
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: 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


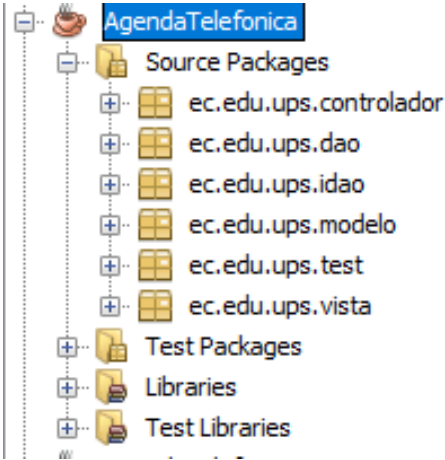
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Clase Genéricas en Java	
OBJETIVO: <ul style="list-style-type: none"> Identificar los cambios importantes de Java Diseñar e Implementar las nuevas técnicas de programación Entender la cada uno de las características nuevas en Java 			
INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):		1. Revisar los conceptos fundamentales de Java	
		2. Establecer las características de Java en programación genérica	
		3. Implementar y diseñar los nuevos componentes de programación genérica	
		4. Realizar el informe respectivo según los datos solicitados.	
ACTIVIDADES POR DESARROLLAR (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			
2. Diseñar e implementar las características de Java para generar una abstracción que permita realizar un CRUD,			
3. Probar su funcionamiento y rendimiento dentro de los equipos de cómputo de programación genérica y ordenar una lista, buscar.			
4. Realizar práctica codificando los codigos de las nuevas características de Java y su uso dentro de una agenda telefónica			
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.			

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		

		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA: Ingeniería en Computación		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	2	TÍTULO PRÁCTICA: Programación genérica	
<ul style="list-style-type: none"> • OBJETIVO ALCANZADO: • Identificar los cambios importantes de Java • Diseñar e Implementar las nuevas técnicas de programación • Entender la cada uno de las características nuevas en Java 			
ACTIVIDADES DESARROLLADAS			
1. Revisar los conceptos fundamentales de Java.			
2. Establecer las características de Java en programación genérica			
3. Implementar y diseñar los nuevos componentes de programación genérica:			
<p>Creamos el proyecto llamado “Agenda Telefónica” con sus respectivos paquetes:</p> 			
<p>En el paquete “ec.edu.ups.modelo” creamos las clases:</p> <p>Clase “Persona”:</p>			

```
public class Persona<T> {

    private T cedula;
    private T nombre;
    private T apellido;
    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 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;
    }

    //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 +

+ "\nTeléfonos:" + 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 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;
}
```

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, I
        this.registrarPersonal = vistaPersona;
        this.listadoTelefono = listadoTelefono;
        this.personaDAO = personaDAO;
        this.telefonoDAO = telefonoDAO;
    }

    public ControladorPersona(VistaPersona vistaPersona, VistaTelefono vistaTelefono, IPersonaDAO personaDAO, ITelefonoDAO te
        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);
    }
}
```

```
//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);
}

// 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 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);
}

//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);
}
```

En el paquete “ec.edu.ups.vista” creamos las clases e interfaz gráfica:

Clase “VistaPersona” para la ejecución por consola:


```
public class VistaPersona {  
  
    private Scanner leer;  
  
    public VistaPersona() {  
        leer=new Scanner(System.in);  
    }  
  
    public Persona ingresarPersona(){  
        leer = new Scanner(System.in);  
        System.out.println("Ingresa los datos");  
        System.out.print("Cédula: ");  
        String cedula = leer.nextLine();  
        System.out.print("Nombre: ");  
        String nombre = leer.next();  
        System.out.print("Apellido: ");  
        String apellido = leer.next();  
        return new Persona(cedula, nombre, apellido);  
    }  
}
```

```
public Persona actualizarPersona() {
    leer = new Scanner(System.in);
    System.out.println("Ingresa la cédula del cliente a actualizar");
    String cedula = leer.nextLine();
    System.out.println("Ingrese los nuevos Datos (nombre, apellido)");
    System.out.print("Nuevo nombre: ");
    String nombre = leer.next();
    System.out.print("Nuevo apellido: ");
    String apellido = leer.next();
    return new Persona(cedula, nombre, apellido);
}

public Persona eliminarPersona() {
    leer = new Scanner(System.in);
    System.out.println("Ingresa la cédula del cliente a eliminar");
    String cedula = leer.nextLine();
    return new Persona(cedula, null, null);
}

public String buscarPersona() {
    leer = new Scanner(System.in);
    System.out.println("Ingresa la cédula del cliente a buscar");
    String cedula = leer.next();
    return cedula;
}

public void verPersona(Persona persona) {
    System.out.println("Datos de la persona: " + persona);
}

public void verPersonas(List<Persona> personas) {
    for (Persona p : personas) {
        System.out.println("Datos de la persona: " + p);
    }
}
```

Clase "VistaTelefono" para la ejecución por consola:

```
public class VistaTelefono {  
  
    private Scanner leer;  
    Persona p=new Persona();  
    public VistaTelefono() {  
  
        leer=new Scanner(System.in);  
    }  
  
    public Telefono ingresarTelefono() {  
        leer = new Scanner(System.in);  
        System.out.println("Ingrese los datos del teléfono");  
  
        System.out.print("Número: ");  
        String numero = leer.nextLine();  
        System.out.print("Tipo: ");  
        String tipo = leer.next();  
        System.out.print("Operadora: ");  
        String operadora = leer.next();  
        return new Telefono(numero, tipo, operadora);  
    }  
}
```

```
public Telefono actualizarTelefono() {
    leer = new Scanner(System.in);
    System.out.println("Ingresa el numero del telefono a actualizar");
    String numero = leer.next();
    System.out.println("Ingrese los nuevos Datos (tipo,operadora)");
    System.out.print("Tipo: ");
    String tipo= leer.next();
    System.out.print("Operadora: ");
    String operadora = leer.next();

    return new Telefono(numero,tipo, operadora);
}


public Telefono eliminarTelefono() {
    leer = new Scanner(System.in);
    System.out.println("Ingresa el número del teléfono a eliminar");
    String numero = leer.next();
    return new Telefono(numero, null, null);
}

public String buscarTelefono() {
    leer = new Scanner(System.in);
    System.out.println("Ingresa el número del teléfono a buscar");
    String numero = leer.next();
    return numero;
}

public void verTelefono( Telefono telefono) {
    System.out.println("Telefono: " + telefono);
}

public void verTelefonos(List<Telefono> telefonos) {
    for (Telefono t : telefonos) {
        System.out.println("Datos del teléfono: " + t);
    }
}
}
```

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
Listar

Interfaz Gráfica “VentanaRegistrarPersona”:



REGISTRO

Cédula:


Nombre:

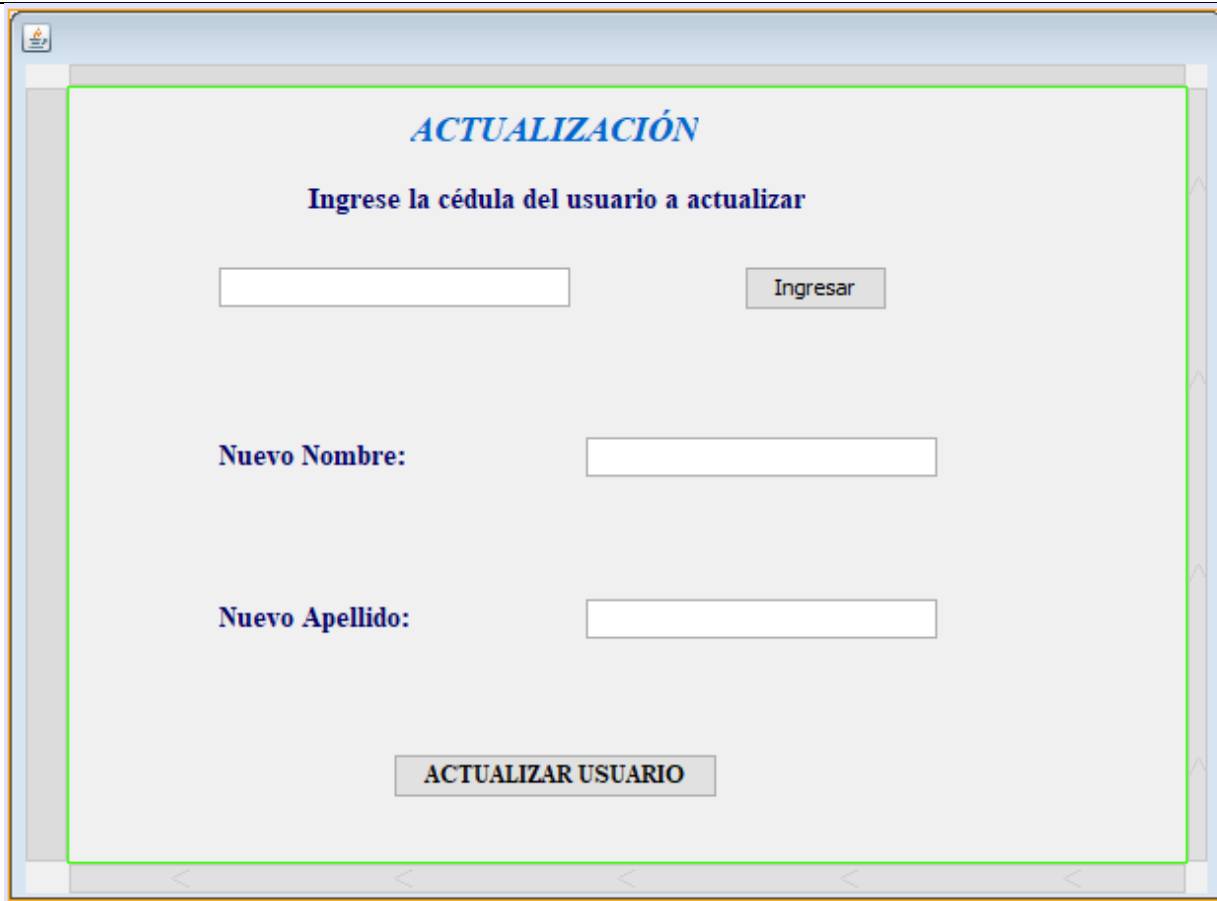
Apellido:

REGISTRAR USUARIO

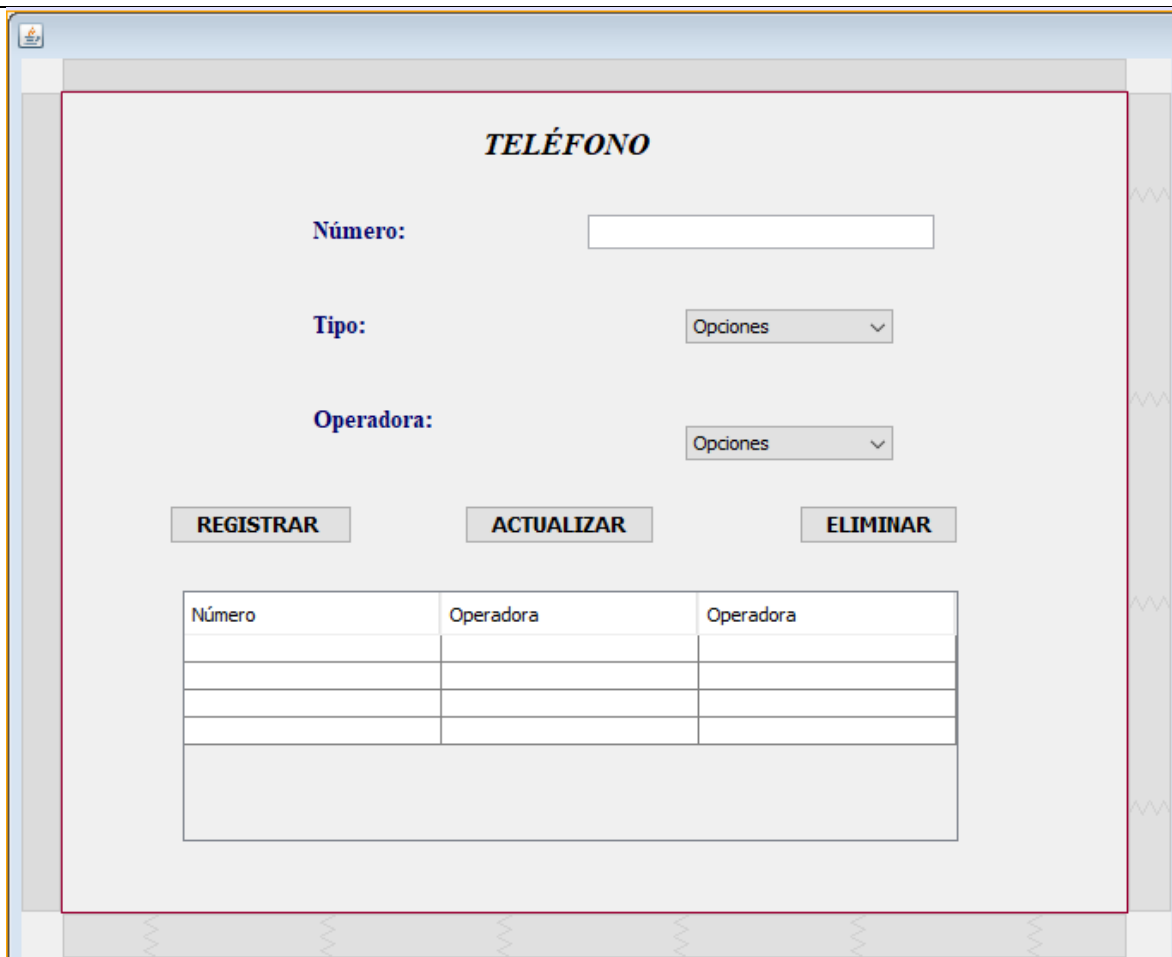
Cedula	Nombre	Apellido

Interfaz Gráfica "VentanActualizarPersona":

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



Interfaz Gráfica “VentanaListado”:



TELÉFONO

Número:


Tipo:

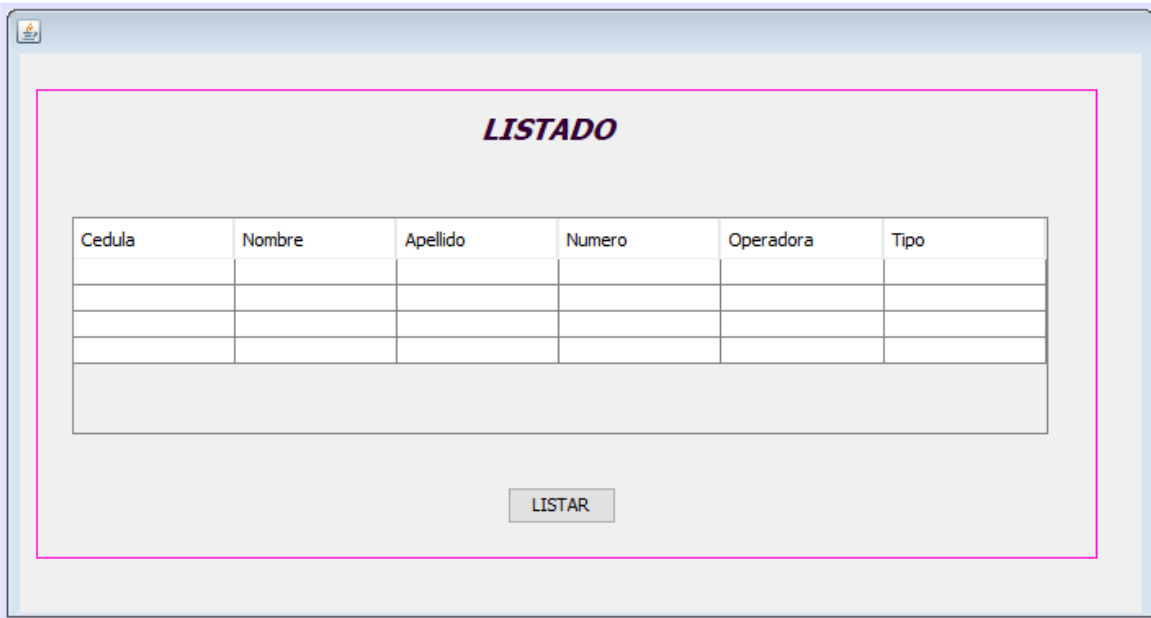
Operadora:

REGISTRAR **ACTUALIZAR** **ELIMINAR**

Número	Operadora	Operadora

Interfaz Gráfica "ListadoGeneral":

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



LISTADO

Cedula	Nombre	Apellido	Numero	Operadora	Tipo

LISTAR

En el paquete “ec.edu.ups.test” creamos la clase:
Clase “Test” para la ejecución por consola:

```
public static void main(String[] args) {  
  
    // vista  
    VistaPersona vistaP = new VistaPersona();  
    VistaTelefono vistaT = new VistaTelefono();  
    //DAOs  
    PersonaDAO personaDAO = new PersonaDAO();  
    TelefonoDAO telefonoDAO = new TelefonoDAO();  
    // controlador  
    ControladorPersona controladorPersona = new ControladorPersona(vistaP,vistaT,personaDAO, telefonoDAO);  
    ControladorTelefono controladorTelefono = new ControladorTelefono(vistaT, telefonoDAO);  
  
    controladorPersona.registrar();  
    controladorPersona.registrar();  
  
    controladorPersona.verPersonas();  
  
    controladorTelefono.registrar();  
    //controladorDireccion.registrar();  
    //controladorDireccion.registrar();  
  
    controladorTelefono.verTelefonos();  
  
    controladorPersona.verPersona();  
    controladorPersona.agregarTelefono();  
  
    controladorPersona.verPersonas();  
    controladorPersona.verPersona();  
    controladorPersona.agregarTelefono();  
}
```


RESULTADO(S) OBTENIDO(S):

Realización procesos de investigación anteriormente dados sobre los cambios importantes de Java como las nuevas versiones del mismo.

Entendimiento de la programación genérica como clases genéricas, streams, expresiones lambda siendo lo principal de este modo de programación.

CONCLUSIONES:

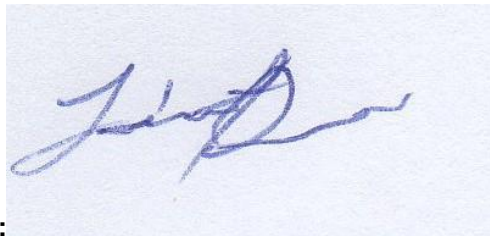
En conclusión, al trabajar con programación genérica hizo más fácil el llevar a cabo el proyecto ya que se pudo ahorrar varias líneas de código con este tipo de programación, además de considerar que mientras se realizaba el proyecto se pudieron observar algunas características de las versiones Java ya investigadas en clases anteriores.

	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: Liseth Reinoso



Firma de estudiante: