

Tópicos Avanzados de Programación

Ingeniería en Sistemas Computacionales

Tema 2. Componentes y librerías.

Maciel Villa Valeria 21010985

Mávil Barojas Luis Enrique 21010994

Vargas Puertos Luis Gerardo 21011056

Grupo 4g2A

27/Marzo/2023

Introducción:

La programación se ha convertido en una herramienta esencial en la creación de software y aplicaciones. Uno de los conceptos más importantes en este ámbito son los componentes, paquetes y librerías.

Los componentes son los bloques de construcción básicos de un programa y se utilizan para realizar tareas específicas. Por su parte, los paquetes son conjuntos de componentes que se agrupan de manera lógica y pueden ser reutilizados en diferentes proyectos. Finalmente, las librerías son conjuntos de código preescrito que proporcionan funciones y herramientas específicas para ser utilizadas en la construcción de aplicaciones.

El uso de librerías proporcionadas por el lenguaje de programación es una práctica muy común en el desarrollo de software, ya que permite ahorrar tiempo y esfuerzo en la creación de aplicaciones. Además, los lenguajes de programación también permiten la creación de componentes definidos por el usuario, tanto visuales como no visuales. Esto significa que se pueden crear componentes personalizados para realizar tareas específicas en el programa.

Asimismo, los paquetes y librerías definidos por el usuario también son una parte importante de la programación, ya que permiten la reutilización de código en diferentes proyectos. Estos paquetes y librerías pueden incluir componentes personalizados, funciones y herramientas específicas para ser utilizados en diferentes aplicaciones.

En resumen, los componentes, paquetes y librerías son conceptos fundamentales en la programación y su uso es esencial para la creación de software eficiente y escalable. Además, la capacidad de crear componentes personalizados y paquetes y librerías definidos por el usuario permite una mayor flexibilidad y reutilización de código en diferentes proyectos.

Competencia específica:

Durante esta unidad la competencia es sobre implementar componentes es decir objetos que tienen una funcionalidad específica y que pueden ser reutilizados en diferentes partes de una aplicación o en diferentes aplicaciones. Los componentes en Java pueden ser visuales o no visuales.

Los componentes visuales en Java son objetos que se utilizan para crear interfaces gráficas de usuario. Estos componentes incluyen botones, etiquetas, campos de texto, listas, menús, entre otros. Los componentes visuales se pueden personalizar y se pueden agregar paneles o marcos para crear una interfaz de usuario completa.

También ocupar las librerías, las librerías son un conjunto de código preescrito que proporciona funciones y herramientas específicas para ser utilizadas en la construcción de aplicaciones. Las librerías son creadas por los desarrolladores y se pueden utilizar para simplificar la programación y ahorrar tiempo en la creación de aplicaciones.

Esto se usa para la reutilización de código que permite optimizar un programa, mejorar su comprensión para otros desarrolladores además de volver la programación un poco más sencilla

Contextualización:

Hoy en día la programación es una herramienta de gran uso a muchos niveles como por ejemplo el desarrollo de nuevas tecnologías, optimizar procesos, creación de software por lo que cada vez buscan de cierta manera facilitar la manera en que se programa.

En este contexto se crean y emplean el uso de interfaces gráficas que es la manera en que un conjunto de programas se comunican de manera visible y amigable con un usuario para ello se usan componentes como ya los habíamos definidos previamente los cuales se encuentran en cualquier programa que vemos a nuestro alrededor y para optimizar aun más se usan las librerías, en este caso se hizo uso

de estos recursos como ejemplos de como funcionan ciertos programas, programas como por ejemplo software para una empresa sobre inventario, ventas, gastos, registros, información sobre los empleados etc, y para poder lograr esto de la mejor manera se bueno conocer bien la manera en como se crean, que herramientas te ofrecen en si mismos estos componentes y como poder usarlos en conjunto.

Material para esta práctica:

- Para el desarrollo de la practica necesitaremos:
Una computadora
- Un IDE de desarrollo, en este caso nosotros usamos Netbeans pero puedes utilizar otros como eclipse por ejemplo
- Conexión a internet para investigar algunas dudas o cuestionamientos (Opcional)

Desarrollo de la practica

Practica Invitados:

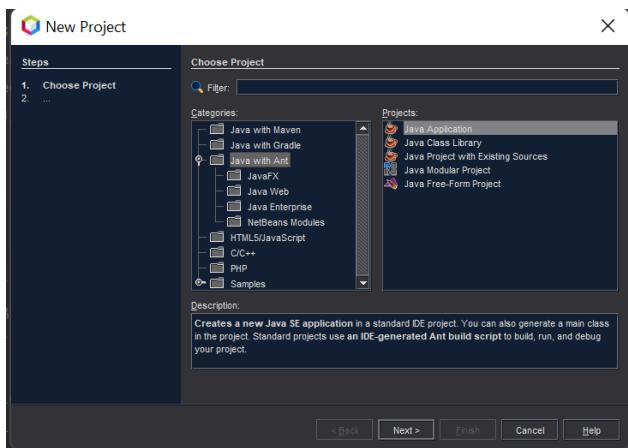
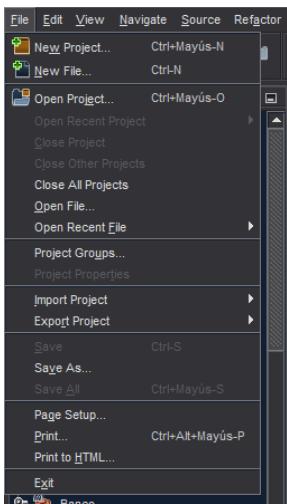
Las indicaciones que se nos han pedido lo siguiente:

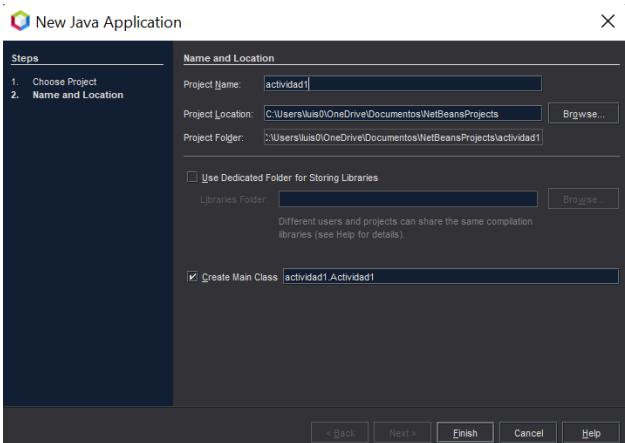
“Implementa una aplicación que permita controlar los asistentes a una fiesta para ello debe agregar la edad sexo masculino o femenino estado civil puede ser soltero o casado viudo o divorciado del asistente luego muestre las siguientes estadísticas:

- Total de asistentes
- Total de personas mayores de edad total de personas menores de edad
- Total de hombres
- Total de mujeres
- Total de solteros
- Total de casados
- Total de viudos
- Total de divorciados
- Porcentaje de hombres y porcentaje de mujeres”

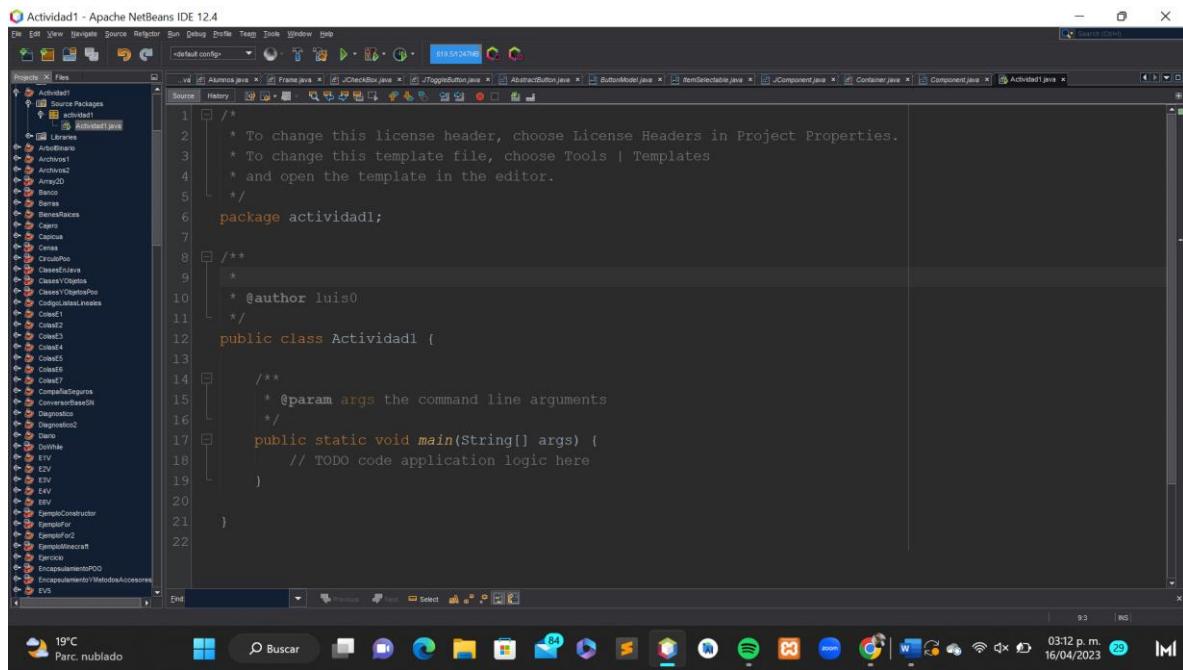
Para empezar nuestra práctica debemos crear un nuevo proyecto que en este caso llamaremos actividad1, pero puede ser el nombre que gustes en casos futuros.

En la parte superior izquierda del IDE hay un menú con diferentes opciones por ahora derechos clic en file, después se nos despliega otro menú donde seleccionaremos “New project” después procederemos a presionar el botón “Next” y en la siguiente ventana poner el nombre, en este caso “Actividad1” y oprimir el botón “Finish”

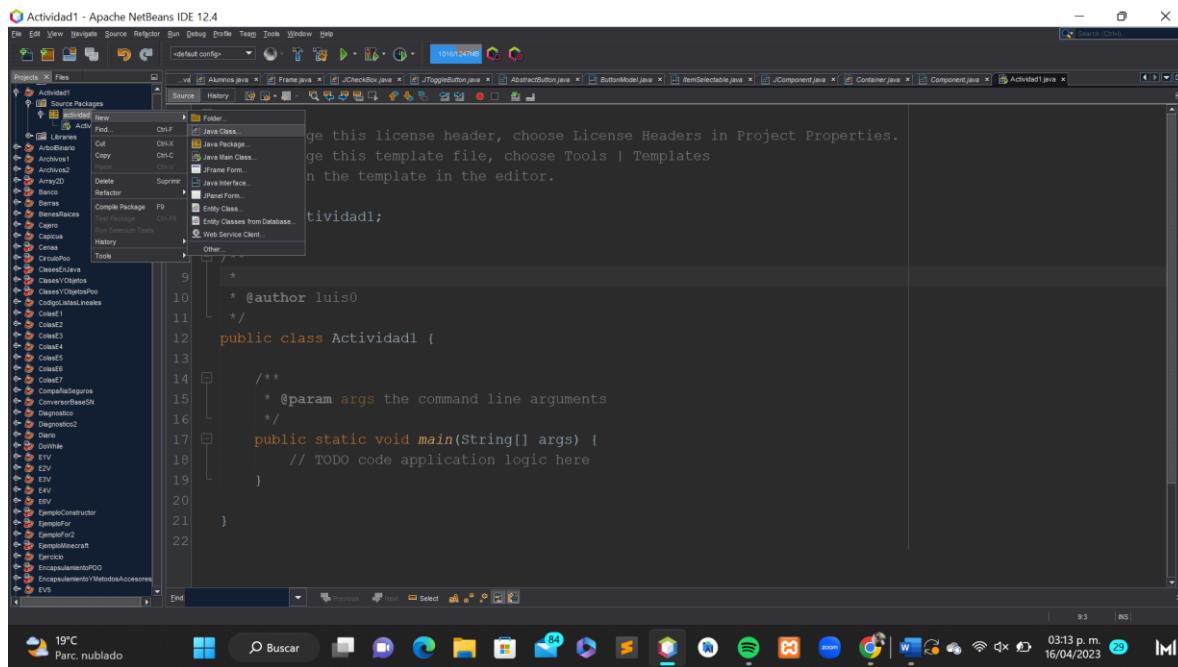




Despues de eso nos aparecera lo siguiente



Despues de eso debemos dar click derecho en donde hay como un paquete y junto del proyecto, se nos despliega un menos, seleccionaremos la opcion “New” y despues la opcion “Java Class”

The screenshot shows the Apache NetBeans IDE interface. In the center, there is a code editor window displaying the following Java code:

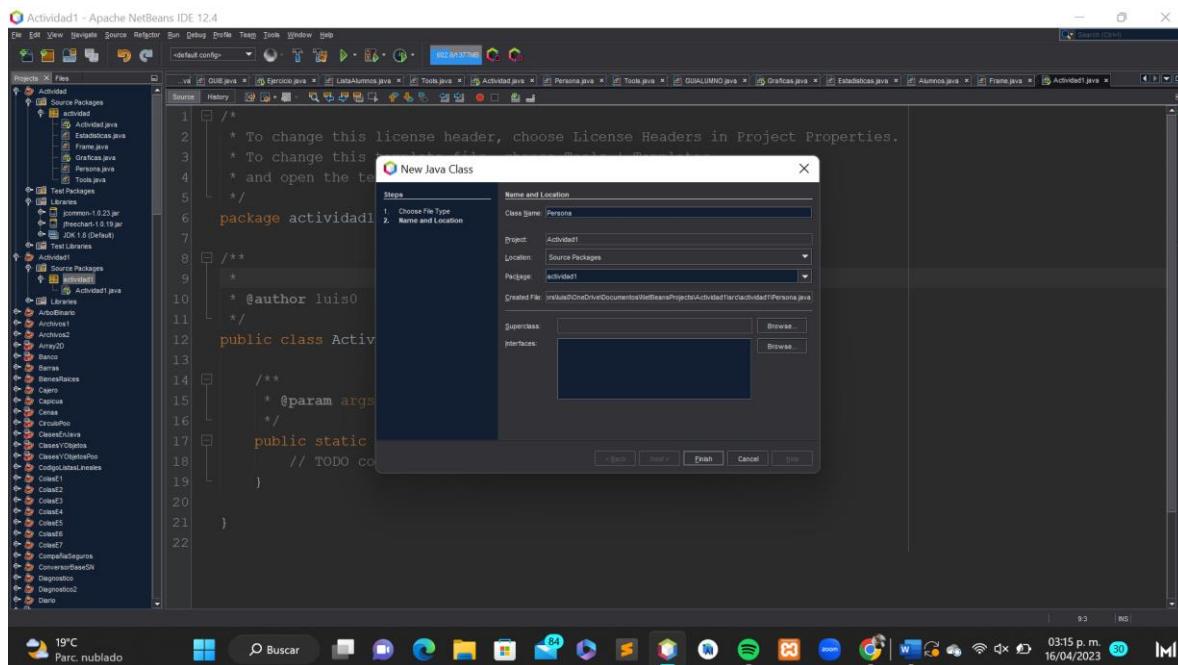
```

 9  *
10 * @author luis0
11 */
12 public class Actividad1 {
13
14     /**
15      * @param args the command line arguments
16     */
17     public static void main(String[] args) {
18         // TODO code application logic here
19     }
20
21 }
22

```

To the left of the code editor, the project tree shows a package named 'Actividad1' containing several Java files like 'Actividad.java', 'Estructuras.java', and 'Frame.java'. A right-click context menu is open over the 'Source Packages' node, with the 'New' option expanded. The 'Java Class' option is highlighted.

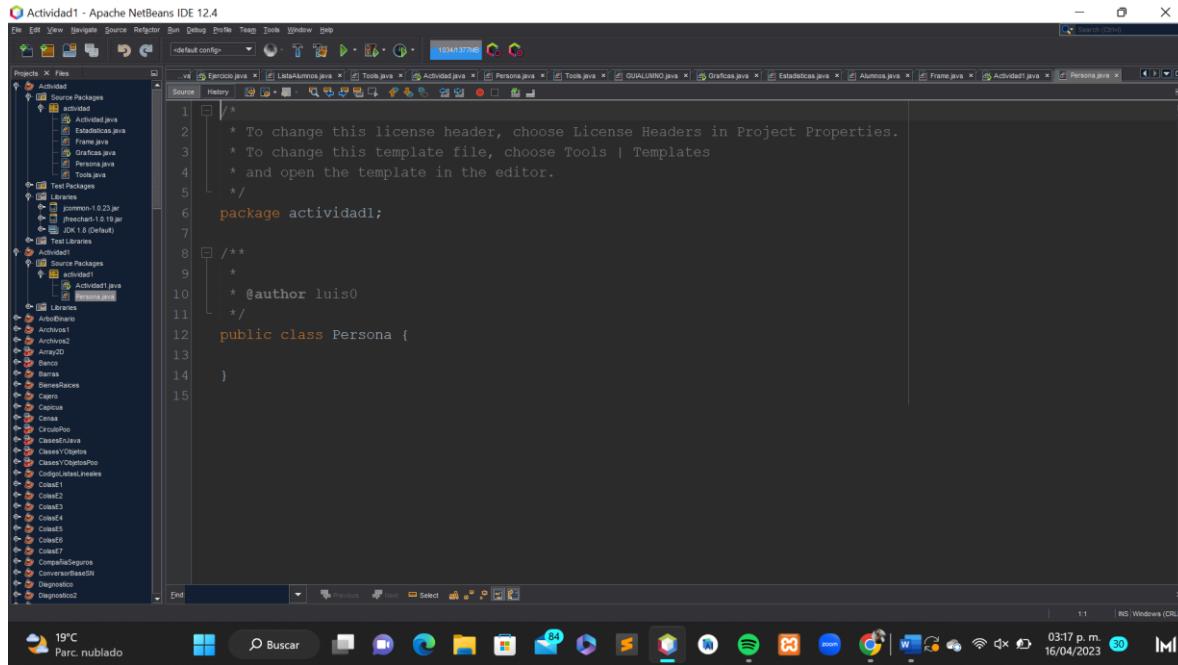
En el apartado “Class name” procedemos a poner el nombre de la clase y después oprimir el botón “Finish”



The screenshot shows the Apache NetBeans IDE interface with the 'New Java Class' dialog box open. The dialog has two tabs: 'Steps' and 'Name and Location'. The 'Name and Location' tab is active, showing the following configuration:

- Class Name:** Person
- Project:** Actividad1
- Location:** Source Packages
- Package:** actividad1
- Created File:** C:\Users\luis0\OneDrive\Documentos\NetBeansProjects\Actividad1\src\actividad1\Person.java

Below these fields, there are 'Superclass' and 'Interfaces' dropdowns, both currently empty. At the bottom of the dialog are buttons for 'Back', 'Next >', 'Finish', 'Cancel', and 'Help'.



Para este trabajo usaremos 5 clases incluyendo la clase “Persona” por lo que repetiremos el procesos anterios poniendo los nombre “Tools”, “Frame”, “Estadisticas” y “Graficas”

En la clase Persona tendra los atributos nombre, sexo, estado civil, edad, tendra su constructor y sus respectivos metodos setters(metodos para cambiar o asignar un valor a los atributos del objeto) y getters (metodos que nos dan el valor de algun atributo en especifico)

Persona

```
1  package actividad;
2
3  public class Persona {
4      private String nombre, sexo, estadoc;
5      private byte edad;
6
7      public Persona() {}
8
9      public String getNombre() {
10         return nombre;
11     }
12
13     public void setNombre(String nombre) {
14         this.nombre = nombre;
15     }
16
17     public String getSexo() {
18         return sexo;
19     }
20
21     public void setSexo(String sexo) {
22         this.sexo = sexo;
23     }
24 }
```

```
25     public String getEstadoc() {
26         return estadoc;
27     }
28
29     public void setEstadoc(String estadoc) {
30         this.estadoc = estadoc;
31     }
32
33     public byte getEdad() {
34         return edad;
35     }
36
37     public void setEdad(byte edad) {
38         this.edad = edad;
39     }
40
41
42 }
```

Después procederemos a crear la clase “Tools” esta clase será un conjunto de métodos que nos ayudaran a validar la información que sea introducida por el usuario ya que el usuario por curiosidad puede meter datos erróneos a propósito para ver como reacciona el programa ante esta situación por lo que, nuestro deber como programador es prever estas situaciones y solucionarlas

Para ello crearemos utilizaremos los métodos que ya hemos visto en prácticas pasadas adecuándolos a nuestras necesidades, para este caso se usaran métodos con los siguientes nombres “validName” y “validEdad”

Únicamente validaran que los datos ingresados vayan de acuerdo a los tipos de datos declarados para los atributos y que concuerden, en el caso del nombre no llevar caracteres especiales ni números, únicamente letras, con la edad, que sea el tipo de dato declarado y aparte no sea negativo, ni tampoco sobrepase la cierta cantidad retornaran ciertos valores que nos ayudaran a darnos una idea de si es correcto o incorrecto lo ingresado

Por lo que procederemos a escribir el siguiente código:

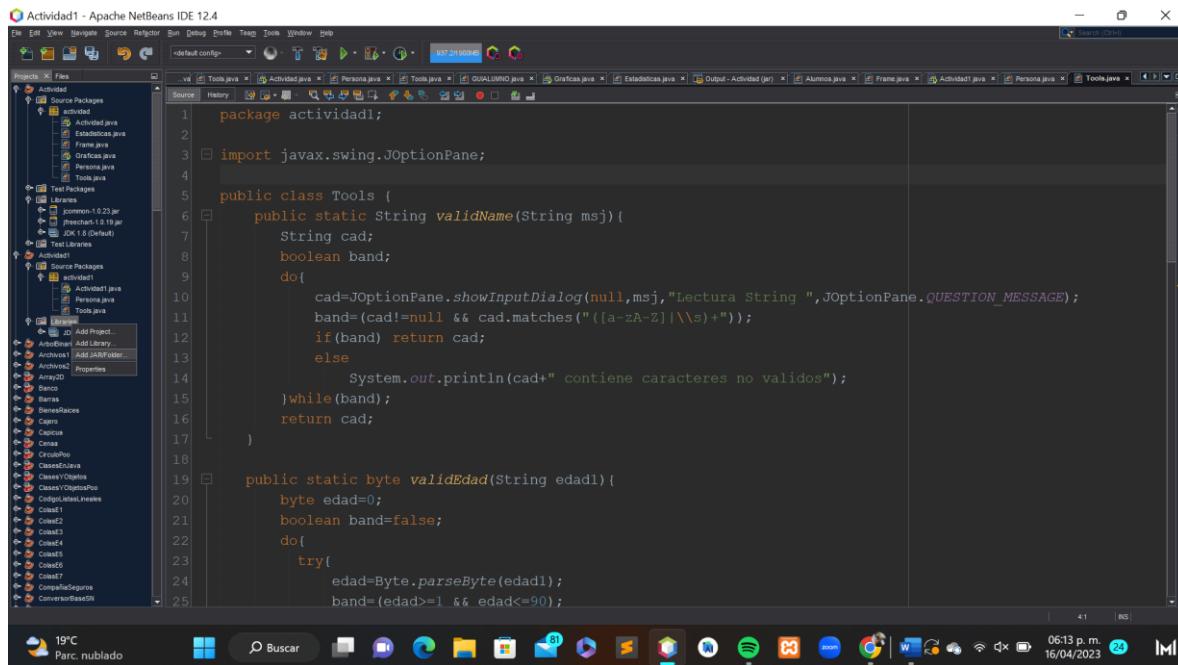
```
1 package actividad;
2
3 public class Tools {
4     public static String validateName(String cad) {
5         boolean band;
6         band=(cad!=null && cad.matches("[a-zA-Z]+\\s+"));
7         if(band) return cad;
8         else return "J";
9     }
10
11    public static byte validateAge(String edad1) {
12        byte edad=0;
13        boolean band=false;
14        try{
15            if(edad1.equals(null)){
16                edad=Byte.parseByte(edad1);
17                band=(edad>=1 && edad<=90);
18                if(band) return edad;
19            }
20        }catch(Exception a){
21        }
22        return edad;
23    }
24 }
```

Despues pasaremos a la clase estadísticas esta es la clase donde se llevarán a cabo la mayor parte de las operaciones es decir crear y añadir a las personas a la lista de invitados también es dónde se llevará a cabo el conteo de las personas el total de viudos divorciados casados solteros así como las personas mayores menores de edad cuántos hombres y mujeres hoy si es necesario eliminará alguno hoy hay para ello crearemos varios métodos los cuales se encargarán realizar todas estas operaciones.

Para mejorar la experiencia del usuario y que las estadísticas sean más comprensibles para las personas que usan este software se ha optado por representar estos datos en una gráfica que nos muestre el total de hombres total de mujeres etcétera para esto ocuparemos lo que comúnmente se le conoce cómo librerías sin embargo para hacer uso de estas librerías tenemos que tener el jar y

además importarlo de manera correcta a nuestro proyecto para poder utilizar las herramientas que contiene a continuación veremos cómo se hace:

En la parte izquierda dónde están nuestros proyectos donde estan nuestras clases hay una parte qué dice “Libraries” ahí haremos clic derecho y seleccionaremos la opción “Add JAR/Folder”



Despues de seleccionar la opción buscaremos la dirección donde hemos guardado previamente las librerías



Actividad1 - Apache NetBeans IDE 12.4

```

  package actividad1;

import javax.swing.JOptionPane;

public class Tools {
    public static String cad;
    boolean band;
    do{
        cad=JOptionPane.showInputDialog("Introduzca su nombre");
        band=(cad.length()>0);
        if(band) JOptionPane.showMessageDialog(null,"Nombre correcto");
        else JOptionPane.showMessageDialog(null,"Nombre incorrecto");
    }while(band);
    return cad;
}

public static byte validEdad(String edad1){
    byte edad=0;
    boolean band=false;
    do{
        try{
            edad=Byte.parseByte(edad1);
            band=(edad>=1 && edad<=90);
        }
        catch(Exception e){
            band=false;
        }
    }while(!band);
    return edad;
}

```

19°C Parc. nublado Buscar 06:14 p.m. 16/04/2023 IMI

Las seleccionamos y ahora las importaremos una por una

Actividad1 - Apache NetBeans IDE 12.4

```

  package actividad1;

import javax.swing.JOptionPane;

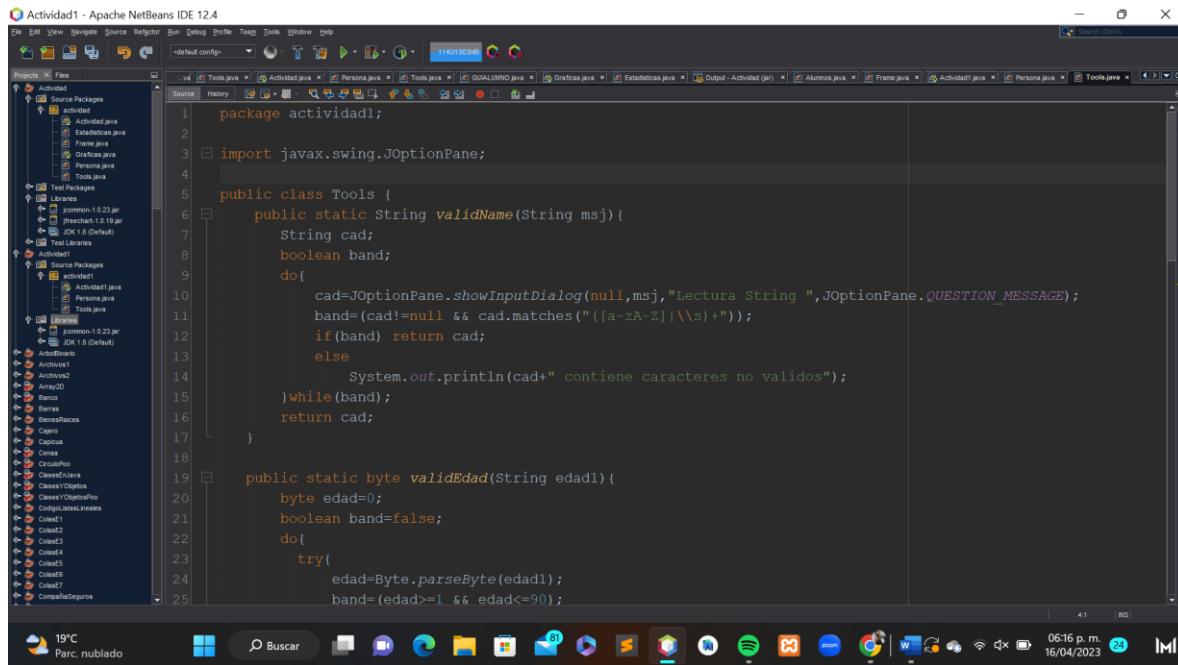
public class Tools {
    public static String cad;
    boolean band;
    do{
        cad=JOptionPane.showInputDialog("Introduzca su nombre");
        band=(cad.length()>0);
        if(band) JOptionPane.showMessageDialog(null,"Nombre correcto");
        else JOptionPane.showMessageDialog(null,"Nombre incorrecto");
    }while(band);
    return cad;
}

public static byte validEdad(String edad1){
    byte edad=0;
    boolean band=false;
    do{
        try{
            edad=Byte.parseByte(edad1);
            band=(edad>=1 && edad<=90);
        }
        catch(Exception e){
            band=false;
        }
    }while(!band);
    return edad;
}

```

19°C Parc. nublado Buscar 06:15 p.m. 16/04/2023 IMI

Presionamos abrir y verificamos del lado izquierdo para ver si se importó, aparecerá un frasco y al lado el nombre de la librería

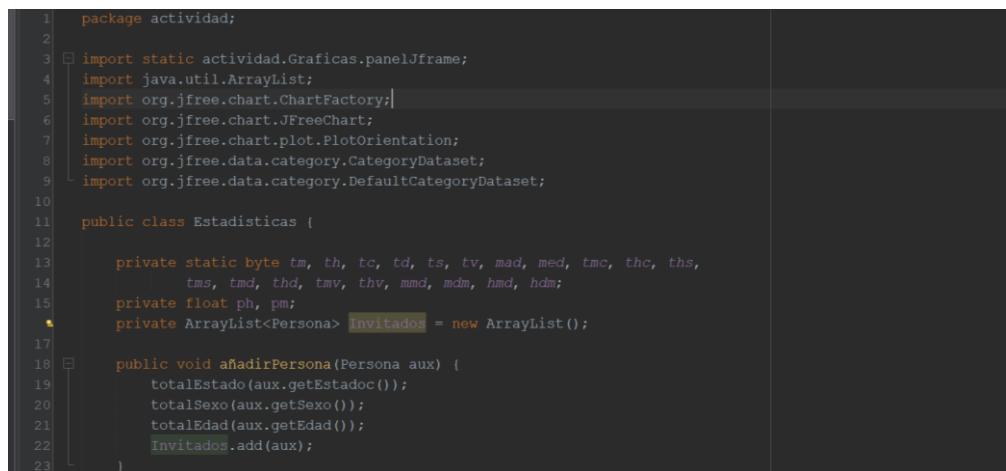
```

 1 package actividad1;
 2
 3 import javax.swing.JOptionPane;
 4
 5 public class Tools {
 6     public static String validarNombre(String msj) {
 7         String cad;
 8         boolean band;
 9         do{
10             cad=JOptionPane.showInputDialog(null,msj,"Lectura String ",JOptionPane.QUESTION_MESSAGE);
11             band=(cad==null || cad.matches("[a-zA-Z]+\\s+"));
12             if(band) return cad;
13             else
14                 System.out.println(cad+" contiene caracteres no validos");
15         }while(band);
16         return cad;
17     }
18
19     public static byte validarEdad(String edad1) {
20         byte edad=0;
21         boolean band=false;
22         do{
23             try{
24                 edad=Byte.parseByte(edad1);
25                 band=(edad>=1 & edad<=90);
26             }
27         }while(!band);
28         return edad;
29     }
30 }
  
```

The screenshot shows the Apache NetBeans IDE interface. The title bar says "Actividad1 - Apache NetBeans IDE 12.4". The left sidebar shows a project structure with packages like "Actividad1", "Test Packages", and "Libraries". The main code editor window displays Java code for a class named "Tools". The code includes methods for validating names and ages. The status bar at the bottom shows system information like weather (19°C), date (16/04/2023), and time (06:16 p.m.).

Repetimos los pasos para la otra librería

Estadísticas



```

 1 package actividad1;
 2
 3 import static actividad1.Graficas.panelJframe;
 4 import java.util.ArrayList;
 5 import org.jfree.chart.ChartFactory;
 6 import org.jfree.chart.JFreeChart;
 7 import org.jfree.chart.plot.PlotOrientation;
 8 import org.jfree.data.category.CategoryDataset;
 9 import org.jfree.data.category.DefaultCategoryDataset;
10
11 public class Estadisticas {
12
13     private static byte tm, th, tc, td, ts, tv, mad, med, tmc, thc, ths,
14         tms, tmd, thd, tmv, thv, mmd, mdm, hmd, hdm;
15     private float ph, pm;
16     private ArrayList<Persona> Invitados = new ArrayList();
17
18     public void añadirPersona(Persona aux) {
19         totalEstado(aux.getEstadoc());
20         totalSexo(aux.getSexo());
21         totalEdad(aux.getEdad());
22         Invitados.add(aux);
23     }
  
```

Como en cualquier código de clase, empieza declarando el paquete después importando las librerías necesarias para el desarrollo del programa como podemos ver importa librerías que no empiezan con java si no con palabras diferentes esto es porque está importando librerías que nosotros momentos antes habíamos importado manualmente seguido de esto sigue la declaración de la clase después

en la línea 13 empiezan la declaración de variables estas como se van a ocupar en un método que veremos más adelante se van a declarar privada estaticas y de tipo byte también tendremos otras de tipo float las cuales se encargarán de guardar los porcentajes por último tenemos la declaración de un array list en el cual estaremos guardando los objetos en este caso personas las cuales están se estarán ingresando por medio de la interfaz gráfica qué desarrollaremos más adelante

Seguido de eso tenemos el método añadir persona el cual recibe como parámetro una variable de tipo persona la cual se obtiene su sexo, edad y estado civil el cual se están pasando como parámetros de otros métodos que se encuentran dentro de la misma clase que en un momento vamos a ver, después de eso procede a añadir al parámetro recibido a el array list

```
24
25  public void eliminarPersona(int a) {
26      Persona aux = Invitados.get(a);
27      char b = aux.getEstadoc().charAt(0);
28      switch (b) {
29          case 'C':
30              tc--;
31              break;
32          case 'D':
33              td--;
34              break;
35          case 'S':
36              ts--;
37              break;
38          case 'V':
39              tv--;
34              break;
40          default:
41      }
42      char c = aux.getSexo().charAt(0);
43      switch (c) {
44          case 'M':
45              tm--;
46              break;
47
48          case 'H':
49              th--;
50              break;
51          default:
52              throw new AssertionError();
53      }
54      byte e = aux.getEdad();
55      if (e >= 18) {
56          mad -= 1;
57      } else {
58          med -= 1;
59      }
60      Invitados.remove(a);
61  }
62
63  public void Actualizar() {
64      for (int i = 0; i < Invitados.size(); i++) {
65          Persona aux = Invitados.get(i);
66          if (aux.getSexo().equals("Mujer")) {
67              char b = aux.getEstadoc().charAt(0);
68              switch (b) {
69                  case 'C':
70                      tmc++;
```

```
70             tmc++;
71             break;
72         case 'D':
73             tmd++;
74             break;
75         case 'S':
76             tms++;
77             break;
78         case 'V':
79             tmv++;
80             break;
81         default:
82
83     }
84     if (aux.getEdad() < 18) {
85         mdm += 1;
86     } else {
87         mmd += 1;
88     }
89 } else {
90     char b = aux.getEstadoc().charAt(0);
91
92     switch (b) {
93         case 'C':
94             thc++;
95             break;
96         case 'D':
97             thd++;
98             break;
99         case 'S':
100            ths++;
101            break;
102        case 'V':
103            thv++;
104            break;
105        default:
106
107    }
108
109    if (aux.getEdad() < 18) {
110        hdm += 1;
111    } else {
112        hmd += 1;
113    }
114 }
115
116 }
```

Tenemos dos métodos, el primero es eliminarPersona el cual recibe un parámetro de tipo int, es int porque está recibiendo número de registro que debe borrar, creamos un objeto de tipo persona y le asignamos la dirección de memoria del registro a borrar, después de ello mediante los métodos get obtiene los datos de la persona y a los totales les empieza a restar según sus datos, por último remueve el elemento de la lista.

El siguiente método que vemos se llama actualizar, actualiza todas las variables de la clase es decir, las cuenta de nuevo, saca de nuevo los porcentajes, cuenta las edades y los totales.

```

117     }
118
119     public void EliminarTodo() {
120         Invitados.clear();
121     }
122
123     public void totalEstado(String a) {
124         char b = a.charAt(0);
125         switch (b) {
126             case 'C':
127                 tc++;
128                 break;
129             case 'D':
130                 td++;
131                 break;
132             case 'S':
133                 ts++;
134                 break;
135             case 'V':
136                 tv++;
137                 break;
138             default:
139         }

```

El metodo eliminarTodo elimina la lista, es decir borra todos los registros del arraylist

El metodo totalEstado lo usamo para calcular el total de personas solteras, casadas, divorciadas o viudas esto lo hace obteniendo la primer letra del estado y comparanco con un switch donde “C” casado, “S” es soltero, “V” viudo y “D” de divorciado

```

140     }
141
142     public void totalSexo(String a) {
143         char b = a.charAt(0);
144         switch (b) {
145             case 'M':
146                 tm++;
147                 break;
148             case 'H':
149                 th++;
150                 break;
151             default:
152                 throw new AssertionError();
153         }
154     }
155
156     public void totalEdad(byte e) {
157         if (e >= 18) {
158             mad += 1;
159         } else {
160             med += 1;
161         }
162     }

```

Los metodos totalSexo y totalEdad se manejan bajo la misma premisa únicamente cambiando los valores correspondientes.

El metodo porcentajes calcula el porcentaje de hombres y mujeres, lo hace aplicando una regla de 3 y despues llenandose bajo una respuesta logica sabemos que si en un lugar solo hay hombres y mujeres, y de esas personas 60% es mujer solamente al

100% le restamos 60% y obtenemos 40%, así no tenemos que volver a hacer la regla de 3 de la proporcionalidad

Despues de ello, tenemos los metodos getters y setters que ya hemos mencionado anteriormente

```
163     }
164
165     public void Porcentajes() {
166         pm = (tm * 100) / invitados.size();
167
168         ph = 100 - pm;
169     }
170
171     public byte getTm() {
172         return tm;
173     }
174
175     public byte getTh() {
176         return th;
177     }
178
179     public byte getTc() {
180         return tc;
181     }
182
183     public byte getTd() {
184         return td;
185     }
```

```
186
187     public byte getTs() {
188         return ts;
189     }
190
191     public byte getTv() {
192         return tv;
193     }
194
195     public byte getTotal() {
196         return (byte) invitados.size();
197     }
198
199     public byte getMad() {
200         return mad;
201     }
202
203     public byte getMed() {
204         return med;
205     }
206
207     public float getPh() {
208         return ph;
```

```

208     return ph;
209 }
210
211     public float getPm() {
212         return pm;
213     }
214
215     public String[] datos() {
216         String datos[] = new String[10];
217         datos[0] = "A la fiesta asistieron " + getTotal() + " personas de las cuales:";
218         datos[1] = "La cantidad de mujeres es: " + getTm();
219         datos[2] = "La cantidad de hombres es: " + getTh();
220         datos[3] = "La cantidad de gente casada es: " + getTc();
221         datos[4] = "La cantidad de gente soltera es: " + getTs();
222         datos[5] = "La cantidad de gente divorciada es: " + getTd();
223         datos[6] = "La cantidad de gente viuda es: " + getTv();
224         datos[7] = "Personas mayores de edad son: " + getMad();
225         datos[8] = "Personas menores de edad son: " + getMed();
226         return datos;
227     }
228
229
230     public CategoryDataset creaDatosCategory() {
231         Actualizar();

```

Con el metodo datos, retorna un vector en el cual en cada posición almacena una cadena con las estadisticas sobre los estados, despues veremos porque se hizo de esta forma y no porque con un simple salto de linea

En el metodo crearDatosCategory() creas los datos y los organizas según tus numeros, es importante entender como ordenar los datos para evitar problemas a la hora de presentar la información en la grafica

```

232     DefaultCategoryDataset datos = new DefaultCategoryDataset(); // se crea el
233
234     datos.setValue(th, "Hombres", "Personas");
235     datos.setValue(tm, "Mujeres", "Personas");
236     if (tms != 0) {
237         datos.setValue(tms, "Mujeres", "Solteras");
238     }
239     if (ths != 0) {
240         datos.setValue(ths, "Hombres", "Solteras");
241     }
242     if (thc != 0) {
243         datos.setValue(thc, "Hombres", "Casadas");
244     }
245     if (tmc != 0) {
246         datos.setValue(tmc, "Mujeres", "Casadas");
247     }
248     if (thd != 0) {
249         datos.setValue(thd, "Hombres", "Divorciadas");
250     }
251     if (tmd != 0) {
252         datos.setValue(tmd, "Mujeres", "Divorciadas");
253     }
254     if (thv != 0) {

```

```

255         datos.setValue(thv, "Hombres", "Viudos");
256     }
257     if (tmv != 0) {
258         datos.setValue(tmv, "Mujeres", "Viudos");
259     }
260     if (mdm != 0) {
261         datos.setValue(mdm, "Mujeres", "Menores de edad");
262     }
263     if (hdm != 0) {
264         datos.setValue(hdm, "Hombres", "Menores de edad");
265     }
266     if (mmd != 0) {
267         datos.setValue(mmd, "Mujeres", "Mayores de edad");
268     }
269     if (hmd != 0) {
270         datos.setValue(hmd, "Hombres", "Mayores de edad");
271     }
272 }
273 return datos;
274 }
275
276 public void grafica3D(CategoryDataset datos) {
277
278     JFreeChart grafico;
279     grafico = ChartFactory.createBarChart3D("Invitados", "Personas",
280                                         "Cantidad", datos, PlotOrientation.VERTICAL, true, true, false);
281     panelJframe(grafico);
282 }
283
284
285 public static void panelJframe(JFreeChart grafica) {
286     // Se crea un Panel para almacenar la grafica generada llamada grafica_Barras, (ChartPanel es el equivalente
287     // de JPanel)
288     ChartPanel panel = new ChartPanel(grafica);
289
290     panel.setMouseWheelEnabled(true);
291     panel.setPreferredSize(new Dimension(500, 300));
292     //Creamos una ventana para presentar la grafica que esta almacenada en el panel
293     JFrame ventana = new JFrame("Grafica"); // titulo del frame
294     ventana.setVisible(true);
295     ventana.setSize(800, 600); // tamaño del JFrame
296     ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
297     ventana.add(panel); // se agrega al JFrame el panel(Grafico)
298 }
299 }
300 }
```

Los ultimos metodos se dedican exclusivamente a la grafica, a hacerla, ponerla en una ventana y mostrarla.

A continuación veremos la clase “Frame” esta clase es donde se desarrolla la interfaz es decir todo lo que ve, observa el usuario, antes de comenzar con el código es importante observar esta tabla de elementos y sus métodos que nos ayudara a entender de mejor forma el código:

Clase	Método	Descripción
JFrame	Frame()	Constructor de la clase JFrame.

	<code>void setTitle(String título)</code>	Establece el título de la ventana con el String especificado
	<code>void setSize(int x, int y)</code>	Cambia el tamaño del componente para que tenga una anchura x y una altura y
	<code>void setResizable(boolean resizable);</code>	Para evitar que se cambie el tamaño de la ventana.
	<code>void setVisible(boolean b)</code>	Muestra u oculta la ventana según el valor del parámetro b.
	<code>void setDefaultCloseOperation (opciones)</code>	Usado para especificar una de las siguientes opciones del botón de cierre: EXIT_ON_CLOSE.
Container	<code>Component add(Component comp)</code>	Añade el componente especificado al final del contenedor.

	<code>void setLayout(LayoutManager mgr)</code>	Establece el layout de la ventana.
Component	<code>void setBounds(int x, int y, int ancho, int alto)</code>	Mueve y cambia el tamaño del componente.
	<code>void addActionListener(this)</code>	Añade un oyente de eventos al componente actual.

Clase	Método	Descripción
JLabel	<code>JLabel()</code>	Constructor de la clase JLabel.
	<code>void setText(String text)</code>	Define una línea de texto que mostrará el componente.
JTextField	<code>JTextField()</code>	Constructor de la clase JTextField.
	<code>String getText()</code>	Retorna el texto contenido en el componente de texto.
JButton	<code>JButton()</code>	Construcción del botón.

	<code>setText(String texto)</code>	Método que establece el texto que tendrá dentro el botón.
ButtonGroup	<code>ButtonGroup()</code>	Construcción del grupo de botones.
	<code>add(Component c)</code>	Método que nos permite añadir botones a un mismo grupo.
	<code>clearSelection()</code>	Método que elimina los botones seleccionados del grupo, dejándolos se inició el programa.
JRadioButton	<code>JRadioButton()</code>	Construcción del botón de opción.
	<code>setText(String texto)</code>	Método que establece el texto que puede tendrá el botón de opciones.
JCheckBox	<code>JCheckBox()</code>	Construcción del botón de la casilla de selección
	<code>setText(String texto)</code>	Método que establece el texto que puede tendrá el botón de opciones.

AbstractButton	isSelected()	Método que devuelve un valor booleano (true o false) si un componente de hijo de esta clase es seleccionado
JList	JList()	Construcción de la lista.
	setSelectionMode(Component c)	Método que establece el modo de selección de la lista.
	setModel(Component c)	Método que establece el modelo de una lista.
DefaultListModel	DefaultListMode()	Construcción del modo de listas por default.
	addElement(Component c)	Método que añade un elemento al modelo establecido.
	clear()	Método que vacía al modelo, dejándolo en su estado por default.
JScrollPane	JScrollPane()	Construcción del panel de scroll.

	<code>setBounds(int x1, int y1, int x2, int y2)</code>	Método que establece la posición y tamaño del panel de scroll.
	<code>setViewportView(Component c)</code>	Método que establece lo que se va a ver a través de este panel de scroll.
<code>ActionListener</code>	<code>ActionListener</code>	Interface que debe ser implementada para gestionar eventos.
	<code>void actionPerformed(ActionEvent e)</code>	Se invoca cuando ocurre un evento.
<code>ItemListener</code>	<code>ItemListener</code>	Interface que debe ser implementada para gestionar eventos.
	<code>itemStateChanged(ItemEvent event)</code>	Se invoca cuando ocurre un evento, es decir se selecciona un item.
<code>ChangeListener</code>	<code>ChangeListener</code>	Interface que debe ser implementada para

		gestionar eventos.
	void stateChanged(ChangeEvent e)	Se invoca cuando ocurre un cambio de estado en un componente que provoca un evento.

Ahora observemos el código e identifiquemos los componentes y sus métodos

Clase Frame

```

1 package actividad;
2
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5 import java.awt.event.ItemEvent;
6 import java.awt.event.ItemListener;
7 import javax.swing.*;
8 import javax.swing.event.ChangeEvent;
9 import javax.swing.event.ChangeListener;
10
11 public class Frame extends JFrame implements ActionListener, ItemListener, ChangeListener{
12
13     private String sexoA, estadoc;
14     private final JLabel nombreL;
15     private final JLabel edadL;
16     private final JLabel sexoL, estado, total;
17     private final JTextField nombre, edad;
18     private final JCheckBox hombre, mujer;
19     private final JRadioButton soltero, casado, viudo, divorciado;
20     private final ButtonGroup grupoBotones = new ButtonGroup();
21     private final ButtonGroup grupoBotones2 = new ButtonGroup();
22     private final JList listaNombres;
23     private final DefaultListModel modelo;
24     private final JScrollPane scrollLista;
25     private final Estadisticas operaciones = new Estadisticas();

```

Empezamos en la línea 1 declarando el paquete después procedemos importar los elementos necesarios para esta clase

Después tenemos la declaración de la clase y podemos ver herencia, la palabra reservada `extends` indica herencia en Java, también tenemos la palabra reservada `implements` se usa para poder acceder o implementar los métodos de una clase, es importante recordar que en Java no hay herencia múltiple pero con la palabra `implements` es una propuesta que nos ofrece.

Despues de ello se declaran las variables de clase en este caso el modificador de acceso sera de tipo private final, es importante notar que no son variables en si, si no objetos, objetos de clases que ya hemos descrito previamente

```
26     setLayout(null);
27     setTitle("Registro");
28     setSize(400, 450);
29     setResizable(false);
30     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31     nombreL = new JLabel("Nombre: ");
32     nombreL.setBounds(10, 10, 200, 40);
33     add(nombreL);
34     nombre = new JTextField();
35     nombre.setBounds(110, 10, 150, 30);
36     add(nombre);
37     edadL = new JLabel("Ingresa tu"
38         + "\n edad: ");
39     edadL.setBounds(10, 50, 200, 40);
40     add(edadL);
41     edad = new JTextField();
42     edad.setBounds(110, 50, 150, 30);
43     add(edad);
44     sexoL = new JLabel("Elige el sexo");
45     sexoL.setBounds(10, 90, 200, 40);
46     add(sexoL);
```

Dentro del constructor de la clase se deben poner todos los elementos que estarán dentro de la parte visual o que tendrán dentro elementos visuales de la interfaz, empezamos con los métodos `setLayout()`, `setTitle()`, etc, al ser métodos que se heredaron se pueden acceder a ellos sin necesidad de crear un objeto de esa clase.

Después los objetos ya declarados, ahora los instanciamos poniendo el nombre del objeto seguido del operador de asignación, que es el igual “=”, después ponemos la palabra reservada “new” junto al constructor de la clase, como podemos ver, cuando pasamos como parámetro una cadena en el constructor esta cadena se convierte en el valor que mostrara el elemento en su visualización.

A continuación en la línea 32 tenemos un ejemplo del método `setBounds()` el cual funciona para cualquier componente, este método asigna el tamaño del componente y también donde estará localizado.

Para terminar con este componente tenemos el método `add()` que pasa como parámetro un componente el cual agregara a la ventana y nos permitirá verlo.

Si el componente no se agrega en el método `add()` no se agregara y no será visible aunque este declarado e instanciado.



Repetimos los mismos pasos con los demás componentes

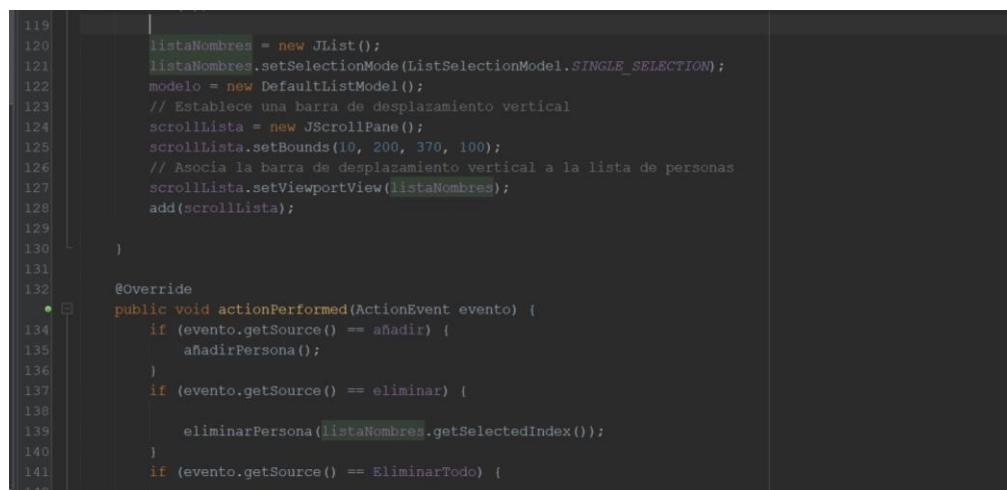
```
50 hombre = new JCheckBox("Hombre");
51 hombre.setBounds(110, 90, 150, 30);
52 hombre.addItemListener(this);
53 grupoBotones.add(hombre);
54 add(hombre);
55
56 mujer = new JCheckBox("Mujer");
57 mujer.setBounds(110, 110, 150, 30);
58 mujer.addItemListener(this);
59 grupoBotones.add(mujer);
60 add(mujer);
61
62 estado = new JLabel("Elige el estado civil: ");
63 estado.setBounds(10, 130, 200, 40);
64 add(estado);
65 setVisible(true);
66
67 soltero = new JRadioButton("Soltero");
68 soltero.setBounds(10, 160, 100, 30);
69 soltero.addChangeListener(this);
70 grupoBotones2.add(soltero);
71 add(soltero);
72
73 casado = new JRadioButton("Casado");
74 casado.setBounds(105, 160, 100, 30);
75 casado.addChangeListener(this);
76 grupoBotones2.add(casado);
77 add(casado);
78
79 divorciado = new JRadioButton("Divorciado");
80 divorciado.setBounds(200, 160, 100, 30);
81 divorciado.addChangeListener(this);
82 grupoBotones2.add(divorciado);
83 add(divorciado);
84
85 viudo = new JRadioButton("Viudo");
86 viudo.setBounds(300, 160, 100, 30);
87 viudo.addChangeListener(this);
88 grupoBotones2.add(viudo);
89 add(viudo);
90
91 añadir = new JButton();
92 añadir.setText("Alta");
93 añadir.setBounds(10, 330, 80, 23);
94 añadir.addActionListener(this);
95 add(añadir);
96
97 eliminar.setBounds(100, 330, 80, 23);
98 eliminar.addActionListener(this);
99 total = new JLabel();
100 total.setText("No.Registro:");
101 total.setBounds(20, 300, 135, 23);
102 add(total);
103
104 add(eliminar);
105 // Establece el botón Borrar list
106 EliminarTodo = new JButton();
107 EliminarTodo.setText("Limpia lista");
108 EliminarTodo.setBounds(190, 330, 120, 23);
109 EliminarTodo.addActionListener(this);
110 add(EliminarTodo);
111
112 e = new JButton();
113 e.setText("Estadisticas");
114 e.setBounds(10, 360, 120, 23);
115 e.addActionListener(this);
116 add(e);
117
118 listaNombres = new JList();
119 listaNombres.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

Podemos ver dos nuevos elementos que son los elementos jCheckBox el, el metodo addItemClickListener() le permite tener un escuchador de eventos el cual se activa invoca el metodo ItemListener() cada vez que cambia la selección

El metodo addChangeListener() y el metodo que invoca ChangeListener() funcionan de la misma manera.

Los ButtonGroup nos sirve para que dentro de un conjunto de botones solamente uno pueda ser seleccionado a la vez

El metodo addActionListener() le proporciona al elemento un escuchador al componente en este caso JButton, cuando algun boton sea presionado llamara al metodo ActionPerformed()



```
119
120     listaNombres = new JList();
121     listaNombres.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
122     modelo = new DefaultListModel();
123     // Establece una barra de desplazamiento vertical
124     scrollLista = new JScrollPane();
125     scrollLista.setBounds(10, 200, 370, 100);
126     // Asocia la barra de desplazamiento vertical a la lista de personas
127     scrollLista.setViewportView(listaNombres);
128     add(scrollLista);
129 }
130
131
132 @Override
133 public void actionPerformed(ActionEvent evento) {
134     if (evento.getSource() == añadir) {
135         añadirPersona();
136     }
137     if (evento.getSource() == eliminar) {
138
139         eliminarPersona(listaNombres.getSelectedIndex());
140     }
141     if (evento.getSource() == EliminarTodo) {
142 }
```

Un JList es un componente en java que le presenta al usuario un grupo de ítems para elegir. Los ítems pueden ser cualquier object, aquí podemos ver la implementación de sus métodos ya explicados

En la línea 132 vemos una syntaxis diferentes, ese “@Override” nos sirve para la sobre escritura de métodos, este método será invocado cada vez sea presionado un JButton el cual recibe como parámetro un objeto de tipo ActionEvent, para saber que botón fue el que lo invoco se usa getSource() que nos devuelve el componente por el cual fue llamado y según el caso se invocaran métodos diferentes

```

143     if (evento.getSource() == e) {
144         Datos();
145     }
146 }
147 }
148 }
149 }
150 }
151 @Override
152 public void itemStateChanged(ItemEvent event) {
153
154     if (hombre.isSelected()) {
155
156         sexoa = "Hombre";
157     }
158     if (mujer.isSelected()) {
159
160         sexoa = "Mujer";
161     }
162 }
163
164 }
```

Al seleccionar un CheckBox mandamos a llamar el método `itemStateChanged()` el cual usando el método `isSelected` preguntara cual de los elementos fue elegido y así dar valor a la variable “`sexoa`” de tipo string

```

165     @Override
166     public void stateChanged(ChangeEvent e) {
167         if (soltero.isSelected()) {
168             estadoc = "Soltero";
169         }
170         if (casado.isSelected()) {
171             estadoc = "Casado";
172         }
173         if (divorciado.isSelected()) {
174             estadoc = "Divorciado";
175         }
176         if (viudo.isSelected()) {
177             estadoc = "Viudo";
178         }
179     }
180 }
181
182     public void añadirPersona() {
183         Persona p = new Persona();
184         p.setNombre(nombre.getText());
185         p.setEdad(Tools.validEdad(edad.getText()));
186         p.setSexo(sexo);
187         p.setEstadoc(estadoc);
188         operaciones.añadirPersona(p);
```

El método `stateChanged()` funciona de la misma manera que el `ItemStateChanged()`

El método `añadirPersona` se invoca cuando el botón alta es presionado

El método crea un objeto de tipo persona el cual por medio del método “`añadirPersona`” de la clase “`Estadísticas`” es agregado al `arrayList`

```

189
190     String cad = nombre.getText() + " " + edad.getText() + " "
191     + sexoa + " " + estadoc;
192     modelo.addElement(cad);
193
194     listaNombres.setModel(modelo);
195     int x = modelo.getSize();
196     total.setText("No.Registro:" + x);
197     edad.setText("");
198     nombre.setText("");
199
200     grupoBotones.clearSelection();
201     grupoBotones2.clearSelection();
202
203 }
204
205 □ public void eliminarPersona(int a) {
206     if (a >= 0) {
207         modelo.removeElementAt(a);
208
209         operaciones.eliminarPersona(a);
210         int x = modelo.getSize();
211         total.setText("No.Registro:" + x);

```

Después de eso crea una cadena con los datos recolectados y lo agrega como un elemento al model “modelo” y asu vez este se le asigna al JList, a continuación el programa procede a limpiar las selecciones echas para así poder crear otro registro.

El método eliminar persona recibe como parámetro un número, este número hace referencia al item del JList que se quiere eliminar, si el número es mayor o igual cero procede a eliminar al elemento del modelo por medio del método removeElementAt(), procede a llamar al método eliminarPersona() de la clase estadísticas para así eliminar a la persona de la lista de invitados, en caso de no haber seleccionado un elemento aparecer un cuadro de dialogo informándole de su error

```

212     } else {
213         JOptionPane.showMessageDialog(null, "Debe seleccionar un elemento");
214     }
215 }
216
217 □ public void EliminarTodo() {
218     operaciones.EliminarTodo();
219     modelo.clear();
220     int x = modelo.getSize();
221     total.setText("No.Registro:" + x);
222 }
223 int count=0;
224 □ public void Datos() {
225
226     operaciones.Porcentajes();
227     if (count==0) {e.setText("Ocultar estadísticas");
228
229     count+=1;
230     agregarEstadisticas();
231     operaciones.grafica3D(operaciones.creaDatosCategory());
232     }else{e.setText("Mostrar estadísticas");
233     eliminarEstadisticas();
234     count-=1;}

```

El método eliminarTodo se invoca cuando el botón con ese mismo nombre es presionado, lo que hace es limpiar el model y después procede a limpiar el arrayList de los invitados por medio del método EliminarTodo() de la clase estadísticas.

El método datos se manda a llamar cuando es presionado el botón estadísticas, en principio actualiza todos los contadores de la clase Estadísticas después procede a preguntar si es la primera vez que se presiona mostrar las estadísticas, si es la segunda, las oculta, después manda llamar al método agregarEstadisticas() y por ultimo muestra la gráfica.

Para eliminar las estadísticas del model, llama al método eliminarEstadisticas()

```
235     }
236     public void agregarEstadisticas(){
237
238         String aux[] = operaciones.datos();
239         for (String aux1 : aux) {
240             modelo.addElement(aux1);
241         }
242     }
243
244     public void eliminarEstadisticas(){
245         String aux[] = operaciones.datos();
246         for (int i = aux.length; i >= 1; i--) {
247             modelo.removeElementAt(i);
248         }
249     }
250 }
251
252
```

El método agregarEstadísticas() crea un arreglo con los valores del arreglo que retorna el método datos() de la clase “Estadísticas” y los agrega elemento por elemento al JList para que sean visibles ya que si se agregan como una sola cadena aunque tengan salto de línea, lo escribe de corrido.

El método eliminarEstadisticas() se encarga de eliminar elemento por elemento las estadísticas del JList según el tamaño del arreglo.

Clase Main

La clase principal arranca la aplicación crea un objeto de tipo Frame y llama al constructor iniciando toda la interfaz grafica

La clase empieza en la línea 1 con la declaración del paquete, después la clase, y por ultimo el método principal

```
1 package actividad;
2
3 public class Actividad {
4
5     public static void main(String[] args) {
6
7         Frame a=new Frame();
8
9     }
10
11 }
12
13
14
```

Practica Alumnos:

Clase	Método	Descripción
JFrame	Frame()	Constructor de la clase JFrame
	void setTitle(String titulo)	Establece “Captura de datos” como el título de la ventana
	void setSize(int x, int y)	Cambia el tamaño del componente para que tenga una anchura de 400 y una altura 450
	void setLocationRelativeTo(Component c)	Establece la ubicación de la ventana en relación con el componente especificado
	void setDefaultCloseOperation(options)	Usado para especificar la función del botón de cierre: EXIT_ON_CLOSE
	void setResizable(boolean resizable)	Evita que el tamaño de la ventana sea cambiado

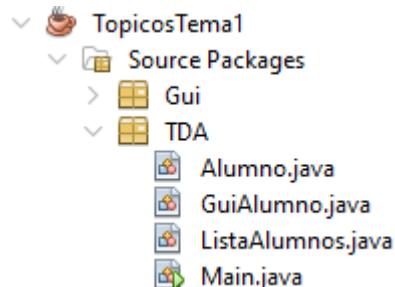
	<code>void setVisible(boolean b)</code>	Muestra la ventana en la pantalla del usuario
ActionListener	ActionListener	La interface debe ser implementada para gestionar eventos
	<code>void actionPerformed (actionEvent e)</code>	Se invoca cuando ocurre un evento
Container	Container <code>getContentPane()</code>	Retorna un objeto ContentPane de la ventana
	<code>void setLayout (LayoutManager mgr)</code>	Establece el layout de la ventana
	<code>Component add (Component comp)</code>	Añade el componente especificado al final del contenedor
Component	<code>void setBounds(int x, int y, int ancho, int alto)</code>	Establece las coordenadas y tamaños del componente
	<code>void addActionListener(this)</code>	Añade un oyente de eventos al componente actual
JLabel	JLabel()	Constructor de la clase JLabel
	<code>void setText(String text)</code>	Define la línea de texto que mostrara el componente
JTextField	JTextField()	Constructor de la clase JTextField
	<code>String getText()</code>	Retorna el texto contenido en el componente de texto.
JButton	JButton()	Constructor de la clase JButton
	<code>void setText(String text)</code>	Define una línea de texto que mostrara este componente
JList	JList()	Constructor de la clase JList

	<code>void setText(String text)</code>	Define línea de texto que mostrara este componente
	<code>void setSelectionMode(int modoSeleccion)</code>	Establece el modo de selección de la lista
	<code>void setModel (ListModel<E>model)</code>	Establece el modelo que representa el contenido de la lista
	<code>int getSelectedIndex()</code>	Devuelve el índice seleccionado
DefaultListModel	<code>DefaultListModel()</code>	Constructor de la clase DefaultListModel
	<code>void addElement (Element e)</code>	Añade un componente específico al final de la lista
	<code>void removeElementAt(int índice)</code>	Elimina el componente seleccionado de la lista.
	<code>void clear()</code>	Elimina todos los elementos de los elementos de la lista
JScrollPane()	<code>JScrollPane()</code>	Constructor de la clase JScrollPane.
	<code>void setViewportView (Component view)</code>	Crea una ventana grafica si es necesario y luego establece la vista

El programa realizado para ayudar a llevar un control al almacenar datos como el número de control, nombre, semestre y edad de un grupo de alumnos con diferentes métodos

Para comenzar con nuestra practica creamos un proyecto que se nombre “TopicosTema1” se crearan 2 package llamados “Gui” “TDA” en esta practica solo utilizaremos el package llamado “TDA”. Creamos 4 clases en el package TDA, y las llamamos:

“Alumno”, “GuiAlumno”, “ListaAlumnos”, “Main”



Comenzamos en la clase Alumno declarando nuestras variables correspondientes para almacenar el número de control, el nombre, la edad y el semestre del alumno, comenzamos declarando variables privadas de tipo String llamadas “numCtrl” y “nomAlu”, de la misma manera se declaran variables privadas de tipo byte llamadas “semAlu” y “edadAlu”. Posteriormente se crea el constructor de la clase Alumno y se crean los métodos setter y getters

```
package TDA;
public class Alumno {
    private String numCtrl;
    private String nomAlu;
    private byte semAlu;
    private byte edadAlu;

    public Alumno() {
    }

    //Alt+insert constructor
    public Alumno(String numCtrl, String nomAlu, byte semAlu, byte edadAlu) {
        this.numCtrl = numCtrl;
        this.nomAlu = nomAlu;
        this.semAlu = semAlu;
        this.edadAlu = edadAlu;
    }
}
```

```
public String getNumCtrl() {
    return numCtrl;
}

public void setNumCtrl(String numCtrl) {
    this.numCtrl = numCtrl;
}

public String getNomAlu() {
    return nomAlu;
}

public void setNomAlu(String nomAlu) {
    this.nomAlu = nomAlu;
}

public byte getSemAlu() {
    return semAlu;
}

public void setSemAlu(byte semAlu) {
    this.semAlu = semAlu;
}

public byte getEdadAlu() {
    return edadAlu;
}

public void setEdadAlu(byte edadAlu) {
    this.edadAlu = edadAlu;
}
```

Posteriormente a esto pasamos a la clase ListaAlumnos, en esta clase usaremos el método ArrayList y crearemos los métodos llamados “añadir”, “eliminar” y “eliminarTodo” que usaremos en la siguiente clase.

```
package TDA;
import java.util.ArrayList;
import TDA.Alumno;
public class ListaAlumnos {
    ArrayList <Alumno> Lista;
    public ListaAlumnos() {
        Lista = new <Alumno> ArrayList();
    }
    public void añadir (Alumno p) {
        Lista.add(e: p);
    }
    public void eliminar(int i) {
        Lista.remove(index: i);
    }
    public boolean eliminarTodo() { //Elimina toda la lista
        return Lista.removeAll(c: Lista);
    }
}
```

Pasamos a la clase GuiAlumno, se declara la clase publica de nombre GuiAlumno y seguido de esto encontramos un término que aprendimos en el tema de herencia, encontramos la palabra reservada extend que indica herencia y seguido de esto tenemos la palabra reservada implements se usa para poder acceder o implementar los métodos de una clase. Se crean las variables de esta clase todas las variables serán privadas, las variables son de tipo ListaAlumnos, Container, JLabel,JTextField, JButton, JList, DefaultListModel y JScrollPane.

Se crea el constructor y se comienza a crear la interfaz, con el método setTitle se le coloca el nombre “Captura datos” como título de la ventana, con el método serSize se le da un tamaño de 400,450 a la interfaz, con el método setLocacionRelativeTo le damos el valor null que significa que la ventana de la interfaz aparecerá en el centro de la pantalla, con el método setDefaultCloseOperation nos especifica que su función es la opción EXIT_ON_CLOSE que significa que su función es cerrar la ventana.

```
public class GuiAlumno extends JFrame implements ActionListener {
    private ListaAlumnos lista;
    private Container contenedor;
    private JLabel edad, total, numCtrl, semestre, nombre;
    private JTextField campoNumero, campoNombre, campoSemestre, campoEdad, campoTotal ;
    private JButton añadir, eliminar, EliminarTodo;
    private JList listaNombres; // Lista de alumnos
    private DefaultListModel modelo; // Objeto que modela la lista
    private JScrollPane scrollLista; // Barra de desplazamiento vertical

    public GuiAlumno(){
        lista = new ListaAlumnos(); // Crea la lista de alumnos
        inicio();
        setTitle(title: "Captura datos ");
        setSize(width: 400, height: 450);
        setLocationRelativeTo(c: null);
        setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
        setResizable(resizable: false);
    }
}
```

Comenzamos con el método privado inicio

El `getContentPand()` es un método que te retorna una instancia de un `JFrame` y por otro lado el método `setLayout()` nos permite establecer un Layout sobre el `JFrame` al que se aplica.

Posterior mente se creara la línea de texto con la clase `JLabel` que en el interfaz le mostrara al usuario donde colocar el número de control y un pequeño contenedor con la clase `JTextField` para colocar la información y nuestro programa lo guarde y almacene, esto mismo ocurre con las líneas de texto asignadas para el nombre, el semestre, la edad del alumno y el número total de alumnos que se ingresan en la máquina, con ayuda del método `setBounds` se colocan en posiciones diferentes sin obstruir una línea a la otra.

```
private void inicio() {
    contenedor = getContentPane();
    contenedor.setLayout(null);
    // Asignacion para el numero de control
    numCtrl = new JLabel();
    numCtrl.setText(text: "Num.Control:");
    numCtrl.setBounds(x: 10, y: 10, width: 100, height: 30);
    campoNumero = new JTextField();
    campoNumero.setBounds(x: 90, y: 10, width: 150, height: 30);

    // Asignacion para el nombre
    nombre = new JLabel();
    nombre.setText(text: "Nombre:");
    nombre.setBounds(x: 10,y: 45,width: 100,height: 30); |
    campoNombre = new JTextField();
    campoNombre.setBounds(x: 90,y: 45,width: 150,height: 30);

    // Asignacion para el semestre
    semestre = new JLabel();
    semestre.setText(text: "Semestre:");
    semestre.setBounds(x: 10,y: 80,width: 100,height: 30);
    campoSemestre = new JTextField();
    campoSemestre.setBounds(x: 90,y: 80,width: 150,height: 30);

    // Asignacion para la edad
    edad = new JLabel();
    edad.setText(text: "Edad:");
    edad.setBounds (x: 10,y: 115,width: 100,height: 30);
    campoEdad = new JTextField();
    campoEdad.setBounds(x: 90,y: 115,width: 150,height: 30);
```

```
//Asignacion para numero de registro
total=new JLabel();
total.setText(text: "No.Registro:");
total.setBounds(x: 20, y: 260, width: 135, height: 23);
campoTotal=new JTextField();
campoTotal.setBounds(x: 90, y: 260, width: 135, height: 23);
```

Se crean 3 botones para dar de alta a el alumno, eliminara un alumno de la lista y borrar la lista completa, el usuario leerá en los botones “Alta”, “Eliminar”, “Limpieza”, el método setBounds en este caso nos ayuda para 2 cosas, para determinar la posición del botón en la ventana y así evitar que los botones se encimen y se obstruyan entre sí, y para determinar el largo y el ancho que tendrán nuestros botones

```
//Crear y asignar el boton
añadir = new JButton();
añadir.setText(text: "Alta");
añadir.setBounds(x: 10, y: 300, width: 80, height: 23);
añadir.addActionListener(l: this);
// Establece el boton Eliminar Alumno
eliminar= new JButton();
eliminar.setText(text: "Eliminar");
eliminar.setBounds(x: 100, y: 300, width: 80, height: 23);
eliminar.addActionListener(l: this);
// Establece el boton Borrar list
EliminarTodo= new JButton();
EliminarTodo.setText(text: "Limpia lista");
EliminarTodo.setBounds(x: 190, y: 300, width: 120, height: 23);
EliminarTodo.addActionListener(l: this);
```

```
// Establece la lista grafica de Alumnos
listaNombres = new JList();
listaNombres.setSelectionMode(selectionMode: ListSelectionModel.SINGLE_SELECTION);
modelo = new DefaultListModel();
// Establece una barra de desplazamiento vertical
scrollLista = new JScrollPane();
scrollLista.setBounds(x: 10, y: 150 ,width: 350, height: 100);
// Asocia la barra de desplazamiento vertical a la lista de personas
scrollLista.setViewportView(view: listaNombres);
```

Las siguientes líneas contenedor.add(comp) quiere decir que son todos los datos que el contenedor imprimirá posteriormente a ingresar los datos del alumno y presionar el botón de “Alta”, el método setVisible como su nombre lo indica es quien se encarga de que el usuario pueda o no ver la ventana del interfaz, si se coloca false en el paréntesis la ventana no será visible, pero si se le coloca true como en este caso la ventana estará visible para el usuario.

```
// Se añade cada componente grafico al contenedor de la ventana
contenedor.add(comp: numCtrl);
contenedor.add(comp: campoNumero);
contenedor.add(comp: nombre);
contenedor.add(comp: campoNombre);
contenedor.add(comp: semestre);
contenedor.add(comp: campoSemestre);
contenedor.add(comp: edad);
contenedor.add(comp: campoEdad);
contenedor.add(comp: total);
contenedor.add(comp: añadir);
contenedor.add(comp: eliminar);
contenedor.add(comp: EliminarTodo);
contenedor.add(comp: scrollLista);

setVisible(b: true);

}
```

Se crea el método ActionPerformed(ActionEvent evento), este método consiste en darle uso a los botones con ayuda de los métodos del ArrayList que se utilizo en la clase ListaAlumno, si se selecciona el botón “Alta” el programa ejecutara el método añadir y lo llevara al método añadirAlumno, si se selecciona el botón de “Eliminar” ejecutara el método de la clase ListaAlumno llamado eliminar y posterior a eso ejecutara el método eliminarAlumno y si se selecciona el botón “Limpiar lista” el programa ejecutara el metodo EliminarTodo de la clase ListaAlumno y luego regresara a la clase eliminarLista.

```
public void actionPerformed(ActionEvent evento) {
    if (evento.getSource() == añadir) { // Si se pulsa el botón añadir
        añadirAlumno(); // Se invoca añadir Alumno
    }
    if (evento.getSource() == eliminar) { // Si se pulsa el botón eliminar
        /* Se invoca el método eliminarNombre que elimina el elemento seleccionado */
        eliminarAlumno(indice: listaNombres.getSelectedIndex());
    }
    if (evento.getSource() == EliminarTodo) { /* Si se pulsa el botón borrar lista */
        eliminarLista(); // Se invoca borrar lista
    }
}
```

```
private void añadirAlumno() {
    /* Se obtienen los campos de texto ingresados y se crea una Alumno */
    Alumno p = new Alumno(numCtrl:campoNumero.getText(), nomalu: campoNombre.getText(),
    (Byte.parseByte(: campoSemestre.getText())), (Byte.parseByte(: campoEdad.getText())));
    lista.add(p); /* Se añade una persona al vector de
    personas */
    String cad = campoNombre.getText() + " " + campoNumero.getText() + " " +
    campoSemestre.getText() + " " + campoEdad.getText();
    modelo.addElement(element:cad); /* Se agrega el texto con los
    datos del Alumno al JList */
    listaNombres.setModel(model: modelo);
    int x = modelo.getSize();
    total.setText("No. Registro:" +x);
    // Se colocan todos los campos de texto nulos, para limpiar el área
    campoNumero.setText(t: "");
    campoNombre.setText(t: "");
    campoSemestre.setText(t: "");
    campoEdad.setText(t: "");
    campoTotal.setText(t: "");
}
```

Por último en la clase “Main” se importa la clase GuiAlumno, se crea un objeto de tipo GuiAlumno y llama al constructor de la misma clase iniciando toda la interfaz grafica

```

package TDA;

import TDA.GuiAlumno;

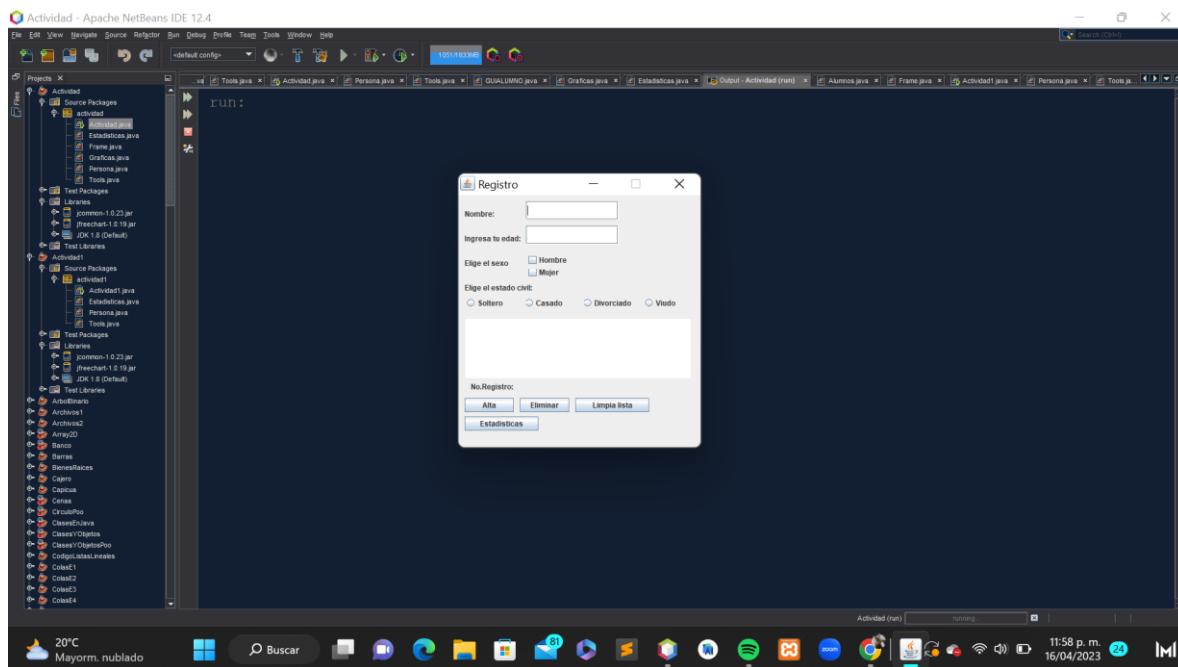
public class Main {

    public static void main (String [] args) {
        GuiAlumno miVentana= new GuiAlumno ();
    }
}

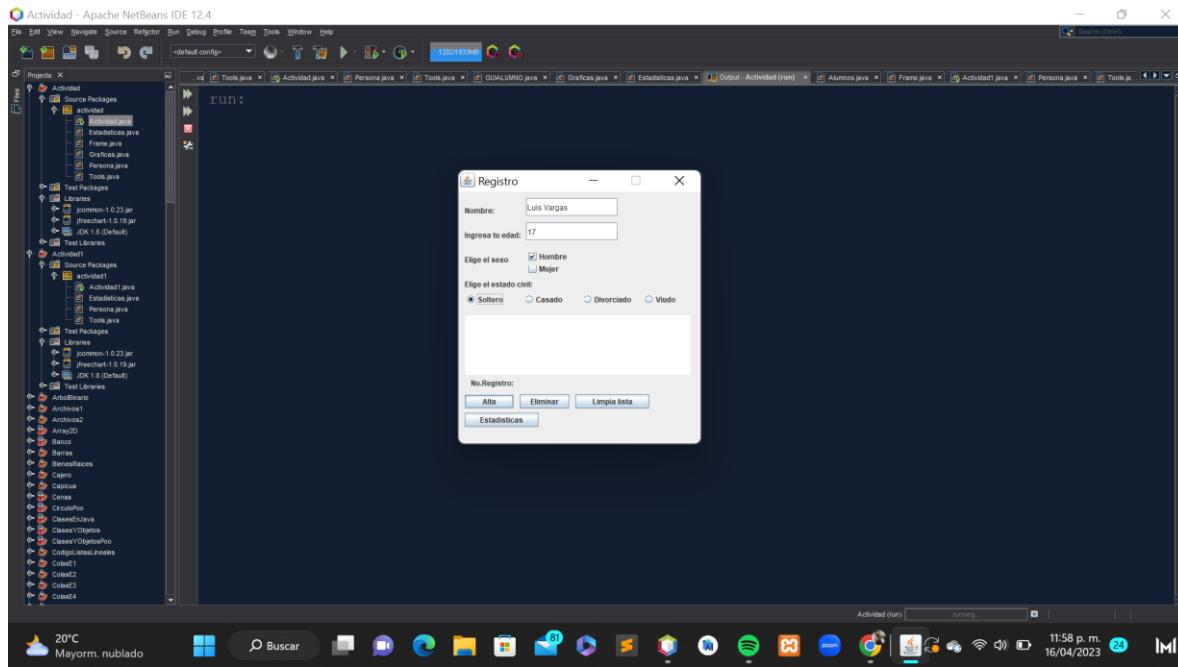
```

RESULTADOS

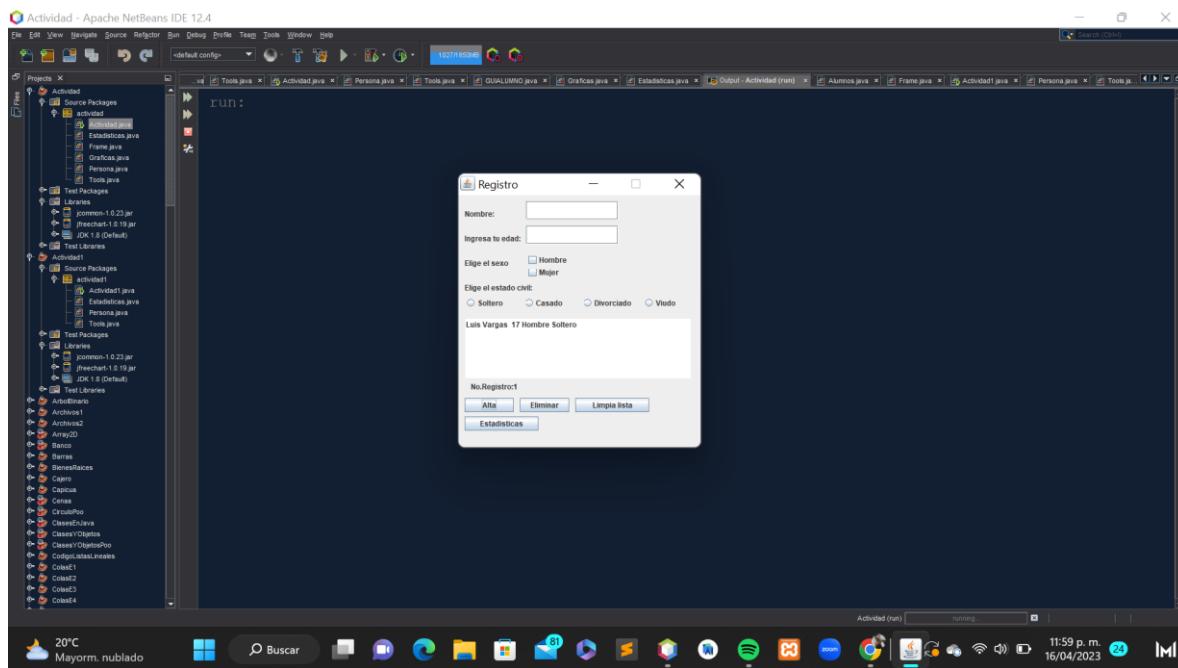
Practica Invitados:



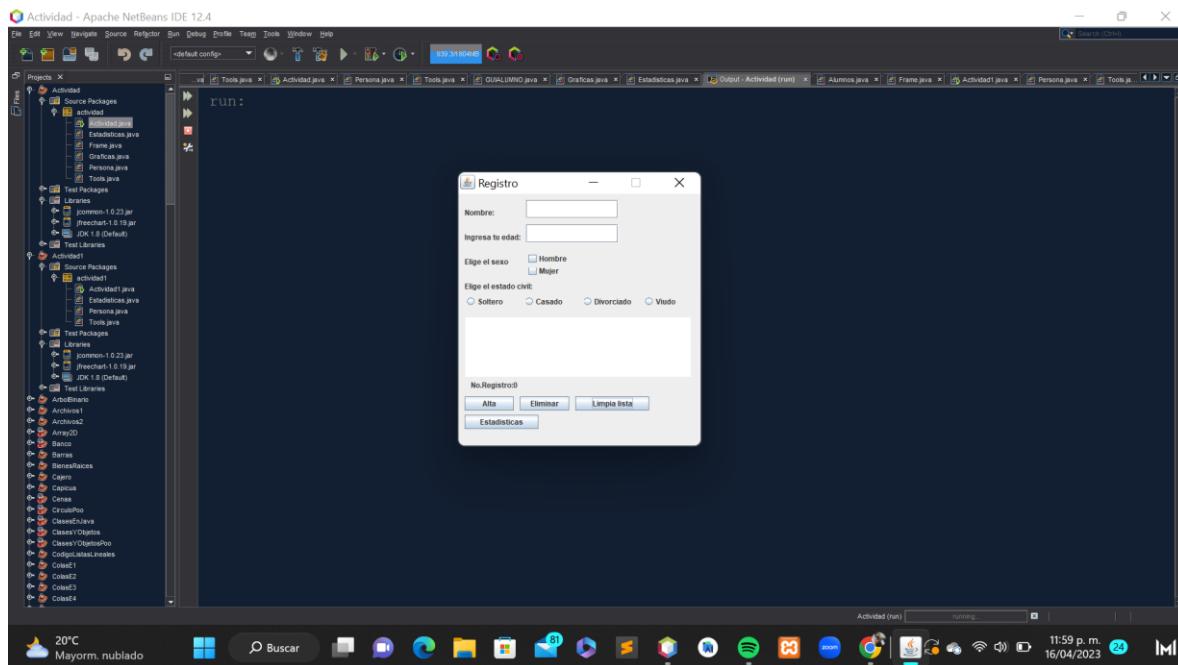
Aquí se muestra la interfaz



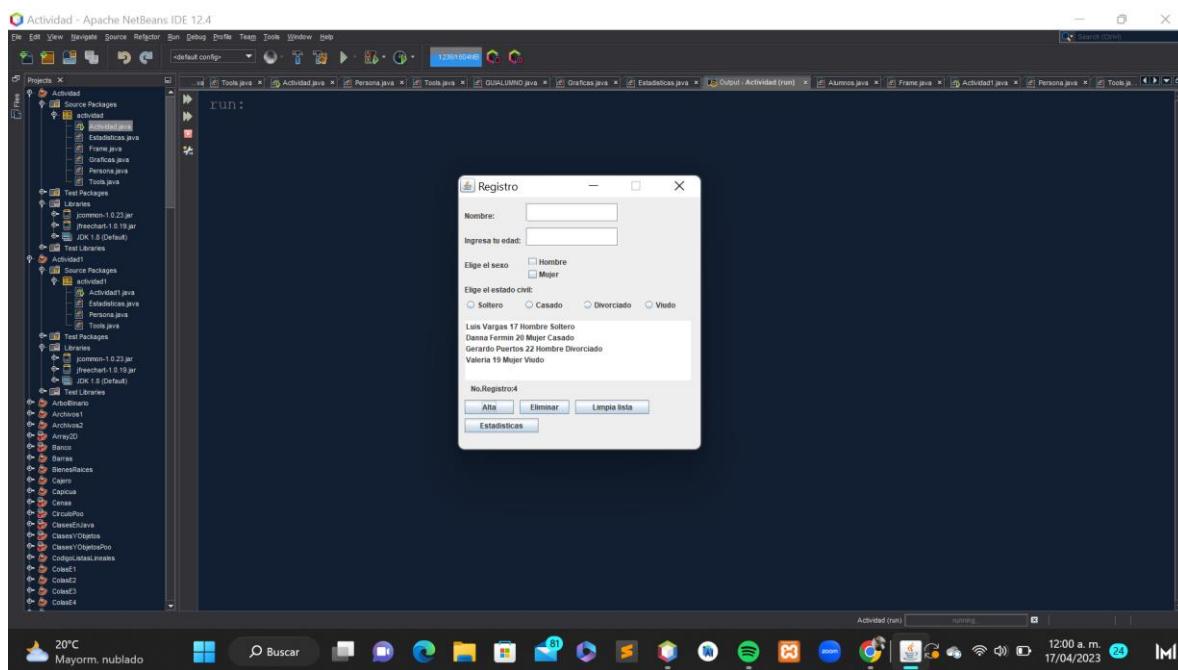
Se ingresan los datos



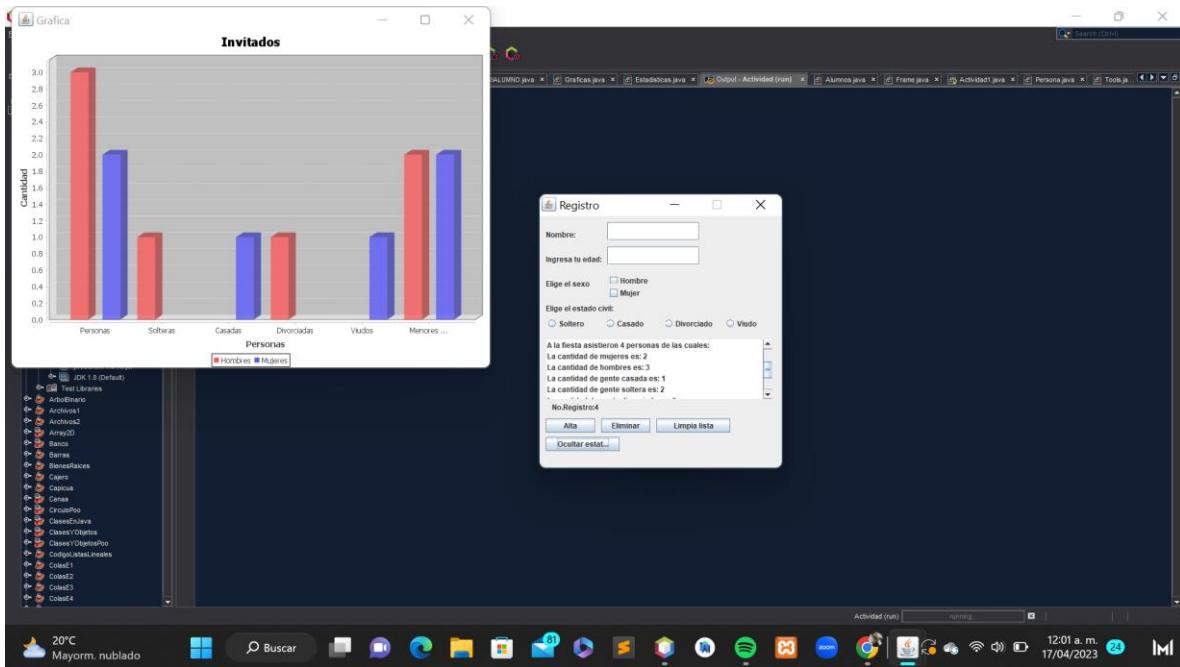
Se hace el alta, es decir el registro



Limpiamos la lista



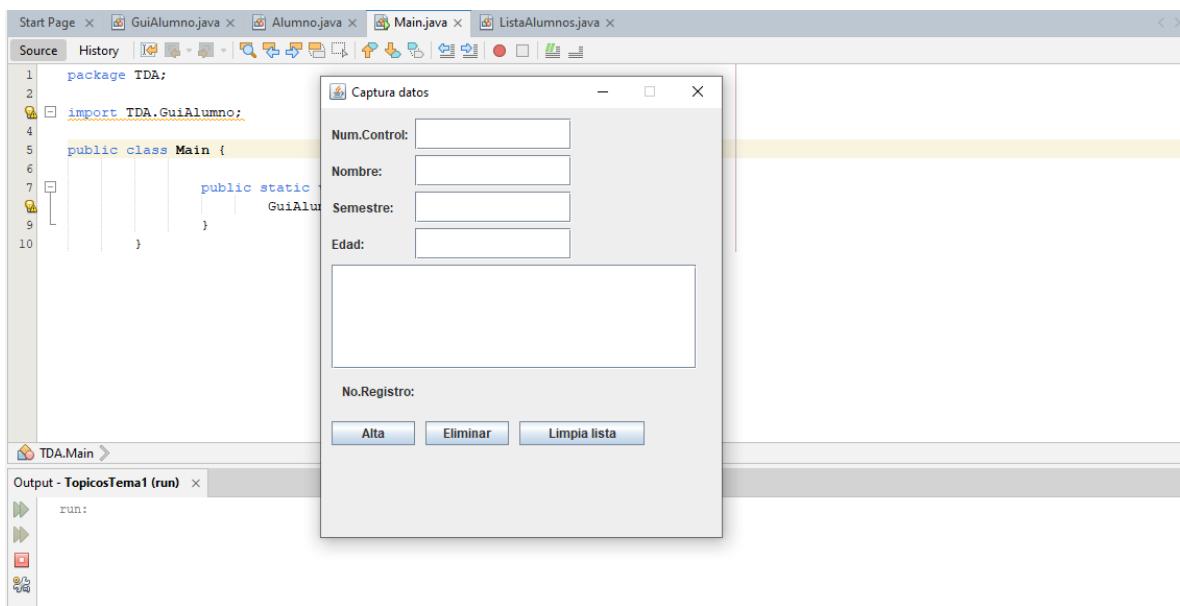
Ingresamos mas registros



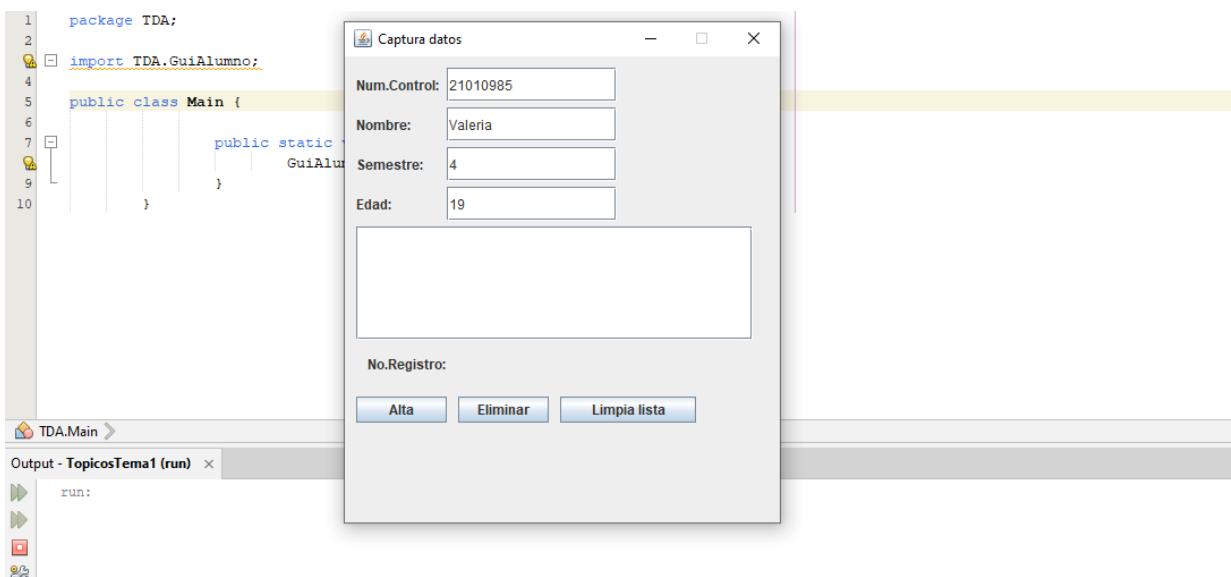
Mostramos las gráficas y las estadísticas

Practica Alumnos:

Así se ve el interfaz grafico del usuario de nuestro programa



Agregamos datos



Se hace el alta y se hace el registro



```
Source History | ☰ 🔍 ↻ ↺ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋
```

```
1 package TDA;
2
3 import TDA.GuiAlumno;
4
5 public class Main {
6
7     public static void main(GuiAlumno arg0) {
8         }
9     }
10 }
```

Captura datos

Num.Control:

Nombre:

Semestre:

Edad:

Valeria 21010985 4 19

No.Registro:1

Alta Eliminar Limpia lista

Agregamos un par de alumnos más y seleccionamos para borrar

The screenshot shows a Java IDE interface with a source code editor and a running application window.

Source Editor:

```
1 package TDA;
2
3 import TDA.GuiAlumno;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         GuiAlumno.main();
9     }
10}
```

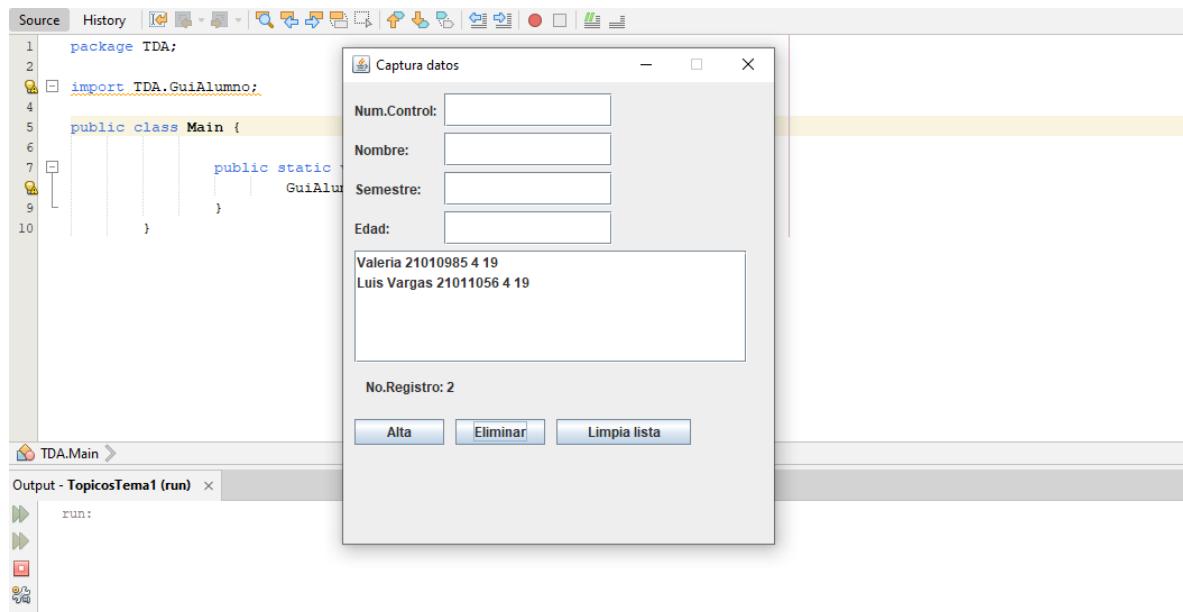
Running Application (Captura datos):

This window contains four text input fields labeled "Num.Control:", "Nombre:", "Semestre:", and "Edad:". Below these fields is a scrollable list box displaying student records:

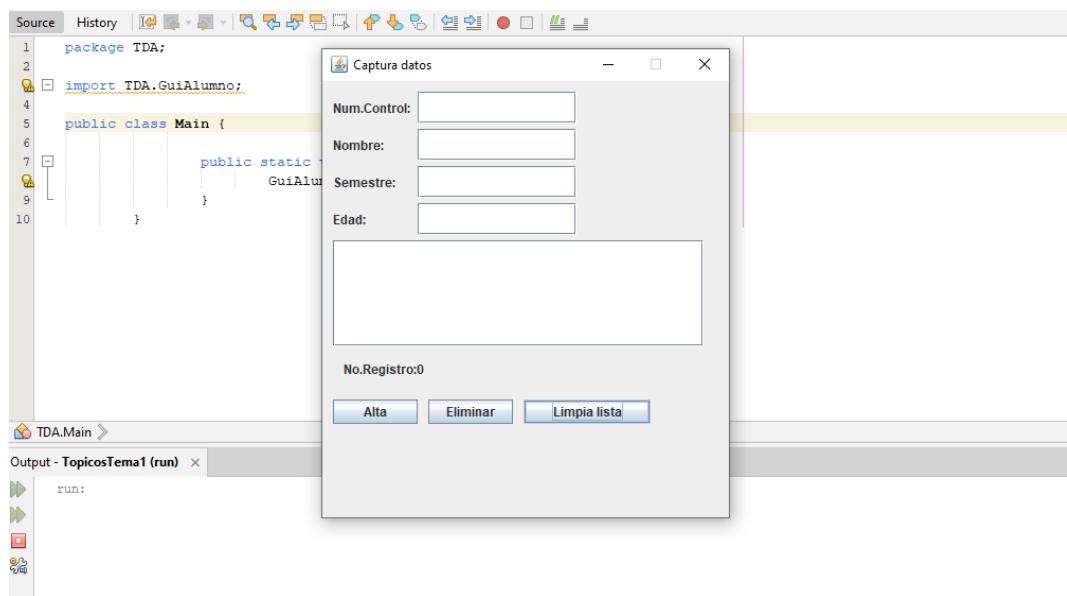
- Valeria 21010985 4 19
- Luis Mavil 21010994 4 19
- Luis Vargas 21011056 4 19

At the bottom of the window, there is a label "No.Registro:3" and three buttons: "Alta", "Eliminar", and "Limpia lista".

Eliminamos



Limpiamos lista



La interfaz queda sin ningún registro

Conclusión

En conclusión, los componentes y las bibliotecas son herramientas fundamentales en el desarrollo de software moderno. Los componentes son piezas modulares de código que se pueden reutilizar en diferentes partes de una aplicación o incluso en diferentes aplicaciones, lo que ahorra tiempo y esfuerzo en el proceso de desarrollo. Las bibliotecas, por otro lado, son conjuntos de código prescrito que tienen funcionalidades específicas y complejas a las aplicaciones.

El uso de componentes y bibliotecas permite a los desarrolladores crear aplicaciones de alta calidad de manera más eficiente, ya que pueden funcionar en la lógica de la aplicación en lugar de tener que crear todos los elementos desde cero. Además, al utilizar componentes y bibliotecas probados y ampliamente utilizados, los desarrolladores pueden confiar en la calidad y confianza del código subyacente.

Sin embargo, es importante tener en cuenta que el uso de componentes y bibliotecas también puede presentar desafíos, como la compatibilidad entre versiones y la dependencia de terceros. Por lo tanto, es importante seleccionar cuidadosamente los componentes y bibliotecas a utilizar y mantenerlos actualizados regularmente para garantizar la seguridad y la eficacia de la aplicación.

En resumen, los componentes y bibliotecas son una parte esencial del proceso de desarrollo de software moderno y pueden mejorar significativamente la eficiencia y la calidad del resultado final, siempre y cuando se utilicen de manera cuidadosa y responsable.