
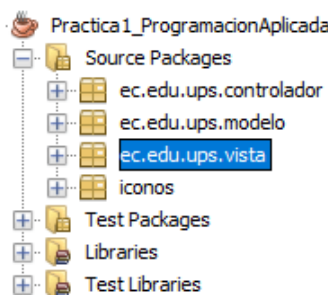
	<b>VICERRECTORADO DOCENTE</b>	<b>Código:</b> GUIA-PRL-001
	CONSEJO ACADÉMICO	<b>Aprobación:</b> 2016/04/06
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

			PRÁCTICA DE LABORATORIO		
CARRERA: COMPUTACIÓN			ASIGNATURA: Programación Aplicada		
NRO. PRÁCTICA:	2	TÍTULO PRÁCTICA: Clase Genéricas en Java			
OBJETIVO ALCANZADO:					
Conocer el funcionamiento y las utilidades que nos brindan las clases genéricas en Java.					
Recordar el patrón de diseño MVC y los métodos CRUD que nos servían para crear, modificar, buscar y eliminar algún tipo de objeto.					
Implementar los métodos CRUD en una clase genérica.					
ACTIVIDADES DESARROLLADAS					
1. Creación de los paquetes modelo, vista, controlador siguiendo el patrón de diseño MVC:					
					
2. Creación de la clase genérica Controlador:					
En esta clase genérica se crearon los métodos CRUD, para el método buscar se uso streams que nos simplifican el código cumpliendo con las mismas funcionalidades que hacerlo sin streams.					
Los Streams en java son un nuevo modelo de datos que nos permite tratar las colecciones como si de etapas de un proceso ETL (“Extract Transform and Load”) se tratara. Esto nos permite utilizar las funciones Map y Reduce tan comunes en esos procesos, especialmente en la etapa de transformación.					
En java 8, toda colección tiene un método stream() que transformará dicha estructura en Stream.					

```
public class Controlador <T>{  
    private List<T> listado;  
    private int tamaño;  
  
    public Controlador() {  
        listado = new ArrayList();  
        tamaño = 0;  
    }  
  
    public Controlador(List<T> listado, int tamaño) {  
        this.listado = listado;  
        this.tamaño = tamaño;  
    }  
  
    public boolean crear(T objeto) {  
        return listado.add(objeto);  
    }  
  
    public List<T> buscar(T objetoBuscado){  
        return listado.stream().filter(objeto -> objeto.equals(objetoBuscado)).collect(Collectors.toList());  
    }  
  
    public boolean actualizar(T objetoActualizado) {  
        for (T objeto : listado) {  
            if (objeto.equals(objetoActualizado)) {  
                listado.set(listado.indexOf(objeto), objetoActualizado);  
                return true;  
            }  
        }  
        return false;  
    }  
  
    public boolean eliminar(T objeto) {  
        return listado.remove(objeto);  
    }  
  
    public List<T> listar() {  
        return listado;  
    }  
}
```

### 3. Se crearon las clases Persona y Teléfono dentro del modelo

En este caso el objeto persona tendrá como atributo un objeto de tipo teléfono

También se agregaron los métodos get() y set() así como también los métodos equals() y hashCode().

```

-  */
public class Persona {

    private String cedula;
    private String nombre;
    private String apellido;
    private String direccion;
    private Telefono telefono;

3   public Persona() {
        this.cedula = "";
        this.nombre = "";
        this.apellido = "";
        this.direccion = "";
        this.telefono = new Telefono();
-   }

3   public Persona(String cedula, String nombre, String apellido, String direccion, Telefono telefono) {
        this.cedula = cedula;
        this.nombre = nombre;
        this.apellido = apellido;
        this.direccion = direccion;
        this.telefono = telefono;
-   }


-  */

public class Telefono {

    private String numero;
    private String operadora;
    private String tipo;

    public Telefono() {
        numero = "";
        operadora = "";
        tipo = "";
    }

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

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

4. Se diseño una interfaz gráfica que permita al usuario agregar actualizar eliminar o buscar registrar números telefónicos

- Ventana principal:



- Ventana para crear un contacto:

**Nuevo Contacto**

Cedula:

Nombre:

Apellido:

Direccion:

Numero Telefonico:

Operadora:

Tipo:

- Ventana para editar contactos:

**Contacto Seleccionado**

Cedula:

Nombre:


Apellido:

Direccion:

Numero Telefonico:

Operadora:

Tipo:

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- Ventana para ver todos los contactos registrados:



##### 5. En el JFrame (Ventana Principal) se instancio todas las ventanas creadas y el Controlador:

```

~ /
public class VentanaPrincipal extends javax.swing.JFrame {

    /**
     * Creates new form VentanaPrincipal
     */

    private VentanaPersona ventanaPersona;
    private VentanaContactosRegistrados ventanaContactosRegistrados;
    private Controlador<Persona> controlador;
    private VentanaEditarContactos ventanaEditarContactos;

    public VentanaPrincipal() {
        initComponents();

        controlador = new Controlador();
        ventanaPersona = new VentanaPersona(controlador);
        ventanaEditarContactos = new VentanaEditarContactos(controlador);
        ventanaContactosRegistrados = new VentanaContactosRegistrados(controlador, ventanaEditarContactos);
    }

```

##### 6. Finalmente se programo cada uno de los botones:

Los más importantes serian:

- Buscar:

```
private void jButtonBuscarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel modelo = (DefaultTableModel) jTableContactos.getModel();
    modelo.setRowCount(0);
    List personasEncontradas;

    if (jComboBoxCondicion.getSelectedItem().toString().equals("Apellido")) {
        personasEncontradas = controlador.buscar(new Persona("", "", jTextFieldBusqueda.getText(), "", null));


        if (personasEncontradas.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No se encontraron Contactos con ese apellido");
        } else {
            for (Object object : personasEncontradas) {
                Persona persona = (Persona) object;
                Object[] rowData = {persona.getCedula(), persona.getNombre(), persona.getApellido(), persona.getDireccion(), persona.getTelefono()};
                modelo.addRow(rowData);
            }
            jTableContactos.setModel(modelo);
        }
    } else {
        personasEncontradas = controlador.listar();

        if (personasEncontradas.isEmpty()) {
            JOptionPane.showMessageDialog(this, "No se encontraron Contactos con ese numero");
        } else {
            for (Object object : personasEncontradas) {
                Persona persona = (Persona) object;
                if (persona.getTelefono().getNumero().equals(jTextFieldBusqueda.getText())) {
                    Object[] rowData = {persona.getCedula(), persona.getNombre(), persona.getApellido(), persona.getDireccion(), persona.getTelefono()};
                    modelo.addRow(rowData);
                }
            }
            jTableContactos.setModel(modelo);
        }
    }
}
}
```

- Actualizar:

```
private void jButtonActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    if (jTextFieldCedula.getText().trim().equals("") || jTextFieldNombre.getText().trim().equals("") || jTextFieldApellido.getText().trim().equals("") ||
        jTextFieldDireccion.getText().trim().equals("") || jTextFieldNumero.getText().trim().equals("") || jTextFieldOperadora.getText().trim().equals("") ||
        jTextFieldTipo.getText().trim().equals("")) {
        JOptionPane.showMessageDialog(this, "Porfavor rellene todos los campos");
    } else {
        Telefono telefono = new Telefono(jTextFieldNumero.getText(), jTextFieldOperadora.getText(), jTextFieldTipo.getText());
        Persona persona = new Persona(jTextFieldCedula.getText(), jTextFieldNombre.getText(), jTextFieldApellido.getText(), jTextFieldDireccion.getText(), telefono);

        if (controlador.actualizar(persona)) {
            JOptionPane.showMessageDialog(this, "Contacto actualizado con éxito");
            this.dispose();
        } else {
            JOptionPane.showMessageDialog(this, "El contacto no se actualizo porfavor vuelva a intentarlo");
        }
    }
}
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- Eliminar:

```

} private void jButtonEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int respuesta = JOptionPane.showConfirmDialog(this, "Esta eguro que decea eliminar este contacto");
    if (respuesta == JOptionPane.YES_OPTION) {
        controlador.eliminar(this.persona);
        JOptionPane.showMessageDialog(this, "El Contacto se ha eliminado con éxito");
        this.dispose();
    }
}

```

- Agregar:

```

private void jButtonGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    if (jTextFieldCedula.getText().trim().equals("") || jTextFieldNombre.getText().trim().equals("") || jTextFieldApellido.getText().trim().equals("")
        || jTextFieldDireccion.getText().trim().equals("") || jTextFieldNumero.getText().trim().equals("") || jTextFieldOperadora.getText().trim().equals("")
        || jTextFieldTipo.getText().trim().equals("")) {
        JOptionPane.showMessageDialog(this, "Porfavor rellene todos los campos");
    } else {
        Telefono telefono = new Telefono(jTextFieldNumero.getText(), jTextFieldOperadora.getText(), jTextFieldTipo.getText());
        Persona persona = new Persona(jTextFieldCedula.getText(), jTextFieldNombre.getText(), jTextFieldApellido.getText(), jTextFieldDireccion.getText(), telefono);

        if(controlador.crear(persona)) {
            JOptionPane.showMessageDialog(this, "Contacto agregado con éxito");
            limpiar();
        } else {
            JOptionPane.showMessageDialog(this, "El contacto no se guardo porfavor vuelva a intentarlo");
        }
    }
}

```


## RESULTADO(S) OBTENIDO(S):

En este trabajo pudimos poner en practica el uso de la clase genérica y recordar algunos conceptos de programación como los patrones de diseño, las interfaces graficas o los métodos CRUD.

## CONCLUSIONES:

En conclusión, diría que las clases genéricas proporcionan los medios para describir el concepto de pila (o cualquier otra clase) en forma independiente de su tipo. Así podemos crear instancias de objetos con tipos específicos de la clase genérica. Esto nos permite una gran oportunidad de reutilizar el software.

Una vez que tenemos una clase genérica, podemos utilizar una notación concisa para indicar el tipo actual a utilizarse en lugar del parámetro de tipo de la clase.

	<b>VICERRECTORADO DOCENTE</b>	<b>Código:</b> GUIA-PRL-001
	CONSEJO ACADÉMICO	<b>Aprobación:</b> 2016/04/06
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

### RECOMENDACIONES:

Realizar el trabajo dentro del tiempo establecido.

**Nombre de estudiante:** Ariel Vazquez

**Firma de estudiante:**

