	<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		


# Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

---

Universidad Politécnica Salesiana

## Vicerrectorado Docente

Código del Formato:	GUIA-PRL-001
Versión:	VF1.0
Elaborado por:	Directores de Área del Conocimiento Integrantes Consejo Académico
Fecha de elaboración:	2016/04/01
Revisado por:	Consejo Académico
Fecha de revisión:	2016/04/06
Aprobado por:	Lauro Fernando Pesántez Avilés Vicerrector Docente
Fecha de aprobación:	2016/14/06
Nivel de confidencialidad:	Interno

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

## Descripción General

### Propósito


El propósito del presente documento es definir un estándar para elaborar documentación de guías de práctica de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana, con la finalidad de lograr una homogenización en la presentación de la información por parte del personal académico y técnico docente.


### Alcance

El presente estándar será aplicado a toda la documentación referente a informes de prácticas de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana.

### Formatos


- Formato de Guía de Práctica de Laboratorio / Talleres / Centros de Simulación – para Docentes
- Formato de Informe de Práctica de Laboratorio / Talleres / Centros de Simulación – para Estudiantes

	<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		


		<b>FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES</b>	
<b>CARRERA:</b> COMPUTACIÓN		<b>ASIGNATURA:</b> Programación Aplicada	
<b>NRO. PRÁCTICA:</b>	1	<b>TÍTULO PRÁCTICA:</b> Reflexión en Java	
<b>OBJETIVO:</b> Identificar los cambios importantes de Java Diseñar e Implementar las nuevas tecnicas de programación Entender cada una de las características nuevas en Java			
<b>INSTRUCCIONES</b> (Detallar las instrucciones que se dará al estudiante):		1. Revisar los conceptos fundamentales de Java	
		2. Establecer las características de Java en reflexión	
		3. Implementar y diseñar los nuevos componentes de reflexión	
		4. Realizar el informe respectivo según los datos solicitados.	
<b>ACTIVIDADES POR DESARROLLAR</b> (Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)			
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 automaticamente.			
3. Probar y modificar el metodo validar para que nos permita utilizar excepciones, ademas de modificar el buscar para controlar el nullpointerexception.			
4. Realizar práctica codificando los codigos de las nuevas características de Java y su uso dentro de una agenda telefónica.			
<b>RESULTADO(S) OBTENIDO(S):</b> 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.			
<b>CONCLUSIONES:</b> Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.			
<b>RECOMENDACIONES:</b> Realizar el trabajo dentro del tiempo establecido.			

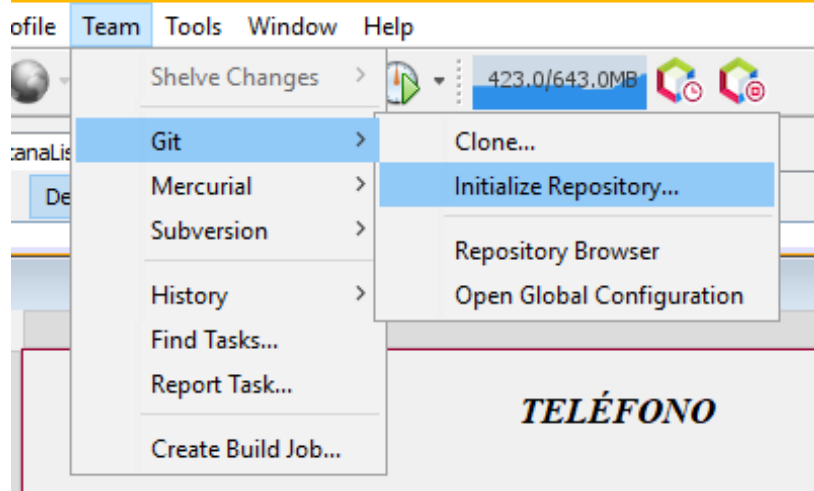
**Docente / Técnico Docente:** \_\_\_\_\_

**Firma:** \_\_\_\_\_

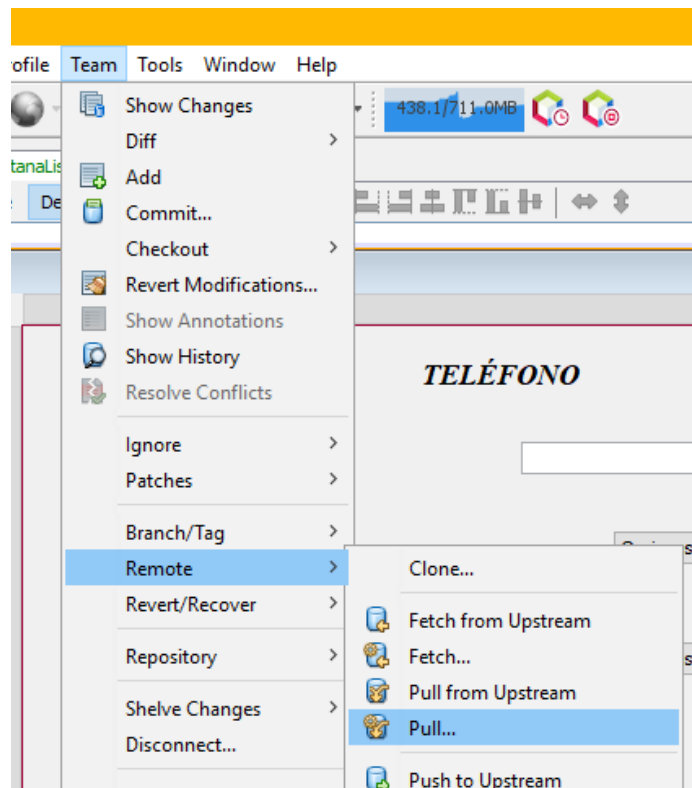
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		<b>FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES</b>	
<b>CARRERA:</b> Ingeniería en Computación		<b>ASIGNATURA:</b> Programación Aplicada	
<b>NRO. PRÁCTICA:</b>	2	<b>TÍTULO PRÁCTICA:</b> Reflexión Java	
<b>OBJETIVO ALCANZADO:</b> Identificar los cambios importantes de Java Diseñar e Implementar las nuevas técnicas de programación Entender cada una de las características nuevas en Java			
<b>ACTIVIDADES DESARROLLADAS</b>			
1. Revisar los conceptos fundamentales de Java			
2. Establecer las características de Java en reflexión			
3. Implementar y diseñar los nuevos componentes de reflexión:  Creamos un repositorio en GitHub con el nombre de la práctica: 			
Iniciamos el repositorio subiendo el proyecto creado con antelación:			

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

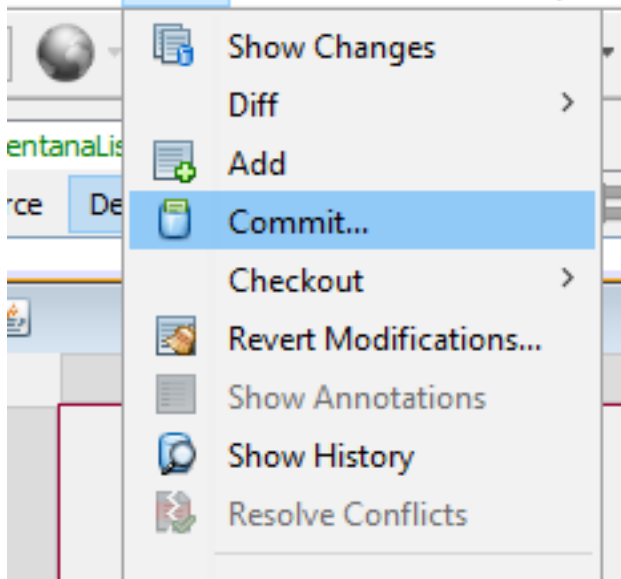


Recuperamos los archivos “.gitignore”, “LICENSE ” y “README.md” haciendo un pull:



Hacemos un commit:

Profile Team Tools Window Help



AgendaTelefonicaG - master

Commit Message:

"Añadiendo al repositorio el proyecto Agenda Telefónica"

Author: LissethReinoso <LissethReinoso30@users.noreply.github.com> Committer: LissethReinoso <LissethReinoso30@users.noreply.github.com>

☐ Amend Last Commit

☒ Files to Commit:


File	Status	Commit Action	Repository Path
build.xml	-/Added	Exclude from Commit	build.xml
manifest.mf	-/Added	Exclude from Commit	manifest.mf
build-impl.xml	-/Added	Exclude from Commit	nbproject/build-impl.xml
genfiles.properties	-/Added	Exclude from Commit	nbproject/genfiles.properties
<input checked="" type="checkbox"/> project.properties	-/Added	Commit	nbproject/project.properties
<input checked="" type="checkbox"/> project.xml	-/Added	Commit	nbproject/project.xml
<input checked="" type="checkbox"/> ControladorPersona.java	-/Added	Commit	src\ec\edu\ups\controlador\ControladorPersona.java
<input checked="" type="checkbox"/> ControladorTelefono.java	-/Added	Commit	src\ec\edu\ups\controlador\ControladorTelefono.java

By right-clicking on a row you may specify some additional Actions.

☒ Update Task

Commit Cancel Help



	<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		

Utilizamos el proyecto anterior realizando algunos cambios:

En el paquete “ec.edu.ups.modelo” creamos las clases:

Clase “Persona”:



```
//creación de una clase genérica
public class Persona<T> {

    private T cedula;
    private T nombre;
    private T apellido;
    private T correoElectronico;
    private T contrasenia;

    private Telefono telf;
    //Agregacion
    private List<Telefono> telefonos;

    public Persona() {
        telefonos = new ArrayList<>();
    }

    public Persona(T cedula, T nombre, T apellido) {
        this.cedula = cedula;
        this.nombre = nombre;
        this.apellido = apellido;

        telefonos = new ArrayList<>();
    }
}
```

```
public Persona(T cedula, T nombre, T apellido, T correoElectronico, T contrasenia) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.correoElectronico = correoElectronico;
    this.contrasenia = contrasenia;
}

public T getCedula() {
    return cedula;
}

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

public T getNombre() {
    return nombre;
}

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

public T getApellido() {
    return apellido;
}

public void setApellido(T apellido) {
    this.apellido = apellido;
}

public T getCorreoElectronico() {
    return correoElectronico;
}

public void setCorreoElectronico(T correoElectronico) {
    this.correoElectronico = correoElectronico;
}

public T getContrasenia() {
    return contrasenia;
}

public void setContrasenia(T contrasenia) {
    this.contrasenia = contrasenia;
}

//métodos de la agregación
public boolean agregarTelefono(Telefono telefono) {
    return this.telefonos.add(new Telefono(telefono.getNumero(), telefono.getTipo(), telefono.getOperadora()));
}
```

```

public boolean actualizarTelefono( String numero,String tipo, String operadora) {

    int pos = posicion(numero);
    if (pos >= 0) {
        Telefono t = this.telefonos.get(pos);
        t.setNumero(numero);
        t.setTipo(tipo);
        t.setOperadora(operadora);
        return this.telefonos.set(pos, t) != null;
    }
    return false;
}

public void eliminarTelefono(Telefono telefono) {
    if (telefonos.contains(telefono)) {
        int index = telefonos.indexOf(telefono);
        telefonos.remove(index);
    }
}

public void listar() {
    telefonos.stream().forEach(a->System.out.println(a));
}

public Telefono buscar(int numero) {

    return telefonos.get(numero);

}

public int posicion(String numero) {
    for (int i = 0; i < this.telefonos.size(); i++) {
        Telefono t = this.telefonos.get(i);
        if (t.getNumero().equals(numero)) {
            return i;
        }
    }
    return -1;
}

@Override
public String toString() {
    return "Persona{" + "cedula=" + cedula + ", nombre=" + nombre + ", apellido=" + apellido + ", correoElectronico=" +

+ correoElectronico + ", contrasenia=" + contrasenia + ", telefonos=" + telefonos + '}';

```

Y la clase "Teléfono":

```
public class Telefono<T> {

    private T numero;
    private T tipo;
    private T operadora;

    public Telefono() {
    }

    public Telefono( T numero, T tipo, T operadora) {

        this.numero = numero;
        this.tipo = tipo;
        this.operadora = operadora;
    }

    public T getNumero() {
        return numero;
    }

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


    public T getTipo() {
        return tipo;
    }

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

    public T getOperadora() {
        return operadora;
    }

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

    @Override
    public String toString() {
        return "Telefono{" + ", numero=" + numero + ", tipo=" + tipo + ", operadora=" + operadora + '}';
    }
}
```

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

En el paquete “ec.edu.ups.idao” creamos las interfaces:

Interface “IPersonaDAO”:

```
public interface IPersonaDAO {
    public boolean create(Persona persona);
    public Persona read(String cedula);
    public void update(Persona persona);
    public Persona iniciarSesion(String correo, String contrasenia);
    public void delete(Persona persona);
    public List<Persona> findAll();
}
```

Interface “ITelefonoDAO”:

```
public interface ITelefonoDAO {

    public void create(Telefono telefono);
    public Telefono read(String numero);
    public void update(Telefono telefono);
    public void delete(Telefono telefono);
    public List<Telefono> findAll();

}
```

En el paquete “ec.edu.ups.dao” creamos las clases:

Clase “PersonaDAO”:

```
public class PersonaDAO implements IPersonaDAO {

    private List<Persona> listaPersona;
    Persona persona;
    PersonaDAO personaDAO;
    public PersonaDAO() {

        listaPersona = new ArrayList<>();
        persona=new Persona();
    }

    @Override
    public boolean create(Persona persona) {

        return this.listaPersona.add(new Persona(persona.getCedula(), persona.getNombre(), persona.getApellido()));
    }

    @Override
    public Persona read(String cedula) {
        for (Persona persona : listaPersona) {
            if (persona.getCedula().equals(cedula)) {
                return persona;
            }
        }
        return null;
    }
}
```

```
@Override
public void update(Persona persona) {
    for (int i = 0; i < listaPersona.size(); i++) {
        Persona p = listaPersona.get(i);
        if (p.getCedula().equals(persona.getCedula())) {
            listaPersona.set(i, persona);
            break;
        }
    }
}

@Override
public void delete(Persona persona) {
    Iterator<Persona> it = listaPersona.iterator();
    while (it.hasNext()) {
        Persona p = it.next();
        if (p.getCedula().equals(persona.getCedula()))
            it.remove();
        break;
    }
}

@Override
public List<Persona> findAll() {
    return listaPersona;
}

@Override
public Persona iniciarSesion(String correo, String contrasenia) {
    for (Persona persona : listaPersona) {
        if (persona.getCorreoElectronico().equals(correo) && persona.getContrasenia().equals(contrasenia)) {
            return persona;
        }
    }
    return null;
}
```

Clase "TelefonoDAO":

```
public class TelefonoDAO implements ITelefonoDAO {  
  
    private List<Telefono> listaTelefono;  
  
    public TelefonoDAO() {  
        listaTelefono=new ArrayList<>();  
    }  
  
    @Override  
    public void create(Telefono telefono) {  
        listaTelefono.add(telefono);  
    }  
  
    @Override  
    public Telefono read(String numero) {  
        for (Telefono telefono : listaTelefono) {  
            if(telefono.getNumero().equals(numero)) {  
                return telefono;  
            }  
        }  
        return null;  
    }  
}
```

```
@Override
public void update(Telefono telefono) {
    for (int i = 0; i < listaTelefono.size(); i++) {
        Telefono t = listaTelefono.get(i);
        if (t.getNumero().equals(telefono.getNumero())) {
            listaTelefono.set(i, telefono);
            break;
        }
    }
}

@Override
public void delete(Telefono telefono) {
    Iterator<Telefono> it = listaTelefono.iterator();
    while (it.hasNext()) {
        Telefono t = it.next();
        if (t.getNumero().equals(telefono.getNumero())) {
            it.remove();
            break;
        }
    }
}

@Override
public List<Telefono> findAll() {
    return listaTelefono;
}
```

En el paquete "ec.edu.ups.controlador" creamos las clases:  
Clase "ControladorPersona":



```

public class ControladorPersona {
    //objetos vista
    private VistaPersona vistaPersona;
    private VistaTelefono vistaTelefono;
    //objetos modelo
    private Persona persona;
    private Telefono telefono;
    //objetos DAO
    private IPersonaDAO personaDAO;
    private ITelefonoDAO telefonoDAO;
    private PersonaDAO pDAO;

    private VentanaRegistrarPersona registrarPersonal;
    private VentanaListado listadoTelefono;

    public ControladorPersona() {
    }

    public ControladorPersona(VentanaRegistrarPersona vistaPersona,VentanaListado listadoTelefono,IPersonaDAO personaDAO,
        this.registrarPersonal = vistaPersona;
        this.listadoTelefono = listadoTelefono;
        this.personaDAO = personaDAO;
        this.telefonoDAO = telefonoDAO;
    }

    public ControladorPersona(VistaPersona vistaPersona,VistaTelefono vistaTelefono,IPersonaDAO personaDAO,ITelefonoDAO t
        this.vistaPersona = vistaPersona;
        this.vistaTelefono = vistaTelefono;
        this.personaDAO = personaDAO;
        this.telefonoDAO = telefonoDAO;
    }

    //llama al DAO para guardar a una persona
    public void registrar() {
        persona = vistaPersona.ingresarPersona();
        personaDAO.create(persona);
    }

    public Persona iniciarSesion(String correo, String contrasenia){
        persona=personaDAO.iniciarSesion(correo, contrasenia);
        return persona;
    }

    //llama al DAO para obtener una persona por la cedula y luego los muestra en la vista
    public void verPersona() {
        String cedula = vistaPersona.buscarPersona();
        persona = personaDAO.read(cedula);
        vistaPersona.verPersona(persona);
    }

    //llama al DAO para actualizar una persona
    public void actualizar() {
        persona = vistaPersona.actualizarPersona();
        personaDAO.update(persona);
    }

    //llama al DAO para eliminar una persona
    public void eliminar() {
        persona = vistaPersona.eliminarPersona();
        personaDAO.delete(persona);
    }
}

```

```
//llama al DAO para obtener todas las personas y luego los muestra en la vista
public void verPersonas() {
    List<Persona> personas;
    personas = personaDAO.findAll();
    vistaPersona.verPersonas(personas);
}

public List<Telefono> listarT(String cedula) {
    persona.listar();
    return null;
}

// agregacion
public void agregarTelefono(){
    String numero = vistaTelefono.buscarTelefono();
    telefono = telefonoDAO.read(numero);
    persona.agregarTelefono(telefono);
    personaDAO.update(persona);
}
```

Clase "ControladorTelefono":

```
public class ControladorTelefono {

    public ControladorTelefono() {
    }

    //objetos vista
    private VistaTelefono vistaTelefono;
    private Telefono telefono;
    private ITelefonoDAO telefonoDAO;
    private VentanaListado listadoTelefono;

    Persona p=new Persona();

    // constructor
    public ControladorTelefono(VistaTelefono vistaTelefono, TelefonoDAO telefonoDAO) {
        this.vistaTelefono = vistaTelefono;
        this.telefonoDAO = telefonoDAO;
    }

    public ControladorTelefono( VentanaListado listadoTelefono, ITelefonoDAO telefonoDAO) {
        this.telefono = telefono;
        this.telefonoDAO = telefonoDAO;
        this.listadoTelefono = listadoTelefono;
    }

    //llama al DAO para guardar un telefono
    public void registrar() {
        telefono = vistaTelefono.ingresarTelefono();
        telefonoDAO.create(telefono);
    }
}
```

```
//llama al DAO para actualizar un telefono
public void actualizar() {
    telefono = vistaTelefono.actualizarTelefono();
    telefonoDAO.update(telefono);
}


//llama al DAO para eliminar un telefono
public void eliminar() {
    telefono= vistaTelefono.eliminarTelefono();
    telefonoDAO.delete(telefono);
}

//llama al DAO para obtener todas las personas y luego los muestra en la vista
public void verTelefonos() {
    List<Telefono> telefonos;
    telefonos = telefonoDAO.findAll();
    vistaTelefono.verTelefonos(telefonos);
}

//llama al DAO para obtener un telefono por el numero y luego los muestra en la vista
public void verTelefono() {
    String numero = vistaTelefono.buscarTelefono();
    telefono = telefonoDAO.read(numero);
    vistaTelefono.verTelefono(telefono);
}
```


En el

paquete "ec.edu.ups.vista" creamos la interfaz gráfica:  
Interfaz Gráfica "VentanaPrincipal":

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

Persona
Telefono
Buscar

Interfaz Gráfica “VentanaRegistrarPersona”:

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



**REGISTRO**

Cédula:

Nombre:

Apellido:


Correo Electrónico:


Contraseña:

**REGISTRAR USUARIO**

Cedula	Nombre	Apellido

Interfaz Gráfica “VentanaActualizarPersona”:

 <b>UNIVERSIDAD POLITÉCNICA SALESIANA</b> ECUADOR	<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		



**ACTUALIZACIÓN**

**Ingrese la cédula del usuario a actualizar**

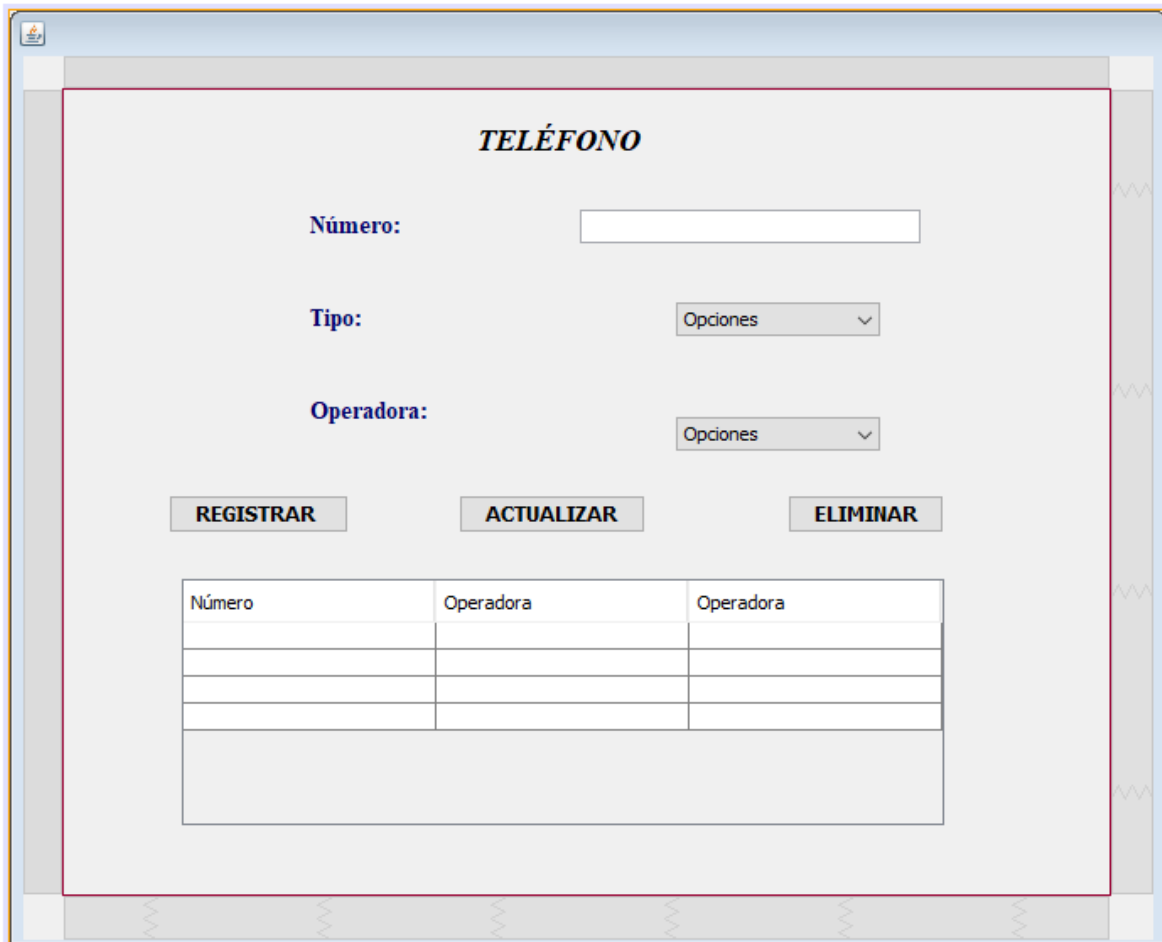
**Nuevo Nombre:**

**Nuevo Apellido:**

**Correo Electrónico:**

**Contraseña:**

Interfaz Gráfica "VentanaListado":



**TELÉFONO**

**Número:**

**Tipo:**

**Operadora:**

**REGISTRAR** **ACTUALIZAR** **ELIMINAR**

Número	Operadora	Operadora

Interfaz Gráfica “ListarTelefonoPorUsuarioVentana”:



### *TELÉFONOS DEL USUARIO*

Cédula:

Correo Electrónico:

Nombre:

Apellido:

**LISTAR**


Número	Tipo	Operadora

#### **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:**

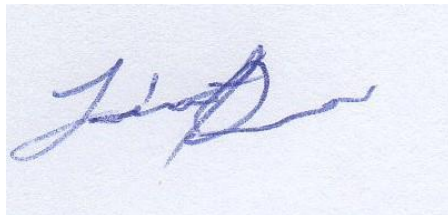
Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.

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

**RECOMENDACIONES:**

No tengo recomendaciones ya que el docente fue claro al dar la tarea.

**Nombre de estudiante:** \_\_\_\_\_ Lisseth Reinoso \_\_\_\_\_



**Firma de estudiante:**