

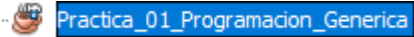














	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


		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: _____

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

Firma: _____

	FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES
CARRERA: Computación	ASIGNATURA: Programación Aplicada
NRO. PRÁCTICA: 2	TÍTULO PRÁCTICA:
OBJETIVO ALCANZADO:	
ACTIVIDADES DESARROLLADAS	
<ol style="list-style-type: none"> 1. Creamos un Proyecto en el NetBeans con el nombre de Practica_01  2. Creamos los paquetes para crear las clases y empezar con el proyecto nuevo creado, creamos los paquetes controlador, modelo e interfaz <div style="margin-left: 20px;">  ec.edu.ups.controlador  ec.edu.ups.interfaz  ec.edu.ups.modelo </div> 3. Dentro de cada paquete creamos las clases respectivas <ol style="list-style-type: none"> a. En el paquete controlador creamos las clases controladorAbstract que será la clase padre de las clases controladorUsuario y controladorTelefono para el teléfono y el usuario y también reutilizar código aplicando programación genérica. <div style="margin-left: 20px;">  ControladorAbstract.java  ControladorTelefono.java  ControladorUsuario.java </div> a. En el paquete modelo creamos las clases usuario y teléfono <div style="margin-left: 20px;">  Telefono.java  Usuario.java </div> b. En el paquete interfaz se encontrará la parte de la interfaz en esta se encontrarán las respectivas ventanas del programa <div style="margin-left: 20px;">  AgregarTelefono.java  EditarUsuario.java  TelefonosporUsuario.java  VentanaIniciarSesion.java  VentanaPrincipal.java  VentanaRegistrarUsuario.java </div> 	
<ol style="list-style-type: none"> 2. Creamos los atributos con sus getters y setters de las clases Usuario y Teléfono, también sus controladores y métodos de comparación como los métodos equals y hashCode. <pre style="margin-left: 20px; color: #4F81BD;">public class Telefono { private int codigo;</pre> 	

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

```

private String Numero;
private String Tipo;
private String Operadora;
//     private Usuario User;

public Telefono(int codigo, String Numero, String Tipo, String Operadora) {
    this.codigo = codigo;
    this.Numero = Numero;
    this.Tipo = Tipo;
    this.Operadora = Operadora;
}

public Telefono() {
}

public int getCodigo() {
    return codigo;
}

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

public String getNumero() {
    return Numero;
}

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

public String getTipo() {
    return Tipo;
}


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

public String getOperadora() {
    return Operadora;
}

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

@Override
public int hashCode() {
    int hash = 3;
    hash = 43 * hash + this.codigo;
    return hash;
}

```

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

```

@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 (this.codigo != other.codigo) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Telefono{" + "codigo=" + codigo + ", Numero=" + Numero + ", Tipo=" +
Tipo + ", Operadora=" + Operadora + '}';
}

}

```

En la clase usuario también creamos los métodos para agregar los teléfonos al usuario respectivo


```

public class Usuario {
    private String cedula;
    private String nombre;
    private String apellido;
    private String correo;
    private String contraseña;
    //Agregacion
    private List<Telefono> listaTelefonos;

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

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

```

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

```

public boolean agregarTelefono(Telefono telefono) {
    return listaTelefonos.add(telefono);
}

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

public int posicion(Telefono telefono) {
    return this.listaTelefonos.stream().filter(bus -> bus.getCodigo() ==
telefono.getCodigo()).findFirst().hashCode();
}

public void actualizarTelefono(Telefono telefono) {
    if (listaTelefonos.contains(telefono)) {
        for (int i = 0; i < listaTelefonos.size(); i++) {
            var us = listaTelefonos.get(i);
            if (us.getCodigo() == telefono.getCodigo()) {
                listaTelefonos.set(i, telefono);
                break;
            }
        }
    } else {
        System.out.println("No existe");
    }
}


public boolean eliminarTelefono(Telefono telefono) {
    if (listaTelefonos.contains(telefono)) {
        for (int i = 0; i < listaTelefonos.size(); i++) {
            var telf = listaTelefonos.get(i);
            if (telefono.getCodigo() == telf.getCodigo()) {
                listaTelefonos.remove(i);
                return true;
            }
        }
    }
    return false;
}

public String getCedula() {
    return cedula;
}

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

public String getNombre() {
    return nombre;
}

```

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

```

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 getContraseña() {
    return contraseña;
}

public void setContraseña(String contraseña) {
    this.contraseña = contraseña;
}


public List<Telefono> getListaTelefonos() {
    return listaTelefonos;
}

public void setListaTelefonos(List<Telefono> listaTelefonos) {
    this.listaTelefonos = listaTelefonos;
}

@Override
public int hashCode() {
    int hash = 3;
    hash = 71 * hash + Objects.hashCode(this.cedula);
    hash = 71 * hash + Objects.hashCode(this.listaTelefonos);
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }

```

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

```

    }
    final Usuario other = (Usuario) obj;
    if (!Objects.equals(this.cedula, other.cedula)) {
        return false;
    }
    if (!Objects.equals(this.listaTelefonos, other.listaTelefonos)) {
        return false;
    }
    return true;
}

public String toString() {
    return "Usuario{" + "cedula=" + cedula + ", nombre=" + nombre + ",
apellido=" + apellido + ", correo=" + correo + ", contrase\u00f1a=" + contraseña +
", listaTelefonos=" + listaTelefonos + '}';
}

```

3. En el paquete controlador, en la clase constructorAbstract creamos todos los métodos que son necesarios para crear, actualizar, leer, y eliminar un usuario o un teléfono, incluso creando métodos abstractos que heredaran a sus clases hijas.

```

public abstract class ControladorAbstract<T> {

    private List<T> listaObjetos;

    public ControladorAbstract() {
        listaObjetos = new ArrayList<>();
    }

    /**
     * Metodo Create para crear al objeto
     *
     * @param objeto
     * @return
     */
    public boolean create(T objeto) {
        if (!listaObjetos.contains(objeto)) {
            return listaObjetos.add(objeto);
        }
        return false;
    }

    /**
     * Metodo read para buscar un objeto
     *
     * @param buscar
     * @return
     */
    public abstract T read(T buscar);

    /**
     * Metodo read para buscar un objeto
     *
     * @param buscar
     * @return
     */
    public T read(T buscar) {
        if (listaObjetos.contains(buscar)) {
            return listaObjetos.stream().filter(objeto ->
objeto.equals(buscar)).findFirst().get();
        }
    }
}

```

```
//      return null;
//    }


/**
 * Metodo update para buscar si existe un objeto
 *
 * @param usuario
 */
public void update(T usuario) {
    if (listaObjetos.contains(usuario)) {
        for (int i = 0; i < listaObjetos.size(); i++) {
            var us = listaObjetos.get(i);
            if (us.equals(usuario)) {
                listaObjetos.set(i, usuario);
                break;
            }
        }
    } else {
        System.out.println("No existe");
    }
}

/**
 * Metodo delete para eliminar un objeto creado
 *
 * @param objeto
 * @return
 */
public boolean delete(T objeto) {
    if (listaObjetos.contains(objeto)) {
        for (int i = 0; i < listaObjetos.size(); i++) {
            var telf = listaObjetos.get(i);
            if (objeto.equals(telf)) {
                listaObjetos.remove(i);
                return true;
            }
        }
    }
    return false;
}

public List<T> findAll() {
    return listaObjetos;
}

public List<T> getListaObjetos() {
    return listaObjetos;
}

public void setListaObjetos(List<T> listaObjetos) {
    this.listaObjetos = listaObjetos;
}
}
```


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

En la clase constructorUsuario creamos métodos para el login del Usuario, comprobar que no se registren usuarios con los mismos correos y los métodos read para buscar datos del usuario mediante cedula y correo.

```
public class ControladorUsuario extends ControladorAbstract<Usuario> {

    public ControladorUsuario() {
        super();
    }

    Usuario sesionIniciada;

    public Usuario getSesionIniciada() {
        return sesionIniciada;
    }

    public boolean login(String email, String password) {
        for (int i = 0; i < getListaObjetos().size(); i++) {
            Usuario us = getListaObjetos().get(i);
            if (email.equals(us.getCorreo()) && password.equals(us.getContraseña()))
        {
                sesionIniciada = us;
                return true;
            }
        }
        return false;
    }

    public boolean comprobarCorreo(String email) {
        for (int i = 0; i < getListaObjetos().size(); i++) {
            var correo = getListaObjetos().get(i);
            if (email.equals(correo.getCorreo())) {
                return true;
            }
        }
        return false;
    }

    @Override
    public Usuario read(Usuario buscar) {
        for (int i = 0; i < getListaObjetos().size(); i++) {
            var us = getListaObjetos().get(i);
            if (buscar.equals(us)) {
                return us;
            }
        }
        return null;
    }

    public Usuario read(String correo) {
        for (int i = 0; i < getListaObjetos().size(); i++) {
            var us = getListaObjetos().get(i);
            if (correo.equals(us.getCorreo())) {
                return us;
            }
        }
    }
}
```

```

    }
    return null;
}

public Usuario readCorreo(String correo) {
    for (int i = 0; i < getListaObjetos().size(); i++) {
        var usuario = getListaObjetos().get(i);
        if (correo.equals(usuario.getCorreo())) {
            return usuario;
        }
    }
    return null;
}
}

```

En la clase constructorTelefono creamos los métodos para obtener el código de los teléfonos, y para comprobar que exista ese teléfono y ese usuario y read para buscar un usuario.

```

public class ControladorTelefono extends ControladorAbstract<Telefono> {

    private Usuario usuario;


    public ControladorTelefono() {
        super();
    }

    public int obtenerCodigo() {
        if (getListaObjetos().size() > 0) return getListaObjetos().size() + 1; return
1;
    }

    public Telefono comprobar(int id){
        for (int i = 0; i < getListaObjetos().size(); i++) {
            var telf = getListaObjetos().get(i);
            if (id == telf.getCodigo()) {
                return telf;
            }
        }
        return null;
    }

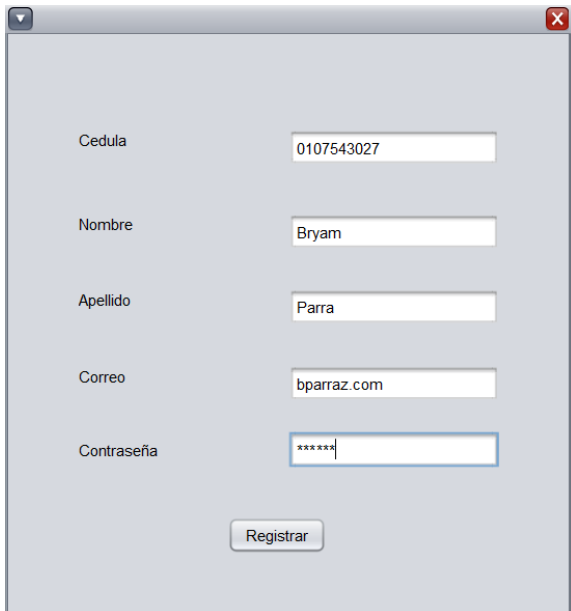
    @Override
    public Telefono read(Telefono buscar) {
        for (int i = 0; i < getListaObjetos().size(); i++) {
            var us = getListaObjetos().get(i);
            if (buscar.equals(us)) {
                return us;
            }
        }
        return null;
    }
}

```

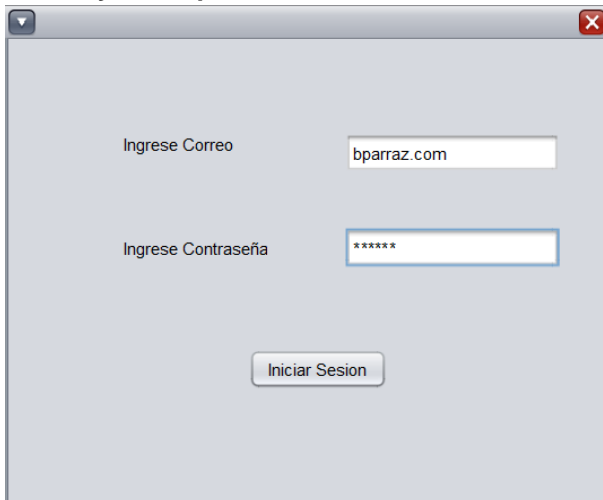
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

4. Funcionamiento aplicación


- a. Registramos a un usuario nuevo, al crear comprobamos si el correo ingresado no existe y así crear al nuevo usuario.

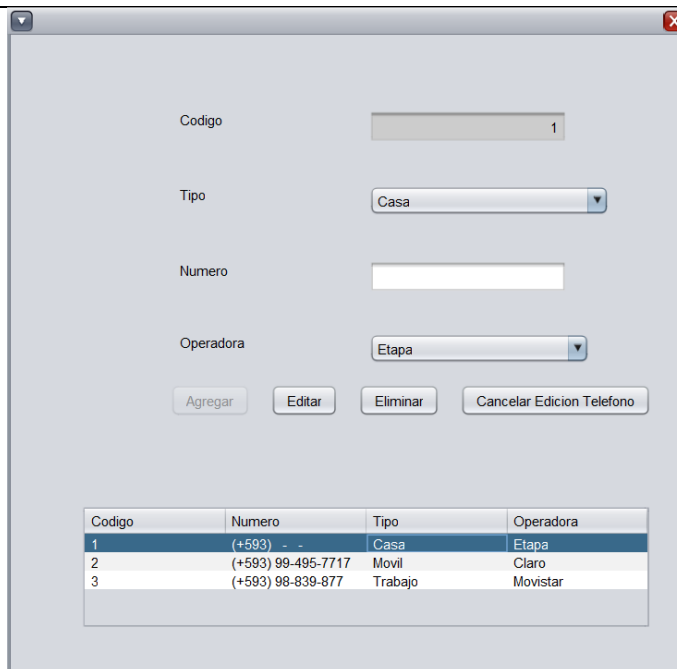


- b. Para iniciar sesión y editar los datos del usuario y agregar teléfonos para ese usuario, al mismo tiempo modificarlos o eliminarlos, si el correo ingresado es incorrecto no iniciará sesión y no se podrá modificar datos ni crear telefonos.



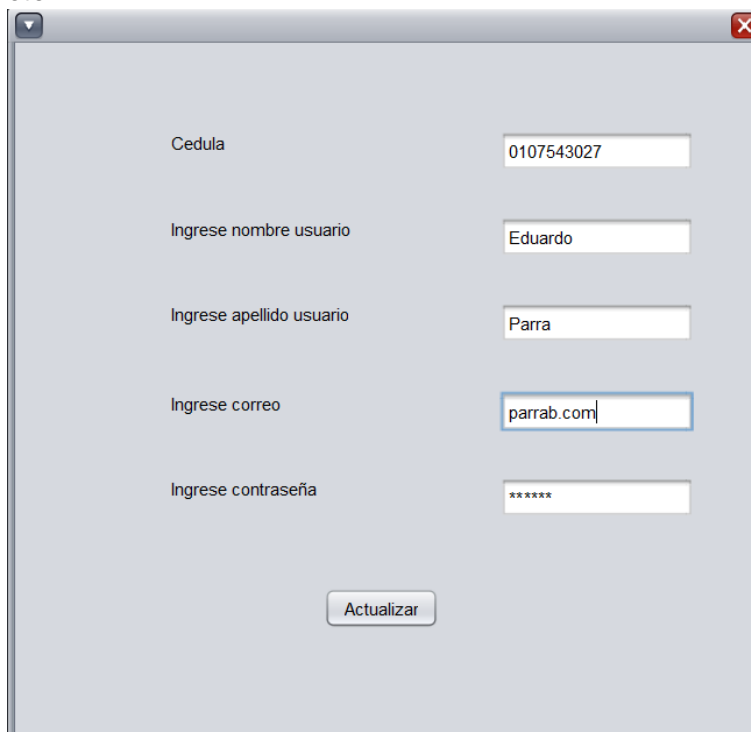
- c. En esta ventana se podrá crear los teléfonos de igual manera se podrá editar al teléfono con solo dar clic en la tabla se mostrar los datos de ese teléfono y se podrá modificar o eliminar.

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




Codigo	Numero	Tipo	Operadora
1	(+593) - -	Casa	Etapa
2	(+593) 99-495-7717	Movil	Claro
3	(+593) 98-839-877	Trabajo	Movistar

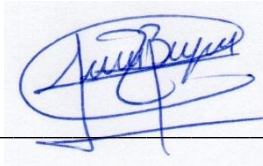
- d. También se podrá actualizar los datos del usuario, cambiar el nombre, correo, contraseña, etc.



- e. En el menú listar teléfonos se podrá mostrar los teléfonos que tiene un usuario y se podrá buscar a ese usuario mediante cedula o correo, también se podrá listar todos los telefonos que existen creados.

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

Nombre de estudiante: Bryam Parra



Firma de estudiante: _____