

TALLER DE DESARROLLO DE APLICACIONES 1

SEMANA: 5

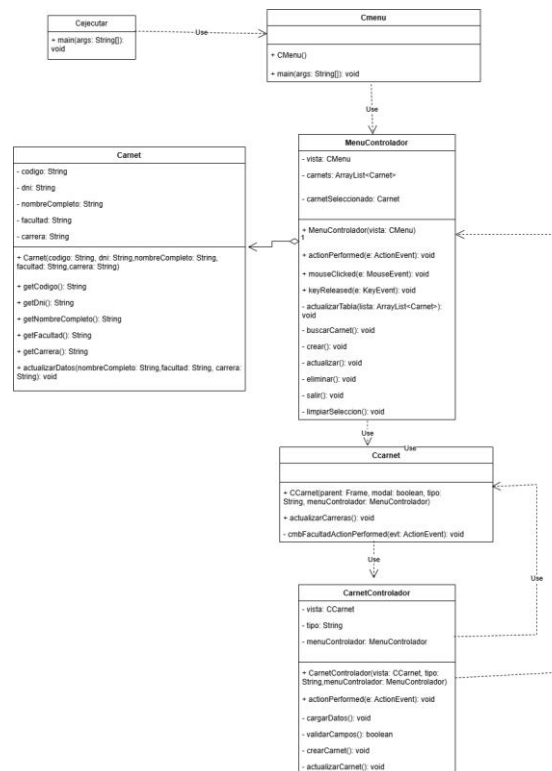
ENTREGADO POR: Mescua Segovia Marino

CÓDIGO: N04058C

CATEDRÁTICO: Fernández Bejarano Raúl Enrique

ACTIVIDAD 1:

DIAGRAMA DE CLASES:



DISEÑO DE LA APLICACIÓN:

CALCULAR CUADRADO

DNI :

CODIGO:

FACULTAD:

Ciencias administrativas y contables

▼

NOMBRES:

APELLIDOS:

CARRERA:

▼

CREAR

VOLVER

SISTEMA CARNET

BUSCA SU CARNET:

Title 1	Title 2	Title 3	Title 4

CREAR

ACTUALIZAR

ELIMINAR

SALIR

CÓDIGO DE LA APLICACIÓN:

```

14 public class CarnetControlador implements ActionListener {
15
16     private final CCarnet vista;
17     private final String tipo;
18     private final MenuControlador menuControlador;
19
20     public CarnetControlador(CCarnet vista, String tipo, MenuControlador menuControlador) {
21         this.vista = vista;
22         this.tipo = tipo;
23         this.menuControlador = menuControlador;
24
25         this.vista.btnCrear.addActionListener(this);
26         this.vista.btnVolver.addActionListener(this);
27
28         if (tipo.equalsIgnoreCase("actualizar")) {
29             cargarDatos();
30         }
31     }
32
33     private void cargarDatos() {
34         Carnet carnet = menuControlador.getCarnetSeleccionado();
35         if (carnet != null) {
36             vista.txtCodigo.setText(carnet.getCodigo());
37             vista.txtDni.setText(carnet.getDni());
38             vista.txtNombre.setText(carnet.getNombreCompleto());
39             vista.cmbFacultad.setSelectedItem(carnet.getFacultad());
40
41             vista.actualizarCarreras(); // Ahora llamas a la vista para actualizar carreras.
42
43             vista.cmbCarrera.setSelectedItem(carnet.getCarrera());
44
45             vista.txtCodigo.setEditable(false);
46             vista.txtDni.setEditable(false);
47         }
48     }
49
50     private boolean validarCampos() {
51         if (vista.txtCodigo.getText().trim().isEmpty()
52             || vista.txtDni.getText().trim().isEmpty()
53             || vista.txtNombre.getText().trim().isEmpty()
54             || vista.cmbFacultad.getSelectedIndex() == -1
55             || vista.cmbCarrera.getSelectedIndex() == -1) {
56             JOptionPane.showMessageDialog(vista, "Todos los campos deben estar completos.", "Error", JOptionPane.ERROR_MESSAGE);
57             return false;
58         }
59         return true;
60     }
61 }

```

```

62 private void crearCarnet() {
63     Carnet nuevo = new Carnet(
64         vista.txtCodigo.getText(),
65         vista.txtDni.getText(),
66         vista.txtNombre.getText(),
67         (String) vista.cmbFacultad.getSelectedItem(),
68         (String) vista.cmbCarrera.getSelectedItem()
69     );
70     menuControlador.agregarCarnet(nuevo);
71     JOptionPane.showMessageDialog(vista, "Carnet creado exitosamente.");
72     vista.dispose();
73 }
74
75 private void actualizarCarnet() {
76     Carnet carnet = menuControlador.getCarnetSeleccionado();
77     if (carnet != null) {
78         carnet.actualizarDatos(
79             vista.txtNombre.getText(),
80             (String) vista.cmbFacultad.getSelectedItem(),
81             (String) vista.cmbCarrera.getSelectedItem()
82         );
83         JOptionPane.showMessageDialog(vista, "Carnet actualizado exitosamente.");
84         vista.dispose();
85     }
86 }
87
88 @Override
89 public void actionPerformed(ActionEvent e) {
90     Object source = e.getSource();
91
92     if (source == vista.btnVolver) {
93         vista.dispose();
94     } else if (source == vista.btnCrear) {
95         if (validarCampos()) {
96             if (tipo.equalsIgnoreCase("crear")) {
97                 crearCarnet();
98             } else if (tipo.equalsIgnoreCase("actualizar")) {
99                 actualizarCarnet();
100             }
101         }
102     }
103 }
104 }
105

```

```

16 public class MenuControlador implements ActionListener, MouseListener, KeyListener {
17
18     private final CMenu vista;
19     private static final ArrayList<Carnet> carnets = new ArrayList<>();
20     private Carnet carnetSeleccionado = null;
21
22     public MenuControlador(CMenu vista) {
23         this.vista = vista;
24
25         this.vista.btnCrear.addActionListener(this);
26         this.vista.btnActualizar.addActionListener(this);
27         this.vista.btnEliminar.addActionListener(this);
28         this.vista.btnSalir.addActionListener(this);
29
30         this.vista.tblDatos.addMouseListener(this);
31         this.vista.txtBuscar.addKeyListener(this);
32
33         this.vista.btnActualizar.setEnabled(false);
34         this.vista.btnEliminar.setEnabled(false);
35
36         actualizarTabla(carnets);
37     }
38
39     private void actualizarTabla(ArrayList<Carnet> lista) {
40         DefaultTableModel modelo = new DefaultTableModel();
41         modelo.addColumn("Código");
42         modelo.addColumn("DNI");
43         modelo.addColumn("Nombre Completo");
44         modelo.addColumn("Facultad");
45         modelo.addColumn("Carrera");
46
47         for (Carnet c : lista) {
48             modelo.addRow(new Object[]{
49                 c.getCodigo(),
50                 c.getDni(),
51                 c.getNombreCompleto(),
52                 c.getFacultad(),
53                 c.getCarrera()
54             });
55         }
56
57         vista.tblDatos.setModel(modelo);
58     }
59

```

```

60 private void buscarCarnet() {
61     String texto = vista.txtBuscar.getText().toLowerCase();
62     ArrayList<Carnet> resultados = new ArrayList<>();
63
64     for (Carnet c : carnets) {
65         if (c.getDni().toLowerCase().contains(texto)
66             || c.getCodigo().toLowerCase().contains(texto)
67             || c.getNombreCompleto().toLowerCase().contains(texto)) {
68             resultados.add(c);
69         }
70     }
71
72     actualizarTabla(resultados);
73 }
74
75 @Override
76 public void actionPerformed(ActionEvent e) {
77     Object source = e.getSource();
78
79     if (source == vista.btnCrear) {
80         crear();
81     } else if (source == vista.btnActualizar) {
82         actualizar();
83     } else if (source == vista.btnEliminar) {
84         eliminar();
85     } else if (source == vista.btnSalir) {
86         salir();
87     }
88 }
89
90 private void crear() {
91     CCarnet dialogo = new CCarnet(vista, true, "crear", this);
92     dialogo.setVisible(true);
93     actualizarTabla(carnets);
94     limpiarSeleccion();
95 }
96
97 private void actualizar() {
98     if (carnetSeleccionado != null) {
99         CCarnet dialogo = new CCarnet(vista, true, "actualizar", this);
100         dialogo.setVisible(true);
101         actualizarTabla(carnets);
102         limpiarSeleccion();
103     } else {
104         JOptionPane.showMessageDialog(vista, "Seleccione un carnet primero.");
105     }
106 }
107

```

```

107
108 private void eliminar() {
109     if (carnetSeleccionado != null) {
110         int confirm = JOptionPane.showConfirmDialog(vista, "¿Seguro de eliminar este carnet?", "Confirmar", JOptionPane.YES_NO_OPTION);
111         if (confirm == JOptionPane.YES_OPTION) {
112             carnets.remove(carnetSeleccionado);
113             actualizarTabla(carnets);
114             carnetSeleccionado = null;
115             limpiarSeleccion();
116         }
117     } else {
118         JOptionPane.showMessageDialog(vista, "Seleccione un carnet primero.");
119     }
120 }
121
122 private void salir() {
123     int respuesta = JOptionPane.showOptionDialog(vista, "¿Estás seguro de salir?", "Salir", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, null, null);
124     if (respuesta == 0) {
125         System.exit(0);
126     }
127     vista.txtBuscar.requestFocus();
128 }
129
130 private void limpiarSeleccion() {
131     carnetSeleccionado = null;
132     vista.btnActualizar.setEnabled(false);
133     vista.btnEliminar.setEnabled(false);
134     vista.tblDatos.clearSelection();
135 }
136
137 @Override
138 public void mouseClicked(MouseEvent e) {
139     int fila = vista.tblDatos.getSelectedRow();
140     if (fila >= 0) {
141         String dniSeleccionado = (String) vista.tblDatos.getValueAt(fila, 1); // columna 1 = DNI
142         for (Carnet c : carnets) {
143             if (c.getDni().equals(dniSeleccionado)) {
144                 carnetSeleccionado = c;
145                 break;
146             }
147         }
148         vista.btnActualizar.setEnabled(true);
149         vista.btnEliminar.setEnabled(true);
150     }
151 }
152
153 }
154
155 // Métodos vacíos que se deben implementar
156 @Override
157 public void mousePressed(MouseEvent e) {
158 }
159
160 @Override
161 public void mouseReleased(MouseEvent e) {
162 }
163
164 @Override
165 public void mouseEntered(MouseEvent e) {
166 }
167
168 @Override
169 public void mouseExited(MouseEvent e) {
170 }
171
172 @Override
173 public void keyPressed(KeyEvent e) {
174 }
175
176 @Override
177 public void keyTyped(KeyEvent e) {
178 }
179
180 @Override
181 public void keyReleased(KeyEvent e) {
182 }
183
184 // Métodos para acceder desde el CarnetControlador
185 public void agregarCarnet(Carnet c) {
186     carnets.add(c);
187 }
188
189 public Carnet getCarnetSeleccionado() {
190     return carnetSeleccionado;
191 }
192
193 }
194

```

```

8
9 public class CEjecutar {
10
11     public static void main(String[] args) {
12         CMenu menu = new CMenu();
13         menu.setVisible(true);
14     }
15 }
16

```

```

6 public class Carnet {
7
8     private String codigo;
9     private String dni;
10    private String nombreCompleto;
11    private String facultad;
12    private String carrera;
13
14
15    public Carnet(String codigo, String dni, String nombreCompleto, String facultad, String carrera) {
16        this.codigo = codigo;
17        this.dni = dni;
18        this.nombreCompleto = nombreCompleto;
19        this.facultad = facultad;
20        this.carrera = carrera;
21    }
22
23    public String getCodigo() {
24        return codigo;
25    }
26
27    public String getDni() {
28        return dni;
29    }
30
31    public String getNombreCompleto() {
32        return nombreCompleto;
33    }
34
35    public String getFacultad() {
36        return facultad;
37    }
38
39    public String getCarrera() {
40        return carrera;
41    }
42
43    public void actualizarDatos(String nombreCompleto, String facultad, String carrera) {
44        this.nombreCompleto = nombreCompleto;
45        this.facultad = facultad;
46        this.carrera = carrera;
47    }
48 }
49

```



```

15 public class CCarnet extends javax.swing.JDialog {
16
17     /**
18      * Creates new form CCarnet
19      */
20     public CCarnet(java.awt.Frame parent, boolean modal, String tipo, MenuControlador menuControlador) {
21         super(parent, modal);
22         initComponents();
23         new CarnetControlador(this, tipo, menuControlador);
24     }
25
26     public void actualizarCarreras() {
27         String facultadSeleccionada = (String) cmbFacultad.getSelectedItem();
28         DefaultComboBoxModel<String> modeloCarreras = new DefaultComboBoxModel<>();
29
30         if (facultadSeleccionada == null) {
31             return;
32         }
33
34         if (facultadSeleccionada.equalsIgnoreCase("Ciencias administrativas y contables")) {
35             modeloCarreras.addElement("Administración");
36             modeloCarreras.addElement("Administración e Inteligencia de Negocios");
37             modeloCarreras.addElement("Administración y Negocios Globales");
38             modeloCarreras.addElement("Administración y Gestión del Talento Humano");
39             modeloCarreras.addElement("Contabilidad y Finanzas");
40         } else if (facultadSeleccionada.equalsIgnoreCase("Derecho y ciencias políticas")) {
41             modeloCarreras.addElement("Derecho");
42             modeloCarreras.addElement("Educación Inicial");
43             modeloCarreras.addElement("Educación Primaria");
44         } else if (facultadSeleccionada.equalsIgnoreCase("Ingeniería")) {
45             modeloCarreras.addElement("Arquitectura");
46             modeloCarreras.addElement("Ingeniería Civil");
47             modeloCarreras.addElement("Ingeniería Industrial");
48             modeloCarreras.addElement("Ingeniería del Medio Ambiente y Desarrollo");
49             modeloCarreras.addElement("Ingeniería de Sistemas y Computación");
50         } else if (facultadSeleccionada.equalsIgnoreCase("Ciencias de la Salud")) {
51             modeloCarreras.addElement("Enfermería");
52             modeloCarreras.addElement("Farmacia y Bioquímica");
53             modeloCarreras.addElement("Medicina Veterinaria y Zootecnia");
54             modeloCarreras.addElement("Nutrición Humana");
55             modeloCarreras.addElement("Obstetricia");
56             modeloCarreras.addElement("Odontología");
57             modeloCarreras.addElement("Psicología");
58             modeloCarreras.addElement("Tecnología Médica");
59         } else if (facultadSeleccionada.equalsIgnoreCase("Medicina Humana")) {
60             modeloCarreras.addElement("Medicina Humana");
61         }
62     }

```

```

13 public class CMenu extends javax.swing.JFrame {
14
15     /**
16      * Creates new form CMenu
17      */
18     public CMenu() {
19         initComponents();
20         new MenuControlador(this);
21     }
22
23     /**
24      * This method is called from within the constructor to initialize the form.
25      * WARNING: Do NOT modify this code. The content of this method is always
26      * regenerated by the Form Editor.
27      */
28     @SuppressWarnings("unchecked")
29     Generated Code
30
31     /**
32      * @param args the command line arguments
33      */

```