
	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


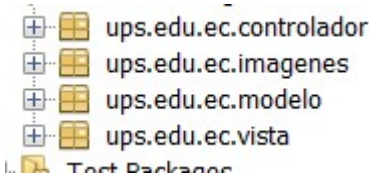
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Reflexión en Java	
OBJETIVO: Identificar los cambios importantes de Java Diseñar e Implementar las nuevas tecnicas de programación Entender cada una 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 reflexión	
		3. Implementar y diseñar los nuevos componentes de reflexión	
		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, 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.			
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: _____

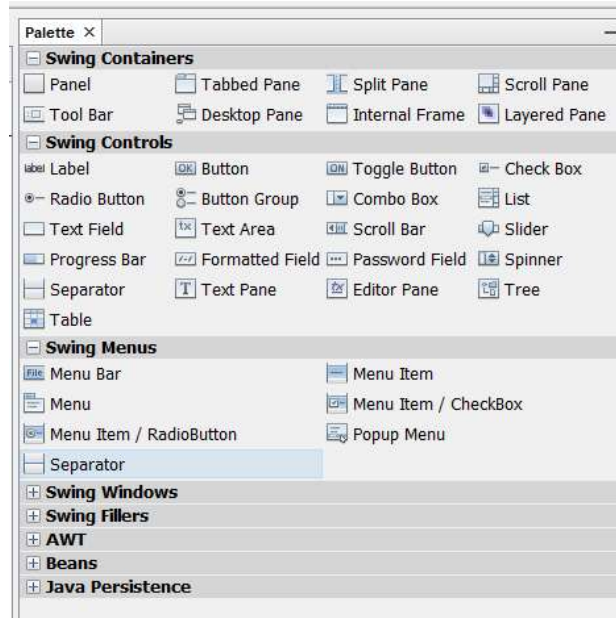
Firma: _____

	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: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Clase Genéricas en Java	
OBJETIVO ALCANZADO: <ul style="list-style-type: none"> • Aplicar e identificar los cambios de java. • Diseñar e Implementar las nuevas técnicas de programación aprendidas en las sesiones virtuales. • Entender la cada uno de las características nuevas en Java 8. 			
ACTIVIDADES DESARROLLADAS			
<p>1. Revisar la teoría y conceptos de Java 8, 9 ,10, 11, 12, 13, 14, 15</p> <p>- Arquitectura MVC:</p> <p>Uno de los conceptos aplicados en el desarrollo de esta práctica es la arquitectura MVC en la cual se separa en tres capas (Modelo, Vista, Controlador), cada una de estas está asignada a una función del programa.</p> <div style="text-align: center;">  </div> <p>- Métodos CRUD:</p> <p>Estos métodos directamente relacionados con el almacenamiento de datos, es decir con la base de datos, el nombre es la abreviación de Create, Read, Update, Delete,</p> <pre> public boolean create(T objeto) {...6 lines } public T read(Predicate<T> id) {...8 lines } public boolean update(T objeto) {...8 lines } public boolean delete(T objeto) {...3 lines } </pre> <p>- Clase Genéricas:</p> <p>Las clases Genéricas son el tipo de clases que sirve para el reutilizamiento de código, disminuyendo la cantidad de líneas de código y encapsula sus operaciones para que no sean utilizadas por un solo tipo de dato.</p> <pre> */ public abstract class AbstractControlador<T> { </pre>			

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

- **Javax Swing:** Paquete de datos donde se incluye los JFrame para las ventanas, paneles, botones, tablas, etc. Para desarrollar una interfaz gráfica y funcione como intermediario entre el usuario y la aplicación.



- **Patrones de creación Singleton:**

Se podría crear el objeto al inicio y colocarlo en un atributo estático. Pero no se podría proporcionar ninguna información de inicialización justo cuando vaya a usarse y no se puede controlar quien accede al objeto

```
public class ControladorLogin {

    private static ControladorLogin instance = new ControladorLogin();

    private List lista = new ArrayList();

    private ControladorLogin() {

    }

    public static ControladorLogin getInstance() {
        return instance;
    }


}
```

- **Reflexión de lista:**

Es un bucle que permite imprimir todos los métodos de la clase sin importar cuál sea el tipo de lista.

```
public void imprimirListaReflexion(List lista){
    for (Object object : lista) {
        Method[] metodos = object.getClass().getMethods();
        for (Method m: metodos) {

            if(m.getName().substring(0,3).equals("get")){
                try {
                    Object variable = (Object) m.invoke(object, null);
                    System.out.print(variable+" ");
                } catch (IllegalAccessException ex) {
                    Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE, null, ex);
                } catch (IllegalArgumentException ex) {
                    Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE, null, ex);
                } catch (InvocationTargetException ex) {
                    Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }
    }
    System.out.println("");
}
```

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

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.

Para generar una lista que imprima todos los componentes que tenga la clase que este guardada en ese Array, podemos utilizar la lista reflexible, esta lista recorre en un bucle for each, obteniendo objeto por objeto para extraer todos sus métodos y guardarlos en un vector de tipo method.. Después de guardar se recorre el vector comparando si comienza con get para imprimir solo los datos que están ingresados y excluir errores de argumentos e impresión.

```
public void imprimirListaReflexion(List lista){
    for (Object object : lista) {
        Method[] metodos = object.getClass().getMethods();
        for(Method m: metodos){

            if(m.getName().substring(0,3).equals("get")){
                try {
                    Object variable = (Object) m.invoke(object, null);
                    System.out.print(variable+" ");
                } catch (IllegalAccessException ex) {
                    Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE, null, ex);
                } catch (IllegalArgumentException ex) {
                    Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE, null, ex);
                } catch (InvocationTargetException ex) {
                    Logger.getLogger(AbstractControlador.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }
        System.out.println("");
    }
}
```


Al crear datos y guardar en la lista se obtuvo esto como resultado.

```
099999999 4 Persona{id=1, cedula=324, nombre=Hola, apellido=Mundo, direccion=Av.) Movistar Celular class ec.edu.ups.modelo.Telefono
2343242 3 Persona{id=1, cedula=324, nombre=Hola, apellido=Mundo, direccion=Av.) Etapa Fijo class ec.edu.ups.modelo.Telefono
3242342323 1 Persona{id=1, cedula=324, nombre=Hola, apellido=Mundo, direccion=Av.) Etapa Fijo class ec.edu.ups.modelo.Telefono
3453534 2 Persona{id=1, cedula=324, nombre=Hola, apellido=Mundo, direccion=Av.) Etapa Fijo class ec.edu.ups.modelo.Telefono
```

En este caso el orden de impresión fue numero de celular, código, Persona al que pertenece, Operadora y tipo. La siguiente línea de código es para generar automáticamente, en el cual se obtuvo la lista mediante el método .get() y se recorrió la lista para obtener el mayor código y retornar el valor sumado uno, para no repetir códigos y se dañe el ID al eliminar un valor.

```
@Override
public int generarId(){
    List<Persona> lista = getList();
    int codigo = 0;
    if (lista.size()>0){
        for (Persona persona : lista) {
            int aux = persona.getId();
            if(aux>codigo){
                codigo=aux;
            }
        }
        return codigo+1;
    }else{
        return 1;
    }
}
```

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

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

Para controlar el nullpointerexeption se puede utilizar lo que son las excepciones de Java. En el caso del buscar cunado no haya nada en la lista y se busque algún objeto va a saltar el error de la lista vacía, interrumpiendo la ejecución del programa.

```
public T read(Predicate<T> id){
    try {
        return lista.stream().filter(id).findFirst().get();
    } catch (NullPointerException e) {
        System.out.println("Lista vacia "+e);
        return null;
    }
}
```

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

Siguiendo las instrucciones del docente se procedió a realizar el programa de la agenda telefónica, aplicando los conceptos aprendidos en estas semanas de clases virtuales, primeramente se aplicó el concepto de MVC y en ellos el CRUD como base de datos, Para el CRUD se utilizó la programación genérica y también los métodos abstractos:

```
public abstract class AbstractControlador<T> {


    private List<T> lista;

    public AbstractControlador() {
        lista = new ArrayList<>();
    }

    public boolean create(T objeto){
        if(validar(objeto)==true){
            return lista.add(objeto);
        }
        return false;
    }

    public T read(Predicate<T> id){
        try {
            return lista.stream().filter(id).findFirst().get();
        } catch (NullPointerException e) {
            System.out.println("Lista vacia "+e);
            return null;
        }
    }

    public boolean update(T objeto){
        int posicion = buscarPosicion(objeto);
        if(posicion >=0){
            lista.set(posicion, objeto);
            return true;
        }
        return false;
    }
}
```


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

```

public boolean delite(T objeto){
    return (lista.contains(objeto))? lista.remove(objeto): false;
}

public abstract boolean validar(T objeto);

public abstract void ordenarLista();

public abstract int generarId();

public int buscarPosicion(T buscar){
    for (int i = 0; i < lista.size(); i++) {
        Object objeto = lista.get(i);
        if(objeto.equals(buscar))
            return i;
    }
    return -1;
}

public List<T> getLista() {
    return lista;
}


public void setLista(List<T> lista) {
    this.lista = lista;
}

public void imprimirListaReflexion(List lista){
    for (Object object : lista) {
        Method[] metodos = object.getClass().getMethods();
        for(Method m: metodos){

            if(m.getName().substring(0,3).equals("get")){
                try {
                    Object variable = (Object) m.invoke(object, null);
                    System.out.print(variable+" ");
                } catch (IllegalAccessException ex) {

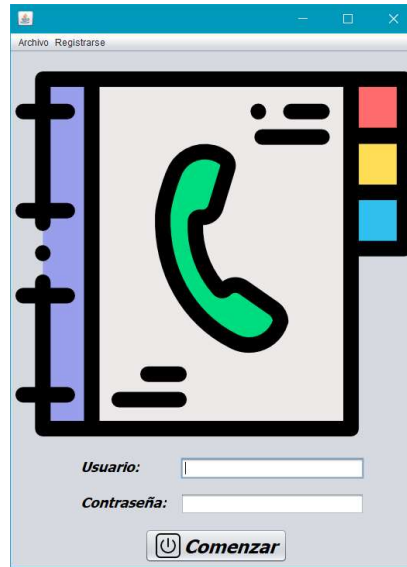
                }
            }
        }
    }
    System.out.println("");
}
}
}

```

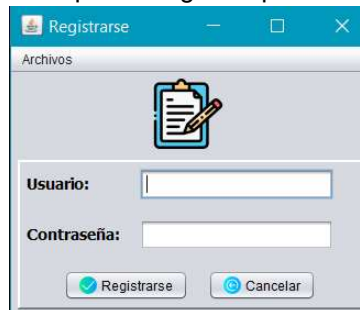

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


Esta clase nos va servir para no repetir las líneas de código en los demás controladores, se utilizaran en los demás los métodos que faltan y no pueden ser genéricos.

Al ejecutar el programa aparece la pantalla principal, en la cual cuenta con un login para entrar a la agenda telefónica.

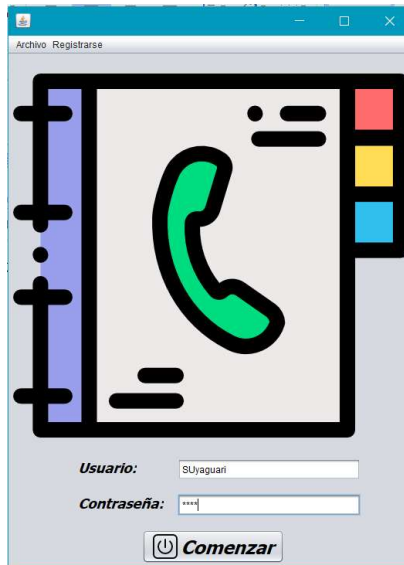


En la parte superior hay un menú en el cual se puede registrar para acceder a la agenda telefónica:



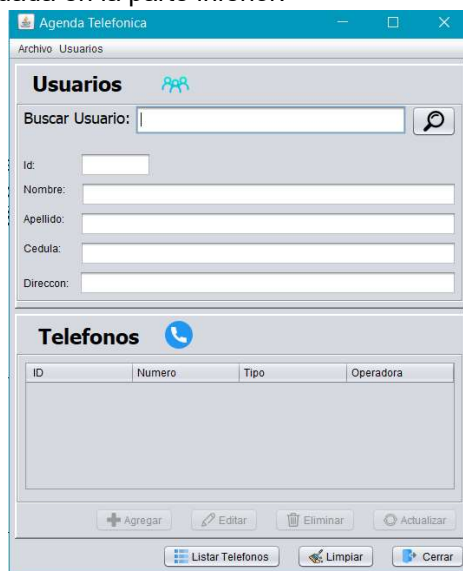
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


Se llena con los datos que desee y se regresa a la principal para llenar los datos:



Una vez llenado los datos se da clic en el botón comenzar y si ha llenado los datos correctamente podrá acceder a la agenda o sino caso contrario le aparece un mensaje de error.

Al ingresar se le aparecerá una pantalla en la que puede buscar al usuario para agregarle los teléfonos correspondientes, al buscar al usuario se le cargara los datos en los Jtext y si tiene agregado números telefónicos se le cargara los datos en la tabla situada en la parte inferior.

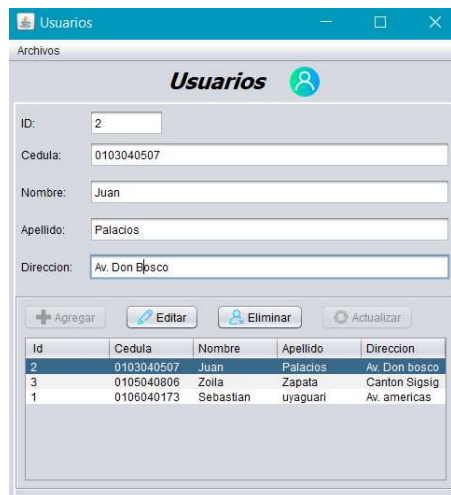


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Si no tiene agregado un usuario puede agregar en el menú de la parte superior, al hacer click en ese menú se le aparecerá una nueva ventana donde puede gestionar a los usuarios.




Si agregamos los usuarios o personas que desea se cargara los datos en la parte inferior, en este caso se agregó tres personas, cabe recalcar que al hacer click en la tabla sobre un usuario este se cargara en los jtext y podrá editarse o ser eliminado:



Id	Cedula	Nombre	Apellido	Direccion
2	0103040507	Juan	Palacios	Av. Don bosco
3	0105040806	Zola	Zapata	Canton Sigsig
1	0106040173	Sebastian	uyaguari	Av. americas

Los ID salen en desorden ya que al ser ordenados según el apellido estos se desordenan, para ordenar se utilizó el método de ordenación burbuja:

```
@Override
public void ordenarLista() {
    List<Persona> lista = getLista();
    for (int i = 0; i < lista.size()-1; i++) {
        for (int j = i+1; j < lista.size(); j++) {
            if(lista.get(i).getApellido().compareTo(lista.get(j).getApellido())>0){
                Persona aux = lista.get(i);
                lista.set(i, lista.get(j));
                lista.set(j, aux);
            }
        }
    }
}
```

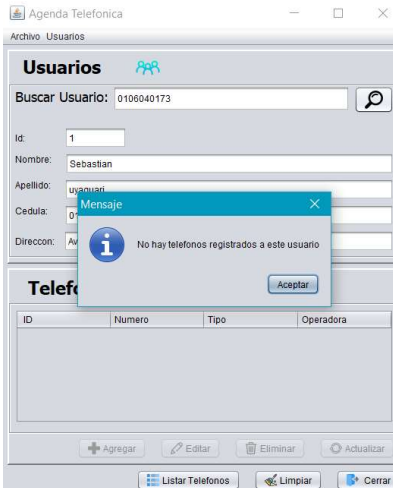
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

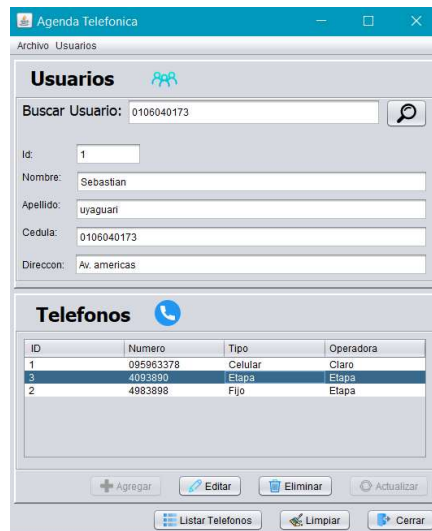
        setLista(lista);
    }

```

Al volver a la agenda principal, buscamos uno de nuestros usuarios registrados:



Al no tener ningún número registrado le aparecerá un mensaje diciendo que no tiene registrado ningún número, Ahora podemos registrar algunos teléfonos a ese usuario.




Una vez agregados se cargara los datos en la tabla inferior, para mostrar los teléfonos de cada usuario se implementó un método en el controlador Teléfono para buscar solo los números afiliados al usuarios:

```

public List buscarTelefonoPersona(int id) {
    List<Telefono> lista = new ArrayList<>();
    List<Telefono> listaTelefonos = getListas();
    for (Telefono listaTelefono : listaTelefonos) {
        Persona persona = listaTelefono.getPersona();
        if (persona.getId() == id) {
            lista.add(listaTelefono);
        }
    }
    return lista;
}

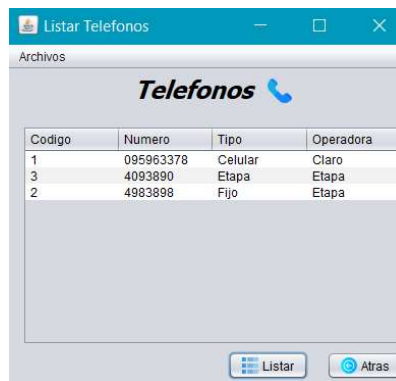
```

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

Igualmente se ordenó a los datos, pero en base a los números telefónicos:

```
@Override
public void ordenarLista() {
    List<Telefono> lista = getLista();
    for (int i = 0; i < lista.size()-1; i++) {
        for (int j = i+1; j < lista.size(); j++) {
            if(lista.get(i).getNumero().compareTo(lista.get(j).getNumero())>0){
                Telefono aux = lista.get(i);
                lista.set(i, lista.get(j));
                lista.set(j, aux);
            }
        }
    }
    setLista(lista);
}
```

Y por último la pantalla en la que se puede visualizar todos los teléfonos agregados:



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:

Este trabajo ha empleado el conocimiento básico adquirido el ciclo pasado de java y los nuevos métodos de programación, pero lo más valioso de este trabajo es que aprendes nuevas funciones que pueden ser aplicadas en el desarrollo de un software.

RECOMENDACIONES:

Mayor uso de las herramientas tecnológicas para buscar información sobre el lenguaje de programación que se desee para ampliar los conocimientos que se tiene o se desea.

Nombre de estudiante: Sebastián Uyaguari

 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		



Firma de estudiante: