

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

CARRERA:

ASIGNATURA: PROGRAMACIÓN APLICADA

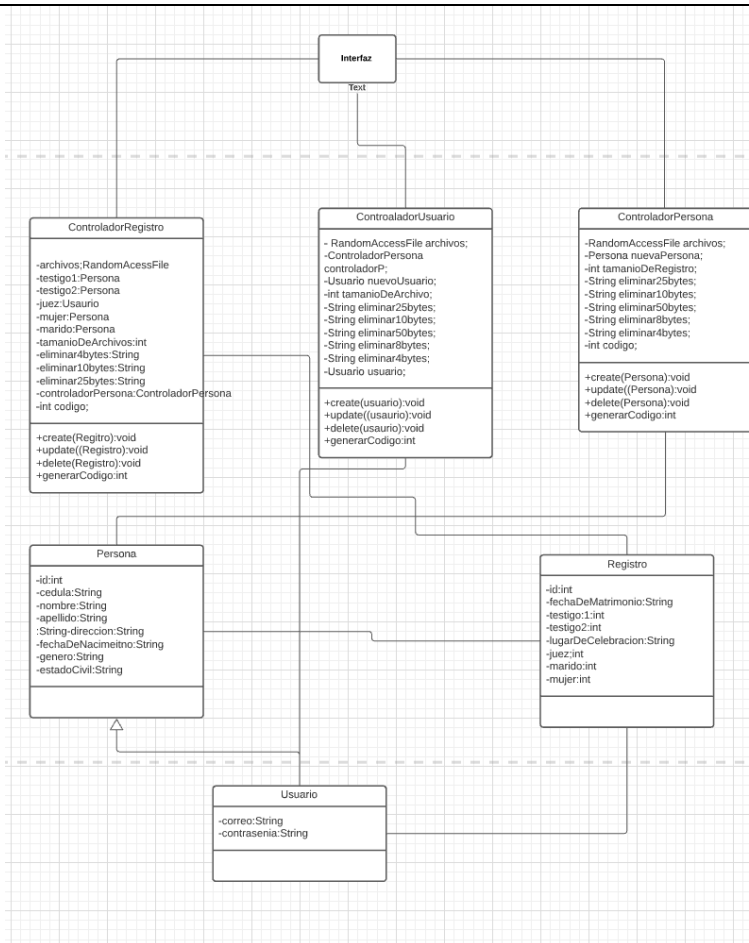
NRO.
PRÁCTICA:

1.1

TÍTULO PRÁCTICA: Prueba Practica 1

OBJETIVO ALCANZADO: Reforzar los conocimientos adquiridos en clase sobre la programación aplicada (Java 8, Programación Genérica, Reflexión y Patrones de Diseño) en un contexto real.

ACTIVIDADES DESARROLLADAS



1. Se nos pidió crear una aplicación de matrimonios que tengan a dos individuos y los testigos. El juez o autoridad tendrá el acceso a la aplicación mientras que los demás solamente se registran.

2. Se crearon 3 paquetes para este proyecto.

- ec.ups.edu.controlador
- ec.ups.edu.modelo
- ec.ups.edu.vista

3. ec.ups.edu.modelo

- el paquete modelo contiene tres clases: Persona, Usuario, Registro.
- En la clase Persona van a tener todos los atributos de una persona incluyendo su estado civil y su genero.
- La clase usuario es una clase que hereda de la clase persona. Solamente tiene como atributos propios el correo y la contraseña.

- La clase registro recibe a varias personas y este cambia el estado de los individuos que se están casando. Aparte que también guarda los datos de los testigos.

4. **Persona**

- Atributos:
 - private int id;
 - private String cedula;
 - private String nombre;
 - private String apellido;
 - private String direccion;
 - private String fechaDeNacimiento;
 - private String genero;
 - private String estadoCivil;
- Esta clase solamente tiene dos constructores. 1. Un constructor vacío 2. Constructor con todos sus atributos.
- Esta clase también tiene declarado todos sus getters y setters.
- Esta clase también tiene instanciada su hashCode y equals.
- Finalmente esta clase tiene creado su último método toString().

5. **Usuario**


- Atributos:
 - private String correo;
 - private String contrasenia;
- Esta clase hereda de la clase persona y solamente tiene dos atributos.
- Esta clase de igual manera tiene solamente creado dos constructores. 1. constructor vacío 2. Constructor con todos sus atributos incluyendo de su clase padre.
- Esta clase también tiene declarado todos sus getters y setters.
- Esta clase también tiene instanciada su hashCode y equals.
- Finalmente esta clase tiene creado su último método toString().

6. **Registro**

- La clase registro va a recibir varios objetos de tipo persona.
- Atributos:
 - private int id;
 - private String fechaDeMatrimonio;
 - private int testigo1;
 - private int testigo2;
 - private String lugarDeCelebracion;
 - private int juez;
 - private int marido;
 - private int mujer;
- Esta clase de igual manera tiene solamente creado dos constructores. 1. constructor vacío 2. Constructor con todos sus atributos
- Esta clase también tiene declarado todos sus getters y setters.
- Esta clase también tiene instanciada su hashCode y equals.
- Finalmente esta clase tiene creado su último método toString().

7. **ec.ups.edu.controlador**

- este paquete contiene todas las clases que van a interactuar con la interfaz.

	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

- Este paquete tiene cuatro clases.
- La clase controladorGenerico es una clase que utilizamos para reutilizar métodos y no tener que escribirlas o instanciarlas nuevamente.
- La clase ControladorPersona nos ayuda a crear una persona y guardarla en la base de datos.
- La clase ControladorUsuario es una clase para dar autoridad al juez para que solamente el pueda utilizar la aplicación y realizar el acto del matrimonio.
- La clase ControladorRegistro nos ayuda a guardar todos los datos de las personas que van a estar presente al momento del matrimonio.

8. controladorPersona

- atributos:
 - private RandomAccessFile archivos;
 - private Persona nuevaPersona;
 - private int tamanoDeRegistro;
 - private String eliminar25bytes;
 - private String eliminar10bytes;
 - private String eliminar50bytes;
 - private String eliminar8bytes;
 - private String eliminar4bytes;
 - private int codigo;
- se calculo que la cantidad de espacio que utilizaría esta clase fue de 156 bytes. Se utilizaron métodos como randomAccessFiles para acceder y crear un archivo para esta clase.
- Métodos:
 - Créate: Este método recibe todos los datos de una persona y los manda a guardar en los archivos.
 - Read: Este método retorna un objeto de tipo persona y los busca mediante su código único.
 - Update: Este método actualiza una persona, recibe un objeto de tipo persona en su parámetro y manda a actualizar a la persona que tenga el mismo código y los actualiza con los nuevos datos que recibió.
 - Delete: Este método simplemente elimina un usuario. En el caso de mi programa llena los espacios en blanco en donde solía estar ubicado el usuario.
 - generarCodigo: Este método solamente consigue el próximo índice para asignarlo a la siguiente persona.

9. ControladorRegistro

- Esta clase solamente sirve para que el juez pueda tener la autoridad de realizar los actos de matrimonios.
- Atributos:
 - private Registro registro;
 - private RandomAccessFile archivos;
 - private Persona testigo1;
 - private Persona testigo2;
 - private Persona juez;
 - private Persona mujer;
 - private Persona marido;
 - private int tamanoDeArchivos;
 - private int eliminar4bytes;
 - private String eliminar10bytes;

private String eliminar25bytes;

private ControladorPersona controladorPersona;

private int codigo;

- **Métodos:**

- Créate: Este método solamente guarda los datos de la pareja recién casada con todos los datos de los testigos.
- Read: Este método sirve para buscar un acto de matrimonio y solamente se debe buscar por el código asignado en cada acta.
- Delete: Este método sirve para eliminar las actas de matrimonio en caso de que las personas se quieran divorciar.
- findAillRegistros: Este método no sirve para listar todas las actas existentes dentro de los archivos

10. **controladorUsuario**

- Esta clase se creao para que solamente los que son jueces puedan acceder a la aplicación. Las demás personas solamente se pueden registrar.

- Atributos:

private RandomAccessFile archivos;

private ControladorPersona controladorP;

private Usuario nuevoUsuario;

private int tamanoDeArchivo;

private String eliminar25bytes;

private String eliminar10bytes;

private String eliminar50bytes;

private String eliminar8bytes;

private String eliminar4bytes;

private Usuario usuario;

- Meotods:

- Registrar: Este método simplemente registra un juez en los archivos para que luego el sistema solo les deje a ellos ingresar en la aplicación.
- Login: Este método solamente recibe el correo y la contraseña que ingresan y verifica si existen en los archivos de los usuarios.
- buscarUsaurio: Este método recibe un objeto de tipo entero y busca al usuario por medio de su código único.

11. **Ec.ups.edu.vista**

- Este paquete contiene todas las interfaces en las cuales el usuario solamente interactuaran con ellos.

12. **MenuPrincipal**

INICIO MENU

<input type="checkbox"/> I. SESSION	shortcut
<input type="checkbox"/> L. MATRIMONIOS	shortcut
<input type="checkbox"/> R. PERSONA	shortcut
<input type="checkbox"/> QUIT	shortcut
<input type="checkbox"/> CERRAR SESSION	shortcut

- Esta interfaz solamente contendrá a todas las demás interfaces en donde el usuario interactuara con la aplicación.

13. ListarDatos

ACTA

FECHA

TESTIGO

TESTIGO

UBICACION

JUEZ

MARIDO

ESPOSA

MOSTRAR

DIVORCIAR

NOMBRE

APELLIDO

JUEZ:

MARIDO:

ESPOSA:

TESTIGO1:

TESTIGO2:

- Esta interfaz permitirá que el usuario pueda listar todas las actas de matrimonio que pueden existir en los archivos.

14. iniciarSesion

INICIAR SESSION

CORREO:

CONTRASEÑA:

INICIO CANCELAR

- Esta interfaz solamente recibirá el correo y la contraseña de un usuario y verificara si existen en los archivos \. Si es que no existen le saldrá una venta de error, caso contrario se habilitaran las demás ventanas.

15. RegistrarPersona



The screenshot shows a web browser window with a registration form titled "REGISTRO". The form contains the following fields and controls:

- ID:** A text input field containing the value "0".
- CEDULA:** A text input field.
- NOMBRE:** A text input field.
- APELLIDO:** A text input field.
- DIRECCION:** A text input field.
- FECHA DE NACIMIENTO:** A date selection field showing "-- --".
- GENERO:** A dropdown menu with the text "--SELECCIONE UNA OPCION--" and a blue arrow icon.
- ESTADO CIVIL:** A dropdown menu with the text "--SELECCIONE UNA OPCION--" and a blue arrow icon.
- CORREO:** A text input field.
- CONTRASEÑA:** A text input field.

At the bottom of the form, there are two buttons: "REGISTRAR" and "CANCELAR".

- Este interfaz permitirá que el usuario pueda registrarse. En el caso de que un usuario seleccione su estado civil como "Juez". Se habilitarán los campos de texto de correo y contraseña. El programa les permitirá registrarse como juez caso contrario una persona que no es "juez" no puede acceder a la aplicación.

16. RegistroCivil

REGISTRO CIVIL

CODIGO DE PROMETIDA/O:	CODIGO DE PROMETIDO/A:	CODIGO DE TESTIGO 1:		FECHA:
<input type="text"/>	<input type="text"/>	<input type="text"/>		<input type="text"/>
CEDULA:	CEDULA:	CEDULA:	CEDULA:	UBICACION:
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
NOMBRE:	NOMBRE:	NOMBRE:	NOMBRE:	CASO:
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="0"/>
APELLIDO:	APELLIDO:	APELLIDO:	APELLIDO:	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
DIRECCION:	DIRECCION:	DIRECCION:	DIRECCION:	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
FECHA DE NACIMIENTO:	FECHA DE NACIMIENTO:	FECHA DE NACIMIENTO:	FECHA DE NACIMIENTO:	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
GENERO:	GENERO:	GENERO:	GENERO:	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
ESTADO CIVIL:	ESTADO CIVIL:	ESTADO CIVIL:	ESTADO CIVIL:	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	

ACEPTAR

- Esta interfaz permitirá que el juez ingrese los datos fácilmente. Todos los campos de texto esta bloqueados excepto los campos de ingreso de código. El “juez” debe ingresar los códigos de los individuos y presionar espacio o Enter. Si es que el usuario existe los datos se cargaran automáticamente en las cajas de texto caso contrario soltara una venta de error diciendo que la persona no esta registrada. Una vez que el juez hay terminado de ingresar los datos, lo único que debe hacer es presionar “aceptar” y se registrara el matrimonio.

RESULTADO(S) OBTENIDO(S):

```
System.out.println(ex);
}
return null;*/
for (Object object : FindAll()) {
    Method[] metodos = object.getClass().getMethods();
    for (Method m1 : metodos) {
        if (m1.getName().equals("getCorreo")) {
            try {
                if (m1.invoke(object, null).equals(correo)) {
                    for (Method m1 : metodos) {
                        if (m1.getName().equals("getContraseña")) {
                            if (m1.invoke(object, null).equals(contraseña)) {
                                usuario = (Usuario) object;
                                return (Usuario) object;
                            }
                        }
                    }
                }
            } catch (IllegalAccessException | IllegalArgumentException | InvocationTargetException ex) {
                Logger.getLogger(ControladorUsuario.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
return null;
}
```



```

*/
public abstract class ControladorGenerico<T> {
    //private RandomAccessFile archivos;
    private int codigo;
    private T tipo;

    public ControladorGenerico() {
        codigo=0;
    }

    public void create(T Objeto){

    }

    public T read(int codigo){
        return null;
    }

    public void delete(T Objeto){

    }

    public void update(T objeto){

    }

    public int generarCodigo(){
        int siguienteCodigo=++codigo;
        return siguienteCodigo;
    }

    public int getCodigo() {
        return codigo;
    }
}

*/
public class ControladorRegistro extends ControladorGenerico<Registro> {
    private Registro registro;
    private RandomAccessFile archivos;
    private Persona testigo1;
    private Persona testigo2;
    private Persona juez;
    private Persona mujer;
    private Persona marido;
    private int tamañoDeArchivos;
    private int eliminar4bytes;
    private String eliminar10bytes;
    private String eliminar25bytes;
    private ControladorPersona controladorPersona;
    private int codigo;

    /*tamaño de archivo
    *private int id| 4 bytes
    *private String fechaDeMatrimonio| 10 bytes +2 bytes
    *private int testigo1| 4 bytes
    *private int testigo2| 4 bytes
    *private String lugarDeCelebracion| 25 bytes+ 2 bytes
    *private int juez| 4 bytes
    *private int marido| 4 bytes
    *private int mujer|4 bytes
    *total=63 bytes
    */
    public ControladorRegistro() {
        try {
            archivos = new RandomAccessFile("datos/Registro.dat", "rw");
            tamañoDeArchivos = 63;
            eliminar4bytes = 0;
            eliminar10bytes = " ";
        }
    }
}

*/
public class ControladorPersona extends ControladorGenerico<Persona> {
    private RandomAccessFile archivos;
    private Persona nuevaPersona;
    private int tamañoDeRegistro;
    private String eliminar25bytes;
    private String eliminar10bytes;
    private String eliminar50bytes;
    private String eliminar8bytes;
    private String eliminar4bytes;
    private int codigo;

    /*calculo de propiedades
    *private int id|4 bytes
    *private String cedula|10 bytes+ (2bytes)
    *private String nombre|25 bytes+ ((2bytes)
    *private String apellido|25 bytes+(2bytes)
    *private String direccion|50 bytes +(2bytes)
    *private String fechaDeNacimiento|10 bytes + 2(bytes)
    *private String genero|10 bytes+ (2bytes)
    *private String estadoCivil|8 bytes +(2bytes)
    *total= 140+14=156
    */
    public ControladorPersona() {
        try {
            archivos = new RandomAccessFile("datos/Persona.dat", "rw");
            tamañoDeRegistro = 156;
            eliminar25bytes = " ";
            eliminar10bytes = " ";
            eliminar4bytes = " ";
            eliminar8bytes = " ";
            eliminar50bytes = " ";
            codigo=0;
        }
    }
}

} catch (FileNotFoundException ex) {
    System.out.println("Error escritura y lectura [ControladorPersona]");
}

```

CONCLUSIONES: En conclusión, al realizar esta practica se debe tener conocimiento sobre las clases genérica, como utilizarlas e implementarlas. También se debe tener conocimiento de como implementar reflexión para acceder a otros métodos en caso de que sea necesario.

RECOMENDACIONES:

Nombre de estudiante: Denys Dutan

Firma de estudiante: 