
Manual técnico

USAC- Delivery

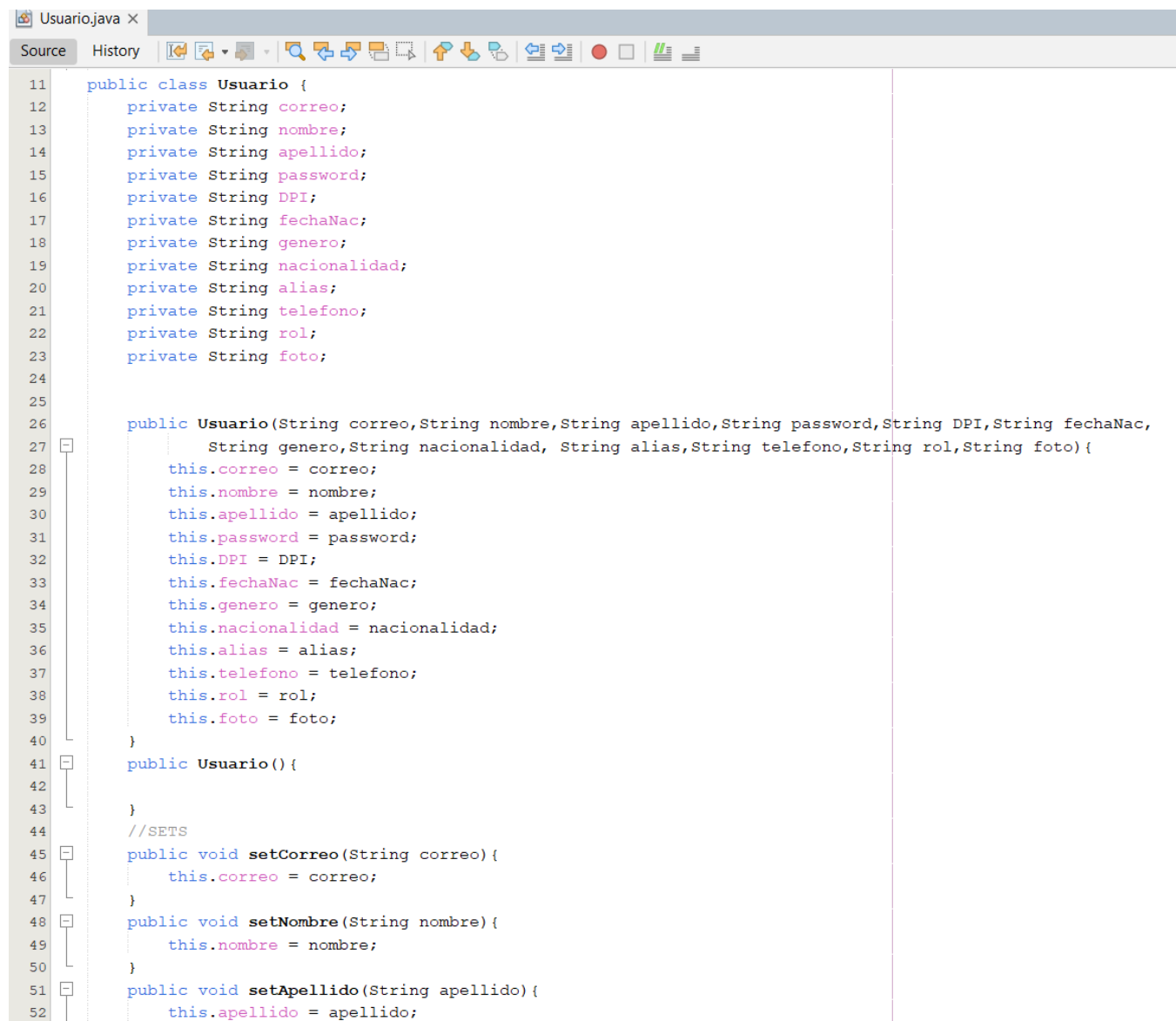
Jeison Estrada

201709409

BEANS

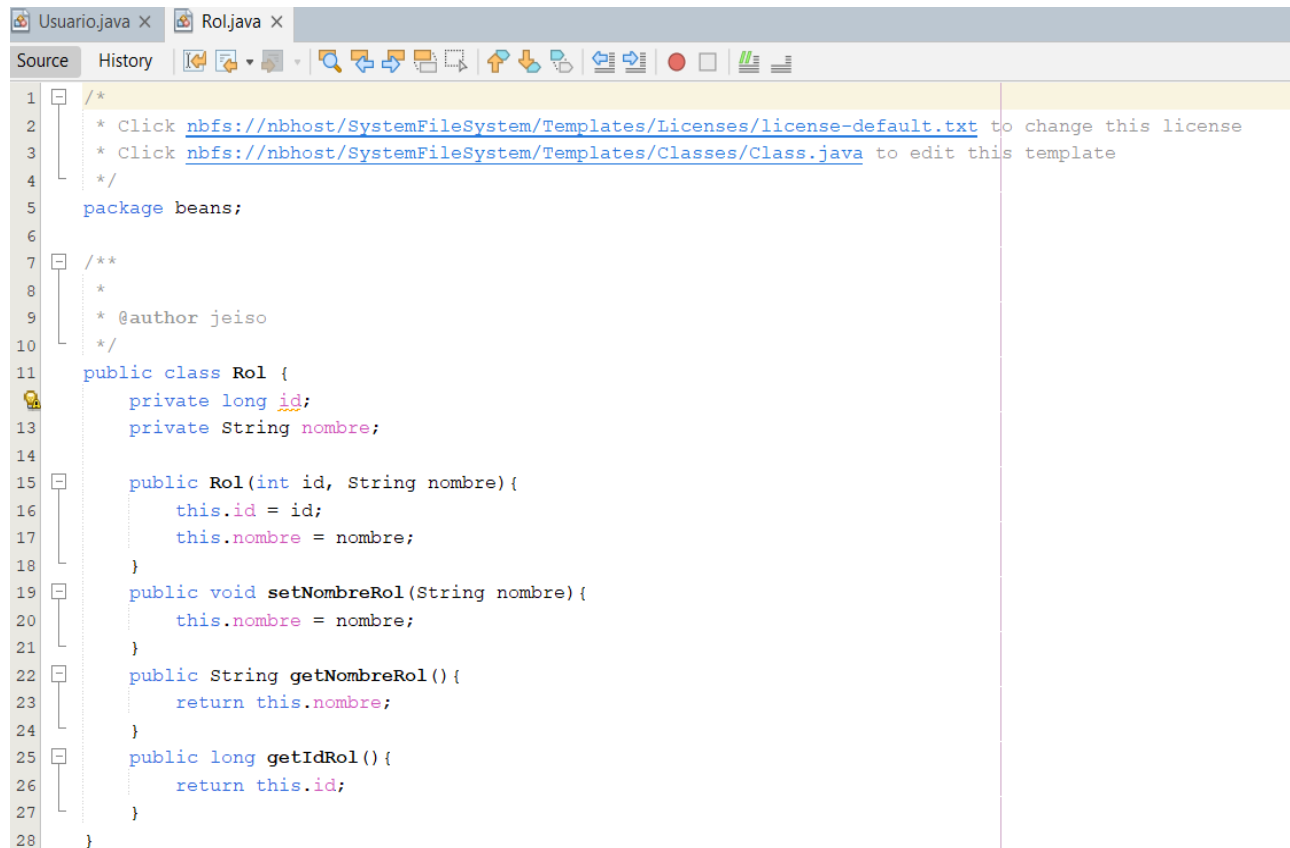
Clases utilizadas para definir y manipular las propiedades de nuestros objetos que nos ayudara a crear nuestra interfaz

USUARIO.JAVA



```
11 public class Usuario {
12     private String correo;
13     private String nombre;
14     private String apellido;
15     private String password;
16     private String DPI;
17     private String fechaNac;
18     private String genero;
19     private String nacionalidad;
20     private String alias;
21     private String telefono;
22     private String rol;
23     private String foto;
24
25
26     public Usuario(String correo,String nombre,String apellido,String password,String DPI,String fechaNac,
27         String genero,String nacionalidad, String alias,String telefono,String rol,String foto){
28         this.correo = correo;
29         this.nombre = nombre;
30         this.apellido = apellido;
31         this.password = password;
32         this.DPI = DPI;
33         this.fechaNac = fechaNac;
34         this.genero = genero;
35         this.nacionalidad = nacionalidad;
36         this.alias = alias;
37         this.telefono = telefono;
38         this.rol = rol;
39         this.foto = foto;
40     }
41     public Usuario(){
42     }
43
44     //SETS
45     public void setCorreo(String correo){
46         this.correo = correo;
47     }
48     public void setNombre(String nombre){
49         this.nombre = nombre;
50     }
51     public void setApellido(String apellido){
52         this.apellido = apellido;
```

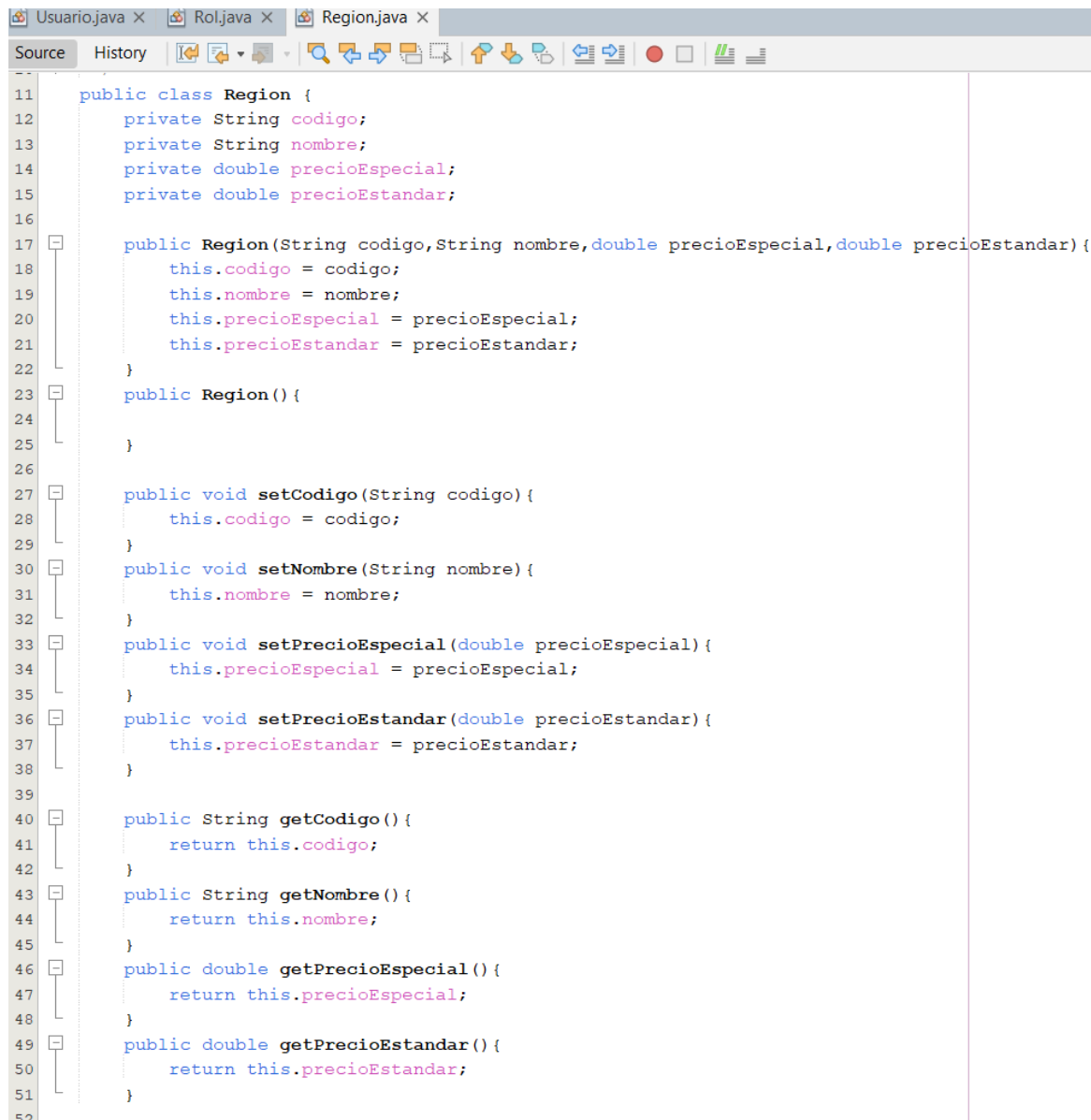
ROL.JAVA



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package beans;
6
7  /**
8   *
9   * @author jeiso
10  */
11  public class Rol {
12      private long id;
13      private String nombre;
14
15      public Rol(int id, String nombre){
16          this.id = id;
17          this.nombre = nombre;
18      }
19      public void setNombreRol(String nombre){
20          this.nombre = nombre;
21      }
22      public String getNombreRol(){
23          return this.nombre;
24      }
25      public long getIdRol(){
26          return this.id;
27      }
28  }
```

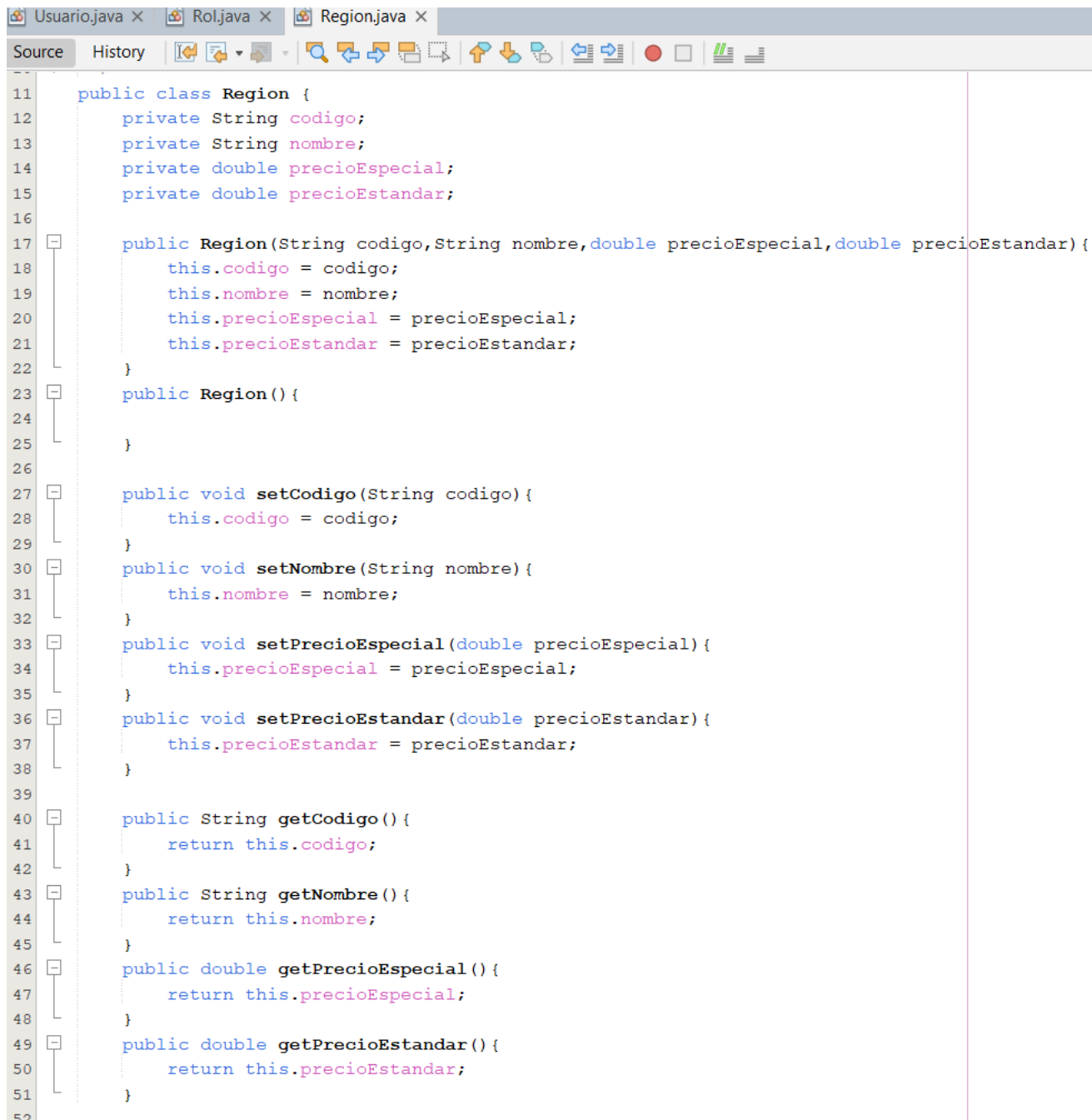
Como se puede observar se implementaron los gets y sets para modelar nuestros objetos.

REGION.JAVA



```
11 public class Region {
12     private String codigo;
13     private String nombre;
14     private double precioEspecial;
15     private double precioEstandar;
16
17     public Region(String codigo,String nombre,double precioEspecial,double precioEstandar){
18         this.codigo = codigo;
19         this.nombre = nombre;
20         this.precioEspecial = precioEspecial;
21         this.precioEstandar = precioEstandar;
22     }
23     public Region() {
24     }
25
26     public void setCodigo(String codigo){
27         this.codigo = codigo;
28     }
29     public void setNombre(String nombre){
30         this.nombre = nombre;
31     }
32     public void setPrecioEspecial(double precioEspecial){
33         this.precioEspecial = precioEspecial;
34     }
35     public void setPrecioEstandar(double precioEstandar){
36         this.precioEstandar = precioEstandar;
37     }
38
39     public String getCodigo(){
40         return this.codigo;
41     }
42     public String getNombre(){
43         return this.nombre;
44     }
45     public double getPrecioEspecial(){
46         return this.precioEspecial;
47     }
48     public double getPrecioEstandar(){
49         return this.precioEstandar;
50     }
51 }
```

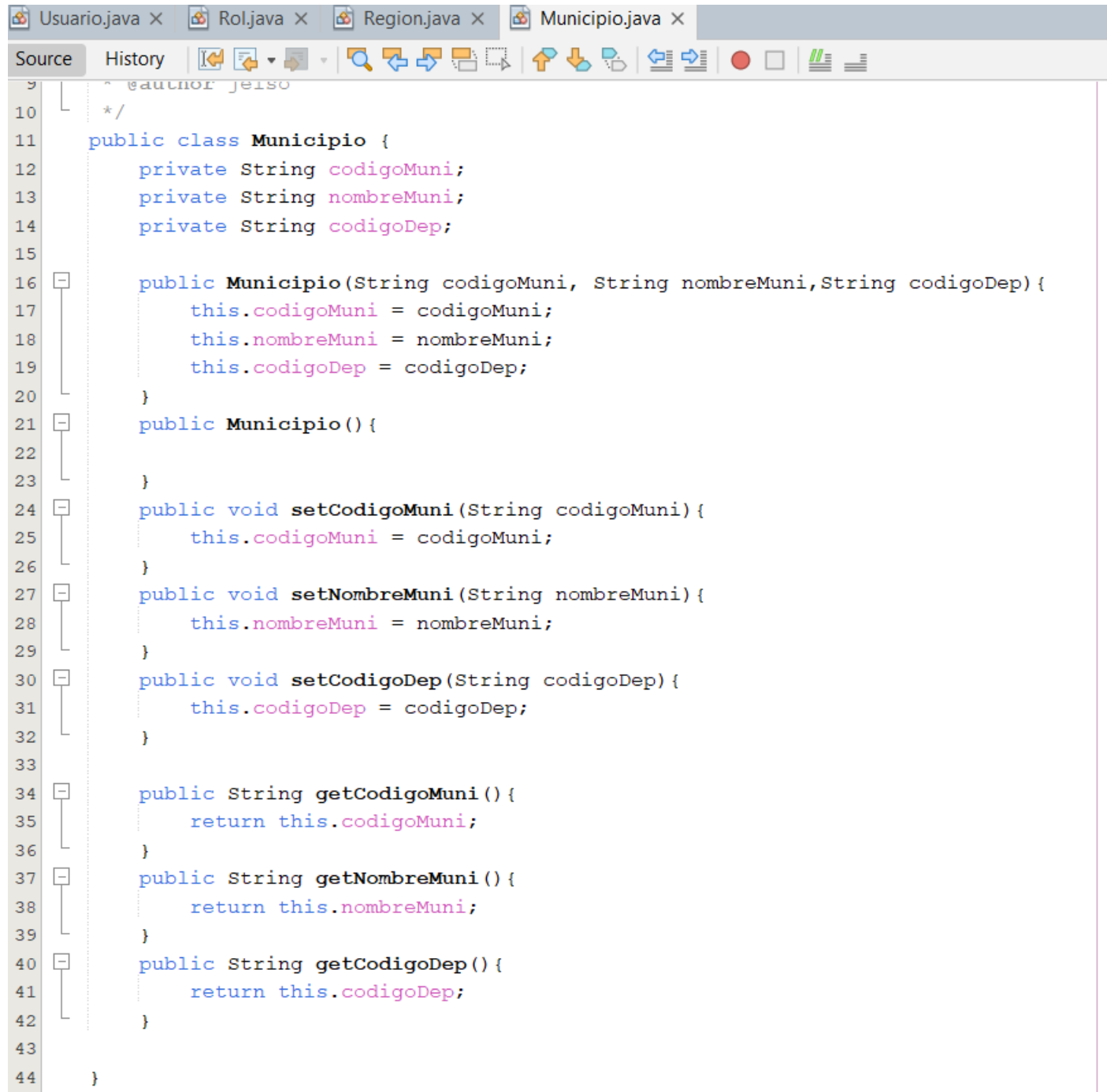
REGION.JAVA



The screenshot shows an IDE window with three tabs: 'Usuario.java', 'Rol.java', and 'Region.java'. The 'Region.java' tab is active, displaying the source code of a Java class named 'Region'. The code includes private attributes for 'codigo', 'nombre', 'precioEspecial', and 'precioEstandar', along with a constructor and several getter and setter methods. The IDE interface includes a toolbar with various icons for file operations, search, and execution. The code is color-coded, with keywords in blue, strings in pink, and comments in green. The line numbers on the left range from 11 to 52.

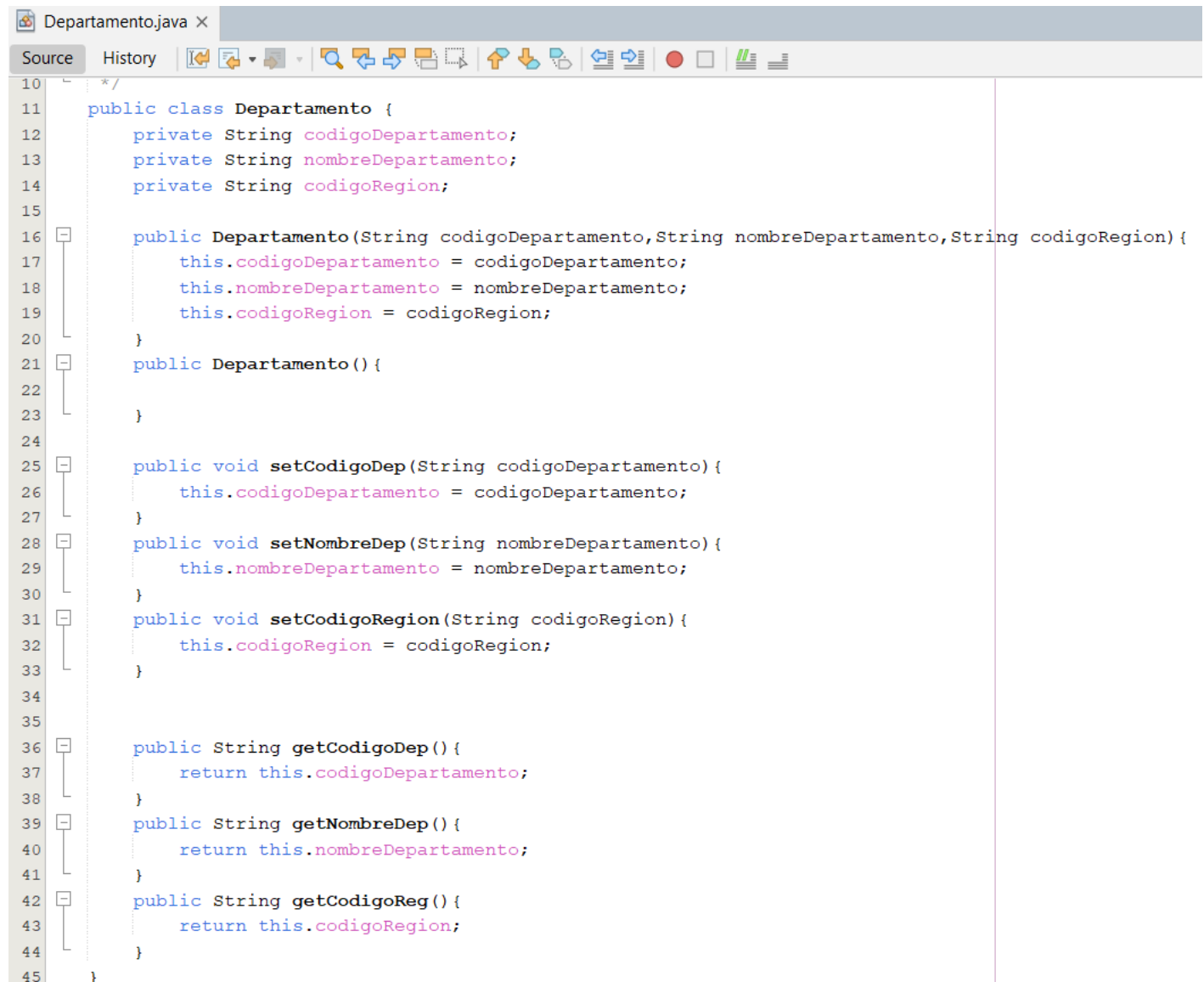
```
11 public class Region {
12     private String codigo;
13     private String nombre;
14     private double precioEspecial;
15     private double precioEstandar;
16
17     public Region(String codigo,String nombre,double precioEspecial,double precioEstandar){
18         this.codigo = codigo;
19         this.nombre = nombre;
20         this.precioEspecial = precioEspecial;
21         this.precioEstandar = precioEstandar;
22     }
23     public Region(){
24
25     }
26
27     public void setCodigo(String codigo){
28         this.codigo = codigo;
29     }
30     public void setNombre(String nombre){
31         this.nombre = nombre;
32     }
33     public void setPrecioEspecial(double precioEspecial){
34         this.precioEspecial = precioEspecial;
35     }
36     public void setPrecioEstandar(double precioEstandar){
37         this.precioEstandar = precioEstandar;
38     }
39
40     public String getCodigo(){
41         return this.codigo;
42     }
43     public String getNombre(){
44         return this.nombre;
45     }
46     public double getPrecioEspecial(){
47         return this.precioEspecial;
48     }
49     public double getPrecioEstandar(){
50         return this.precioEstandar;
51     }
52 }
```

MUNICIPIO.JAVA



```
9  /*
10  */
11  public class Municipio {
12      private String codigoMuni;
13      private String nombreMuni;
14      private String codigoDep;
15
16      public Municipio(String codigoMuni, String nombreMuni, String codigoDep) {
17          this.codigoMuni = codigoMuni;
18          this.nombreMuni = nombreMuni;
19          this.codigoDep = codigoDep;
20      }
21      public Municipio() {
22      }
23
24      public void setCodigoMuni(String codigoMuni) {
25          this.codigoMuni = codigoMuni;
26      }
27      public void setNombreMuni(String nombreMuni) {
28          this.nombreMuni = nombreMuni;
29      }
30      public void setCodigoDep(String codigoDep) {
31          this.codigoDep = codigoDep;
32      }
33
34      public String getCodigoMuni() {
35          return this.codigoMuni;
36      }
37      public String getNombreMuni() {
38          return this.nombreMuni;
39      }
40      public String getCodigoDep() {
41          return this.codigoDep;
42      }
43
44  }
```

DEPARTAMENTO.JAVA

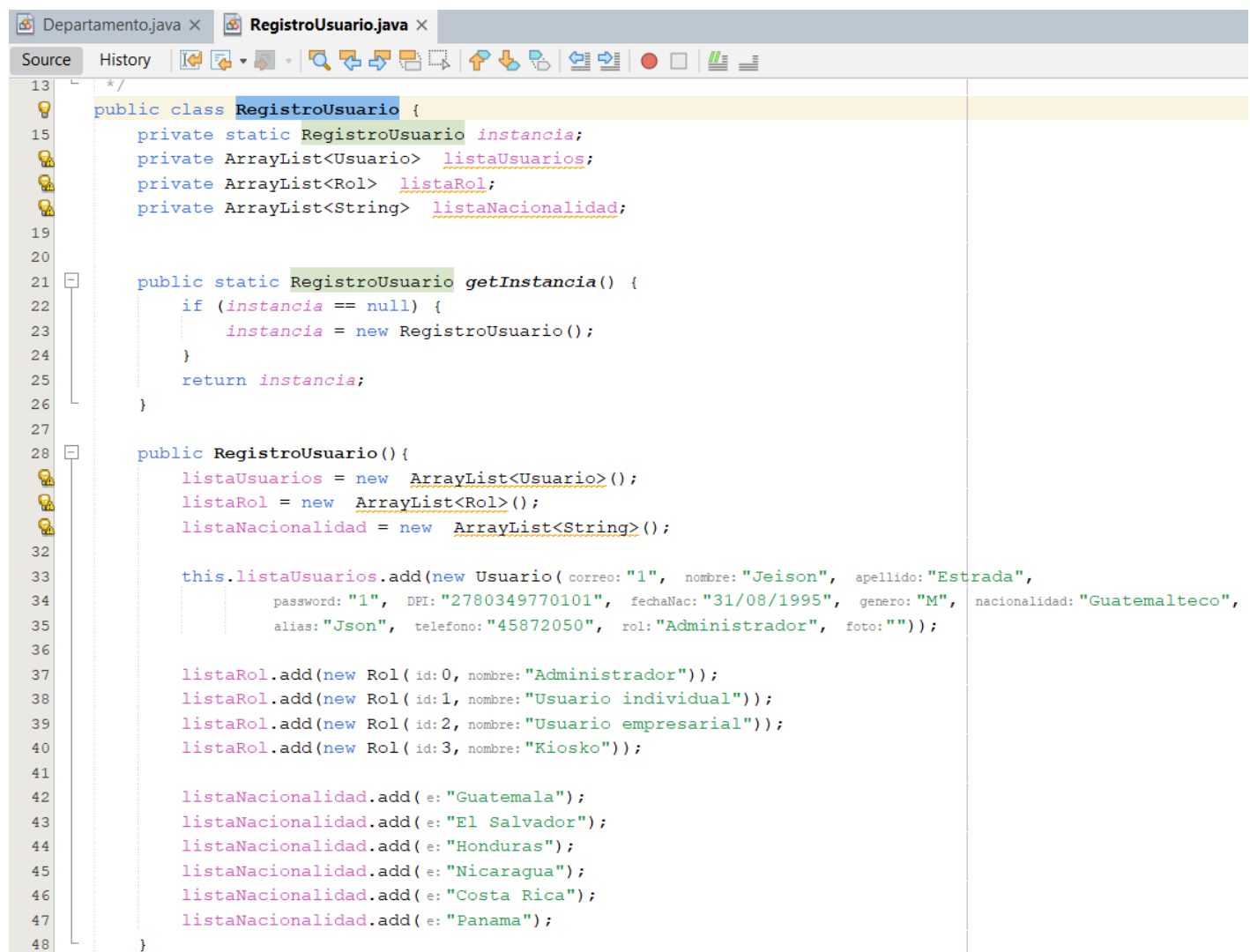


```
10  */
11  public class Departamento {
12      private String codigoDepartamento;
13      private String nombreDepartamento;
14      private String codigoRegion;
15
16      public Departamento(String codigoDepartamento,String nombreDepartamento,String codigoRegion){
17          this.codigoDepartamento = codigoDepartamento;
18          this.nombreDepartamento = nombreDepartamento;
19          this.codigoRegion = codigoRegion;
20      }
21      public Departamento() {
22      }
23
24
25      public void setCodigoDep(String codigoDepartamento){
26          this.codigoDepartamento = codigoDepartamento;
27      }
28      public void setNombreDep(String nombreDepartamento){
29          this.nombreDepartamento = nombreDepartamento;
30      }
31      public void setCodigoRegion(String codigoRegion){
32          this.codigoRegion = codigoRegion;
33      }
34
35
36      public String getCodigoDep(){
37          return this.codigoDepartamento;
38      }
39      public String getNombreDep(){
40          return this.nombreDepartamento;
41      }
42      public String getCodigoReg(){
43          return this.codigoRegion;
44      }
45  }
```

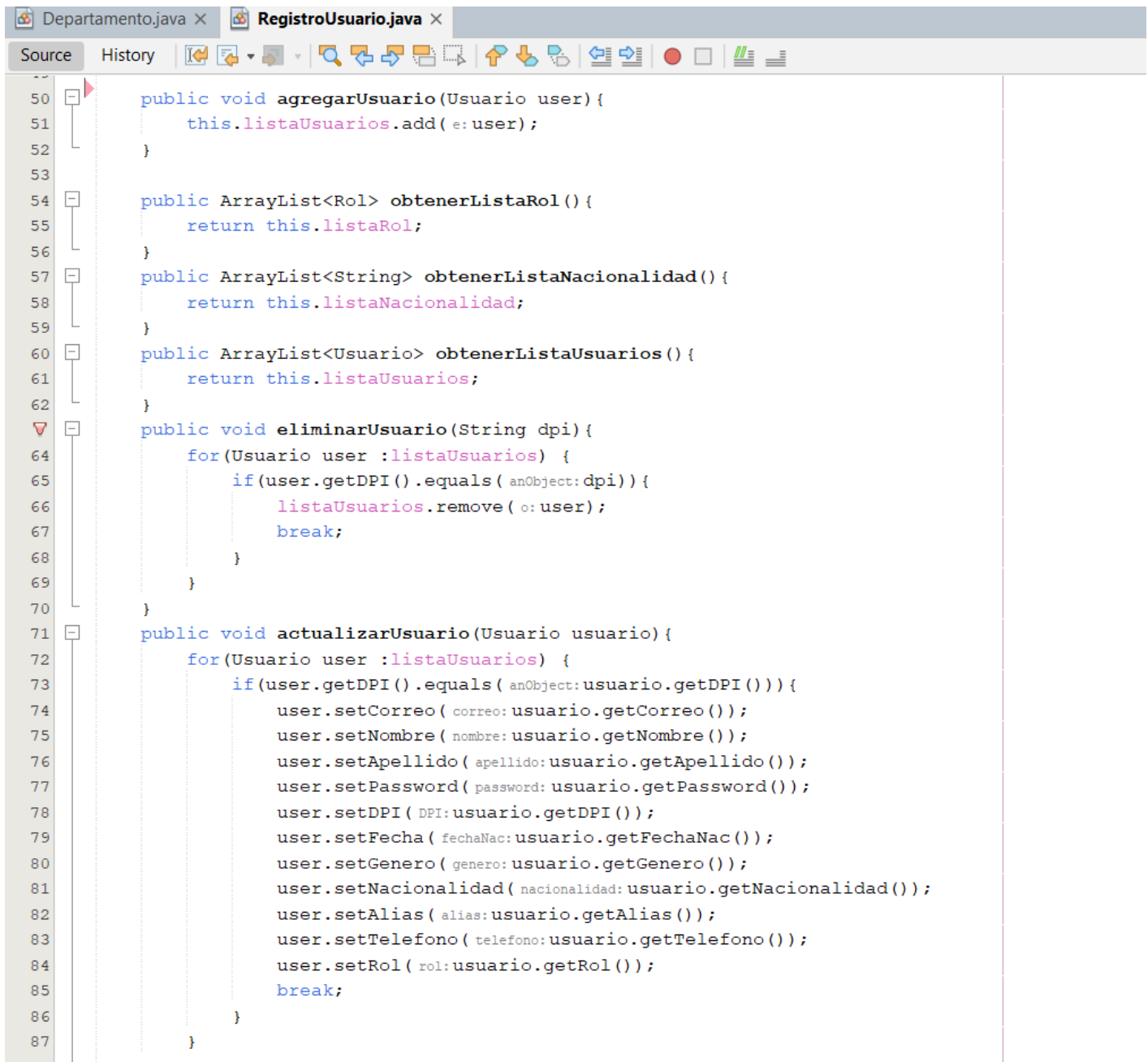
CONTROLADORES

Estas clases están destinadas almacenar, modificar y eliminar información relevante para los usuarios. Estas clases contienen las listas donde se almacena la información y también contiene los métodos para manipular dichas listas dinámicas.

REGISTROUSUARIO.JAVA



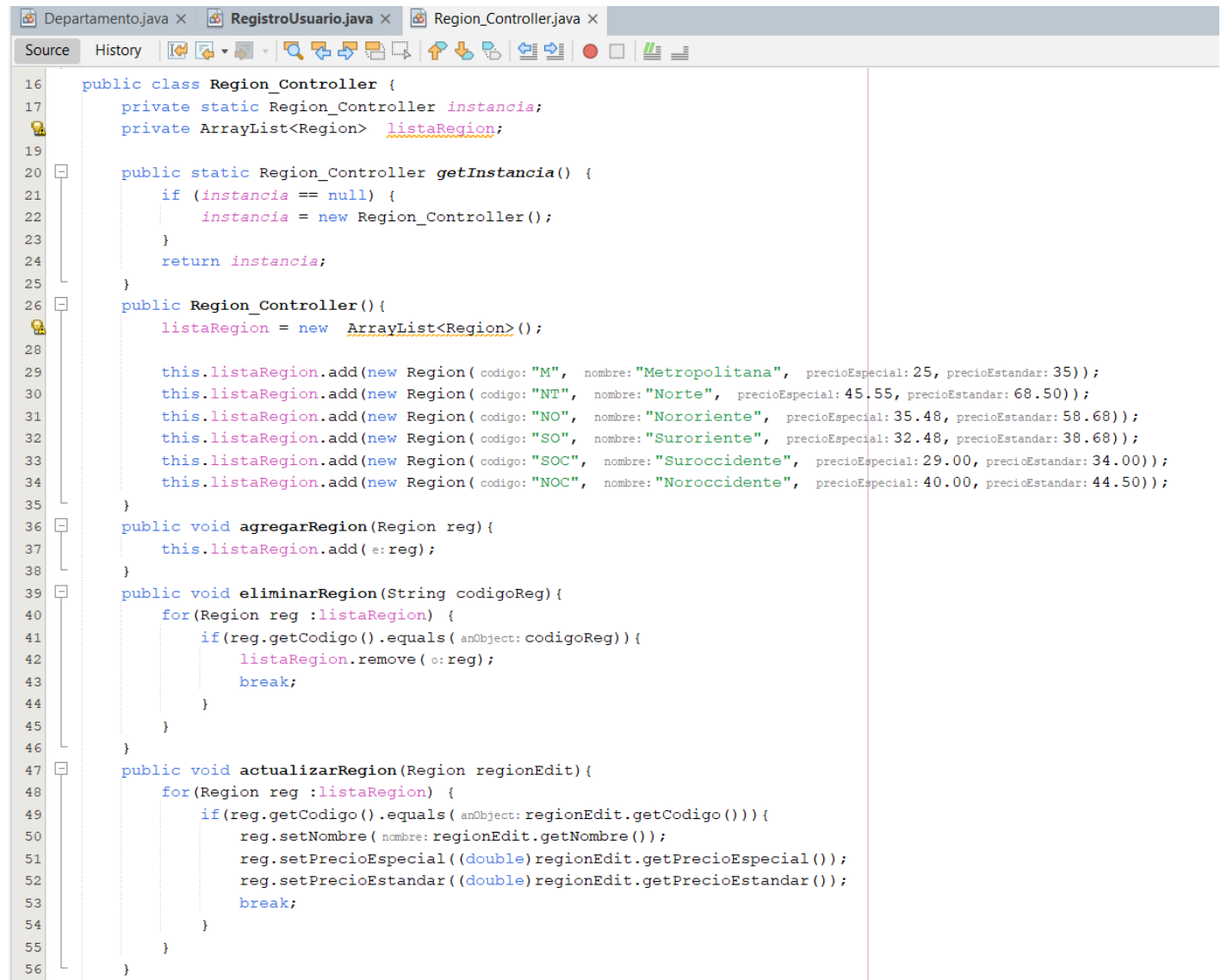
```
13  */
14  public class RegistroUsuario {
15      private static RegistroUsuario instancia;
16      private ArrayList<Usuario> listaUsuarios;
17      private ArrayList<Rol> listaRol;
18      private ArrayList<String> listaNacionalidad;
19
20
21      public static RegistroUsuario getInstancia() {
22          if (instancia == null) {
23              instancia = new RegistroUsuario();
24          }
25          return instancia;
26      }
27
28      public RegistroUsuario() {
29          listaUsuarios = new ArrayList<Usuario>();
30          listaRol = new ArrayList<Rol>();
31          listaNacionalidad = new ArrayList<String>();
32
33          this.listaUsuarios.add(new Usuario( correo: "1", nombre: "Jeison", apellido: "Estrada",
34              password: "1", DPI: "2780349770101", fechaNac: "31/08/1995", genero: "M", nacionalidad: "Guatemalteco",
35              alias: "Json", telefono: "45872050", rol: "Administrador", foto: ""));
36
37          listaRol.add(new Rol( id: 0, nombre: "Administrador"));
38          listaRol.add(new Rol( id: 1, nombre: "Usuario individual"));
39          listaRol.add(new Rol( id: 2, nombre: "Usuario empresarial"));
40          listaRol.add(new Rol( id: 3, nombre: "Kiosko"));
41
42          listaNacionalidad.add( e: "Guatemala");
43          listaNacionalidad.add( e: "El Salvador");
44          listaNacionalidad.add( e: "Honduras");
45          listaNacionalidad.add( e: "Nicaragua");
46          listaNacionalidad.add( e: "Costa Rica");
47          listaNacionalidad.add( e: "Panama");
48      }
```

```
Departamento.java x RegistroUsuario.java x
Source History

50 public void agregarUsuario(Usuario user) {
51     this.listaUsuarios.add( e: user );
52 }
53
54 public ArrayList<Rol> obtenerListaRol() {
55     return this.listaRol;
56 }
57 public ArrayList<String> obtenerListaNacionalidad() {
58     return this.listaNacionalidad;
59 }
60 public ArrayList<Usuario> obtenerListaUsuarios() {
61     return this.listaUsuarios;
62 }
63 public void eliminarUsuario(String dpi) {
64     for(Usuario user : listaUsuarios) {
65         if(user.getDPI().equals( anObject: dpi )) {
66             listaUsuarios.remove( o: user );
67             break;
68         }
69     }
70 }
71 public void actualizarUsuario(Usuario usuario) {
72     for(Usuario user : listaUsuarios) {
73         if(user.getDPI().equals( anObject: usuario.getDPI() )) {
74             user.setCorreo( correo: usuario.getCorreo() );
75             user.setNombre( nombre: usuario.getNombre() );
76             user.setApellido( apellido: usuario.getApellido() );
77             user.setPassword( password: usuario.getPassword() );
78             user.setDPI( DPI: usuario.getDPI() );
79             user.setFecha( fechaNac: usuario.getFechaNac() );
80             user.setGenero( genero: usuario.getGenero() );
81             user.setNacionalidad( nacionalidad: usuario.getNacionalidad() );
82             user.setAlias( alias: usuario.getAlias() );
83             user.setTelefono( telefono: usuario.getTelefono() );
84             user.setRol( rol: usuario.getRol() );
85             break;
86         }
87     }
88 }
```

REGION_CONTROLLER.JAVA



```
16 public class Region_Controller {
17     private static Region_Controller instancia;
18     private ArrayList<Region> listaRegion;
19
20     public static Region_Controller getInstancia() {
21         if (instancia == null) {
22             instancia = new Region_Controller();
23         }
24         return instancia;
25     }
26     public Region_Controller() {
27         listaRegion = new ArrayList<Region>();
28
29         this.listaRegion.add(new Region( codigo: "M",  nombre: "Metropolitana",  precioEspecial: 25, precioEstandar: 35));
30         this.listaRegion.add(new Region( codigo: "NT",  nombre: "Norte",  precioEspecial: 45.55, precioEstandar: 68.50));
31         this.listaRegion.add(new Region( codigo: "NO",  nombre: "Nororiente",  precioEspecial: 35.48, precioEstandar: 58.68));
32         this.listaRegion.add(new Region( codigo: "SO",  nombre: "Suroriente",  precioEspecial: 32.48, precioEstandar: 38.68));
33         this.listaRegion.add(new Region( codigo: "SOC",  nombre: "Suroccidente",  precioEspecial: 29.00, precioEstandar: 34.00));
34         this.listaRegion.add(new Region( codigo: "NOC",  nombre: "Noroccidente",  precioEspecial: 40.00, precioEstandar: 44.50));
35     }
36     public void agregarRegion(Region reg) {
37         this.listaRegion.add( e: reg);
38     }
39     public void eliminarRegion(String codigoReg){
40         for(Region reg : listaRegion) {
41             if(reg.getCodigo().equals( anObject: codigoReg)){
42                 listaRegion.remove( o: reg);
43                 break;
44             }
45         }
46     }
47     public void actualizarRegion(Region regionEdit){
48         for(Region reg : listaRegion) {
49             if(reg.getCodigo().equals( anObject: regionEdit.getCodigo())){
50                 reg.setNombre( nombre: regionEdit.getNombre());
51                 reg.setPrecioEspecial((double) regionEdit.getPrecioEspecial());
52                 reg.setPrecioEstandar((double) regionEdit.getPrecioEstandar());
53                 break;
54             }
55         }
56     }
57 }
```

DEPARTAMENTO_CONTROLLER.JAVA

```
Departamento_Controller.java X
Source History
13
14 public class Departamento_Controller {
15     private static Departamento_Controller instancia;
16     private ArrayList<Departamento> listaDepartamento;
17
18     public static Departamento_Controller getInstancia() {
19         if (instancia == null) {
20             instancia = new Departamento_Controller();
21         }
22         return instancia;
23     }
24
25     public Departamento_Controller(){
26         listaDepartamento = new ArrayList<Departamento>();
27         this.listaDepartamento.add(new Departamento ( codigoDepartamento: "GT", nombreDepartamento: "Guatemala", codigoRegion: "M"));
28         this.listaDepartamento.add(new Departamento ( codigoDepartamento: "CHI", nombreDepartamento: "Chiquimula", codigoRegion: "NO"));
29         this.listaDepartamento.add(new Departamento ( codigoDepartamento: "ZAP", nombreDepartamento: "Zacapa", codigoRegion: "NO"));
30     }
31
32     public void agregarDepartamento(Departamento dep){
33         this.listaDepartamento.add ( e: dep );
34     }
35
36     public void eliminarDepartamento(String codigoDep){
37         for(Departamento reg : listaDepartamento) {
38             if(reg.getCodigoDep().equals ( anObject: codigoDep)){
39                 listaDepartamento.remove ( o: reg );
40                 break;
41             }
42         }
43
44     public void actualizarDepartamento(Departamento departamentoEdit){
45         for(Departamento dep : listaDepartamento) {
46             if(dep.getCodigoDep().equals ( anObject: departamentoEdit.getCodigoDep())){
47                 dep.setNombreDep ( nombreDepartamento: departamentoEdit.getNombreDep ());
48                 dep.setCodigoRegion ( codigoRegion: departamentoEdit.getCodigoReg ());
49                 break;
50             }
51         }
52     }
53
54     public ArrayList<Departamento> obtenerListaDepartamento () {
55         return this.listaDepartamento;
56     }
57 }
```

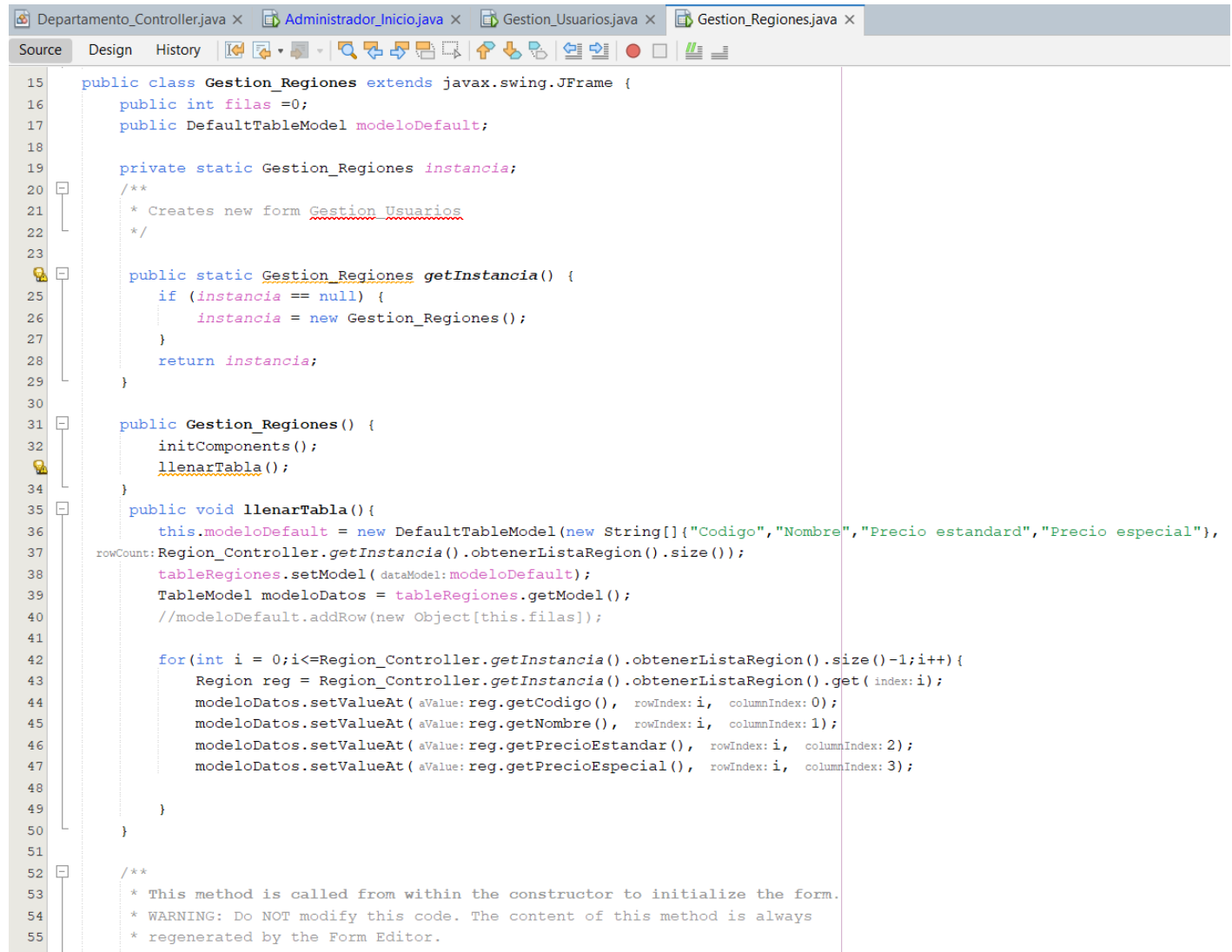
VISTAS

Estas clases son las que contiene todos los componentes o elementos que el usuario puede observar y con los cuales puede interactuar. Es aquí donde se construyen todos los menus.

GESTION_USUARIOS.JAVA

```
Departamento_Controller.java x Administrador_Inicio.java x Gestion_Usuarios.java x
Source Design History
14
15 public class Gestion_Usuarios extends javax.swing.JFrame {
16     public int filas = 0;
17     public DefaultTableModel modeloDefault;
18
19     private static Gestion_Usuarios instancia;
20     /**
21      * Creates new form Gestion_Usuarios
22      */
23
24     public static Gestion_Usuarios getInstancia() {
25         if (instancia == null) {
26             instancia = new Gestion_Usuarios();
27         }
28         return instancia;
29     }
30
31     public Gestion_Usuarios() {
32         initComponents();
33         llenarTabla();
34     }
35
36     public void llenarTabla() {
37         this.modeloDefault = new DefaultTableModel(new String[]{"Correo", "Nombre", "Apellido", "Password", "DPI", "Fecha nac", "Genero", "Nacion", "Rol"}, 0);
38         rowCount: RegistroUsuario.getInstancia().obtenerListaUsuarios().size();
39         tableUsuarios.setModel(dataModel: modeloDefault);
40         TableModel modeloDatos = tableUsuarios.getModel();
41         //modeloDefault.addRow(new Object[this.filas]);
42
43         for(int i = 0; i <= RegistroUsuario.getInstancia().obtenerListaUsuarios().size()-1; i++) {
44             Usuario user = RegistroUsuario.getInstancia().obtenerListaUsuarios().get(index: i);
45             modeloDatos.setValueAt(aValue: user.getCorreo(), rowIndex: i, columnIndex: 0);
46             modeloDatos.setValueAt(aValue: user.getNombre(), rowIndex: i, columnIndex: 1);
47             modeloDatos.setValueAt(aValue: user.getApellido(), rowIndex: i, columnIndex: 2);
48             modeloDatos.setValueAt(aValue: user.getPassword(), rowIndex: i, columnIndex: 3);
49             modeloDatos.setValueAt(aValue: user.getDPI(), rowIndex: i, columnIndex: 4);
50             modeloDatos.setValueAt(aValue: user.getFechaNac(), rowIndex: i, columnIndex: 5);
51             modeloDatos.setValueAt(aValue: user.getGenero(), rowIndex: i, columnIndex: 6);
52             modeloDatos.setValueAt(aValue: user.getNacionalidad(), rowIndex: i, columnIndex: 7);
53             modeloDatos.setValueAt(aValue: user.getAlias(), rowIndex: i, columnIndex: 8);
54             modeloDatos.setValueAt(aValue: user.getTelefono(), rowIndex: i, columnIndex: 9);
55             modeloDatos.setValueAt(aValue: user.getRol(), rowIndex: i, columnIndex: 10);
56         }
57     }
58 }
```

GESTION_REGIONES.JAVA

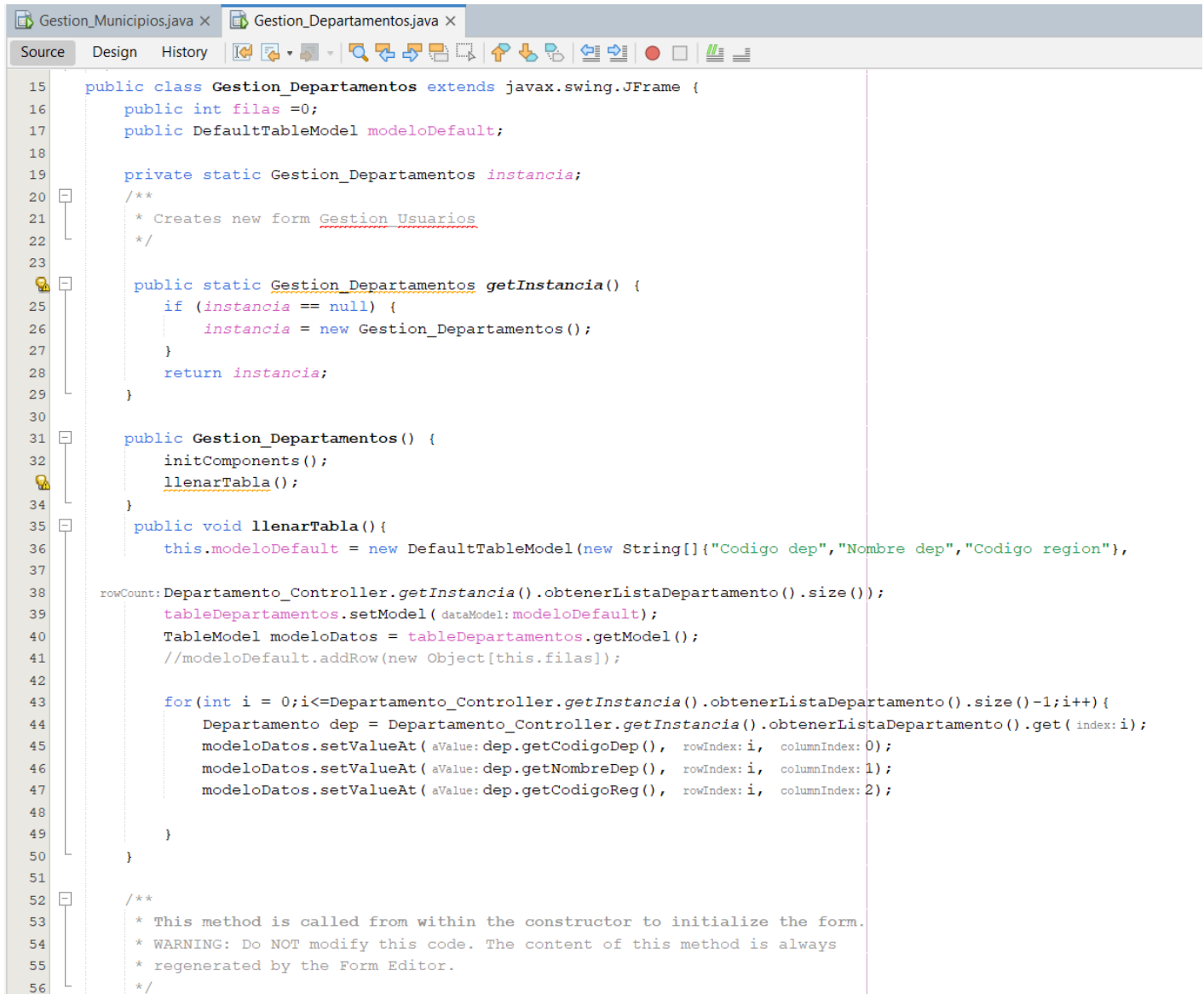


```
15 public class Gestion_Regiones extends javax.swing.JFrame {
16     public int filas =0;
17     public DefaultTableModel modeloDefault;
18
19     private static Gestion_Regiones instancia;
20     /**
21      * Creates new form Gestion Usuarios
22      */
23
24     public static Gestion_Regiones getInstancia() {
25         if (instancia == null) {
26             instancia = new Gestion_Regiones();
27         }
28         return instancia;
29     }
30
31     public Gestion_Regiones() {
32         initComponents();
33         llenarTabla();
34     }
35
36     public void llenarTabla(){
37         this.modeloDefault = new DefaultTableModel(new String[]{"Codigo", "Nombre", "Precio estandar", "Precio especial"},
38         rowCount: Region_Controller.getInstancia().obtenerListaRegion().size());
39         tableRegiones.setModel( dataModel: modeloDefault);
40         TableModel modeloDatos = tableRegiones.getModel();
41         //modeloDefault.addRow(new Object[this.filas]);
42
43         for(int i = 0;i<=Region_Controller.getInstancia().obtenerListaRegion().size()-1;i++){
44             Region reg = Region_Controller.getInstancia().obtenerListaRegion().get( index: i);
45             modeloDatos.setValueAt( aValue: reg.getCodigo(),   rowIndex: i,   columnIndex: 0);
46             modeloDatos.setValueAt( aValue: reg.getNombre(),   rowIndex: i,   columnIndex: 1);
47             modeloDatos.setValueAt( aValue: reg.getPrecioEstandar(),   rowIndex: i,   columnIndex: 2);
48             modeloDatos.setValueAt( aValue: reg.getPrecioEspecial(),   rowIndex: i,   columnIndex: 3);
49         }
50     }
51
52     /**
53      * This method is called from within the constructor to initialize the form.
54      * WARNING: Do NOT modify this code. The content of this method is always
55      * regenerated by the Form Editor.
56      */
57 }
```


GESTION_MUNICIPIOS.JAVA

```
Gestion_Municipios.java x
Source Design History
15 public class Gestion_Municipios extends javax.swing.JFrame {
16     public int filas = 0;
17     public DefaultTableModel modeloDefault;
18
19     private static Gestion_Municipios instancia;
20     /**
21      * Creates new form Gestion_Usuarios
22      */
23
24     public static Gestion_Municipios getInstancia() {
25         if (instancia == null) {
26             instancia = new Gestion_Municipios();
27         }
28         return instancia;
29     }
30
31     public Gestion_Municipios() {
32         initComponents();
33         llenarTabla();
34     }
35     public void llenarTabla() {
36         this.modeloDefault = new DefaultTableModel(new String[]{"Codigo muni", "Nombre muni", "Codigo dep"},
37
38         rowCount: Municipio_Controller.getInstancia().obtenerListaMunicipios().size());
39         tableMunicipios.setModel(dataModel: modeloDefault);
40         TableModel modeloDatos = tableMunicipios.getModel();
41         //modeloDefault.addRow(new Object[this.filas]);
42
43         for(int i = 0; i <= Municipio_Controller.getInstancia().obtenerListaMunicipios().size()-1; i++) {
44             Municipio muni = Municipio_Controller.getInstancia().obtenerListaMunicipios().get(index: i);
45             modeloDatos.setValueAt(aValue: muni.getCodigoMuni(), rowIndex: i, columnIndex: 0);
46             modeloDatos.setValueAt(aValue: muni.getNombreMuni(), rowIndex: i, columnIndex: 1);
47             modeloDatos.setValueAt(aValue: muni.getCodigoDep(), rowIndex: i, columnIndex: 2);
48         }
49     }
50 }
51
52 /**
53  * This method is called from within the constructor to initialize the form.
54  * WARNING: Do NOT modify this code. The content of this method is always
55  * regenerated by the Form Editor.
56  */
```

GESTION_DEPARTAMENTOS.JAVA



```
15 public class Gestion_Departamentos extends javax.swing.JFrame {
16     public int filas = 0;
17     public DefaultTableModel modeloDefault;
18
19     private static Gestion_Departamentos instancia;
20     /**
21      * Creates new form Gestion Usuarios
22      */
23
24     public static Gestion_Departamentos getInstance() {
25         if (instancia == null) {
26             instancia = new Gestion_Departamentos();
27         }
28         return instancia;
29     }
30
31     public Gestion_Departamentos() {
32         initComponents();
33         llenarTabla();
34     }
35     public void llenarTabla() {
36         this.modeloDefault = new DefaultTableModel(new String[]{"Codigo dep", "Nombre dep", "Codigo region"},
37
38         rowCount: Departamento_Controller.getInstance().obtenerListaDepartamento().size());
39         tableDepartamentos.setModel(dataModel: modeloDefault);
40         TableModel modeloDatos = tableDepartamentos.getModel();
41         //modeloDefault.addRow(new Object[this.filas]);
42
43         for(int i = 0; i <= Departamento_Controller.getInstance().obtenerListaDepartamento().size()-1; i++) {
44             Departamento dep = Departamento_Controller.getInstance().obtenerListaDepartamento().get(index: i);
45             modeloDatos.setValueAt(aValue: dep.getCodigoDep(), rowIndex: i, columnIndex: 0);
46             modeloDatos.setValueAt(aValue: dep.getNombreDep(), rowIndex: i, columnIndex: 1);
47             modeloDatos.setValueAt(aValue: dep.getCodigoReg(), rowIndex: i, columnIndex: 2);
48
49         }
50     }
51
52     /**
53      * This method is called from within the constructor to initialize the form.
54      * WARNING: Do NOT modify this code. The content of this method is always
55      * regenerated by the Form Editor.
56      */
```