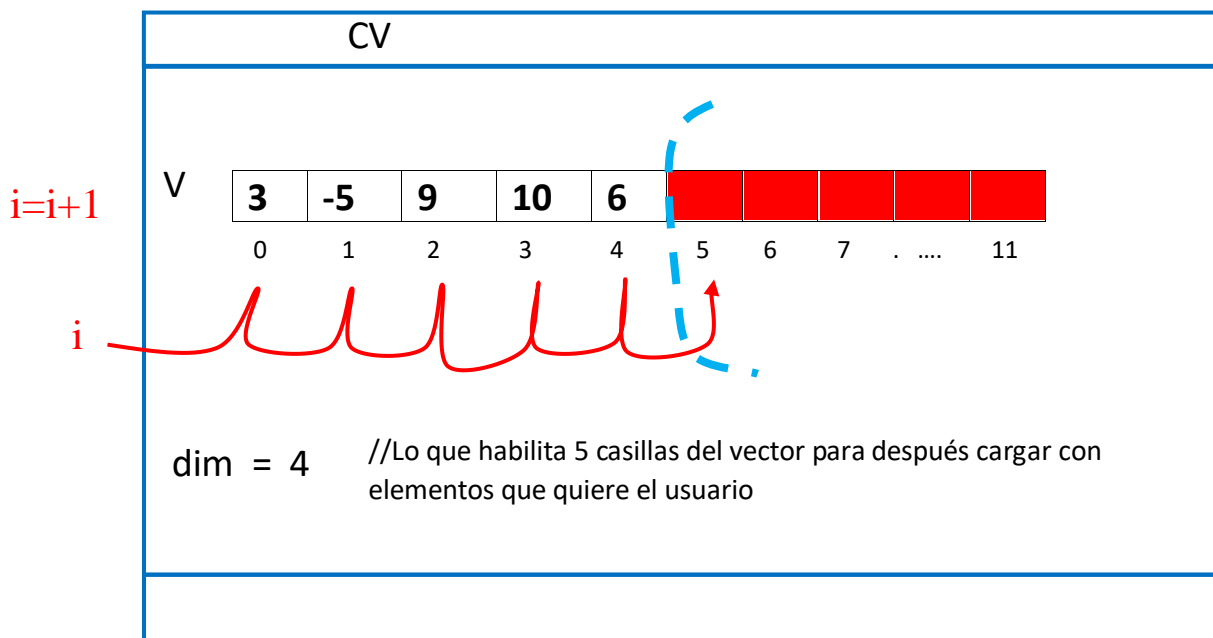
Source del JFrame

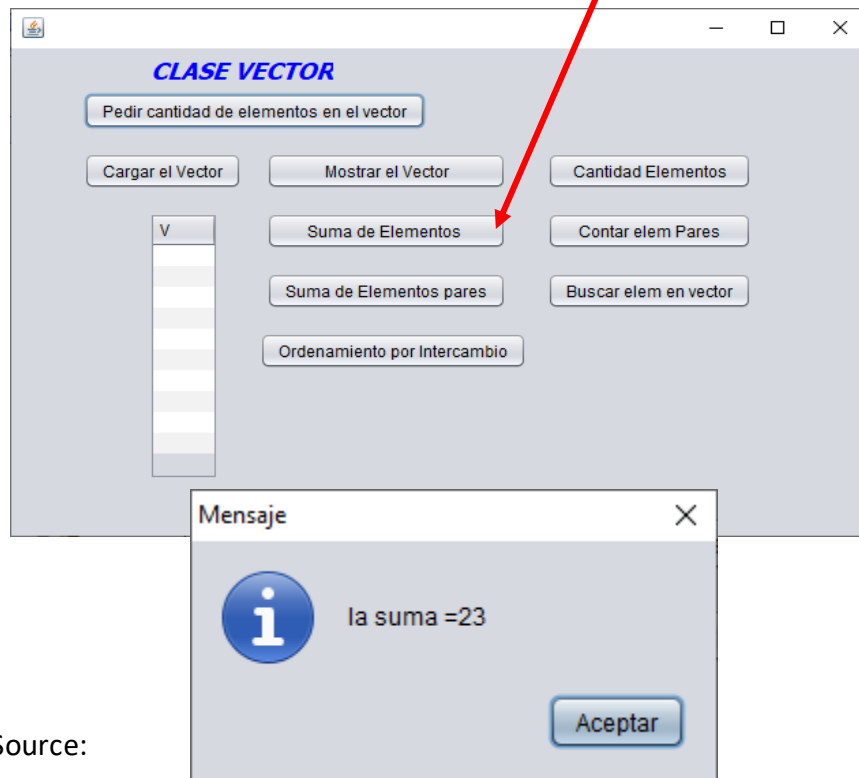
```
ans IDE 7.3
jar Source Regresar Run Debug Profile Team Herramientas Window Ayuda
<configuración prede...
...ave JFrameCNumero.java x CNumero.java x CString.java x JFString.java x JFVector.java x CVector.java x CapasPrueba.java x Main.java x C...
Source Design History
199
200 private void jBtCantidadElemActionPerformed(java.awt.event.ActionEvent evt) {
201     // TODO add your handling code here:
202     int aux =cv.Cantidad();
203     JOptionPane.showMessageDialog(rootPane,"CANTIDAD DE ELEM = "+aux);
204 }
205
```

Se llama a la función desde la Clase CVector:

```
proypruebacapas2 30
ProyString 31
ProyVector2019 32
Paquetes de fi 33
proyvector 34
CVect
JFVect
Main,j
public int Cantidad() {
    return (dim + 1);
}
```

SUMA DE ELEMENTOS DEL VECTOR





En el Source:

```

203
204 private void jButtonSumaActionPerformed(java.awt.event.ActionEvent evt) {
205     // TODO add your handling code here:
206     int aux = cv.Suma();
207     JOptionPane.showMessageDialog(rootPane, "la suma =" + aux);
208 }
209

```

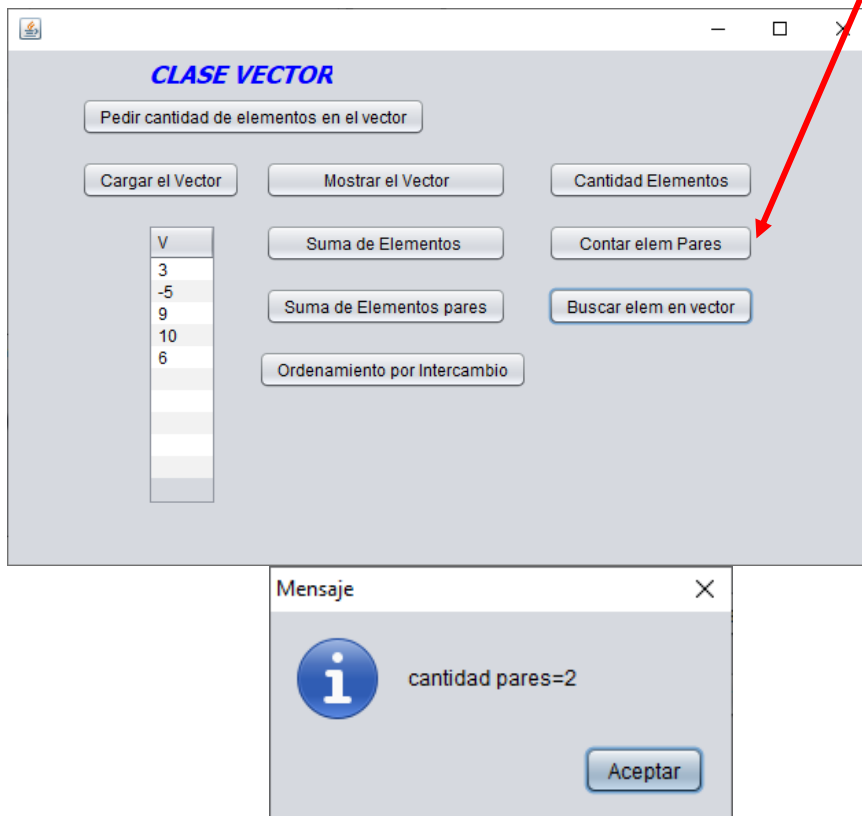
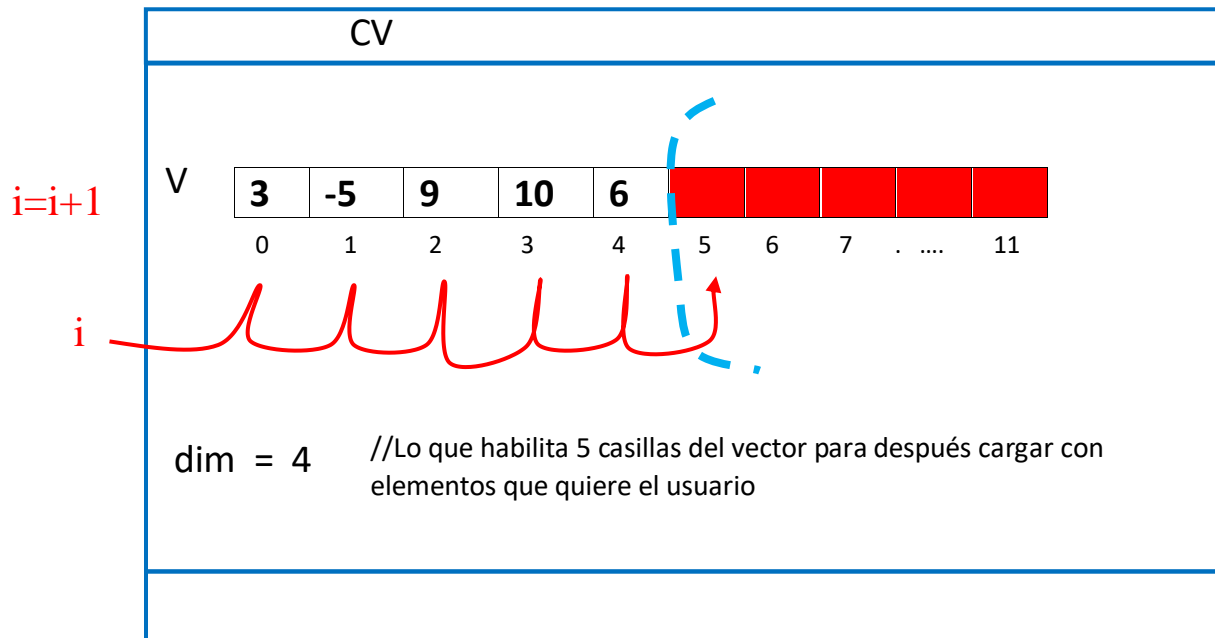
En la Clase:

```

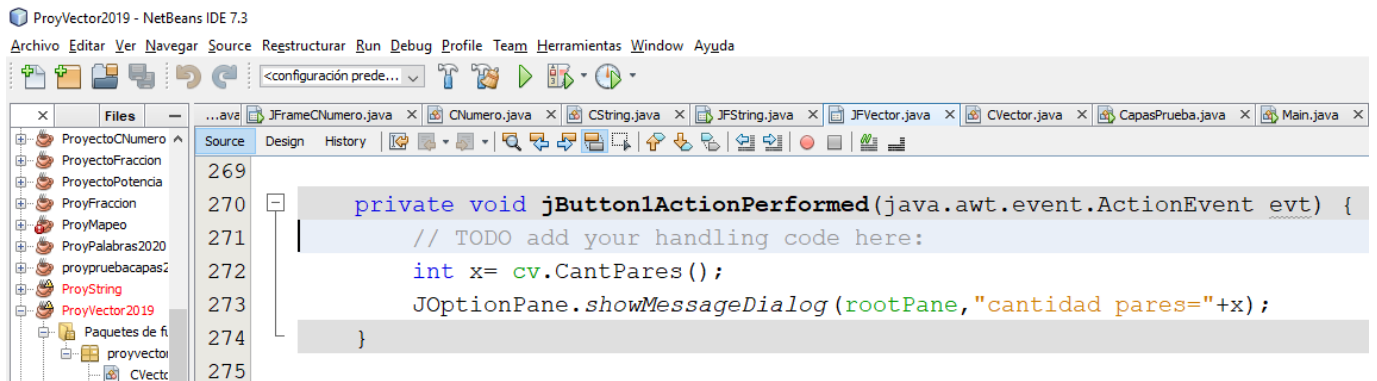
73 public int Suma() {
74     int s, valor;
75     s = 0;
76     for (int i = 0; i <= this.GetDim(); i++) {
77         valor = this.GetValor(i);
78         s = s + valor;
79     }
80     return s;
81 }
82

```

CANTIDAD DE ELEMENTOS PARES DEL VECTOR



En el Source:



ProyVector2019 - NetBeans IDE 7.3

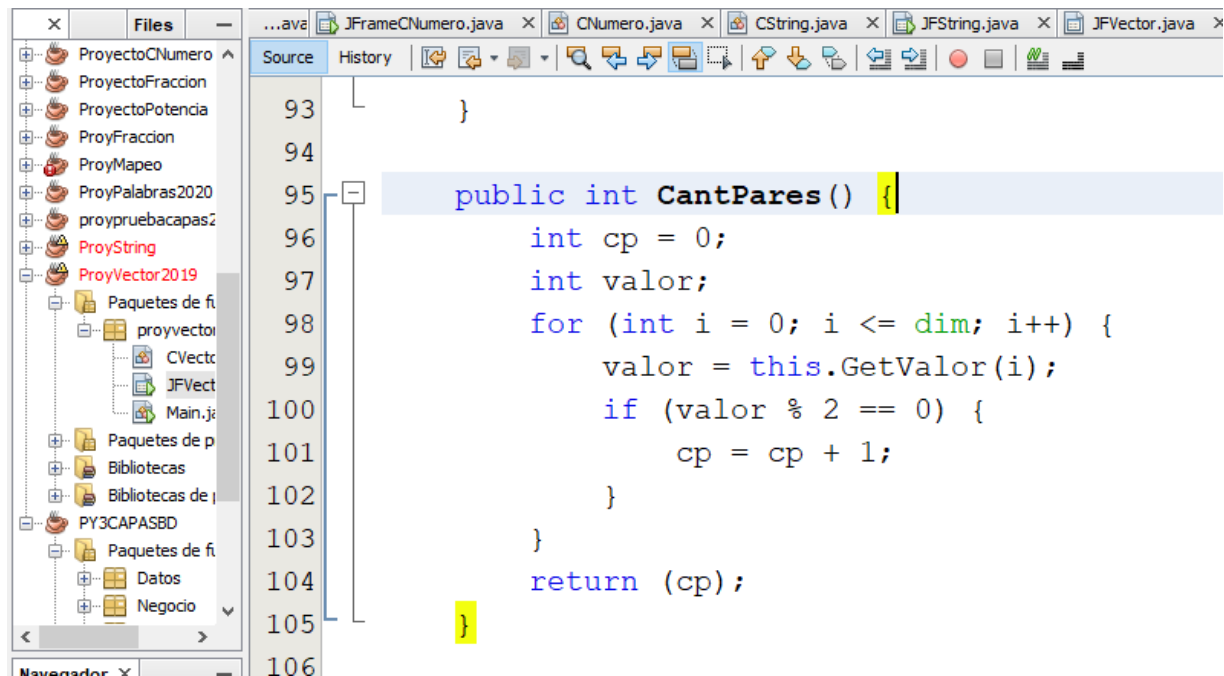
Archivo Editar Ver Navegar Source Regresar Run Debug Profile Team Herramientas Window Ayuda

Files

Source

```
269
270 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
271     // TODO add your handling code here:
272     int x= cv.CantPares();
273     JOptionPane.showMessageDialog(rootPane,"cantidad pares="+x);
274 }
275
```

En la Clase:

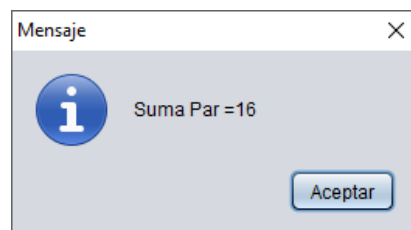
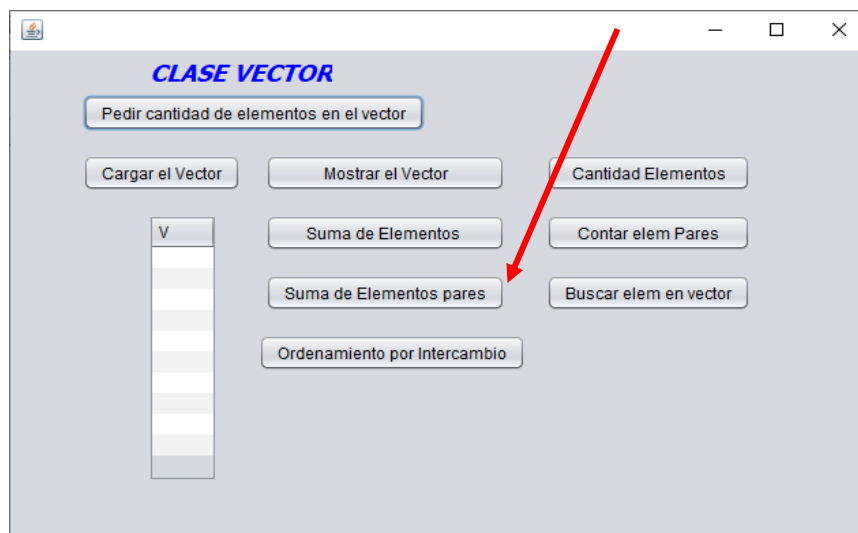
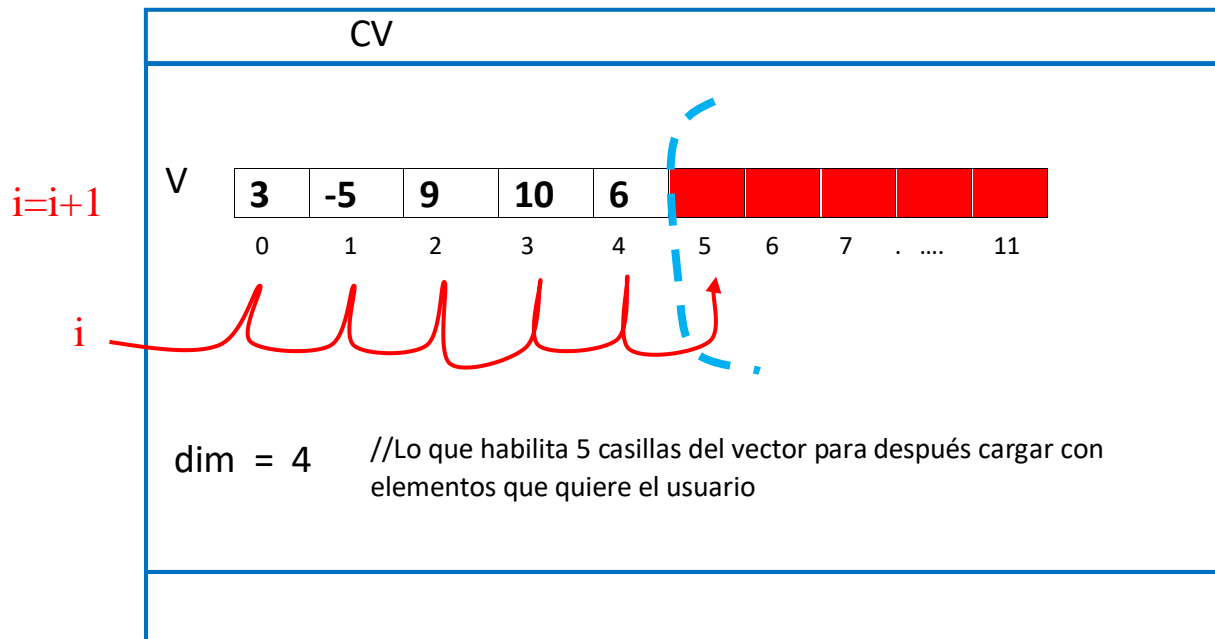


Files

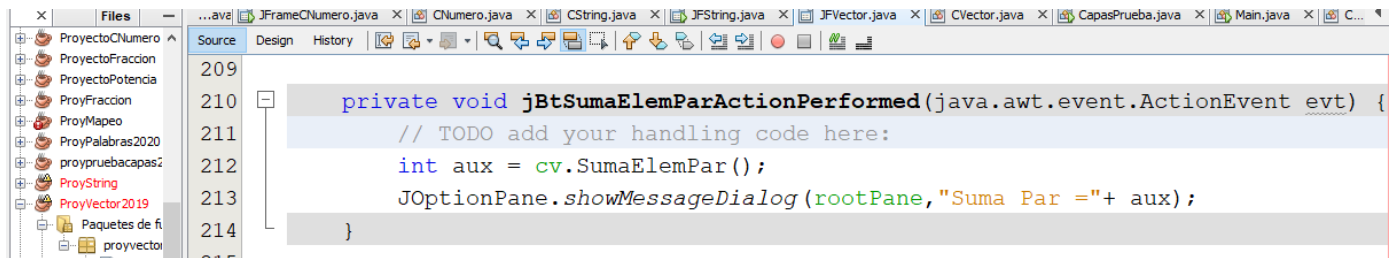
Source

```
93
94
95 public int CantPares() {
96     int cp = 0;
97     int valor;
98     for (int i = 0; i <= dim; i++) {
99         valor = this.GetValor(i);
100         if (valor % 2 == 0) {
101             cp = cp + 1;
102         }
103     }
104     return (cp);
105 }
106
```

SUMA DE ELEMENTOS PARES DEL VECTOR

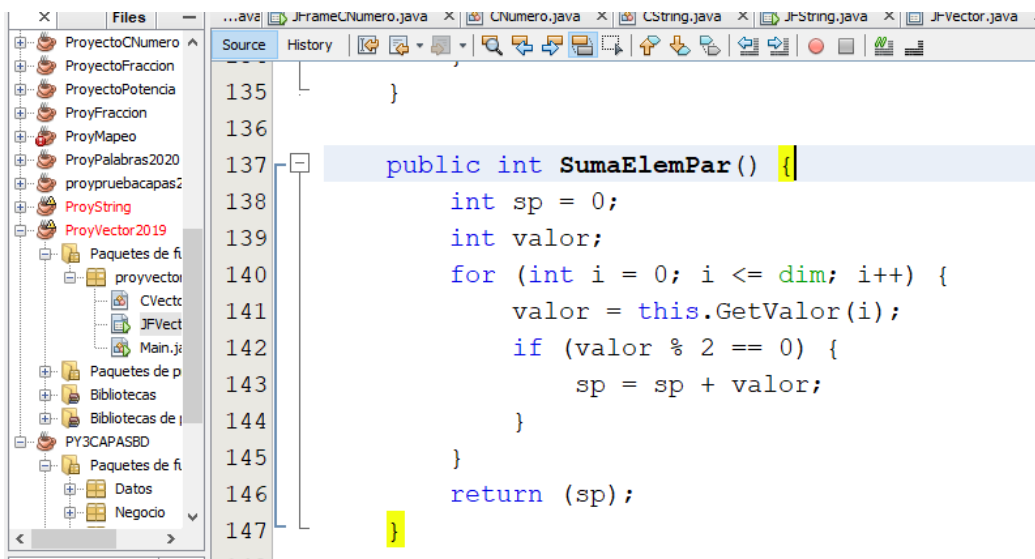


En el Source:



```
209 private void jBtSumaElemParActionPerformed(java.awt.event.ActionEvent evt) {  
210     // TODO add your handling code here:  
211     int aux = cv.SumaElemPar();  
212     JOptionPane.showMessageDialog(rootPane,"Suma Par =" + aux);  
213 }  
214
```

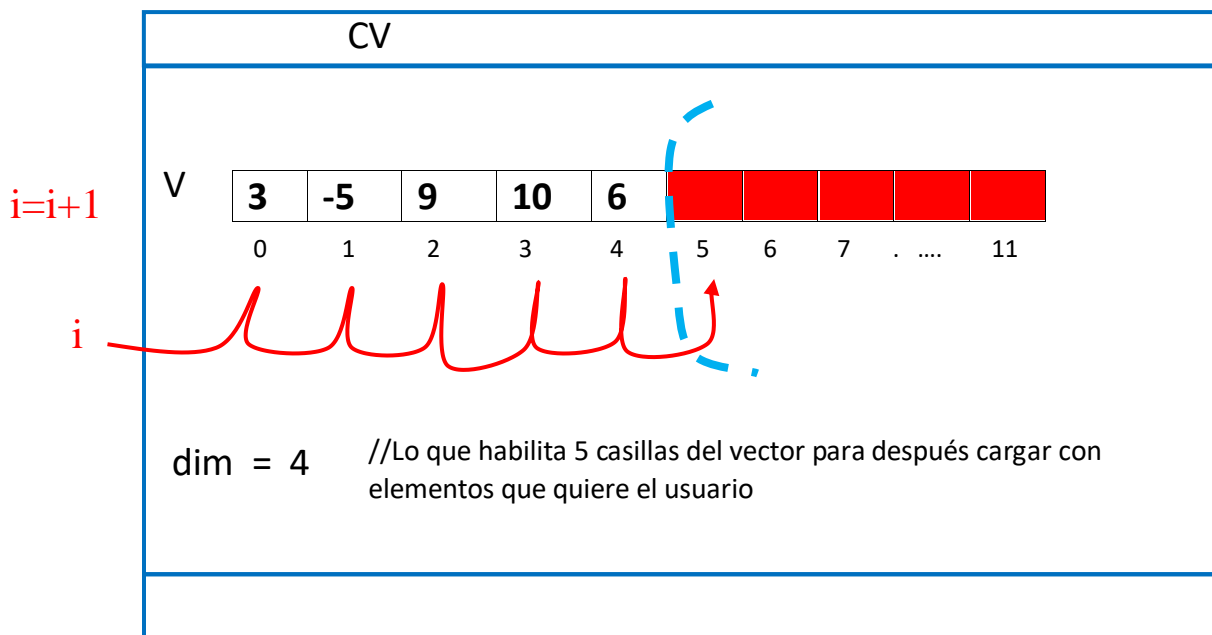
En la Clase:



```
135 }  
136  
137 public int SumaElemPar() {  
138     int sp = 0;  
139     int valor;  
140     for (int i = 0; i <= dim; i++) {  
141         valor = this.GetValor(i);  
142         if (valor % 2 == 0) {  
143             sp = sp + valor;  
144         }  
145     }  
146     return (sp);  
147 }
```

BUSCAR ELEMENTO EN EL VECTOR

1º Forma de Búsqueda Secuencial



CLASE VECTOR

Pedir cantidad de elementos en el vector


Cargar el Vector Mostrar el Vector Cantidad Elementos

Suma de Elementos Contar elem Pares

Suma de Elementos pares **Buscar elem en vector**

Ordenamiento por Intercambio

V
3
-5
9
10
6



Entrada

Que elem desea buscar ?

9

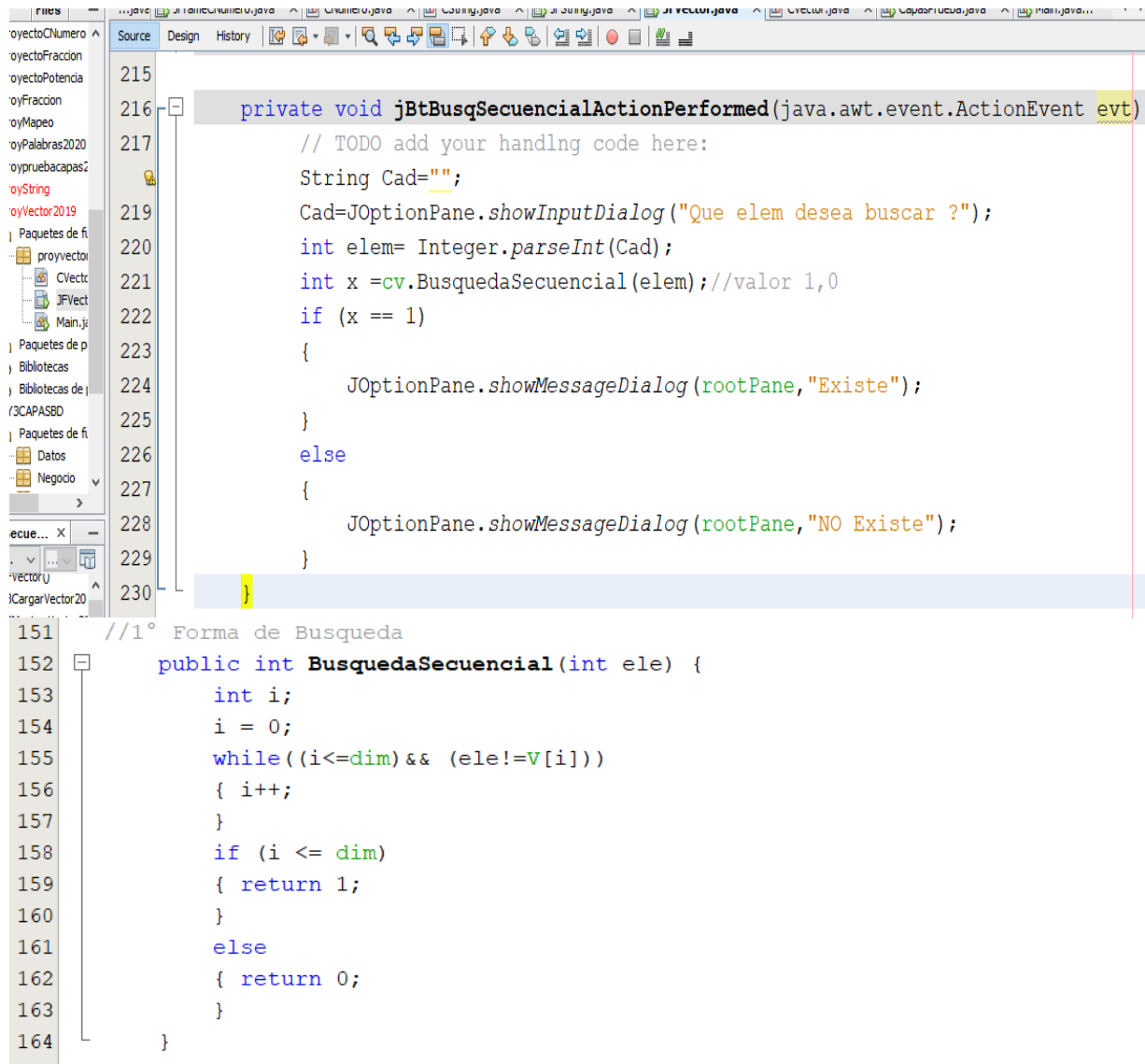
Aceptar Cancelar

Mensaje

Existe

Aceptar

En el Source:



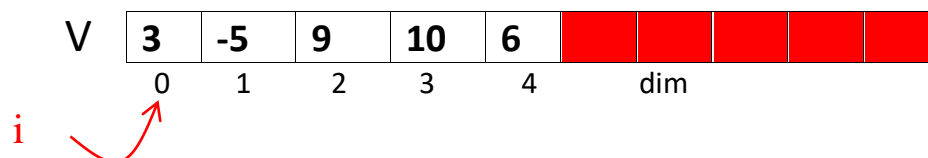
```
215
216 private void jBtBusqSecuencialActionPerformed(java.awt.event.ActionEvent evt)
217     // TODO add your handling code here:
218     String Cad="";
219     Cad=JOptionPane.showInputDialog("Que elem desea buscar ?");
220     int elem= Integer.parseInt(Cad);
221     int x =cv.BusquedaSecuencial(elem);//valor 1,0
222     if (x == 1)
223     {
224         JOptionPane.showMessageDialog(rootPane,"Existe");
225     }
226     else
227     {
228         JOptionPane.showMessageDialog(rootPane,"NO Existe");
229     }
230 }

151 //1° Forma de Busqueda
152 public int BusquedaSecuencial(int ele) {
153     int i;
154     i = 0;
155     while((i<=dim) && (ele!=v[i]))
156     { i++;
157     }
158     if (i <= dim)
159     { return 1;
160     }
161     else
162     { return 0;
163     }
164 }
```

En la Clase:

ele =3

Si se encuentra en la primera posición:



Si el elemento a buscar es **ele = 9**

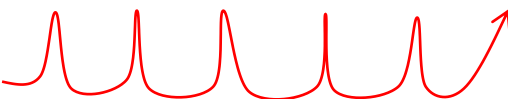
V	3	-5	9	10	6
	0	1	2	3	4

i 

```
public int BusquedaSecuencial(int ele) {  
    int i;  
    i = 0;  
    while((i<=dim) && (ele!=V[i]))  
    { i++;  
    }  
    if (i <= dim)  
    { return 1;  
    }  
    else  
    { return 0;  
    }  
}
```

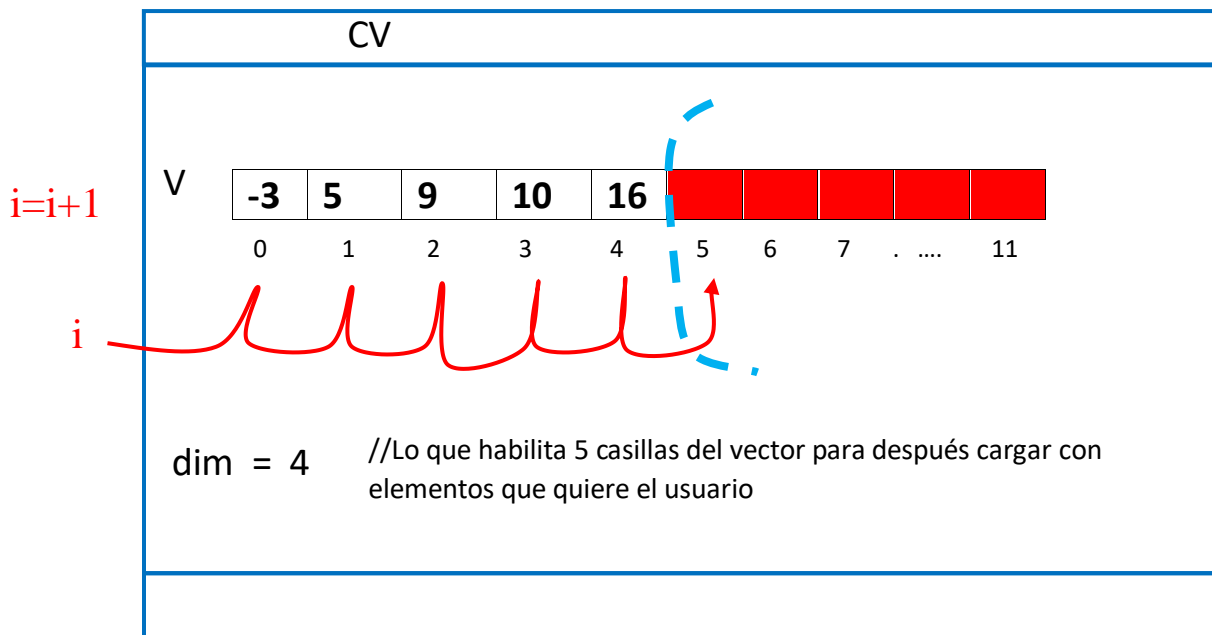
Si el elemento a buscar no existe **ele = 11**

V	3	-5	9	10	6
	0	1	2	3	4

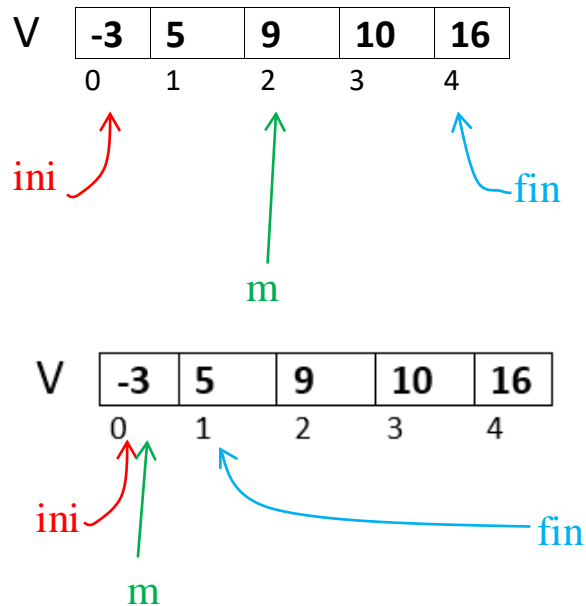
i 

```
public int BusquedaSecuencial(int ele) {  
    int i;  
    i = 0;  
    while((i<=dim) && (ele!=V[i]))  
    { i++;  
    }  
    if (i <= dim)  
    { return 1;  
    }  
    else  
    { return 0;  
    }  
}
```

2° Forma de Búsqueda Binaria



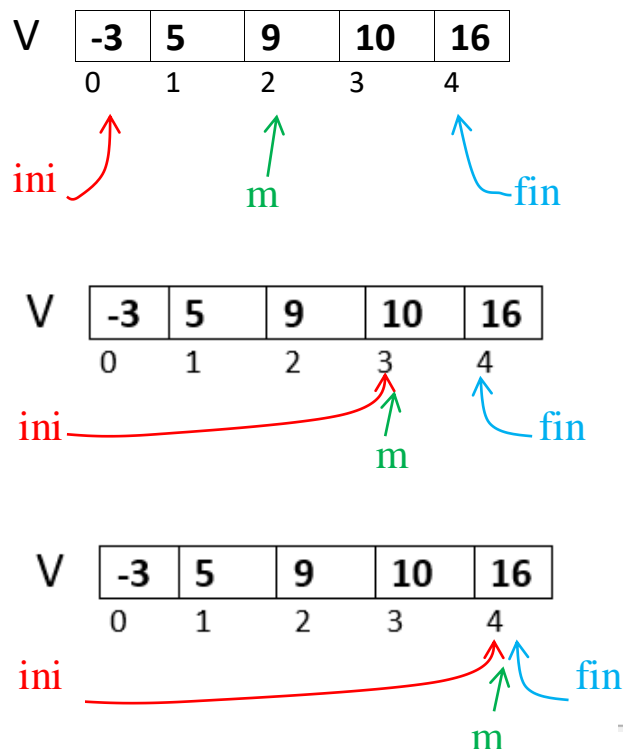
Si el elemento a buscar es **Ele = -3**



```
//2° Forma de Búsqueda: BINARIA los elem
// deben estar |ordenados ASC O DESC
// EN ESTE CASO ESTAN ORDENADOS ASC
```

```
public int BusqBinaria(int Ele) {
    int ini, fin, m;
    //fin = this.GetDim();
    fin = dim;
    ini = 0;
    while (ini <= fin) {
        m = (ini + fin) / 2;
        if (Ele == V[m])
            { return 1; }
        else
            {
                if (Ele < V[m])
                    { fin = m - 1; }
                else
                    { ini = m + 1; }
            }
    }
    if (ini > fin)
        { return 0; }
    else
        { return 1; }
}
```

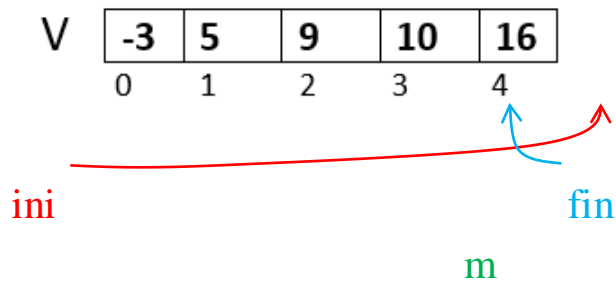
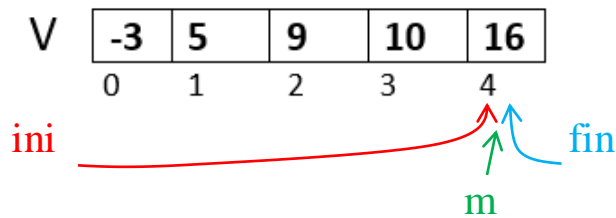
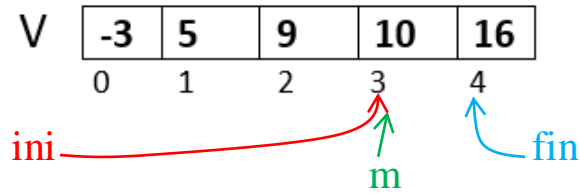
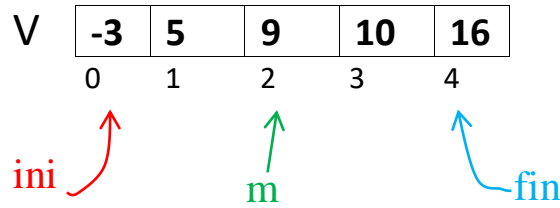
Si el elemento a buscar es **Ele = 16**



```
//2° Forma de Búsqueda: BINARIA los elem
// deben estar |ordenados ASC O DESC
// EN ESTE CASO ESTAN ORDENADOS ASC
```

```
public int BusqBinaria(int Ele) {
    int ini, fin, m;
    //fin = this.GetDim();
    fin = dim;
    ini = 0;
    while (ini <= fin) {
        m = (ini + fin) / 2;
        if (Ele == V[m])
            { return 1; }
        else
            {
                if (Ele < V[m])
                    { fin = m - 1; }
                else
                    { ini = m + 1; }
            }
    }
    if (ini > fin)
        { return 0; }
    else
        { return 1; }
}
```

Si el elemento a buscar NO EXISTE Ele = 27



//2° Forma de Búsqueda: BINARIA los elem
// deben estar |rdenados ASC O DESC
// EN ESTE CASO ESTAN ORDENADOS ASC

```
public int BusqBinaria(int Ele) {
    int ini, fin, m;
    //fin = this.GetDim();
    fin = dim;
    ini = 0;
    while (ini <= fin) {
        m = (ini + fin) / 2;
        if (Ele == v[m])
            { return 1; }
        else
            { if (Ele < v[m])
                { fin = m - 1; }
              else
                { ini = m + 1; }
            }
    }
    if (ini > fin)
        { return 0; }
    else
        { return 1; }
}
```

Cuando ini es mayor que fin entonces no existe el elemento

ORDENAR ELEMENTOS EN EL VECTOR (METODO DE INTERCAMBIO)

CV

$i = i + 1$

V

3	-5	9	10	6							
0	1	2	3	4	5	6	7	11	

i

$\text{dim} = 4$ //Lo que habilita 5 casillas del vector para después cargar con elementos que quiere el usuario

CLASE VECTOR

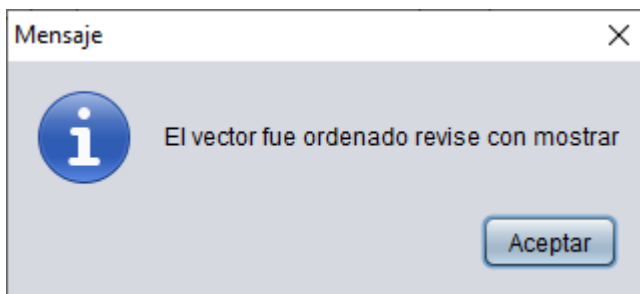
Pedir cantidad de elementos en el vector

Cargar el Vector Mostrar el Vector Cantidad Elementos

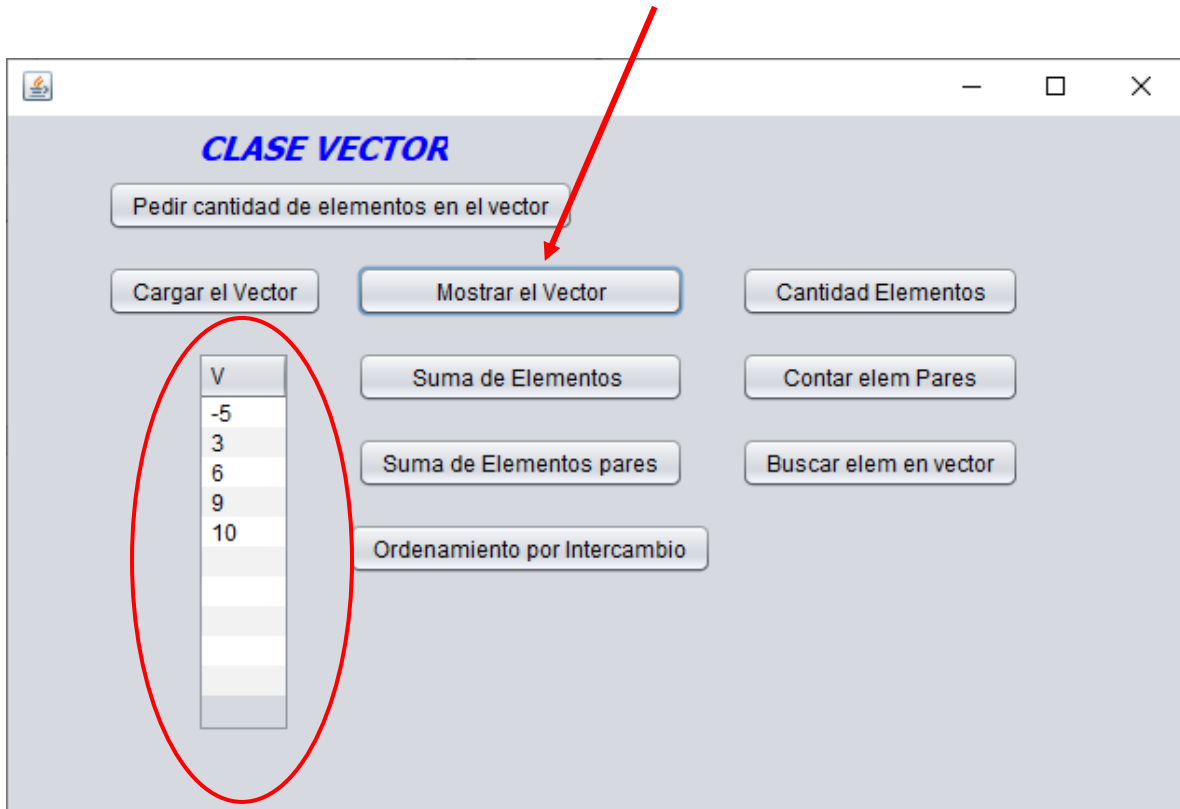
Suma de Elementos Contar elem Pares

Suma de Elementos pares Buscar elem en vector

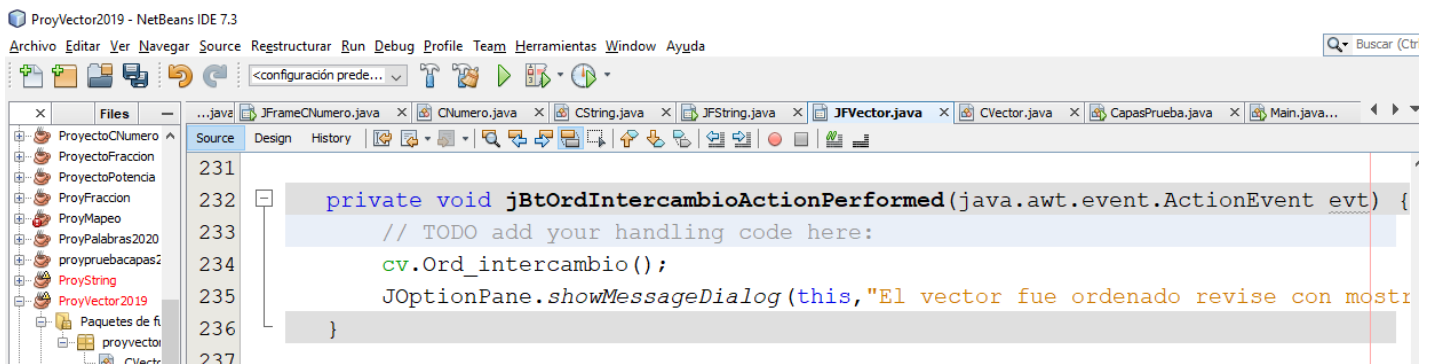
Ordenamiento por Intercambio



Mostrar el vector para actualizar por el ordenamiento:



En el Source:



En la Clase:

```

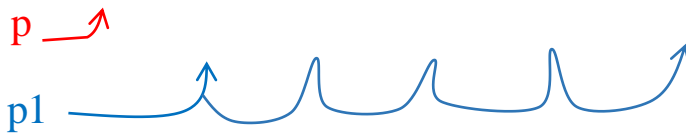
205 // 1° Metodo de Ordenamiento
206 public void Ord_intercambio() {
207     int aux;
208     for (int p = 0; p <= (dim- 1); p++)
209     {
210         for (int p1 = p + 1; p1 <= (this.GetDim()); p1++)
211         {
212             if (V[p1] < V[p]) {
213                 aux = V[p1];
214                 V[p1] = V[p];
215                 V[p] = aux;
216             }
217         }
218     }
219 }
    
```

INTERCAMBIO ASCENDENTE

ordenamientos

V

3	15	-9	7	10
0	1	2	3	4



V

-9	15	3	7	10
0	1	2	3	4



V

-9	3	15	7	10
0	1	2	3	4



```

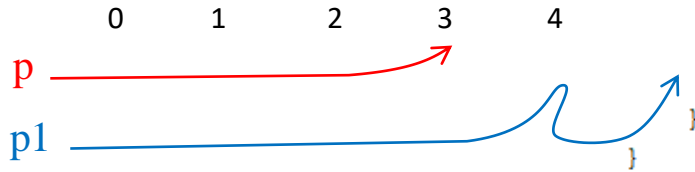
public void Ord_intercambio() {
    int aux;
    for (int p = 0; p <= (dim- 1); p++)
    {
        for (int p1 = p + 1; p1 <= (dim); p1++)
        {
            if (V[p1] < V[p]) {
                aux = V[p1];
                V[p1] = V[p];
                V[p] = aux;
            }
        }
    }
}
    
```

```

public void Ord_intercambio() {
    int aux;
    for (int p = 0; p <= (dim- 1); p++)
    {
        for (int p1 = p + 1; p1 <= (dim); p1++)
        {
            if (V[p1] < V[p]) {
                aux = V[p1];
                V[p1] = V[p];
                V[p] = aux;
            }
        }
    }
}
    
```


V

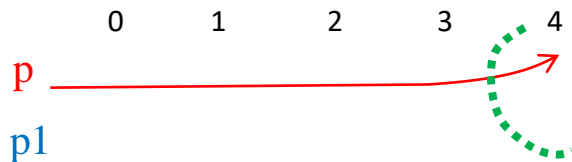
-9	3	7	15	10
----	---	---	----	----



```
public void Ord_intercambio() {
    int aux;
    for (int p = 0; p <= (dim- 1); p++)
    { for (int p1 = p + 1; p1 <= (dim); p1++)
        { if (V[p1] < V[p]) {
            aux = V[p1];
            V[p1] = V[p];
            V[p] = aux;
        }
    }
}
```

V

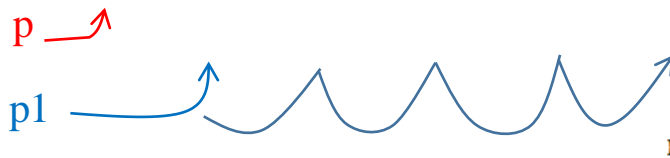
-9	3	7	10	15
----	---	---	----	----



```
public void Ord_intercambio() {
    int aux;
    for (int p = 0; p <= (dim- 1); p++)
    { for (int p1 = p + 1; p1 <= (dim); p1++)
        { if (V[p1] < V[p]) {
            aux = V[p1];
            V[p1] = V[p];
            V[p] = aux;
        }
    }
}
```

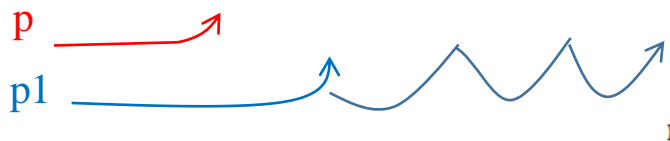
INTERCAMBIO DESCENDENTE

V	3	15	-9	7	10
	0	1	2	3	4



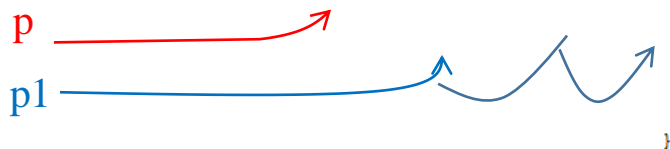
```
public void Ord_intercambioDesc() {
    int aux;
    for (int p = 0; p <= (this.GetDim() - 1); p++) {
        for (int pl = p + 1; pl <= (this.GetDim()); pl++) {
            if (V[pl] > V[p]) {
                aux = V[pl];
                V[pl] = V[p];
                V[p] = aux;
            }
        }
    }
}
```

V	15	10	-9	3	7
	0	1	2	3	4

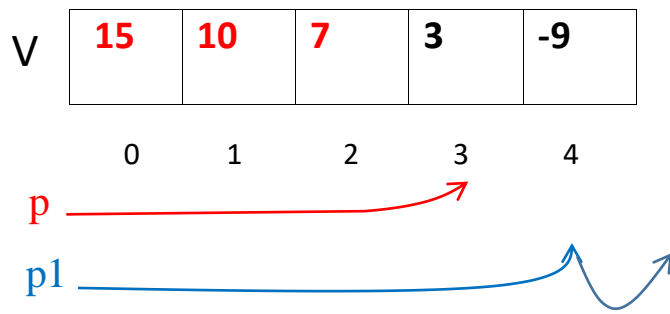


```
public void Ord_intercambioDesc() {
    int aux;
    for (int p = 0; p <= (this.GetDim() - 1); p++) {
        for (int pl = p + 1; pl <= (this.GetDim()); pl++) {
            if (V[pl] > V[p]) {
                aux = V[pl];
                V[pl] = V[p];
                V[p] = aux;
            }
        }
    }
}
```

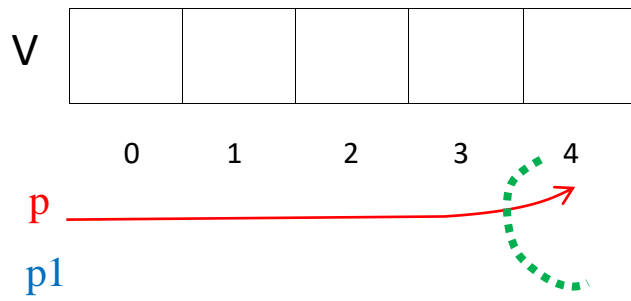
V	15	10	7	-9	3
	0	1	2	3	4



```
public void Ord_intercambioDesc() {
    int aux;
    for (int p = 0; p <= (this.GetDim() - 1); p++) {
        for (int pl = p + 1; pl <= (this.GetDim()); pl++) {
            if (V[pl] > V[p]) {
                aux = V[pl];
                V[pl] = V[p];
                V[p] = aux;
            }
        }
    }
}
```



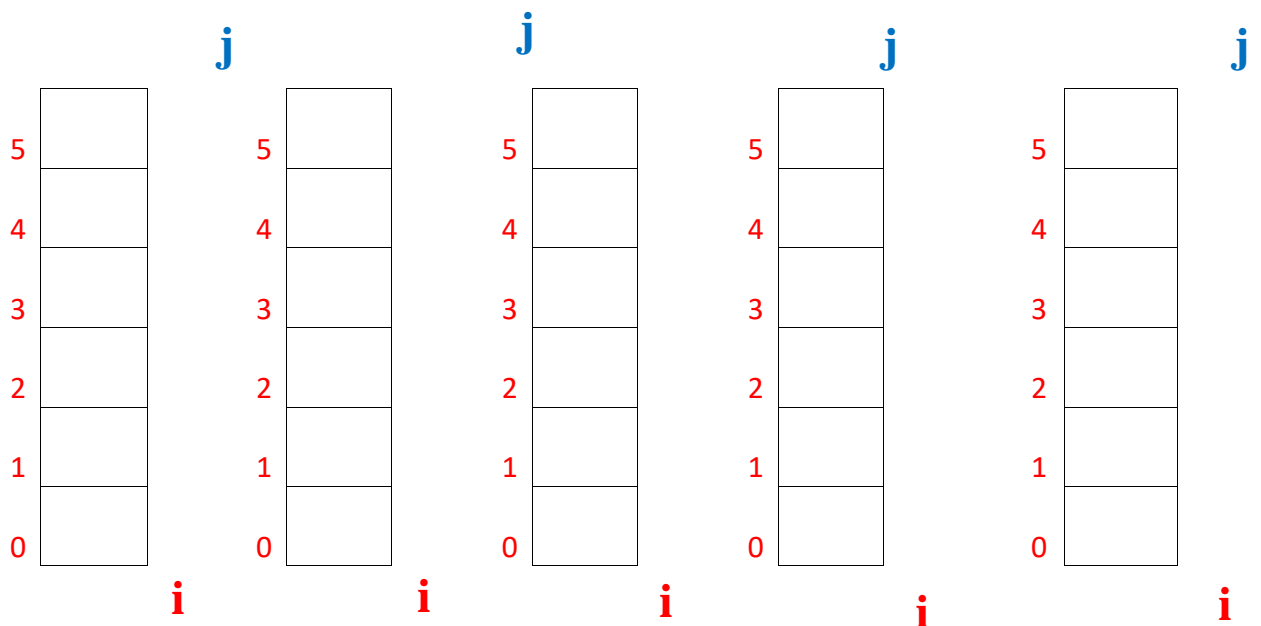
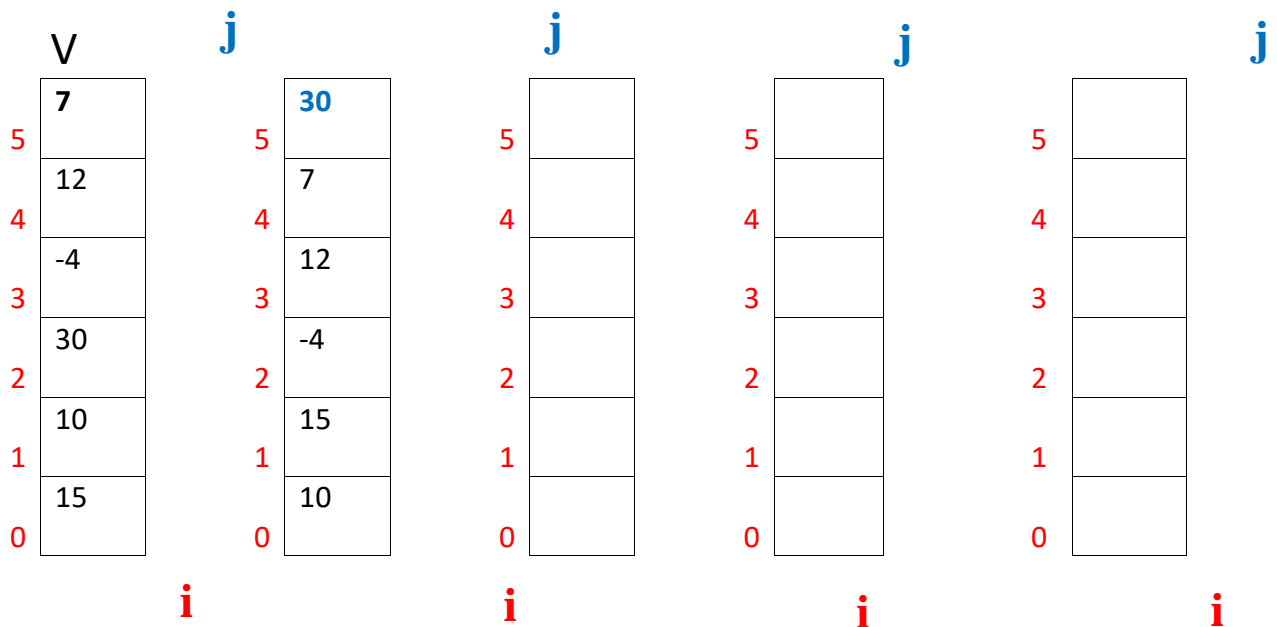
```
public void Ord_intercambioDesc() {
    int aux;
    for (int p = 0; p <= (this.GetDim() - 1); p++) {
        for (int p1 = p + 1; p1 <= (this.GetDim()); p1++)
            if (V[p1] > V[p]) {
                aux = V[p1];
                V[p1] = V[p];
                V[p] = aux;
            }
        }
    }
}
```



Metodo de Ordenamiento

ORDENAMIENTO BURBUJA

```
public void Ord_Burbuja() {
    int aux;
    for (int j = dim; j >= 1; j--) {
        for (int i = 0; i <= j - 1; i++) {
            if (V[i] > V[i + 1]) {
                aux = V[i];
                V[i] = V[i + 1];
                V[i + 1] = aux;
            }
        }
    }
}
```



ORDENAMIENTO SELECCIÓN

V	3	15	-9	7	10
---	---	----	----	---	----

i 

k 

j 

--	--	--	--	--

i 

k 

j 

--	--	--	--	--

i 

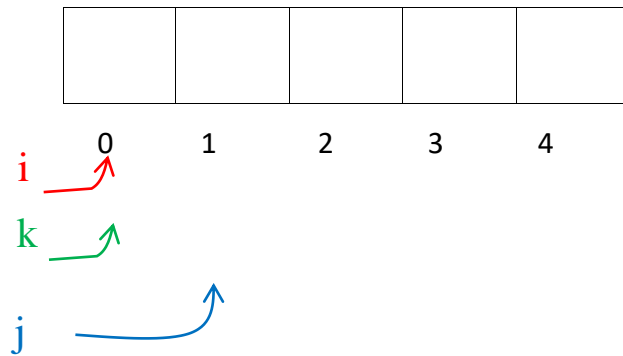
k 

j 

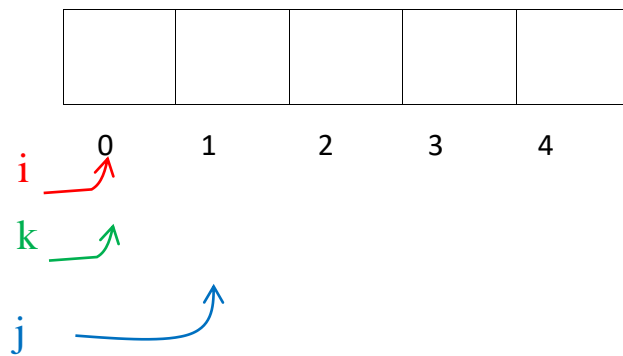
```
public void Ord_Seleccion() {
    int aux, k;
    for (int i = 0; i <= (this.GetDim() - 1); i++)
    { k = i;
        for (int j = k + 1; j <= this.GetDim(); j++)
        { if (V[j] < V[k])
            { k = j;
            }
        }
        if (k != i)
        { aux = V[i];
          V[i] = V[k];
          V[k] = aux;
        }
    }
}
```

```
public void Ord_Seleccion() {
    int aux, k;
    for (int i = 0; i <= (this.GetDim() - 1); i++)
    { k = i;
        for (int j = k + 1; j <= this.GetDim(); j++)
        { if (V[j] < V[k])
            { k = j;
            }
        }
        if (k != i)
        { aux = V[i];
          V[i] = V[k];
          V[k] = aux;
        }
    }
}
```

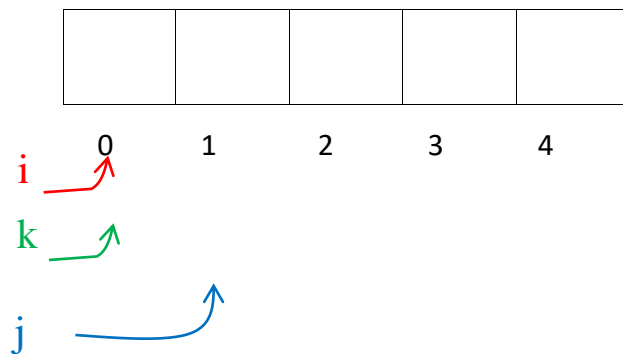
```
public void Ord_Seleccion() {
    int aux, k;
    for (int i = 0; i <= (this.GetDim() - 1); i++)
    { k = i;
        for (int j = k + 1; j <= this.GetDim(); j++)
        { if (V[j] < V[k])
            { k = j;
            }
        }
        if (k != i)
        { aux = V[i];
          V[i] = V[k];
          V[k] = aux;
        }
    }
}
```



```
public void Ord_Seleccion() {
    int aux, k;
    for (int i = 0; i <= (this.GetDim() - 1); i++)
    { k = i;
      for (int j = k + 1; j <= this.GetDim(); j++)
      { if (V[j] < V[k])
        { k = j;
        }
      }
      if (k != i)
      { aux = V[i];
        V[i] = V[k];
        V[k] = aux;
      }
    }
}
```



```
public void Ord_Seleccion() {
    int aux, k;
    for (int i = 0; i <= (this.GetDim() - 1); i++)
    { k = i;
      for (int j = k + 1; j <= this.GetDim(); j++)
      { if (V[j] < V[k])
        { k = j;
        }
      }
      if (k != i)
      { aux = V[i];
        V[i] = V[k];
        V[k] = aux;
      }
    }
}
```



```
public void Ord_Seleccion() {
    int aux, k;
    for (int i = 0; i <= (this.GetDim() - 1); i++)
    { k = i;
      for (int j = k + 1; j <= this.GetDim(); j++)
      { if (V[j] < V[k])
        { k = j;
        }
      }
      if (k != i)
      { aux = V[i];
        V[i] = V[k];
        V[k] = aux;
      }
    }
}
```

--	--	--	--	--

i → 0
k →
j →

```

public void Ord_Seleccion() {
    int aux, k;
    for (int i = 0; i <= (this.GetDim() - 1); i++)
    {
        k = i;
        for (int j = k + 1; j <= this.GetDim(); j++)
        {
            if (V[j] < V[k])
            {
                k = j;
            }
        }
        if (k != i)
        {
            aux = V[i];
            V[i] = V[k];
            V[k] = aux;
        }
    }
}
  
```

V

34	-4	8	5	10	7	45	-3	6	5
-----------	-----------	----------	----------	-----------	----------	-----------	-----------	----------	----------

```

public void Ord_intercambioDesc() {
    int aux;
    for (int p = 0; p <= (this.GetDim() - 1); p++) {
        for (int pl = p + 1; pl <= (this.GetDim()); pl++) {
            if (V[pl] > V[p]) {
                aux = V[pl];
                V[pl] = V[p];
                V[p] = aux;
            }
        }
    }
}
  
```

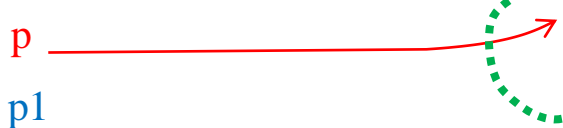
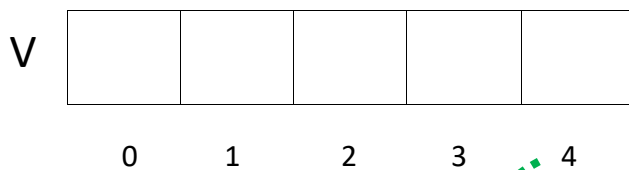
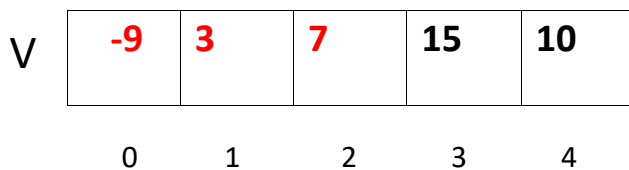
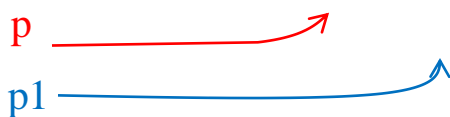
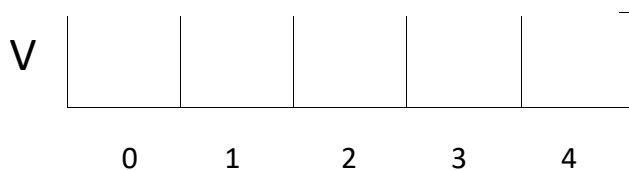
```

' Metodo de Ordenamiento
public void Ord_Burbuja() {
    int aux;
    for (int j = dim; j >= 1; j--) {
        for (int i = 0; i <= j - 1; i++) {
            if (V[i] > V[i + 1]) {
                aux = V[i];
                V[i] = V[i + 1];
                V[i + 1] = aux;
            }
        }
    }
}
  
```

```

public void Ord_intercambioDesc() {
    int aux;
    for (int p = 0; p <= (this.GetDim() - 1); p++) {
        for (int pl = p + 1; pl <= (this.GetDim()); pl++) {
            if (V[pl] > V[p]) {
                aux = V[pl];
                V[pl] = V[p];
                V[p] = aux;
            }
        }
    }
}

```



```

public void Ord_Seleccion() {
    int aux, k;
    for (int i = 0; i <= (this.GetDim() - 1); i++)
    { k = i;
      for (int j = k + 1; j <= this.GetDim(); j++)
      { if (V[j] < V[k])
        { k = j;
        }
      }
      if (k != i)
      { aux = V[i];
        V[i] = V[k];
        V[k] = aux;
      }
    }
}

```


CORTE DE CONTROL

Este algoritmo funciona solo si el VECTOR esta ORDENADO

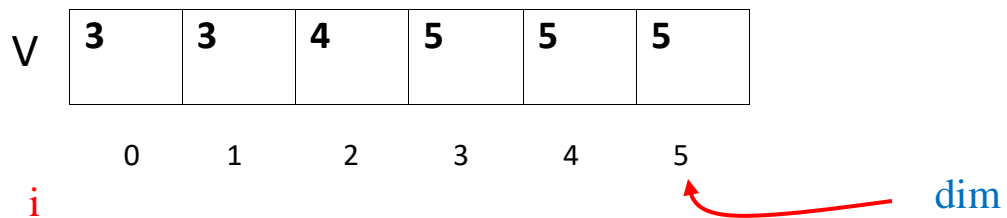
```
public void CorteControl() {  
    int i = 0;  
    int Aux, c;  
    while (i <= dim) {  
        Aux = V[i];  
        c = 0;  
        while ((i <= dim) && (Aux == V[i])) {  
            c++;  
            i++;  
        }  
        JOptionPane.showMessageDialog  
            (null, "elemento " + Aux + " Se repite " + c);  
    }  
}
```

V	3	3	4	5	5	5
	0	1	2	3	4	5
i						dim

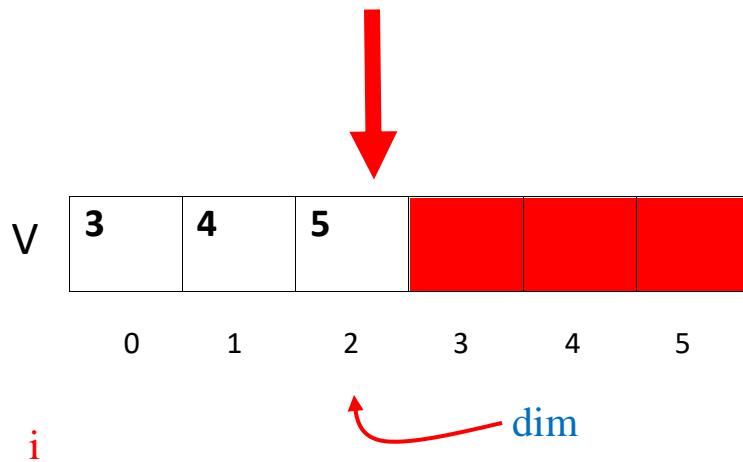
```

public void Purga() {
    int i, A, j, c;
    i = 0;
    j = -1;
    int n = dim;
    while (i <= n) {
        A = v[i];
        while ((i <= n) && (A == v[i])) {
            i++;
        }
        j++;
        v[j] = A;
    }
    dim=j;
}

```



El resultado de la purga será:



V

3	3	4	5	5	5
----------	----------	----------	----------	----------	----------

0 1 2 3 4 5

i

dim

V

3	3	4	5	5	5
----------	----------	----------	----------	----------	----------

0 1 2 3 4 5

i

dim