

**CARRERA:** COMPUTACIÓN

**ASIGNATURA:** Programación Aplicada

**NRO. PRÁCTICA:**

2

**TÍTULO PRÁCTICA:** Reflexión en Java

**OBJETIVO ALCANZADO:**

- Identificar los cambios importantes de Java
- Diseñar e Implementar las nuevas técnicas de programación
- Entender cada una de las características nuevas en Java para llegar a aplicarlas

**ACTIVIDADES DESARROLLADAS**

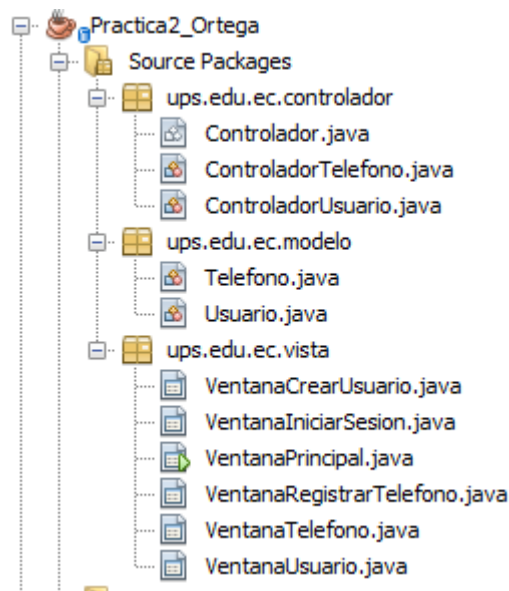
1. Revisar la teoría y conceptos de Java 8, 9, 10, 11, 12, 13, 14, 15.

2. Diseñar e implementar las características de Java para generar la impresión de cualquier lista, de los modelos que tengan el campo id generar automáticamente.

3. Probar y modificar el método validar para que nos permita utilizar excepciones, además de modificar el buscar para controlar el nullpointerexception.

4. Realizar práctica codificando los códigos de las nuevas características de Java y su uso dentro de una agenda telefónica.

**Paquetes del proyecto:**



**Controladores:**

```

package ups.edu.ec.controlador;

import java.util.ArrayList;
import java.util.List;

/**
 * @author Usuario
 * @param <L>
 */
public abstract class Controlador<L> {
    private List<L> listaGenerica;

    public Controlador() {
        listaGenerica = new ArrayList<>();
    }

    public boolean create(L objCrear) {
        if (!listaGenerica.contains(objCrear)) {
            return listaGenerica.add(objCrear);
        }
        return false;
    }

    public L read(L objBuscar) {
        if (listaGenerica.contains(objBuscar)) {
            return (L) listaGenerica.stream().filter(objeto -> objeto.equals(objBuscar)).findFirst().get();
        } else {
            return null;
        }
    }

    public L update(L objAnterior, L objNuevo) {
        int index = listaGenerica.indexOf(objAnterior);
        return listaGenerica.set(index, objNuevo);
    }

    public boolean delete(L objEliminar) {
        return listaGenerica.remove(objEliminar);
    }

    public List<L> findAll() {
        return listaGenerica;
    }
}

package ups.edu.ec.controlador;

import java.util.List;
import ups.edu.ec.modelo.Telefono;

/**
 * @author Usuario
 */
public class ControladorTelefono extends Controlador{

    public ControladorTelefono() {
    }

    @Override
    public int actualizar() {
        var copiaListaTelefonos = (List<Telefono>) List.copyOf(findAll());
        if (copiaListaTelefonos.isEmpty()) {
            return 1;
        } else {
            return copiaListaTelefonos.get(copiaListaTelefonos.size() - 1).getCodigo() + 1
        }
    }
}

```

```

package ups.edu.ec.controlador;

import java.util.List;
import ups.edu.ec.modelo.Telefono;
import ups.edu.ec.modelo.Usuario;

/**
 * @author Usuario
 */
public class ControladorUsuario extends Controlador{

    public ControladorUsuario() {
    }

    @Override
    public int actualizar() {
        if (findAll().size() > 0) {
            return findAll().size() + 1;
        } else {
            return 1;
        }
    }

    public Usuario iniciarSesion(String correo, String clave) {
        for (var usuario : (List<Usuario>) findAll()) {
            if (usuario.getCorreo().equals(correo) && usuario.getClave().equals(clave)) {
                return usuario;
            }
        }
        return null;
    }

    public Usuario readCedula(Usuario usuarioBuscar) {
        var copiaListaUsuarios = (List<Usuario>) List.copyOf(findAll());
        return copiaListaUsuarios.stream().filter(usuario -> usuario.getCedula().equals(usuarioBuscar.getCedula())).
    }

    public Usuario readNumero(Telefono telefonoBuscar) {
        for (var usuario : (List<Usuario>) findAll()) {
            if (usuario.buscarTelefono(telefonoBuscar) != null) {
                return usuario;
            }
        }
        return null;
    }
}

```

**Modelo**

**Telefono:**

```
public class Telefono {
    private int codigo;
    private String tipo;
    private String numero;
    private String operadora;

    public Telefono() {
    }

    public Telefono(int codigo, String tipo, String numero, String operadora) {
        this.codigo = codigo;
        this.tipo = tipo;
        this.numero = numero;
        this.operadora = operadora;
    }

    public Telefono(String numero) {
        this.numero = numero;
    }

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    public String getNumero() {
        return numero;
    }

    public void setNumero(String numero) {
        this.numero = numero;
    }

    public String getOperadora() {
        return operadora;
    }

    public void setOperadora(String operadora) {
        this.operadora = operadora;
    }

    @Override
    public int hashCode() {
        int hash = 3;
        return hash;
    }
}
```

```

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Telefono other = (Telefono) obj;
    if (!Objects.equals(this.numero, other.numero)) {
        return false;
    }
    return true;
}

```

Usuario:

```
package ups.edu.ec.modelo;
```

```

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

```

```

/**
 *
 * @author Usuario
 */

```

```

public class Usuario {
    private int id;
    private String cedula;
    private String nombre;
    private String apellido;
    private String correo;
    private String clave;
    private List<Telefono> listaTelefonos;

```

```

    public Usuario(int id, String cedula, String nombre,
        String apellido, String correo, String clave) {
        this.id = id;
        this.cedula = cedula;
        this.nombre = nombre;
        this.apellido = apellido;
        this.correo = correo;
        this.clave = clave;
        listaTelefonos = new ArrayList<>();
    }

```

```

public Usuario(String cedula, String nombre, String apellido,
    String correo, String clave, List<Telefono> listaTelefonos) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.correo = correo;
    this.clave = clave;
    this.listaTelefonos = listaTelefonos;
}

public Usuario(int id, String cedula, String nombre, String apellido,
    String correo, String clave, List<Telefono> listaTelefonos) {
    this.id = id;
    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.correo = correo;
    this.clave = clave;
    this.listaTelefonos = listaTelefonos;
}

public Usuario(String apellido) {
    this.apellido = apellido;
}

public Usuario() {
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {
    this.cedula = cedula;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

```

```
public String getApellido() {  
    return apellido;  
}  
  
public void setApellido(String apellido) {  
    this.apellido = apellido;  
}  
  
public String getCorreo() {  
    return correo;  
}  
  
public void setCorreo(String correo) {  
    this.correo = correo;  
}  
  
public String getClave() {  
    return clave;  
}  
  
public void setClave(String clave) {  
    this.clave = clave;  
}  
  
public List<Telefono> getListatelefonos() {  
    return listaTelefonos;  
}
```

```

public void agregarTelefono(Telefono telefono){
    this.listaTelefonos.add(telefono);
}

public Telefono buscarTelefono(Telefono tlfBuscar){
    return this.listaTelefonos.stream().filter(t ->
        t.getNumero().equals(tlfBuscar.getNumero())).findFirst().get();
}

public void actualizarTelefono(Telefono tlfNuevo){
    var tlfActualizar = listaTelefonos.stream().filter(t ->
        t.getCodigo() == tlfNuevo.getCodigo()).findFirst().get();
    int index = listaTelefonos.indexOf(tlfActualizar);
    listaTelefonos.set(index, tlfNuevo);
}

public void eliminarTelefono(int codigo){
    var tlfEliminar = listaTelefonos.stream().filter(telefono ->
        telefono.getCodigo() == codigo).findFirst().get();
    listaTelefonos.remove(tlfEliminar);
}

@Override
public int hashCode() {
    int hash = 3;
    hash = 17 * hash + Objects.hashCode(this.apellido);
    return hash;
}

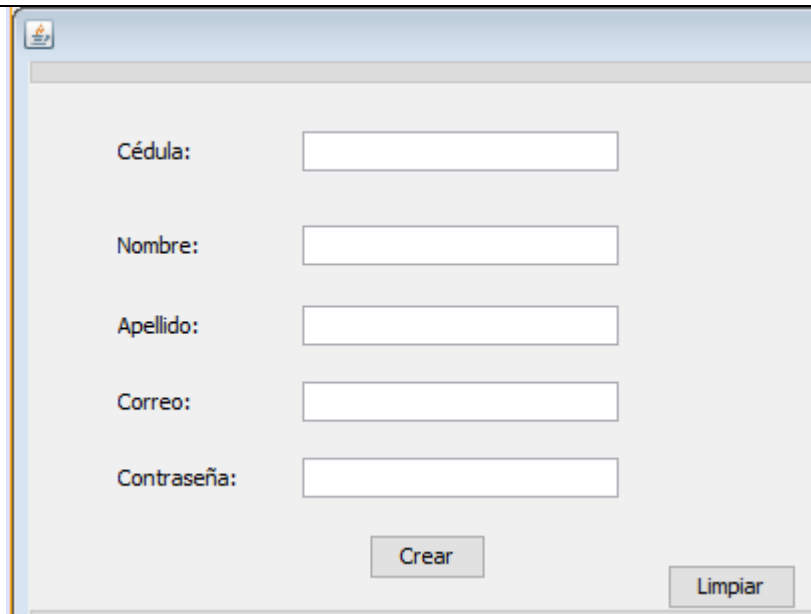
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Usuario other = (Usuario) obj;
    if (!Objects.equals(this.apellido, other.apellido)) {
        return false;
    }
    return true;
}

```

Ventanas/Vista

VentanaCrearUsuario:





Cédula:

Nombre:

Apellido:

Correo:

Contraseña:

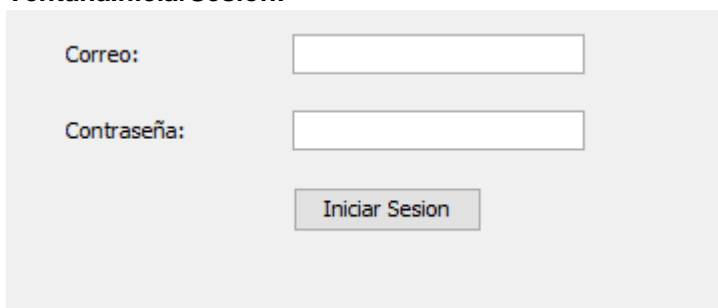
```
private void txtApellidoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    limpiar();
}

private void btnCrearActionPerformed(java.awt.event.ActionEvent evt) {
    String cedula = (String) txtCedula.getValue();
    String nombre = txtNombre.getText().trim();
    String apellido = txtApellido.getText().trim();
    String correo = txtCorreo.getText();
    String clave = String.valueOf(txtClave.getPassword());

    if (cedula.length() != 10 || nombre.isBlank() || apellido.isBlank() || correo.isBlank() || clave.isBlank())
        JOptionPane.showMessageDialog(this, "Por favor llene todos los campos correctamente");
    else {
        var usuario = controladorUsuario.comprobarMayusculas(new Usuario(controladorUsuario.actualizar(),
            (String) txtCedula.getValue(),
            txtNombre.getText(), txtApellido.getText(), txtCorreo.getText(),
            String.valueOf(txtClave.getPassword())));
        controladorUsuario.create(usuario);
        JOptionPane.showMessageDialog(this, "Se ha registrado el usuario");
        this.hide();
    }
}
```

### VentanaIniciarSesion:



Correo:

Contraseña:

```

public class VentanaIniciarSesion extends javax.swing.JInternalFrame {

    private ControladorUsuario controladorUsuario;

    private VentanaPrincipal ventanaPrincipal;
    private VentanaTelefono ventanaTelefono;
    private VentanaUsuario ventanaUsuario;

    /**
     * Creates new form VentanaIniciarSesion
     * @param controladorUsuario
     * @param ventanaPrincipal
     * @param ventanaTelefono
     * @param ventanaUsuario
     */
    public VentanaIniciarSesion(ControladorUsuario controladorUsuario, VentanaPrincipal ventanaPrincipal,
        VentanaTelefono ventanaTelefono, VentanaUsuario ventanaUsuario) {
        initComponents();

        this.controladorUsuario = controladorUsuario;
        this.ventanaPrincipal = ventanaPrincipal;
        this.ventanaTelefono = ventanaTelefono;
        this.ventanaUsuario = ventanaUsuario;
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        Usuario usuarioEncontrado = controladorUsuario.iniciarSesion(txtCorreo.getText(),
            String.valueOf(txtClave.getPassword()));
        System.out.println(usuarioEncontrado);
        if (usuarioEncontrado != null) {
            JOptionPane.showMessageDialog(this, "Sesión iniciada");
            ventanaTelefono.setUsuario(usuarioEncontrado);
            ventanaUsuario.setUsuario(usuarioEncontrado);
            ventanaPrincipal.getMenuGestion().setEnabled(true);
            ventanaPrincipal.getMenuItemCerrarSesion().setEnabled(true);
            ventanaPrincipal.getMenuItemCrearUsuario().setEnabled(false);
            ventanaPrincipal.getMenuItemIniciarSesion().setEnabled(false);
            txtCorreo.setText("");
            txtClave.setText("");
            this.hide();

        } else {
            JOptionPane.showMessageDialog(this, "Usuario no encontrado");
        }
    }

    private void formInternalFrameClosed(javax.swing.event.InternalFrameEvent evt) {
        txtCorreo.setText("");
        txtClave.setText("");
    }

    private void formComponentHidden(java.awt.event.ComponentEvent evt) {
        txtCorreo.setText("");
        txtClave.setText("");
    }
}

```

**VentanaRegistrarTelefono:**

Buscar Por:

----

▼

Buscar

Limpiar

Nombre:

Apellido:

Cédula:

Correo:

Listar Todo

Código	Número	Tipo	Operadora
--------	--------	------	-----------

```

public VentanaRegistrarTelefono(ControladorUsuario controladorUsuario,
    ControladorTelefono controladorTelefono) {
    initComponents();
    comboOpc.setSelectedIndex(0);
    this.controladorUsuario = controladorUsuario;
    this.controladorTelefono = controladorTelefono;
}

/**
private void btnListarActionPerformed(java.awt.event.ActionEvent evt) {
    var listaTelefono = (List<Telefono>) controladorTelefono.findAll();

    if (listaTelefono.isEmpty()) {
        JOptionPane.showMessageDialog(this, "No existen teléfonos");
    } else {
        llenarTabla(listaTelefono);
    }
}
}

```

```

public void llenarTabla(List<Telefono> listaTelefonos) {
    DefaultTableModel modeloTabla = (DefaultTableModel) tblTelf.getModel();
    modeloTabla.setRowCount(0);

    for (Telefono tlf : listaTelefonos) {
        Object[] rowData = {tlf.getCodigo(), tlf.getNumero(), tlf.getTipo(), tlf.getOperadora()};
        modeloTabla.addRow(rowData);
    }

    tblTelf.setModel(modeloTabla);
}

```

### VentanaTelefono:

The screenshot shows a Java Swing window titled "VentanaTelefono". It contains a form with four input fields: "Código:" (text), "Tipo:" (dropdown menu), "Número:" (text), and "Operadora:" (dropdown menu). Below the form are four buttons: "Crear", "Actualizar", "Eliminar", and "Limpiar". At the bottom of the window is a table with four columns: "Código", "Número", "Tipo", and "Operadora". The table is currently empty.

Código	Número	Tipo	Operadora
--------	--------	------	-----------

```

public class VentanaTelefono extends javax.swing.JInternalFrame {

    private ControladorUsuario controladorUsuario;
    private ControladorTelefono controladorTelefono;

    private Usuario usuario;
    private Telefono telefono;

    /**
     * Creates new form VentanaTelefono
     * @param controladorUsuario
     * @param controladorTelefono
     */
    public VentanaTelefono(ControladorUsuario controladorUsuario,
        ControladorTelefono controladorTelefono) {
        initComponents();
        this.controladorTelefono = controladorTelefono;
        this.controladorUsuario = controladorUsuario;
        telefono = new Telefono();
        comboTipo.setSelectedIndex(0);
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }

    private void btnCrearActionPerformed(java.awt.event.ActionEvent evt) {
        int codigo = Integer.parseInt(txtCodigo.getText());
        String numero = (String) txtNumero.getValue();
        String tipo = (String) comboTipo.getSelectedItem();
        String operadora = (String) comboOperadora.getSelectedItem();
        Telefono tlf = new Telefono(codigo, tipo, numero, operadora);

        if (numero.isEmpty() || tipo.equals("----") || operadora.equals("----")) {
            JOptionPane.showMessageDialog(this, "Llene todos los campos para crear un teléfono");
        } else {
            controladorTelefono.create(tlf);
            usuario.agregarTelefono(tlf);
            controladorUsuario.update(usuario, usuario);
            JOptionPane.showMessageDialog(this, "Teléfono creado");
            llenarTablaTelefono();
            limpiar();
        }
    }
}

```

```

private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    int codigo = Integer.parseInt(txtCodigo.getText());
    String numero = (String) txtNumero.getValue();
    String tipo = (String) comboTipo.getSelectedItem();
    String operadora = (String) comboOperadora.getSelectedItem();
    Telefono tlfActualizado = new Telefono(codigo, tipo, numero, operadora);

    if (numero.isEmpty() || tipo.equals("----") || operadora.equals("----"))
        JOptionPane.showMessageDialog(this, "Por favor llene todos los campos");
    else {
        controladorTelefono.update(this.telefono, tlfActualizado);
        usuario.actualizarTelefono(tlfActualizado);
        controladorUsuario.update(usuario, usuario);
        JOptionPane.showMessageDialog(this, "Teléfono actualizado");
        llenarTablaTelefono();
        limpiar();
    }
}

```

```

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int codigo = Integer.parseInt(txtCodigo.getText());
    String numero = (String) txtNumero.getValue();
    String tipo = (String) comboTipo.getSelectedItem();
    String operadora = (String) comboOperadora.getSelectedItem();
    Telefono tlf = new Telefono(codigo, tipo, numero, operadora);

    controladorTelefono.delete(tlf);
    usuario.eliminarTelefono(codigo);
    controladorUsuario.update(usuario, usuario);
    JOptionPane.showMessageDialog(this, "Teléfono eliminado");
    llenarTablaTelefono();
    limpiar();
}

```

```

private void comboTipoActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String item = (String) comboTipo.getSelectedItem();
        if (item.equals("Casa")) {
            txtNumero.setEditable(true);
            txtNumero.setFormatterFactory(
                new javax.swing.text.DefaultFormatterFactory(
                    new javax.swing.text.MaskFormatter("(593) 0##-####-###")
                )
            );
        } else if (item.equals("Celular")) {
            txtNumero.setEditable(true);
            txtNumero.setFormatterFactory(
                new javax.swing.text.DefaultFormatterFactory(
                    new javax.swing.text.MaskFormatter("(593) 0##-####-####")
                )
            );
        } else {
            txtNumero.setEditable(false);
            txtNumero.setFormatterFactory(
                new javax.swing.text.DefaultFormatterFactory(
                    new javax.swing.text.MaskFormatter("seleccione tipo")
                )
            );
        }
    } catch (java.text.ParseException ex) {
        JOptionPane.showMessageDialog(this, "Formato del número del teléfono erroneo");
    }
}

```

```

private void jTableMouseClicked(java.awt.event.MouseEvent evt) {
    int fila = jTable.getSelectedRow();
    int codigo = (int) jTable.getValueAt(fila, 0);
    String numero = (String) jTable.getValueAt(fila, 1);
    String tipo = (String) jTable.getValueAt(fila, 2);
    String operadora = (String) jTable.getValueAt(fila, 3);

    this.telefono = new Telefono(codigo, tipo, numero, operadora);

    txtCodigo.setText(String.valueOf(codigo));
    comboTipo.setSelectedItem(tipo);
    txtNumero.setValue(numero);
    comboOperadora.setSelectedItem(operadora);

    comboTipo.setEnabled(false);

    btnCrear.setEnabled(false);
    btnActualizar.setEnabled(true);
    btnEliminar.setEnabled(true);
}

```

#### VentanaUsuario:

Nombre:

Apellido:

Cédula:

Correo:

Contraseña:

```

private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    String nombre = txtNombre.getText();
    String apellido = txtApellido.getText();
    String correo = txtCorreo.getText();
    String clave = txtClave.getText();

    if (nombre.isBlank() || apellido.isBlank() || correo.isBlank() || clave.isBlank()) {
        JOptionPane.showMessageDialog(this, "Por favor llene todas las casillas");
    } else {
        var usuarioNuevo = controladorUsuario.comprobarMayusculas(new Usuario(usuario.getId(),
            txtCedula.getText(), nombre, apellido, correo, clave, usuario.getListaTelefonos()));

        controladorUsuario.update(usuario, usuarioNuevo);
        this.usuario = usuarioNuevo;
        llenarDatos();

        JOptionPane.showMessageDialog(this, "Datos Actualizados");
    }
}

```

## VentanaPrincipal:

Menú Gestión

```
public class VentanaPrincipal extends javax.swing.JFrame {

    private ControladorUsuario controladorUsuario;
    private ControladorTelefono controladorTelefono;
    private VentanaCrearUsuario ventanaRegistrarUsuario;
    private VentanaIniciarSesion ventanaIniciarSesion;
    private VentanaRegistrarTelefono ventanaRegistrarTelefono;
    private VentanaTelefono ventanaTelefono;
    private VentanaUsuario ventanaUsuario;

    /**
     * Creates new form VentanaPrincipal
     */
    public VentanaPrincipal() {
        initComponents();
        controladorUsuario = new ControladorUsuario();
        controladorTelefono = new ControladorTelefono();

        ventanaTelefono = new VentanaTelefono(controladorUsuario, controladorTelefono);
        ventanaUsuario = new VentanaUsuario(controladorUsuario);
        ventanaRegistrarUsuario = new VentanaCrearUsuario(controladorUsuario);
        ventanaIniciarSesion = new VentanaIniciarSesion(controladorUsuario, this, ventanaTelefono, ventanaUsuario);
        ventanaRegistrarTelefono = new VentanaRegistrarTelefono(controladorUsuario, controladorTelefono);

        jDesktopPanel.add(ventanaRegistrarUsuario);
        jDesktopPanel.add(ventanaIniciarSesion);
        jDesktopPanel.add(ventanaRegistrarTelefono);
        jDesktopPanel.add(ventanaTelefono);
        jDesktopPanel.add(ventanaUsuario);

        this.setExtendedState(VentanaPrincipal.MAXIMIZED_BOTH);
    }
}
```



```

public JMenu getMenuGestion() {
    return jMenu2;
}

public JMenuItem getMenuItemCerrarSesion() {
    return menuItemCerrarSesion;
}

public JMenuItem getMenuItemCrearUsuario() {
    return menuItemCrearUsuario;
}

public JMenuItem getMenuItemIniciarSesion() {
    return menuItemIniciarSesion;
}

public void cerrarVentanas() {
    ventanaRegistrarUsuario.setVisible(false);
    ventanaIniciarSesion.setVisible(false);
    ventanaRegistrarTelefono.setVisible(false);
    ventanaTelefono.setVisible(false);
    ventanaUsuario.setVisible(false);
}

private void menuItemExitActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

private void menuItemCrearUsuarioActionPerformed(java.awt.event.ActionEvent evt) {
    cerrarVentanas();
    ventanaRegistrarUsuario.setVisible(true);
}

private void menuItemIniciarSesionActionPerformed(java.awt.event.ActionEvent evt) {
    cerrarVentanas();
    ventanaIniciarSesion.setVisible(true);
}

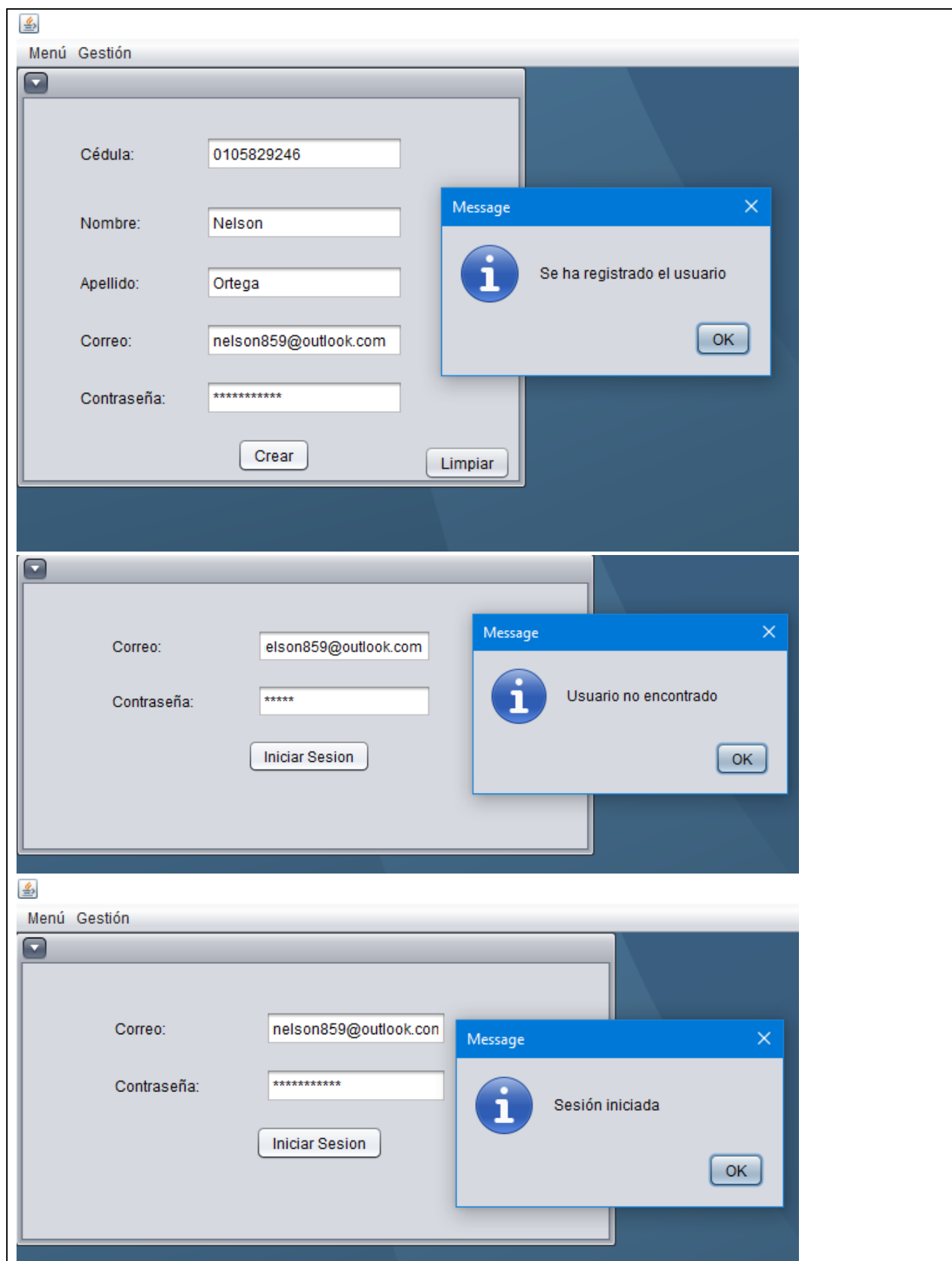
private void menuItemCerrarSesionActionPerformed(java.awt.event.ActionEvent evt) {
    cerrarVentanas();
    jMenu2.setEnabled(false);
    menuItemCerrarSesion.setEnabled(false);
    menuItemCrearUsuario.setEnabled(true);
    menuItemIniciarSesion.setEnabled(true);
}

private void menuItemTelefonoActionPerformed(java.awt.event.ActionEvent evt) {
    cerrarVentanas();
    ventanaRegistrarTelefono.setVisible(true);
}

private void menuItemGestionUsuarioActionPerformed(java.awt.event.ActionEvent evt) {
    cerrarVentanas();
    ventanaUsuario.setVisible(true);
}

```

Screenshots de corrida del programa:



Menú Gestión

Nombre: Paul

Apellido: Ortega

Cédula: 0105829246

Correo: nelson859@outlook.com

Contraseña: UniversidadEjemplo

Cambiar Datos

Limpiar

Message

i

Datos Actualizados

OK

Menú Gestión

Código: 2

Tipo: ---- ▾

Número: seleccione tipo

Operadora: ---- ▾


Crear

Actualizar

Eliminar

Limpiar

Código	Número	Tipo	Operadora
1	(593)02-5689-744	Casa	Claro


Menú Gestión

▼

Buscar Por:

Cédula

▼

0105829246

Buscar

Limpiar

Nombre:

Nelson

Apellido:

Ortega

Cédula:

0105829246

Correo:

nelson859@outlook.com

Listar Todo

Código	Número	Tipo	Operadora
1	(593)02-5689-744	Casa	Claro

**RESULTADO(S) OBTENIDO(S):**

- Realizar procesos de investigación sobre los cambios importantes de Java
- Entender las aplicaciones de codificación de las nuevas características en base a la programación genérica
- Entender las funcionalidades adicionales de Java.

## CONCLUSIONES:

- Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.

### RECOMENDACIONES:

- Realizar el trabajo dentro del tiempo establecido.
- Consultar webs oficiales para estar al tanto de las nuevas versiones de Java y lo que ofrece

Nombre de estudiante: Nelson Paul Ortega Segarra

Firma de estudiante:

A close-up of a handwritten letter 'B' on a yellow background with a faint grid pattern. The letter is drawn with a thick black marker. There is a small red mark above the letter.

