

UNIVERSIDAD DE CORDOBA

FACULTAD DE INGENIERIAS

PROGRAMA INGENIERIA DE SISTEMAS

CURSO: Programación II

TEMA: Ordenamiento de arreglos por el método de selección.

DESCRIPCION: A lo largo de este documento se presenta la explicación del método de ordenamiento de selección, aplicándolo al ordenamiento de arreglos unidimensionales o vectores, considerando en primera instancia su descripción textual y en segunda instancia se complementa con la gráfica del funcionamiento de este método, paso a paso y de manera muy detallada. De esta manera, se presenta de manera visual un arreglo con su respectivo contenido, señalando en cada paso del método el estado del arreglo y las operaciones de comparación e intercambio que el método de selección efectúa para ordenar el arreglo; de modo que se muestran las parejas de elementos correspondiente con la posición de cada menor encontrado y la posición de inicio de su búsqueda, que es en esencia el fundamento del ordenamiento por selección. En cuanto al ejemplo práctico de demostración de este método, se usará el lenguaje Java con una aplicación de ventana hecha en el IDE *NetBeans* y basada en un enunciado textual y su respectivo diseño de clases UML.

OBJETIVO: Diseñar e implementar una aplicación de ventana con componentes del paquete swing, aplicando programación orientada a objetos y diseño de clases en UML, de modo que se demuestre la codificación y uso del método de selección, usando como lenguaje de programación a Java y como IDE a *NetBeans*, posterior al análisis textual del funcionamiento del método de selección, así como también a la representación gráfica de su operatividad.

PALABRAS CLAVES: Métodos de ordenamiento de vectores, ordenamiento por el método de selección, UML y clases en Java, arreglos en Java, aplicaciones de ventana en Java con *NetBeans* y controles de la paleta swing.

1. Método de Ordenamiento Por Selección.

Para exponer el funcionamiento de este método consideremos que debemos ordenar un arreglo en forma ascendiente; es decir, de menor a mayor. Para este efecto, el método de selección inicia buscando la posición de elemento menor dentro del arreglo, considerando todo el arreglo completo; es decir, buscando la posición del menor elemento desde el puesto cero hasta la última posición. Una vez localizada la posición del menor elemento, se intercambian los elementos de la posición del elemento menor con el elemento que ocupa la primera posición en el arreglo; es decir, con el elemento de la posición cero, puesto que a partir de esta posición (la cero) se inició la búsqueda del menor elemento en el arreglo.

Luego buscamos la posición del siguiente elemento menor, pero esta vez considerando que lo buscamos desde la posición 1 hasta la última posición; una vez encontrado, intercambiamos los elementos de la posición 1 con el que ocupa la posición de nuevo menor encontrado. Esta vez se inició la búsqueda del nuevo menor desde la posición 1 y no de la 0, puesto que en la posición 0 ya está el elemento más pequeño y ninguno de los otros le quitará ese lugar; esto además reduce en 1 el número de comparaciones que deben hacer los ciclos en cada una de sus iteraciones; es decir, el ciclo interno hará una iteración menos por cada iteración hecha por el ciclo externo que lo contiene, y en consecuencia hace más eficiente el proceso.

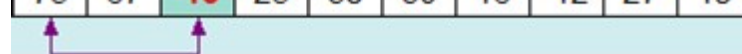
El proceso se repite continuando la búsqueda del siguiente menor en orden, pero desde la posición 2 hasta la última e intercambiándolo con el elemento de la posición 2. Luego se efectúa la búsqueda desde la posición 3 y se intercambia los elementos de esta posición (3) con el elemento de la posición del menor y así sucesivamente hasta que la búsqueda del elemento menor comprenda solo las dos últimas posiciones del arreglo.

2. Representación gráfica del método de selección.

Como ejemplo consideremos el siguiente vector de números enteros:

0	1	2	3	4	5	6	7	8	9
75	67	-15	25	38	30	16	-12	27	19

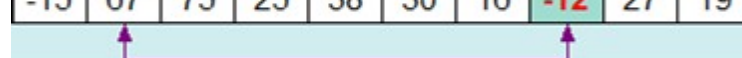
0	1	2	3	4	5	6	7	8	9
75	67	-15	25	38	30	16	-12	27	19



Buscamos el menor entre los elementos de la posición 0 a la 9, el cual es -15 y se encuentra en la posición 2. Intercambiamos las posiciones 0 y 2 quedando:

0	1	2	3	4	5	6	7	8	9
-15	67	75	25	38	30	16	-12	27	19

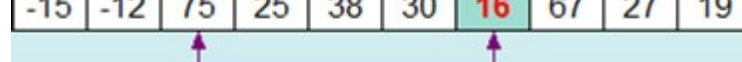
0	1	2	3	4	5	6	7	8	9
-15	67	75	25	38	30	16	-12	27	19



Buscamos el menor entre los elementos de la posición 1 a la 9, el cual es -12 y se encuentra en la posición 7. Intercambiamos las posiciones 1 y 7 quedando:

0	1	2	3	4	5	6	7	8	9
-15	-12	75	25	38	30	16	67	27	19

0	1	2	3	4	5	6	7	8	9
-15	-12	75	25	38	30	16	67	27	19



Buscamos el menor entre los elementos de la posición 2 a la 9, el cual es 16 y se encuentra en la posición 6. Intercambiamos las posiciones 2 y 6 quedando:

0	1	2	3	4	5	6	7	8	9
-15	-12	16	25	38	30	75	67	27	19

0	1	2	3	4	5	6	7	8	9
-15	-12	16	25	38	30	75	67	27	19

Buscamos el menor entre los elementos de la posición 3 a la 9, el cual es 19 y se encuentra en la posición 9. Intercambiamos las posiciones 3 y 9 quedando:

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	38	30	75	67	27	25

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	38	30	75	67	27	25

Buscamos el menor entre los elementos de la posición 4 a la 9, el cual es 25 y se encuentra en la posición 9. Intercambiamos las posiciones 4 y 9 quedando:

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	30	75	67	27	38

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	30	75	67	27	38

Buscamos el menor entre los elementos de la posición 5 a la 9, el cual es 27 y se encuentra en la posición 8. Intercambiamos las posiciones 5 y 8 quedando:

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	27	75	67	30	38

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	27	75	67	30	38

Buscamos el menor entre los elementos de la posición 6 a la 9, el cual es 30 y se encuentra en la posición 8. Intercambiamos las posiciones 6 y 8 quedando:

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	27	30	67	75	38

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	27	30	67	75	38

Buscamos el menor entre los elementos de la posición 7 a la 9, el cual es 38 y se encuentra en la posición 9. Intercambiamos las posiciones 7 y 9 quedando:

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	27	30	38	75	67

0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	27	30	38	75	67

Buscamos el menor entre los elementos de la posición 8 a la 9, el cual es 67 y se encuentra en la posición 9. Intercambiamos las posiciones 8 y 9 quedando:

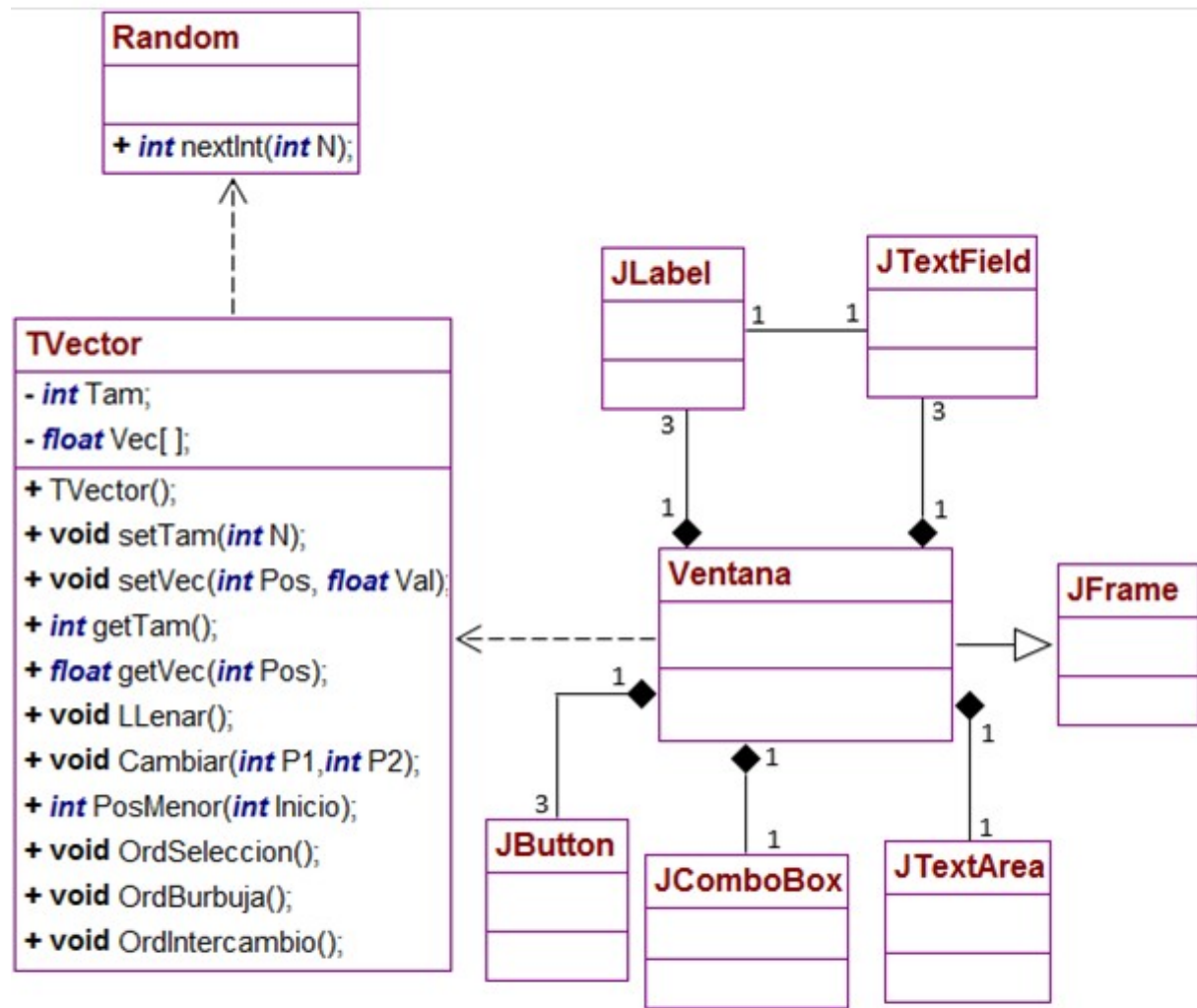
0	1	2	3	4	5	6	7	8	9
-15	-12	16	19	25	27	30	38	67	75

3. Presentación ejemplo método de selección.

Diseñar una clase en UML e implementarla en Java con una aplicación de ventana en NetBeans, que contenga un arreglo de números reales de tamaño N (dado por el usuario); de modo que permita llenar el arreglo tanto con valores ingresados por el usuario, como con valores aleatorios (para este último caso vale cualquier rango de valores), además de habilitar el ordenamiento del arreglo de manera ascendente por el método de

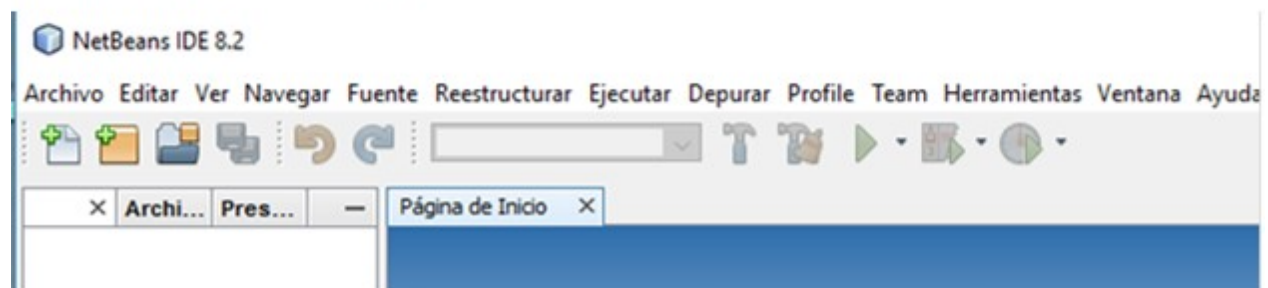
selección; también ordenar de forma descendente solo la primera mitad usando el método de burbuja y ascendentemente la segunda mitad aplicando el método de ordenamiento de intercambio.

4. Diseño diagrama de clases en UML.

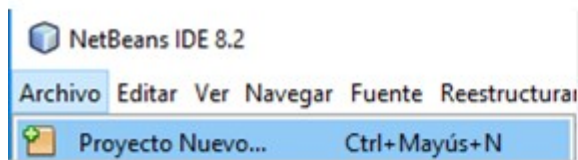


5. Creación del proyecto en NetBeans.

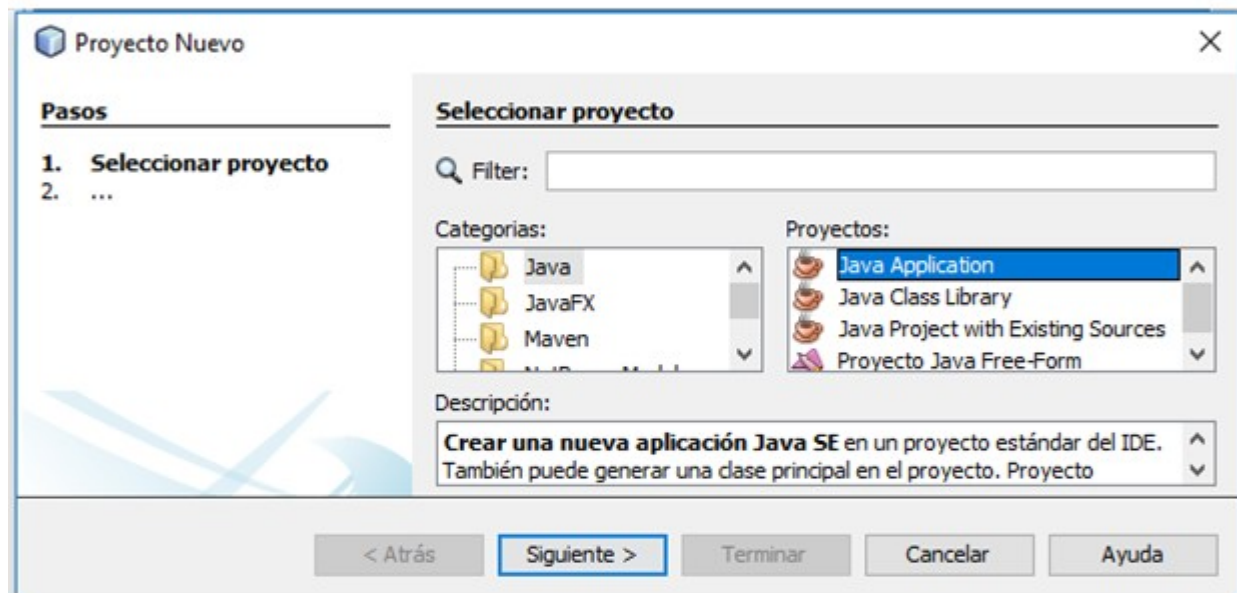
a) Entre al IDE de NetBeans



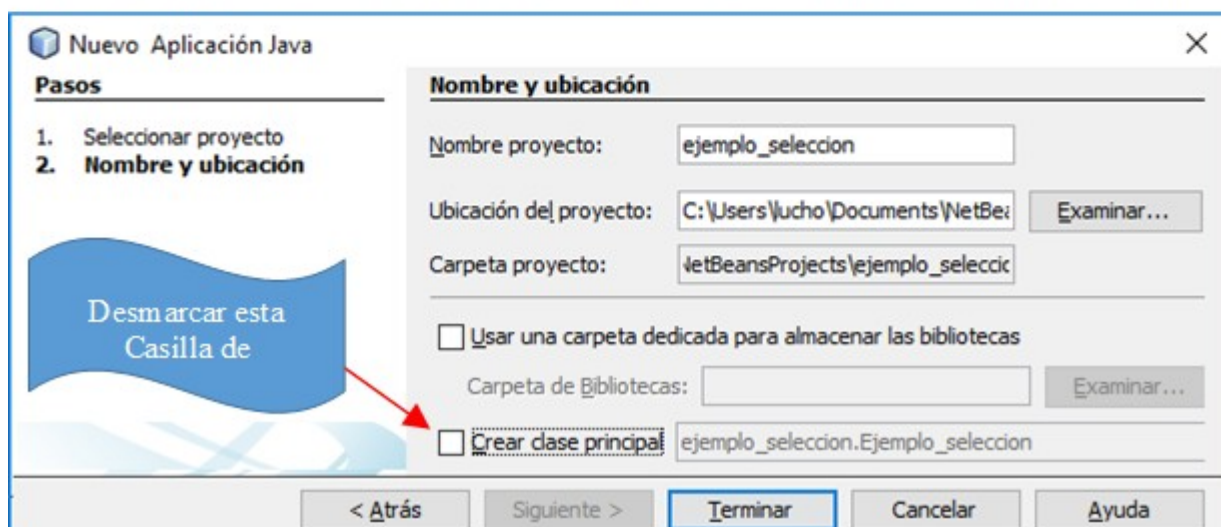
b) Crear un proyecto nuevo haciendo click en el menú **Archivo** y luego en la opción **Proyecto nuevo**, tal como se muestra en la siguiente imagen:



c) En la ventana mostrada abajo, seleccione el nodo **java** del panel de **categorías** y el nodo **Java Application** del panel **proyectos**; a aquí continuación haga click en el botón **Siguiente**:

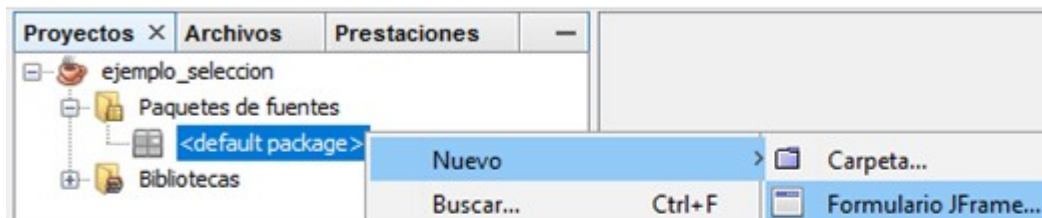


d) Ahora ingrese el nombre del proyecto (**ejemplo_seleccion**), desmarque la casilla de verificación titulada como "**Crear clase principal**" y haga click en el botón **Terminar**.

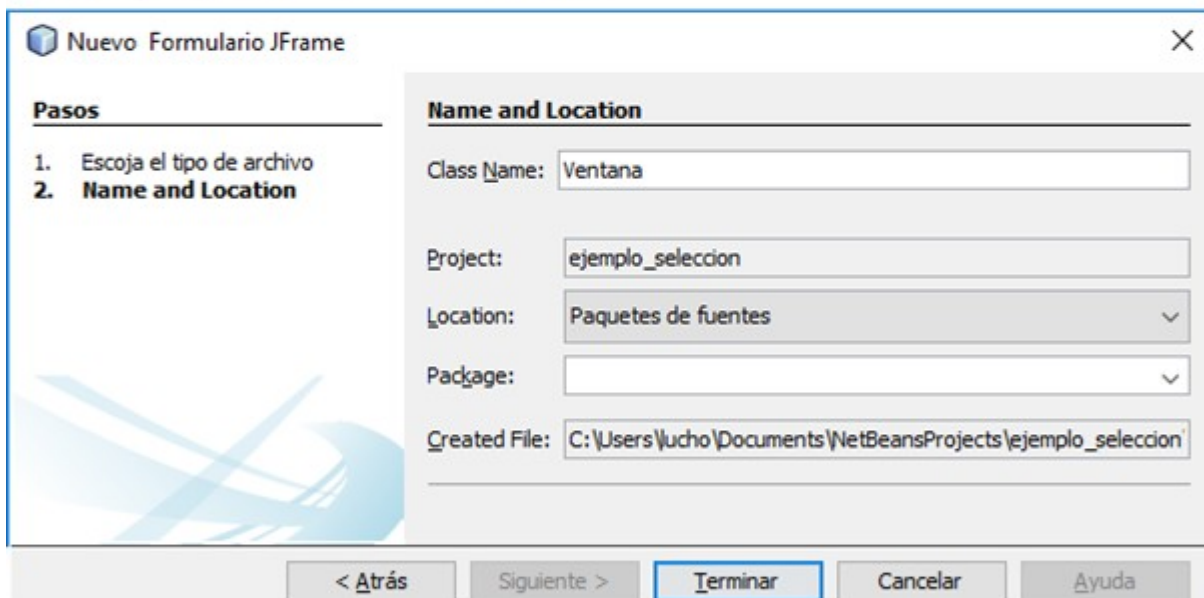


Nota: Al desmarcar la casilla “Crear clase principal” indicamos que no se cree una clase principal (que tiene el método main) para el proyecto, ya que la clase para la ventana (que creamos después) tiene su propio método main ; en consecuencia es también una clase principal.

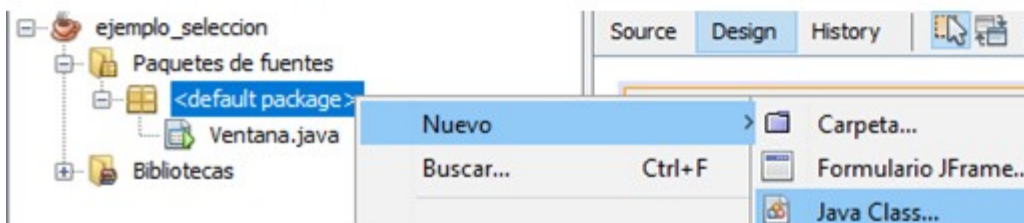
e) Crear una nueva ventana. Para ello haga click derecho en nodo **Paquetes fuentes** del proyecto y escoja la opción **Nuevo** (o **New** si el IDE esta en inglés) y luego **Formulario JFrame**, tal como se muestra en la siguiente imagen:



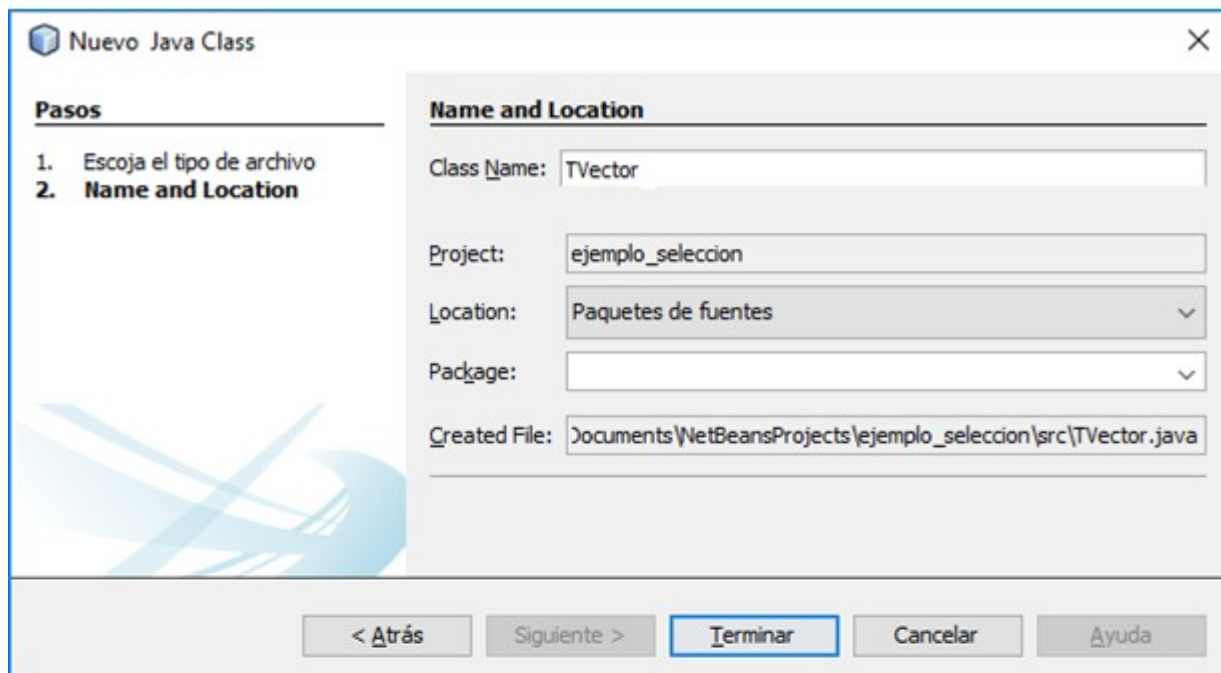
f) En la ventana desplegada, en la entrada **Class Name**, ingrese el nombre para la clase (**Ventana**) haga click en el botón **Terminar**.



g) Ahora haga click derecho en nodo **Paquetes fuentes** del proyecto y escoja la opción **Nuevo** y luego **Java Class**, tal como se muestra en la siguiente imagen:



h) En la ventana desplegada, en la entrada **Class Name**, ingrese el nombre para la clase (**TVector**) haga click en el botón **Terminar**.



Nuevo Java Class

Pasos

1. Escoja el tipo de archivo
2. **Name and Location**

Name and Location

Class Name: TVector

Project: ejemplo_seleccion

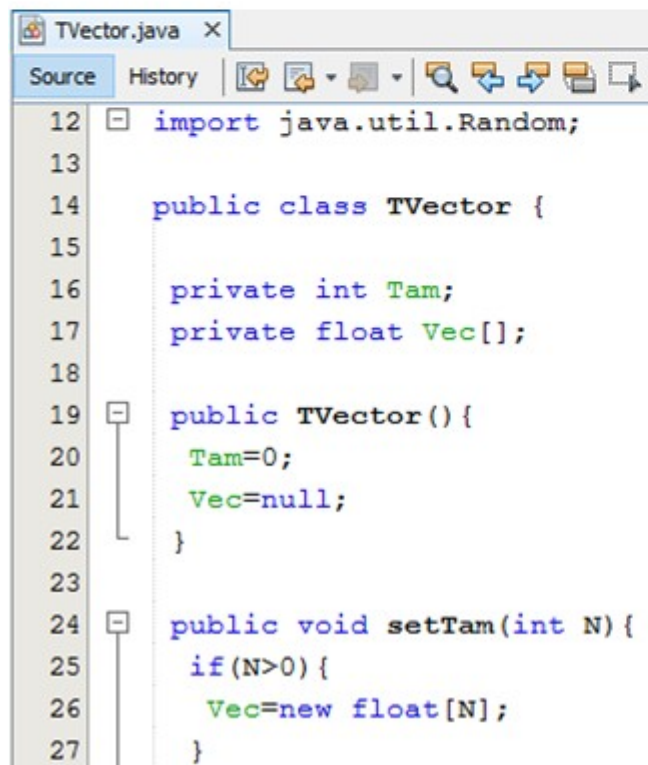
Location: Paquetes de fuentes

Package:

Created File: Documents\NetBeansProjects\ejemplo_seleccion\src\TVector.java

< Atrás Siguiete > **Terminar** Cancelar Ayuda

i) Implemente la clase **TVector** de la siguiente manera:



```
12 import java.util.Random;
13
14 public class TVector {
15
16     private int Tam;
17     private float Vec[];
18
19     public TVector() {
20         Tam=0;
21         Vec=null;
22     }
23
24     public void setTam(int N) {
25         if(N>0) {
26             Vec=new float[N];
27         }
28     }
29 }
```

```

28     else{
29         Vec=null;
30     }
31     Tam=N;
32 }
33
34 public void setVec(int Pos,float Val){
35     Vec[Pos]=Val;
36 }
37
38 public int getTam(){
39     return Tam;
40 }
41
42 public float getVec(int Pos){
43     return Vec[Pos];
44 }
45
46 public void Llenar(){
47     int i;
48     Random R;
49     R=new Random();
50     for(i=0;i<Tam;i++){
51         Vec[i]=(R.nextInt(1000)+90)/100f;
52     }
53 }
54
55 public void Cambiar(int P1,int P2){
56     float tem;
57     tem=Vec[P1];
58     Vec[P1]=Vec[P2];
59     Vec[P2]=tem;
60 }
61
62 public int PosMenor(int Inicio){
63     int i;
64     float min;
65     int posmin;
66     posmin=Inicio;
67     min=Vec[Inicio];

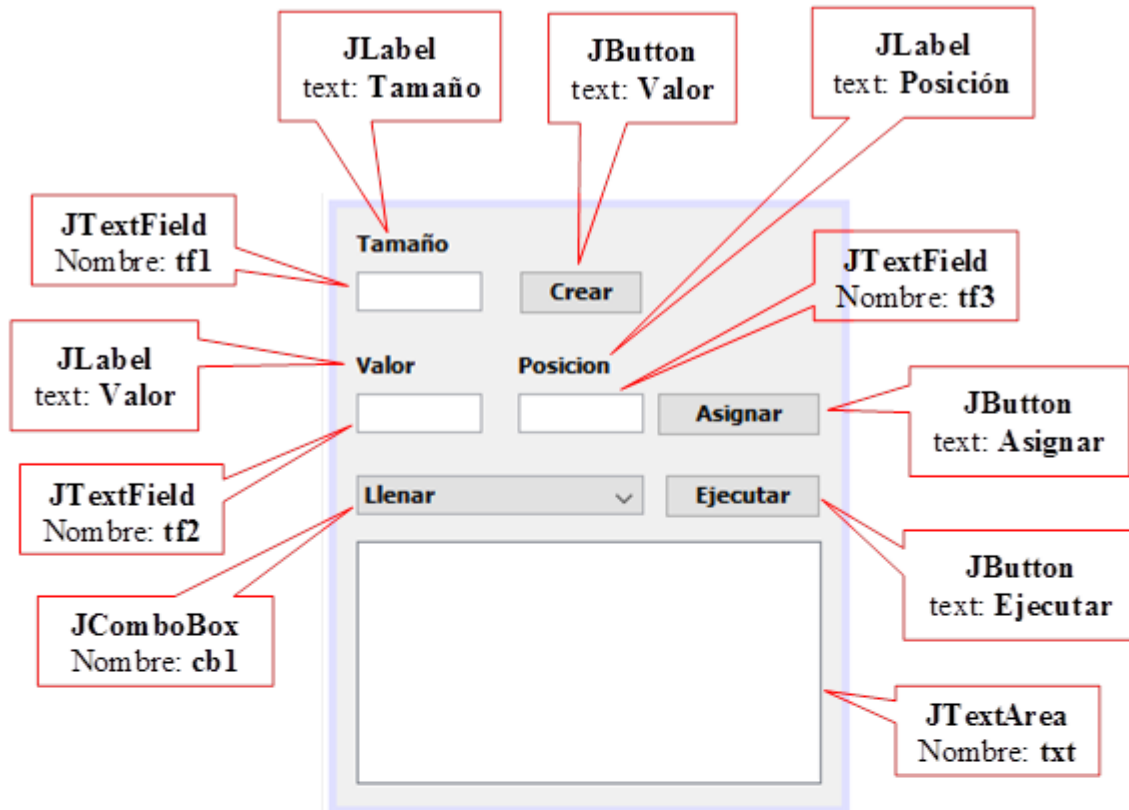
```

```

68     for(i=Inicio + 1;i<Tam;i++)
69         if(Vec[i]<min){
70             posmin=i;
71             min=Vec[i];
72         }
73     return posmin;
74 }
75
76 □ public void OrdSeleccion(){
77     int i;
78     for(i=0;i<Tam;i++)
79         Cambiar(i,PosMenor(i));
80 }
81
82 □ public void OrdBurbuja(){
83     int i,j,n;
84     n=Tam/2;
85     for(i=1;i<n;i++){
86         for(j=0;j<n-i;j++){
87             if(Vec[j]<Vec[j+1]){
88                 Cambiar(j,j+1);
89             }
90         }
91     }
92 }
93
94 □ public void OrdIntercambio(){
95     int i,j,n;
96     n=Tam/2;
97     for(i=n;i<Tam;i++){
98         for(j=i+1;j<Tam;j++){
99             if(Vec[i]>Vec[j]){
100                 Cambiar(i,j);
101             }
102         }
103     }
104 }
105
106 }

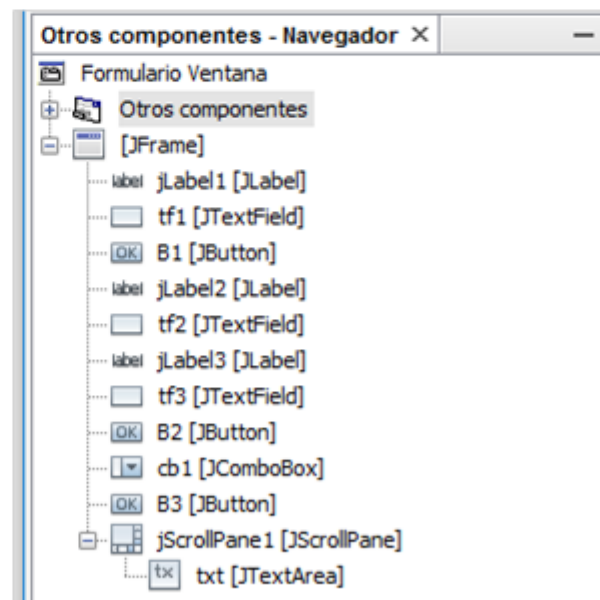
```


j) Diseñe la ventana de la siguiente manera:



También puede guiarse por la siguiente imagen del navegador de componentes, presentada a la derecha, la cual muestra el árbol de componentes añadidos a la ventana. Considere además, que el **JComboBox** (cb1) en la propiedad **model** debe tener las siguientes opciones (una debajo de la otra):

Llenar
Ordenar todo
Ordenar primera mitad
Ordenar segunda mitad



El **JTextArea** (txt) debe tener la propiedad **editable** desmarcada, para que sea de solo lectura, quedando así: **editable** ☐

k) Implemente el código fuente de la ventana de la siguiente manera:

- Vaya al código fuente de la ventana y debajo de la línea de declaración de la clase añada la instancia de la clase *TVector* y el método *Mostrar* (código dentro del cuadro rojo):

```
public class Ventana extends javax.swing.JFrame {  
  
    private TVector Ve=new TVector();  
  
    private void Mostrar(){  
        int i;  
        txt.setText("");  
        for(i=0;i<Ve.getTam();i++){  
            txt.append(String.format("Vec[%d]=%.2f\n",i,Ve.getVec(i)));  
        }  
    }  
}
```

- Haga doble click sobre el botón **Crear** e implemente el evento click (*actionPerformed*) con las dos siguientes líneas de código:

```
Ve.setTam(Integer.parseInt(tf1.getText()));  
txt.setText("");
```

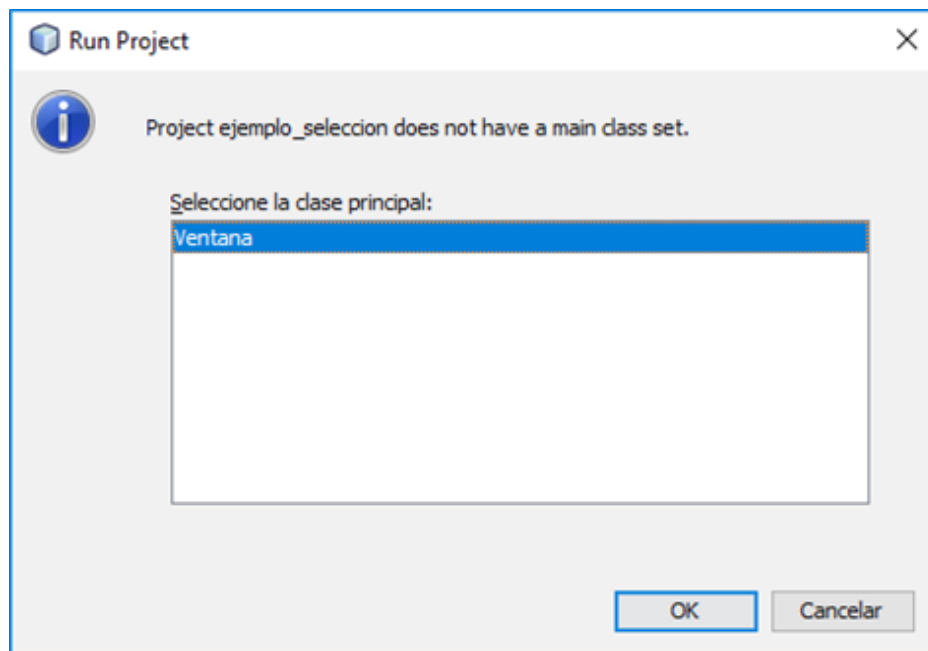
- El evento click para el botón **Asignar** es el siguiente:

```
float val=Float.parseFloat(tf2.getText());  
int pos=Integer.parseInt(tf3.getText());  
Ve.setVec(pos, val);  
Mostrar();
```

- El evento click para el botón **Ejecutar** es el siguiente:

```
switch(cb1.getSelectedIndex()){  
    case 0:Ve.Llenar();break;  
    case 1:Ve.OrdSeleccion();break;  
    case 2:Ve.OrdBurbuja();break;  
    case 3:Ve.OrdIntercambio();break;  
}  
Mostrar();
```

Finalmente compile y corrija los errores si los hubiere, luego ejecute el programa, con lo cual le aparecerá la siguiente ventana:



Esto ocurre solo la primera vez que se ejecuta el programa, ya que no hay clase principal NetBeans nos pregunta si deseamos usar la clase **Ventana** (el **JFrame**) como clase principal, por lo tanto haremos click en botón **OK**.

Ejercicios propuestos

1. Desarrollar una aplicación de ventana en *NetBeans*, con el respectivo diseño en UML e implementación en Java para una clase, que tiene tres arreglos de tamaño N y de tipo entero, de modo que el primer arreglo se ordene ascendentemente por el método de burbuja, el segundo descendientemente por el método de selección y para el tercero, ordene el primer tercio por el método de intercambio descendientemente, el segundo tercio ascendentemente por el método de burbuja y el tercer tercio descendientemente por el método de selección. Cada arreglo debe llenarse aleatoriamente con valores en el rango de 1000 a 9999.
2. Desarrollar proyecto de ventana NetBeans, para el diseño en UML e implementación en java de una clase, que debe permitir ordenar ascendentemente los caracteres de una cadena usando el método de intercambio y que la ordene descendientemente por el método de selección.