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:
Identificar los cambios importantes de Java
Diseñar e Implementar las nuevas tecnicas de programación
Entender la cada uno de las caracteristicas 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 caracteristicas 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*: _____**

FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES

CARRERA: Computaciónón		**ASIGNATURA**: Programación Aplicada
NRO. PRÁCTICA:	2	**TÍTULO PRÁCTICA**: Programación Genérica

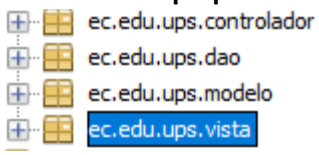
OBJETIVO ALCANZADO:

- Identificar los cambios importantes de Java
- Diseñar e Implementar las nuevas técnicas de programación
- Entender la cada uno de las características nuevas en Java

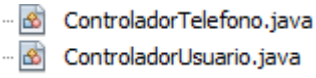
ACTIVIDADES DESARROLLADAS

1. **Creamos un Proyecto en el netbeans con el nombre de Practica_01**

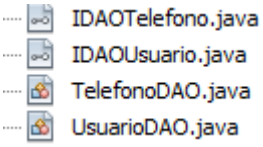
2. **Creamos sus paquetes respectivos, creamos el paquete controlador, dao, modelo y vista**



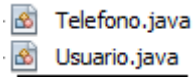
3. **Dentro de cada paquete creamos las clases respectivas**
   a. **En el paquete controlador creamos las clases controlador para el teléfono y el usuario**



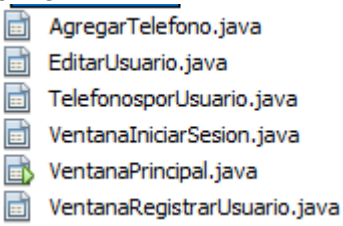
   a. **En el paquete dao creamos las clases interface de usuario y telefono y conotroladores del dao.**



   b. **En el paquete modelo creamos las clases usuario y teléfono**



   c. **En el paquete vista se encontrara la para la intervas, estarán las respectivas ventanas del programa**



3. **En las clases Usuario y Telefono creamos los atributos con sus respectivos getters y setter, y su hashCode y equals**

```
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() {
        listaTelefonos = new ArrayList<>();
    }

    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<>();
        //this.listaTelefonos = listaTelefonos;
    }

    public String getCedula() {
        return cedula;
    }

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

    public String getNombre() {
        return nombre;
    }

    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;
    }

    //metodos agregacion
    public void agregarTelefono(Telefono telefono){
        listaTelefonos.add(telefono);
    }

    public void actualizarTelefono(Telefono telefono){
        for (int i = 0; i < listaTelefonos.size(); i++) {
            Telefono telf = listaTelefonos.get(i);
            if(telf.getCodigo() == telefono.getCodigo()){
                listaTelefonos.set(i, telefono);
                break;
            }
        }
    }

    public void eliminarTelefono(Telefono telefono){
        for (int i = 0; i < listaTelefonos.size(); i++) {
            Telefono telf = listaTelefonos.get(i);
            if(telf.getCodigo() == telefono.getCodigo()){
                listaTelefonos.remove(listaTelefonos.get(i));
                break;
            }
        }
    }

    public Telefono buscarTelefono(Telefono telefono){
        if(listaTelefonos.contains(telefono)){
            int index = listaTelefonos.indexOf(telefono);
            return listaTelefonos.get(index);
        }else{
            return null;
        }
    }

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

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

    @Override
    public int hashCode() {
        int hash = 7;
        hash = 37 * hash + Objects.hashCode(this.cedula);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
```

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

    @Override
    public String toString() {
        return "Usuario: " + "\nCedula=" + cedula + " Nombre=" + nombre + ",
Aellido=" + apellido + ", Correo=" + correo + ", Contrase\u00f1a=" + contraseña +
'}';
          }
```

4. En el paquete Dao creamos la clase interfaz para los daos

```
public interface IDAOUsuario<U> {
    public void crearUsuario(U usuario);
    public Usuario leerUsuarios(String cedula);
    public void actualizarUsuario(Usuario usuario);
    public void eliminarUsuario(Usuario usuario);
    public List<Usuario> findAll();

      }
```

5. En ese mismo paquete creamos las clases ocntroladoras para los DAO'S

```
public class UsuarioDAO implements IDAOUsuario{
    private List<Usuario> listaUsuarios;
    private Usuario usuario;

    public UsuarioDAO() {
        listaUsuarios = new ArrayList<>();
    }

    @Override
    public void crearUsuario(Object usuario) {
        listaUsuarios.add((Usuario) usuario);
    }

    @Override
    public Usuario leerUsuarios(String cedula) {
        for (Usuario usuario : listaUsuarios) {
            if(usuario.getCedula().equals(cedula)){
                return usuario;
            }
        }
        return null;
    }

    @Override
    public void actualizarUsuario(Usuario usuario) {
        for (int i = 0; i < listaUsuarios.size(); i++) {
            Usuario us = listaUsuarios.get(i);
```

```
            if (us.getApellido() == usuario.getCedula()) {
                listaUsuarios.set(i, usuario);
                break;
            }
        }
    }

    @Override
    public void eliminarUsuario(Usuario usuario) {
        for (int i = 0; i < listaUsuarios.size(); i++) {
            Usuario us = listaUsuarios.get(i);
            if (us.getApellido() == usuario.getCedula()) {
                listaUsuarios.remove(listaUsuarios.get(i));
                break;
            }
        }
    }

    @Override
    public List<Usuario> findAll() {
        return listaUsuarios;
    }

    public Usuario comprobarUsuario(String correo, String contraseña){
        for (Usuario usuario : listaUsuarios) {
            if(usuario.getCorreo().equals(correo) &&
usuario.getContraseña().equals(contraseña)){
                return usuario;
            }
        }
        return null;
    }
      }
```

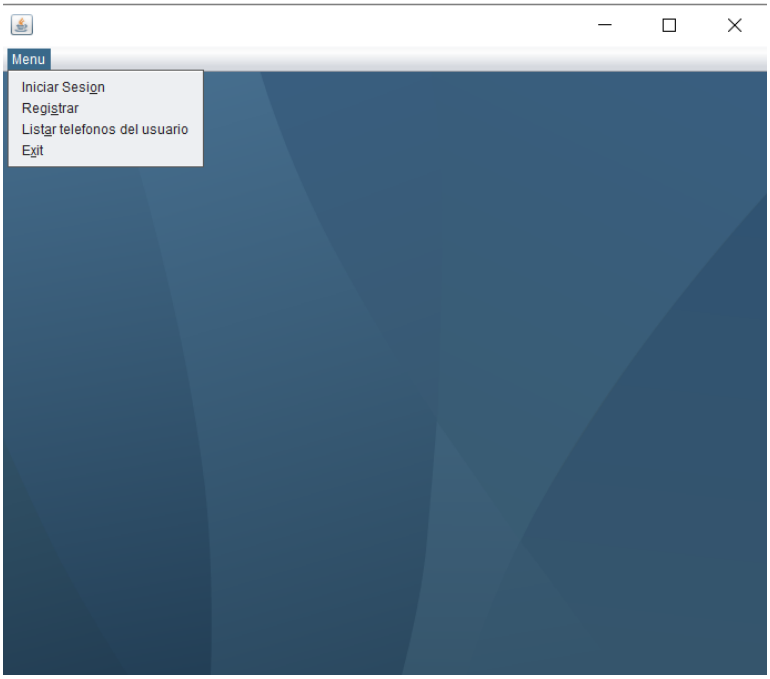
6. En el paquete controlador creamos las clases controlador que será enlazado con las ventanas para ingresar, leer, actualizar o eliminar datos

```
public class ControladorUsuario {
    //modelo
    private Usuario usuario;
    private Telefono telefono;
    //DAO
    private IDAOUsuario daoUs;
    private IDAOTelefono daotelf;

    private int contadorTelf;

    public ControladorUsuario(IDAOUsuario daoUs, IDAOTelefono daotelf) {
        this.daoUs = daoUs;
        this.daotelf = daotelf;

        contadorTelf = 0;
    }

    public void registrarUsuario(String cedula, String nombre, String apellido,
String correo, String contraseña){
        //this.usuario = usuario;
        usuario = new Usuario(cedula, nombre, apellido, correo, contraseña);
        daoUs.crearUsuario(usuario);
```

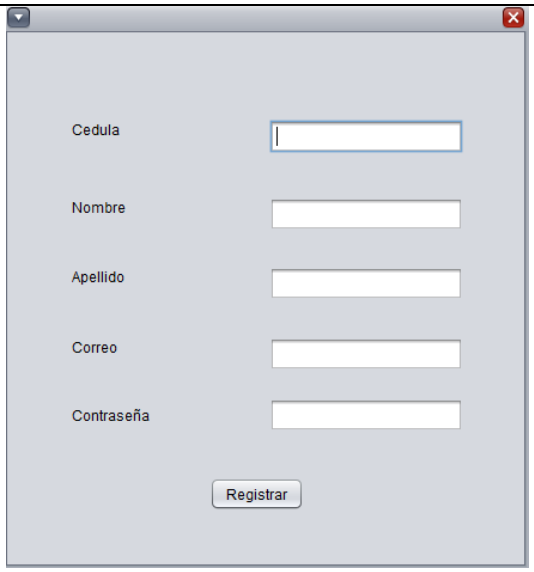
```
    }

    public Usuario verUsuario(){
        return usuario;
    }

    public void actualizarUsuario(String cedula, String nombre, String apellido,
String correo, String contraseña){
        usuario.setCedula(cedula);
        usuario.setNombre(nombre);
        usuario.setApellido(apellido);
        usuario.setCorreo(correo);
        usuario.setContraseña(contraseña);

        daoUs.actualizarUsuario(usuario);
    }

    public Usuario buscarUsuario(String cedula){
        usuario = daoUs.leerUsuarios(cedula);
        if(usuario == null){
            return null;
        }else{
            return usuario;
        }
    }

//    public void eliminarUsuario(){
//
//    }

    //Agregacion
    public void agregarTelefono(String numero, String tipo, String operadora){
        telefono = new Telefono(numero, tipo, operadora);
        daotelf.crearTelefono(telefono);
        usuario.agregarTelefono(telefono);
        daoUs.actualizarUsuario(usuario);
    }

    public void actualizarTelefono(String numero, String tipo, String operadora, int
codigo){
        telefono = new Telefono(codigo, numero, tipo, operadora);
        daotelf.actualizarTelefono(telefono);
        usuario.actualizarTelefono(telefono);
        daoUs.actualizarUsuario(usuario);
    }

    public void eliminarTelefono(int codigo){
        telefono = daotelf.leerTelefono(codigo);
        if(telefono != null){
            daotelf.eliminarTelefono(telefono);
            usuario.eliminarTelefono(telefono);
            daoUs.actualizarUsuario(usuario);
            //telef
        }
    }

    public List<Telefono> listarTelefono(){
```

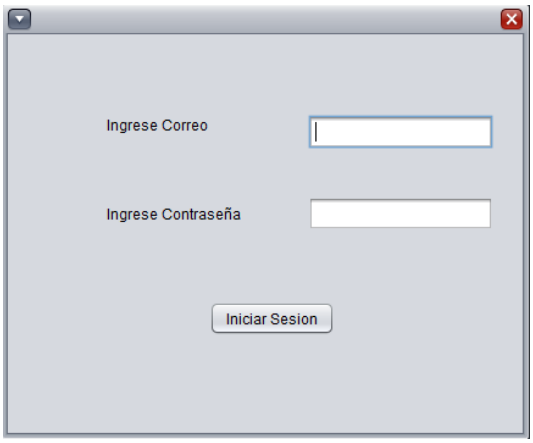
```
        return usuario.getListaTelefonos();
    }

    public List<Telefono> listarTodosTelefonos(){
        return daotelf.findAll();
    }

    public int codigoTelefono(){
        int cont = daotelf.codigoTelefono();
        return cont++;
    }

    public boolean comprobar(String contraseña, String correo){
        if(usuario != null){
            return true;
        }else{
            return false;
        }
    }
      }
```

7. **Creamos las ventanas para el funcionamiento del programa**
    a. **La ventana principal en la que al desplegar un menú se obtendrán las opciones para registrar e ingresar un usuario y listar los teléfonos por usuario.**

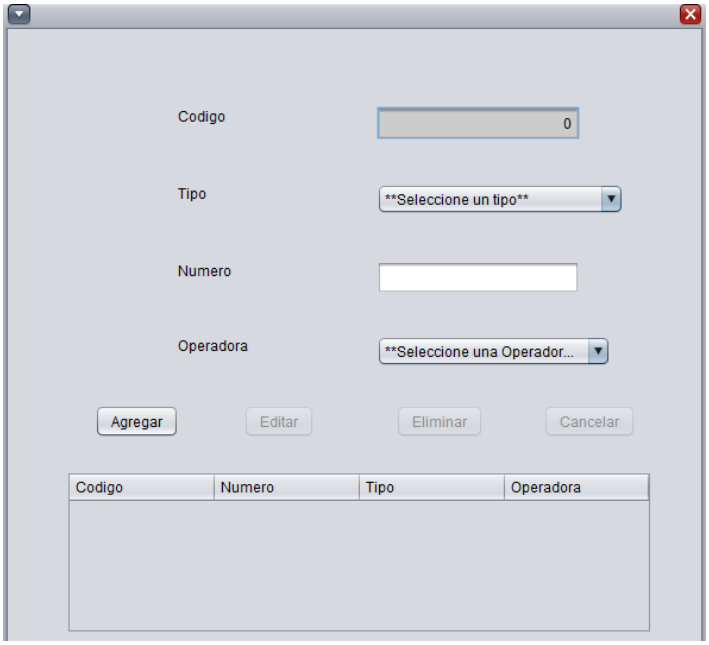


    b. **La ventana registrar será para ingresar un usuario y la ventana iniciar sesión servirá para que el usuario registrado anteriormente pueda crear sus respectivos teléfonos a su agenda telefónica.**

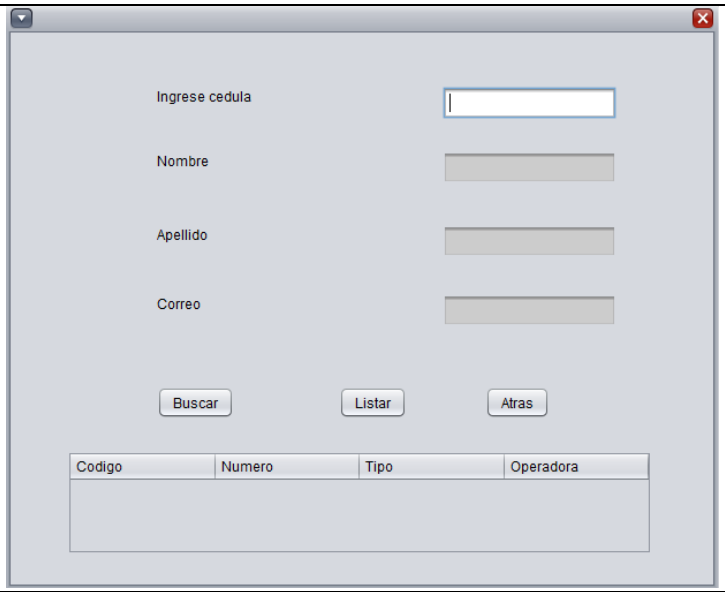




c. **Una vez ingresado el usuario, este podrá crear sus teléfonos e ir agregando a su agenda telefonica**



d. **En la barra de menú se encuentra una opción listar teléfonos por usuario, en esta ventana podrá visualizar los teléfonos que tiene esa persona si en caso de no encontrar datos enviara un mensaje informando que ese usuario buscado o no existe o no tiene 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:
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.

***Nombre de estudiante*: Bryam Parra**

***Firma de estudiante*: _____**