Laboratorio N° 7

Escuela Profesional: Ingeniería de Sistemas. **Asignatura:** Programación orientada a objetos
Docente: Ing. Loncán Salazar, Pierre Paul

Sesión 7: Estructuras Estáticas

I. OBJETIVOS

Al término de esta experiencia, el estudiante será capaz de:

1. Registrar y recuperar datos desde arreglos unidimensionales y bidimensionales

II. EQUIPOS Y MATERIALES

- Computador
- Guía de Laboratorio
- Material impreso con la información de la sesión de aprendizaje.

III. METODOLOGIA Y ACTIVIDADES

a) Teoría de Arreglo de Objetos
b) Teoría de manipulación de archivos de texto

IV. IMPORTANTE

Antes de iniciar con el desarrollo del Laboratorio, crearemos siempre, una carpeta, donde se guardará toda la información del presente laboratorio. Para ello realice lo siguiente:

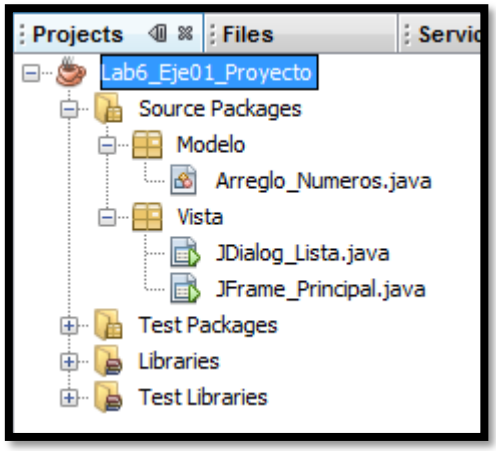
- Ingrese al Explorador del Windows (puede hacerlo dando clic derecho sobre el Botón Inicio de la Barra de Tareas y seleccione la opción Explorar).
- La ventana del Explorador esta dividida en dos columnas, en la columna de la izquierda busque hacia abajo la unidad de almacenamiento (D:) y de un clic izquierdo sobre él. Luego dirija el mouse hacia la columna de la derecha y en un sector vacío, presione clic derecho, seleccione la opción Nuevo y luego la opción Carpeta.
- Aparecerá una carpeta amarilla con un texto: Nueva Carpeta sombreado en azul, digite sobre él, el nombre para su carpeta (este puede ser L7_POO_(Turno Apellido)), luego de digitar presione la tecla Enter. Listo, ya tiene su carpeta dentro de la cual guardará todo lo que trabaje a continuación.
- Cierre la ventana del Explorador del Windows.

V. PROCEDIMIENTO

a) Encender el computador.
b) Crear carpeta donde guardará el documento son su información.
c) Ingresar al software Microsoft Word y allí crear los cuadros de doble entrada y los diagramas de clases y objetos solicitados. Word
d) Ingresar al software NetBeans IDE y allí crear las aplicaciones propuestas
e) Presentar avances al docente para la calificación correspondiente.
f) Guardar la carpeta de sus archivos a sus memorias y enviar por correo una copia del archivo al docente del curso.
g) Retirarse del laboratorio de forma ordenada.

Ejercicio N° 1:

1. Cree un proyecto con la siguiente estructura



2. En la clase llamada **Arreglo_Numeros**.
   a. Declare una variable tipo arreglo unidimensional llamada **Arreglo** de tamaño 7 para almacenar números enteros positivos y una variable tipo entera llamada **i** inicializada en 0.

Arreglo_Numeros.java

```
1  package Modelo;
2
3  import javax.swing.JOptionPane;
4
5  public class Arreglo_Numeros
6  {
7      private int[] Arreglo = new int[7];
8      int i=0;
9
```

   b. Implemente un método llamado **setDato** para agregar datos enteros dentro de este arreglo.

```
10     public void setDato(int Dato)
11     {
12         if(i<Arreglo.length)
13         {
14             if(Dato > 0)
15             {
16                 Arreglo[i] = Dato;
17                 i++;
18             }
```

```
19                else
20                {
21                    JOptionPane.showMessageDialog(null, "Sólo se aceptan" +
22                                                        "valores mayores a 0");
23                }
24            }
25            else
26            {
27                JOptionPane.showMessageDialog(null, "Lista Llena");
28            }
29        }
30
```

c. Implemente un método llamado **getDato** para leer uno de los datos del arreglo según la posición.

```
31        public int getDato(int Pos)
32        {
33            if(Pos<i)
34            {
35                return Arreglo[Pos];
36            }
37            return -1;
38        }
39
```

d. Implemente un método llamado **Buscar_Dato** que reciba un dato entero y busque dicho dato dentro del arreglo ya creado. Devolverá la posición del dato si lo encuentra y devolverá -1 si no lo encuentra.

```
40        public int Buscar_Dato(int Dato)
41        {
42            for (int Pos=0 ; Pos<i ; Pos++)
43            {
44                if(Dato == Arreglo[Pos])
45                    return Pos;
46            }
47            return -1;
48        }
```

e. Cambia la programación del método **setDato** para que ahora utilice el método **Buscar_Dato** y así evitar que se registren datos repetidos.

```
10        public void setDato(int Dato)
11        {
12            if(i<Arreglo.length)
13            {
14                if(Dato > 0)
15                {
16                    if(Buscar_Dato(Dato) == -1)
17                    {
18                        Arreglo[i] = Dato;
19                        i++;
20                    }
```

```
21                  else
22                  {
23                      JOptionPane.showMessageDialog(null,
24                              "No se pueden registrar datos duplicados");
25                  }
26              }
27              else
28              {
29                  JOptionPane.showMessageDialog(null,
30                          "Sólo se aceptan valores mayores a 0");
31              }
32          }
33          else
34          {
35              JOptionPane.showMessageDialog(null, "Lista Llena");
36          }
37      }
38
```

f. Implemente ahora un método llamado **OrdenarMayorMenor_Metodo01** que utilice el algoritmo estándar de ordenación por burbuja. La ordenación se realizará de mayor a menor.

```
58      public void OrdenarMayorMenor_Metodo01()
59      {
60          int Tmp;
61          for(int y=0 ; y<i-1 ; y++)
62          {
63              for(int x=0 ; x<i-1 ; x++)
64              {
65                  if(Arreglo[x] < Arreglo[x+1])
66                  {
67                      Tmp = Arreglo[x];
68                      Arreglo[x] = Arreglo[x+1];
69                      Arreglo[x+1] = Tmp;
70                  }
71              }
72          }
73      }
```

g. Implemente dos métodos llamados **OrdenarMayorMenor_Metodo02** y **OrdenarMayorMenor_Metodo03** que utilice un algoritmo mejorado de ordenación por burbuja de tal forma que se detenga cuando detecte que la lista esta ordenada. La ordenación se realizará de mayor a menor.

```
75      public void OrdenarMayorMenor_Metodo02()
76      {
77          int Tmp;
78          boolean Estado = false;
79          for(int y=0 ; y<i-1 && Estado==false ; y++)
80          {
81              Estado = true;
82              for(int x=0 ; x<i-1 ; x++)
83              {
84                  if(Arreglo[x] < Arreglo[x+1])
```

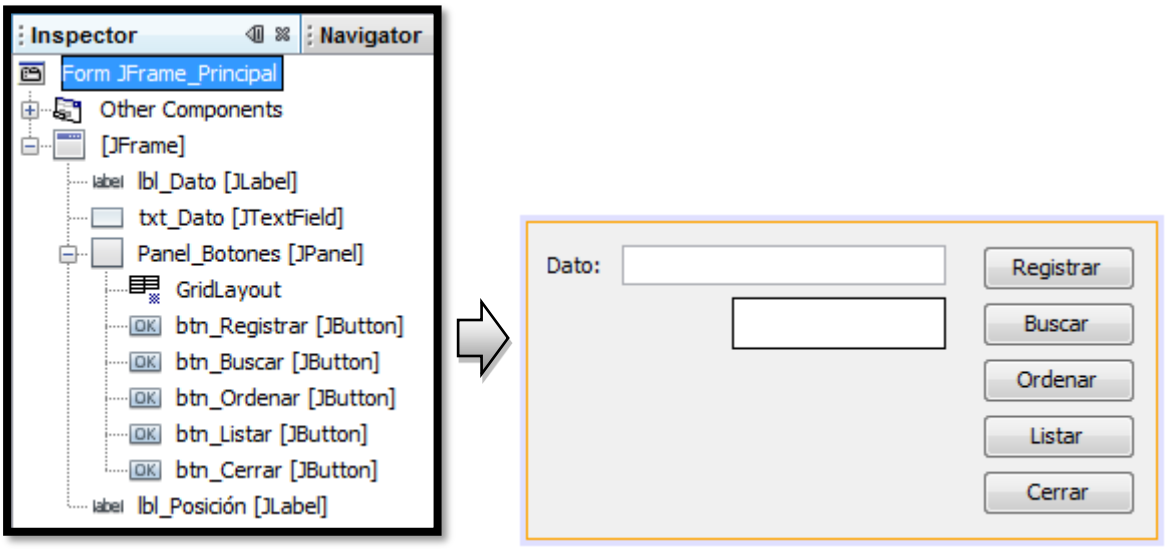
```
85                  {
86                      Tmp = Arreglo[x];
87                      Arreglo[x] = Arreglo[x+1];
88                      Arreglo[x+1] = Tmp;
89                      Estado = false;
90                  }
91              }
92          }
93      }
94
```

```
95      public void OrdenarMayorMenor_Metodo03()
96      {
97          int Tmp, y=0;
98          boolean Estado = false;
99          while(y<i-1 && Estado==false)
100         {
101             Estado = true;
102             for(int x=0 ; x<i-1 ; x++)
103             {
104                 if(Arreglo[x] < Arreglo[x+1])
105                 {
106                     Tmp = Arreglo[x];
107                     Arreglo[x] = Arreglo[x+1];
108                     Arreglo[x+1] = Tmp;
109                     Estado = false;
110                 }
111             }
112             y++;
113         }
114     }
115
```

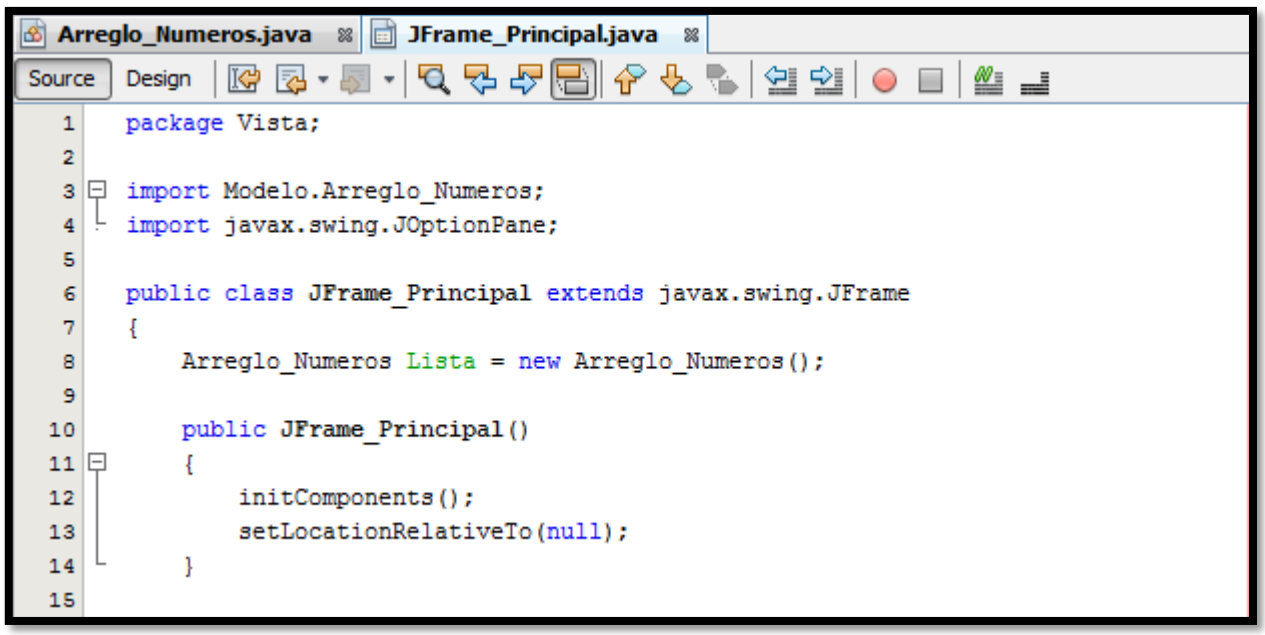
h. Implemente ahora un método llamado **getRegistrados** que retorne el total de elementos registrados en el arreglo.

```
116     public int getRegistrados()
117     {
118         return i;
119     }
```

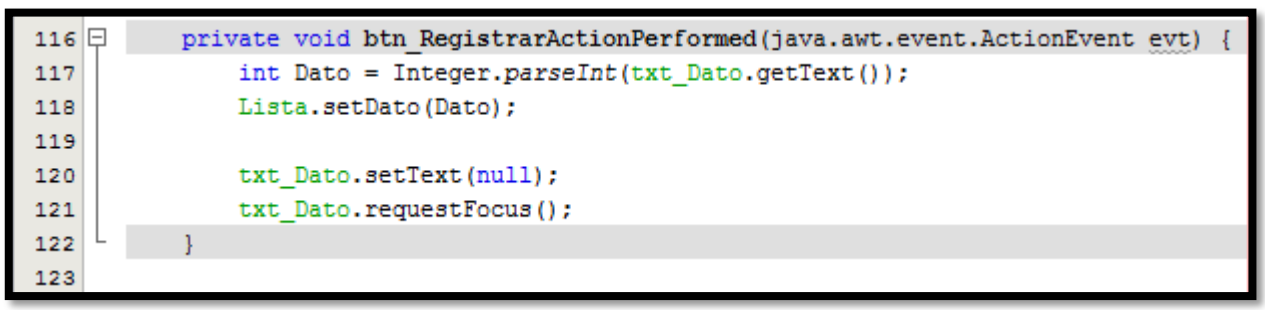
3. En la clase **JFrame_Principal**
   a. Diseñe un formulario con la siguiente apariencia y estructura.



   b. Cree una instancia de la clase **Arreglo_Numeros** llamada **Lista**, y en el constructor coloque un código que centre el formulario.



   c. Implemente el código del control **btn_Registrar**



d. Implemente el código del control **btn_Buscar**

```
124     private void btn_BuscarActionPerformed(java.awt.event.ActionEvent evt) {
125         String Rpta = JOptionPane.showInputDialog("Ingrese Dato a Buscar:");
126 
127         if(Rpta != null)
128         {
129             int Dato = Integer.parseInt(Rpta);
130 
131             int Pos = Lista.Buscar_Dato(Dato);
132 
133             if(Pos != -1)
134             {
135                 lbl_Posición.setText(Pos + "");
136             }
137             else
138             {
139                 JOptionPane.showMessageDialog(this, "Dato no existe");
140             }
141         }
142     }
143 
```

e. Implemente el código del control **btn_Ordenar**

```
144     private void btn_OrdenarActionPerformed(java.awt.event.ActionEvent evt) {
145         Lista.OrdenarMayorMenor_Metodo01();
146     }
147 
```

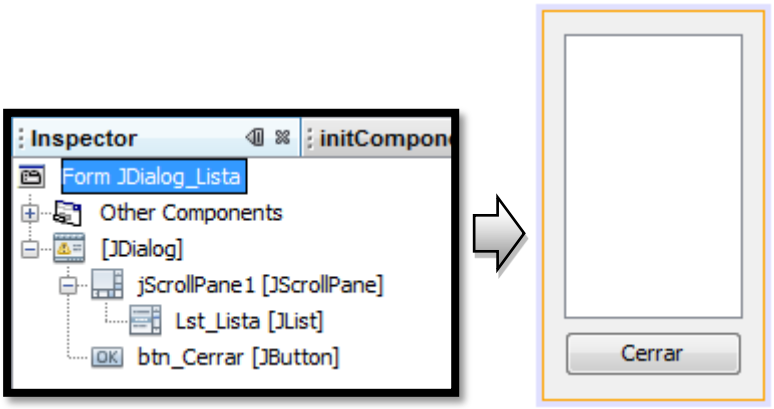
f. Implemente el código del control **btn_Listar**

```
148     private void btn_ListarActionPerformed(java.awt.event.ActionEvent evt) {
149         JDialog_Lista Dialogo = new JDialog_Lista(this, true);
150         Dialogo.Llenar_Lista(Lista);
151         Dialogo.setVisible(true);
152     }
153 
```

g. Implemente el código del control **btn_Cerrar**

```
154     private void btn_CerrarActionPerformed(java.awt.event.ActionEvent evt) {
155         System.exit(0);
156     }
157 
```

4. En la clase **JDialog_Lista**
   a. Diseñe un dialogo con la siguiente apariencia y estructura



   b. Cree una instancia de la clase **DefaultListModel** llamada **modLista**, y en el constructor coloque un código que centre el diálogo y asigne el modelo al control **Lst_Lista**.

```
Arreglo_Numeros.java | JFrame_Principal.java | JDialog_Lista.java
Source | Design
1   package Vista;
2
3   import Modelo.Arreglo_Numeros;
4   import javax.swing.DefaultListModel;
5
6   public class JDialog_Lista extends javax.swing.JDialog
7   {
8       DefaultListModel modLista = new DefaultListModel();
9
10      public JDialog_Lista(java.awt.Frame parent, boolean modal)
11      {
12          super(parent, modal);
13          initComponents();
14
15          setLocationRelativeTo(parent);
16          Lst_Lista.setModel(modLista);
17      }
```

   c. Implemente el código del método **Llenar_Lista**

```
19      public void Llenar_Lista(Arreglo_Numeros Lista)
20      {
21          for(int x=0 ; x<Lista.getRegistrados() ; x++)
22          {
23              modLista.addElement(Lista.getDato(x));
24          }
25      }
```

   d. Implemente el código del control **btn_Cerrar**

```
70      private void btn_CerrarActionPerformed(java.awt.event.ActionEvent evt) {
71          dispose();
72      }
73
```

Ejercicios Propuestos:

Cree las siguientes aplicaciones:

1. Cree una aplicación que permita registrar los nombres de personas que desean ir a un viaje de excursión:
    a. Registrar Nombres y los almacene en un arreglo unidimensional tipo String de tamaño 20.
    b. Listar y mostrar todos los nombres registrados.
    c. Contar cuantos nombres se han registrado.
    d. No permitir que se ingrese 2 veces el mismo nombre.
    e. Listar y mostrar los nombres que contengan ciertos caracteres. Por ejemplo: todos los nombres que terminen con las letras **AS**.
2. Cree una aplicación que permita registrar las edades de una familia:
    a. Registrar Edades en una de 3 listas, cada una con una capacidad máxima de 10.
        i. Edades menores a 12 se registran en la lista 1.
        ii. Edades mayores o iguales a 12 pero menores a 25 se registran en la lista 2.
        iii. Edades mayores o iguales a 25 se registran en la lista 3.
    b. Mostrar las 3 listas
    c. Las listas siempre deberán de mostrarse ordenadas de menor a mayor.
3. Cree una aplicación que permita registrar ventas:
    a. Para cada venta se piensa registrar el número de comprobante y el monto de la venta.
        i. El número de comprobante se registra en la lista 1.
        ii. El monto de la venta se registra en la lista 2.
    b. Consultar ventas:
        i. Se solicita el número de comprobante y se busca en la lista 1.
            - Si existe se muestra en monto que se encuentre en la lista 2 en la misma posición en la que se encontró el número de comprobante en la lista 1.
    c. Mostrar la suma total de todos los montos ingresados
    d. Eliminar cualquier venta:
        i. Se solicita el número de comprobante y se busca en la lista 1.
            - Si existe se eliminará el número de comprobante y el monto asociado a dicho comprobante.

RUBRICA:

Inicio 0-10	Proceso 11-13	Logro previsto 14-17	Logro satisfactorio 18-20
Desarrollo correctamente del laboratorio hasta un 50 %	Desarrollo correctamente del laboratorio hasta un 60 %	Desarrollo correctamente del laboratorio hasta un 80 %	Desarrollo correctamente del laboratorio hasta un 100%

Bibliografía:

- THOMAS WU C. Introducción a la programación orientada a objetos con Java. 1ª Edición. España. McGraw-Hill Interamericana de España. 2008. 15-22pp. ISBN: 978-0-07-352339-2
- LEOBARDO LOPEZ. Román. Metodología de la programación orientada a objetos. 1ª Edición. México. Alafomega grupo editor de México. 2006. 241-253pp ISBN: 970-15-1173-5
- HERBERT SHILDT. JAVA 2 v5.0. España. Ediciones Anaya multimedia.2005. 79-99pp ISBN: 84-415-1865-3