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*: Ing. Diego Quisi**

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

CARRERA: Computación		**ASIGNATURA**: Programación Aplicada
NRO. PRÁCTICA:	1	**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. 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

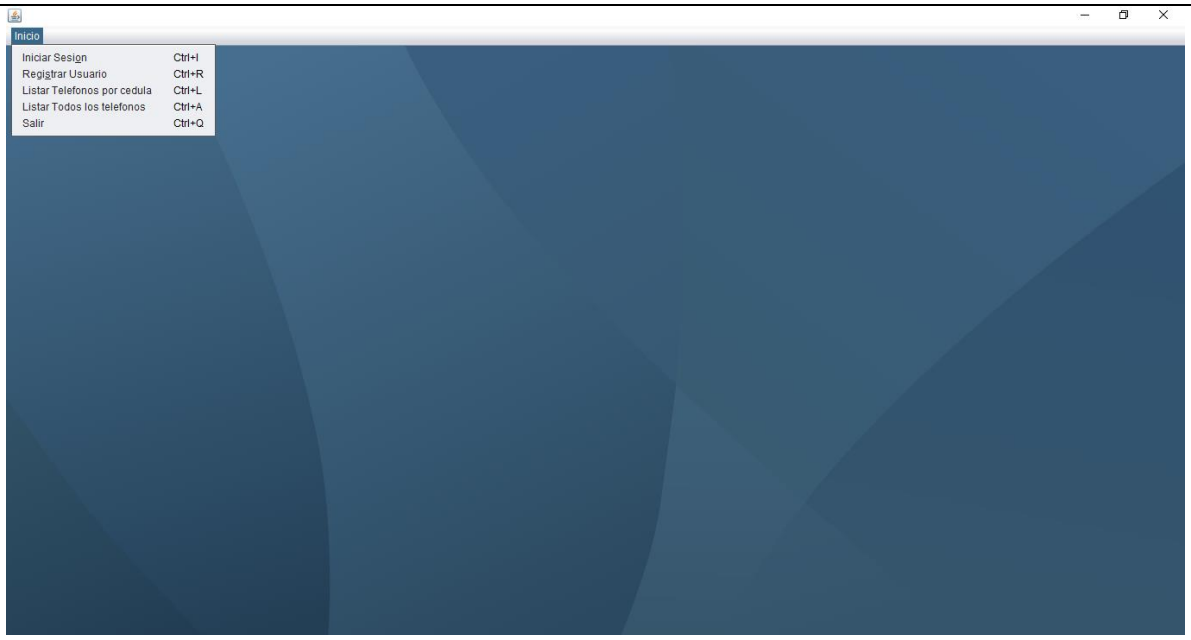
```
1
2   package ec.edu.ups.controlador;
3
4   import ec.edu.ups.modelo.Telefono;
5   import ec.edu.ups.modelo.Usuario;
6   import java.util.ArrayList;
7   import java.util.Collection;
8   import java.util.Iterator;
9   import java.util.List;
10  /**
11   *
12   * @author Anahi
13   */
14  public  class Controlador <E> {
15
16
17      private List<E> listado;
18
19      public Controlador() {
20          listado= new ArrayList();
21      }
22
23
24
25
26      public boolean crear (E obj){
27
28          return listado.add(obj);
```

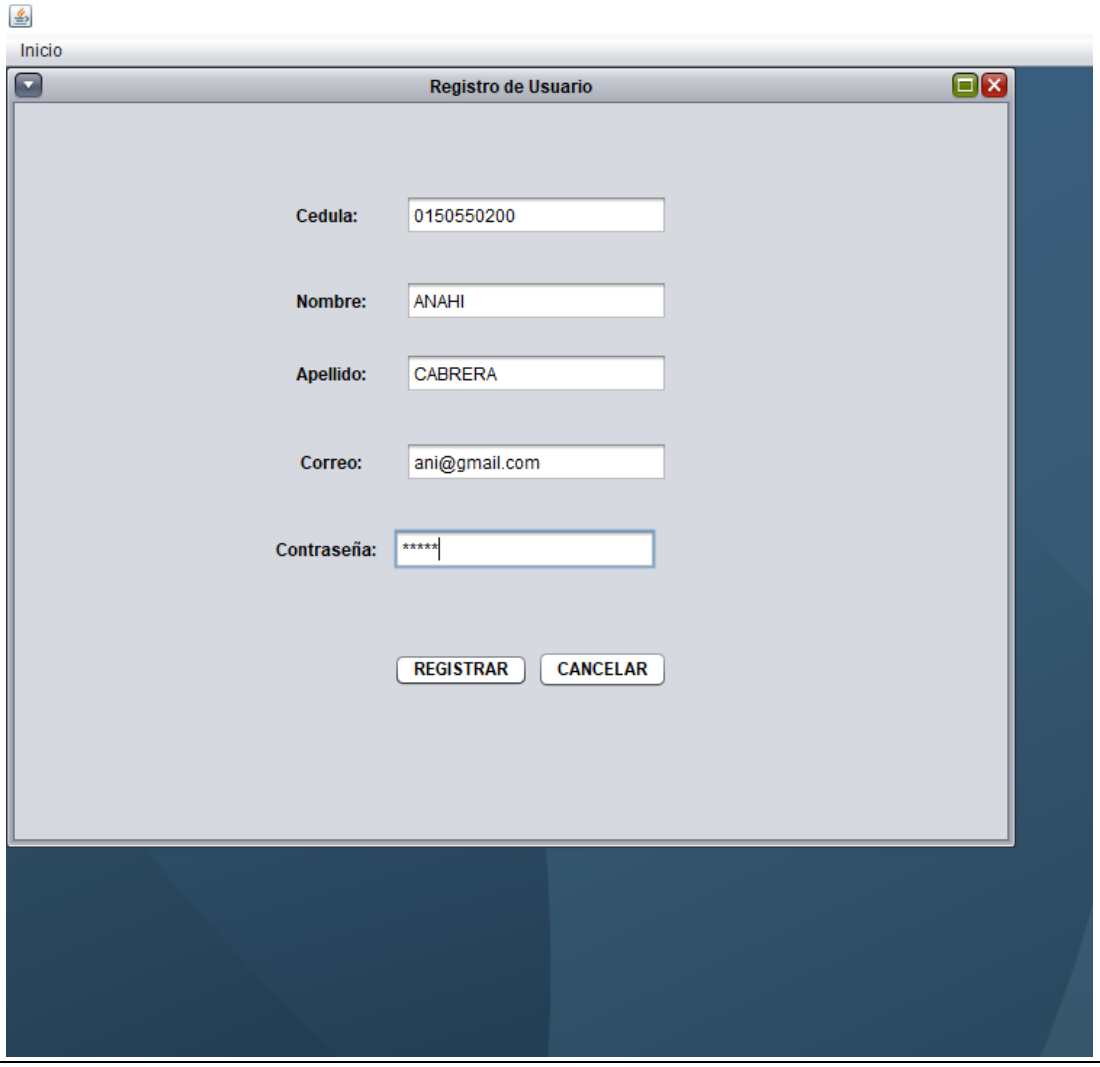
```
31      public E buscar (E ob){
32
33          return this.listado.stream().filter(obj -> ob.equals(obj)).findFirst().get() ;
34
35      }
36      public Telefono buscarTelf (int id){
37         List <Telefono> listaT = (List <Telefono>)  List.copyOf(listado);
38
39          return (Telefono) listaT.stream().filter(Telefono -> Telefono.getCodigo
40          ()==id).findFirst().get();
41
42
43      }
44      public Telefono buscarTelfUsu (String  cedula){
45          List <Telefono> listaT = (List <Telefono>)  List.copyOf(listado);
46
47          return (Telefono) listaT.stream().filter(Telefono -> Telefono.getUsuario
48          ().getCedula().equals(cedula)).findFirst().get();
49
50      }
51
52      public Usuario buscarUsuarios (String cedula){
53
54           List <Usuario> listaU = (List <Usuario>)  List.copyOf(listado);
55          return (Usuario) listaU.stream().filter(Usuario ->
56                  Usuario.getCedula().equals(cedula)).findFirst().get();
57
58      }
```

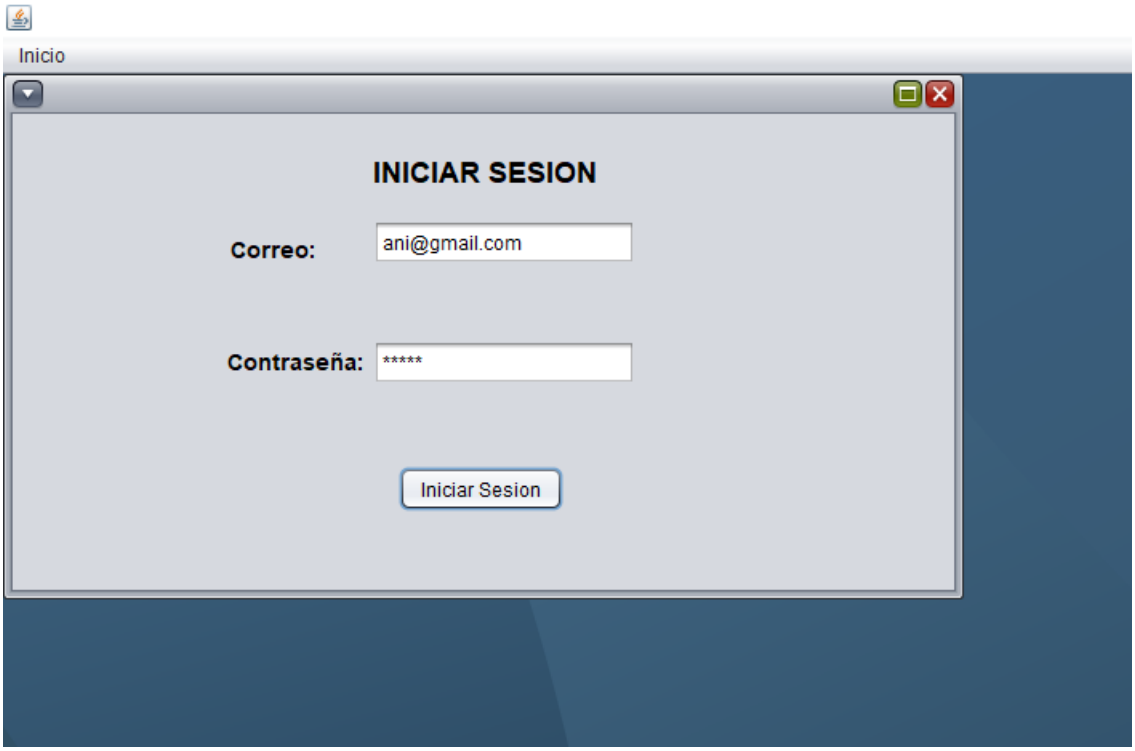
```
    public boolean eliminar(E obj) {
        return listado.remove(obj);
    }
    public void actualizar(E obj, E obj2) {

        int posicion = (listado.indexOf(obj2));
        listado.remove(posicion);

        listado.add(posicion, obj);


    }
public Usuario iniciarSesion(String correo, String pass){
    List <Usuario> listaU = (List <Usuario>)  List.copyOf(listado);
        return (Usuario) listaU.stream().filter(Usuario ->
                Usuario.getCorreo().equals(correo)&&Usuario.getContrasena().
                        equals(pass)).findFirst().get();


}
  public List<E> findAll() {
        return listado;
    }
    public List<Telefono> telefonos (){
       List <Telefono> listaT= new ArrayList();
        Telefono telefono;
        Iterator i  = listado.iterator();
         while(i.hasNext()){
   telefono = (Telefono)i.next();
```

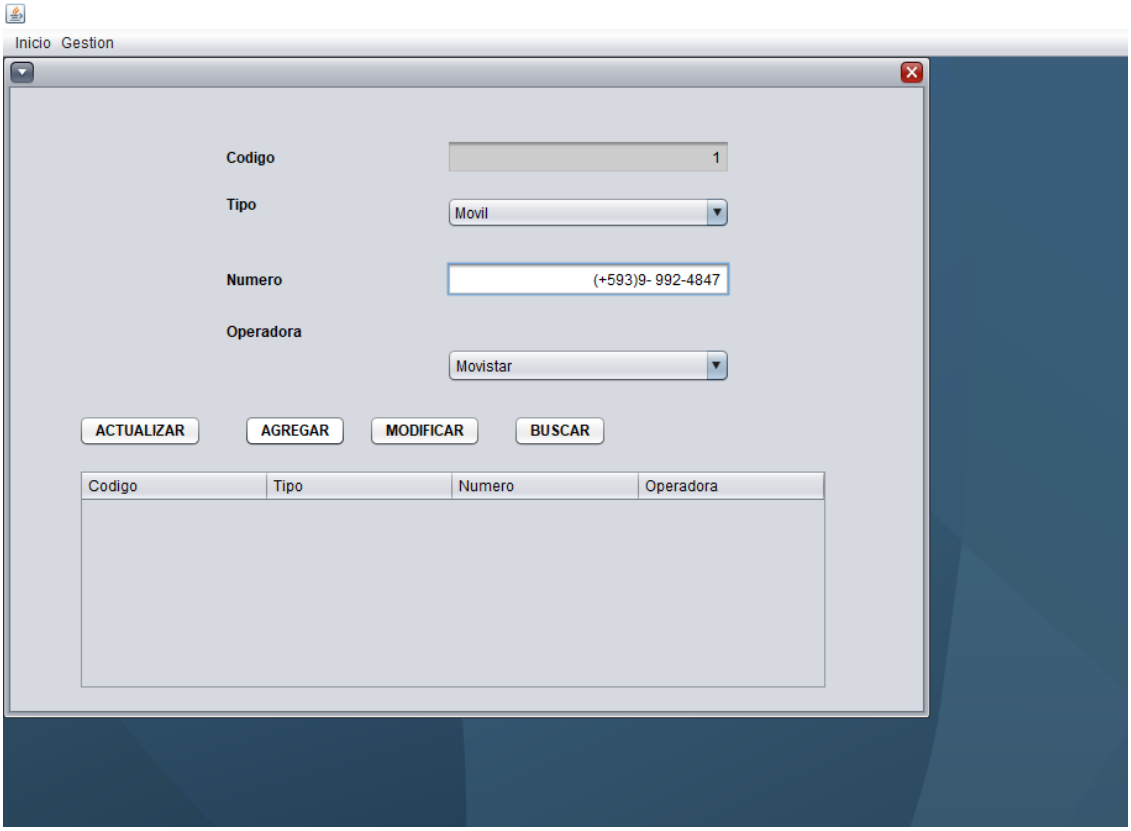
```
92
93      }
94      public List<Usuario> usuarios(){
95          List <Usuario> listaU= new ArrayList();
96          Usuario usuario;
97          Iterator i  = listado.iterator();
98           while(i.hasNext()){
99        usuario = (Usuario)i.next();
100       listaU.add(usuario);
101
102   }
103           return listaU;
104
105     }
106      public int generarId() {
107         if(this.listado.size()>0){
108
109             return (int) listado.size()-1;
110         }
111
112         return 1;
113     }
114     public List<E> getListado() {
115         return listado;
116     }
117
118     public void setListado(List<E> listado) {
119         this.listado = listado;
120     }
```

3. Probar su funcionamiento y rendimiento dentro de los equipos de cómputo de programación genérica y ordenar una lista, buscar.

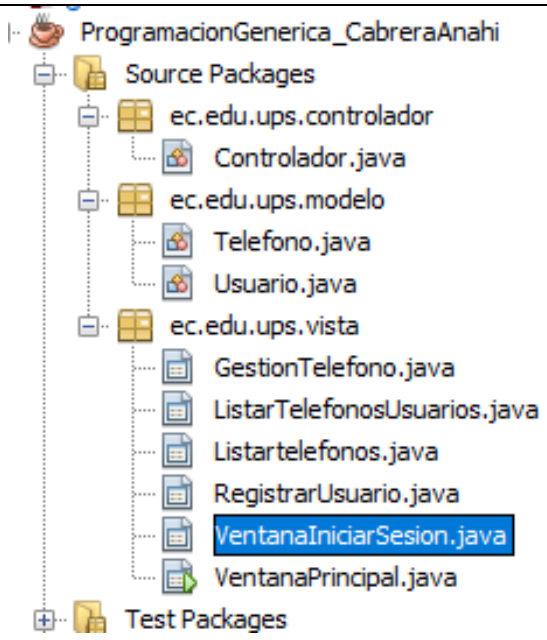

Inicio
Iniciar Sesion Ctrl+I
Registrar Usuario Ctrl+R
Listar Telefonos por cedula Ctrl+L
Listar Todos los telefonos Ctrl+A
Salir Ctrl+Q


Inicio
Registro de Usuario
Cedula:
0150550200
Nombre:
ANAHI
Apellido:
CABRERA
Correo:
ani@gmail.com
Contraseña:
*****
REGISTRAR
CANCELAR





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



Clase Usuario

```
package ec.edu.ups.modelo;

import java.util.Objects;

/**
 *
 * @author Anahi
 */
public class Usuario {

    private int id;
    private String Cedula;
    private String nombre;
    private String apellido;
    private String correo;
    private String contrasena;

    public Usuario() {

    }

    public Usuario(int id,String Cedula, String nombre, String apellido, String
correo, String contrasena) {
        this.setId(id);
        this.setCedula(Cedula);
        this.setNombre(nombre);
        this.setApellido(apellido);
        this.setCorreo(correo);
        this.setContrasena(contrasena);

    }

    public int getId() {
        return id;
```

```
    }

    public void setId(int id) {
        this.id = id;
    }

    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 getContrasena() {
        return contrasena;
    }

    public void setContrasena(String contrasena) {
        this.contrasena = contrasena;
    }

    @Override
    public int hashCode() {
        int hash = 5;
        hash = 89 * hash + this.id;
        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.id, other.id)) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Usuario{" + "Cedula=" + Cedula + ", nombre=" + nombre + ",
apellido=" + apellido + ", correo=" + correo + ", contrasena=" + contrasena + '}';
    }

}
```

Clase Teléfono

```
package ec.edu.ups.modelo;

/**
 *
 * @author Anahi
 */
public class Telefono {

    private int codigo;
    private String numero;
    private String tipo;
    private String operadora;

    private Usuario usuario;

    public Telefono() {
    }

    public Telefono(int codigo, String numero, String tipo, String operadora) {
        this.setCodigo(codigo);
        this.setNumero(numero);
        this.setTipo(tipo);
        this.setOperadora(operadora);

    }

    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;
    }

    public Usuario getUsuario() {
        return usuario;
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }




    @Override
    public int hashCode() {
        int hash = 5;
        hash = 37 * hash + this.codigo;
        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 Telefono other = (Telefono) obj;
        if (this.codigo != other.codigo) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "DATOS TELEFONO ** " + "codigo: " + codigo + " numero: " + numero + "
tipo: " + tipo + " operadora: " + 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:

Con las actualizaciones de java 9,10,11 y 12 se puede trabajar de una manera más eficiente donde podemos reducir código por si en un futuro próximos programadores quisiera modificarlo sería de una manera más fácil.

RECOMENDACIONES:

Utilizar buenas prácticas de programación

***Nombre de estudiante*: Edith Anahí Cabrera Bermeo**