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

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN/INGENIERÍA DE SISTEMAS		ASIGNATURA: PROGRAMACIÓN APLICADA	
NRO. PROYECTO:	1.1	TÍTULO PROYECTO: Prueba Practica 1 Desarrollo e implementación de un sistema de gestión de matrimonios de la ciudad de Cuenca	
OBJETIVO: Reforzar los conocimientos adquiridos en clase sobre la programación aplicada (Java 8, Programación Genérica, Reflexión y Patrones de Diseño) en un contexto real.			
INSTRUCCIONES:		1. Revisar el contenido teórico y práctico del tema	
		2. Profundizar los conocimientos revisando los libros guías, los enlaces contenidos en los objetos de aprendizaje Java y la documentación disponible en fuentes académicas en línea.	
		3. Deberá desarrollar un sistema informático para la gestión de matrimonios, almacenar en archivos y una interfaz gráfica.	
		4. Deberá generar un informe de la práctica en formato PDF y en conjunto con el código se debe subir al GitHub personal.	
		5. Fecha de entrega: El sistema debe ser subido al git hasta 27 de noviembre del 2020 – 23:55.	
ACTIVIDADES POR DESARROLLAR			

1. Enunciado:

Realizar el diagrama de clase y el programa para gestionar los matrimonios de la ciudad de Cuenca empleando las diferentes tecnicas de programación revisadas en clase.

Problema: De cada matrimonio se almacena la fecha, el lugar de la celebración y los datos personales (nombre, apellido, cédula, dirección, genero y fecha de nacimiento) de los contrayentes. Es importante validar la equidad de genero.

Igualmente se guardar los datos personales de los dos testigos y de la autoridad civil (juez o autoridad) que formalizan el acto. Ademas de gestionar la seguridad a traves de un sistema de Usuarios y Autentificación.

Calificación:

- ⑩ Diagrama de Clase 20%
- ⑩ MVC: 20%
- ⑩ Patrón de Diseño aplicado : 30%
- ⑩ Tecnicas de Programación aplicadas (Java 8, Reflexión y Programación Generica): 20%
- ⑩ Informe: 10%A

2. Informe de Actividades:


- Planteamiento y descripción del problema.
- Diagramas de Clases.
- Patrón de diseño aplicado
- Descripción de la solución y pasos seguidos.
- Conclusiones y recomendaciones.
- Resultados.

RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones gráficas en sistemas.
- Los estudiantes están en la capacidad de implementar la persistencia en archivos.

	Computación	Docente: Diego Quisi Peralta
	Programacion Aplicada	Período Lectivo: Septiembre 2020 – Febero 2021

RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la prueba.**

BIBLIOGRAFIA:

[1]: <https://www.ups.edu.ec/evento?calendarBookingId=98892>

Docente / Técnico Docente: Ing. Diego Quisi Peralta Msc.

Firma: _____



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

CARRERA: Computacion

ASIGNATURA: Programacion Aplicada.

NRO. PRÁCTICA:

TÍTULO PRÁCTICA: Prueba Practica 1

OBJETIVO ALCANZADO:

Reforzar los conocimientos adquiridos en clase sobre la programacion aplicada (Java 8, Programación Generica, Reflexión y Patrones de Diseño) en un contexto real.

ACTIVIDADES DESARROLLADAS

1.Revisar el contenido teórico y práctico del tema

Patrón diseño aplicado: el patrón de diseño que se implemento es el singleton con el cual solo tenemos que instanciarlo una vez.

```
public static ControladorJuez getInstance() {  
  
    if (instancia == null) {  
  
        instancia = new ControladorJuez();  
  
    }  
    return instancia;  
}
```

```
controladorPersona = controladorPersona.getInstance();  
controladorUsuario = controladorUsuario.getInstance();  
controladorJuez = controladorJuez.getInstance();
```

2. Profundizar los conocimientos revisando los libros guías, los enlaces contenidos en los objetos de aprendizaje Java y la documentación disponible en fuentes académicas en línea.

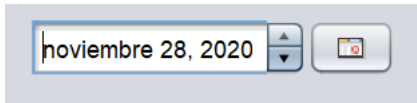
Se ha utilizado el patrón de diseño modelo vista controlador (MVC)

Modelo: Tiene la parte lógica del programa

Vista: o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.

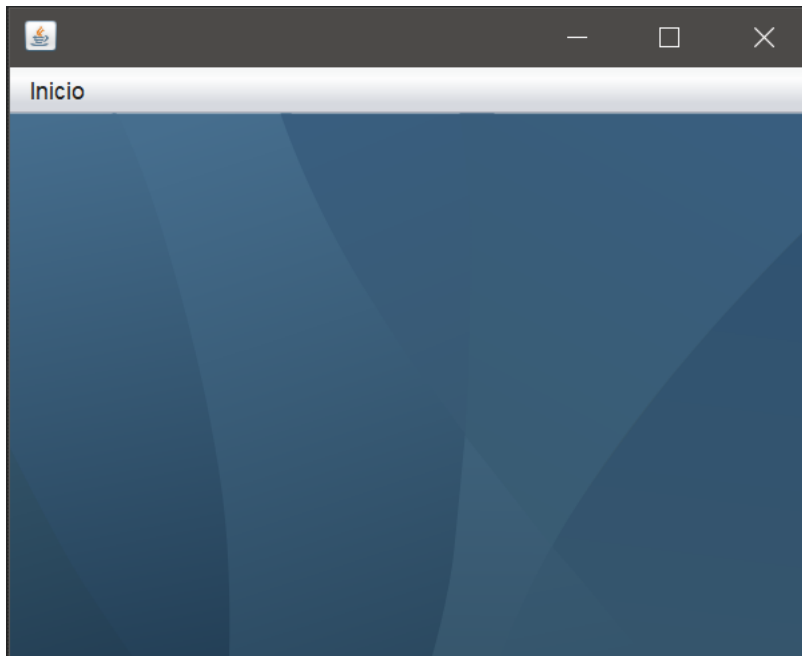
Controlador: que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Se ha utilizado la librería JCalendar con el cual se han agregado las fechas



3. Deberá desarrollar un sistema informático para la gestión de matrimonios, almacenar en archivos y una interfaz gráfica.

Se ha creado una pantalla de inicio donde el usuario elegirá que desea hacer.



Una vez iniciada sesión el usuario es capaz de ingresar a las demás opciones que se encuentran en el programa



Y podrá registrar a los conyugues y como a su vez a los dos testigos y al juez.

The screenshot shows a web application window titled 'Datos Conyugues'. At the top right, there is a date selector showing 'noviembre 9, 2020' and a calendar icon. The form contains several input fields: 'Cedula' (0100513589), 'Nombre' (Santiago), 'Apellido' (Cabrera), 'Direccion' (Manuel Guillen), 'Genero' (masculino), and 'Fecha' (empty). A modal message box is overlaid in the center, displaying an information icon, the text 'registrado con exito', and an 'OK' button. Below the form, there are three buttons: 'Actualizar', 'Registrar', and 'Listar'.

Cedula	Nombre	Apellido	Direccion	Genero	Fecha
--------	--------	----------	-----------	--------	-------

A su vez podrá listar los datos que se hayan guardado en los archivos.

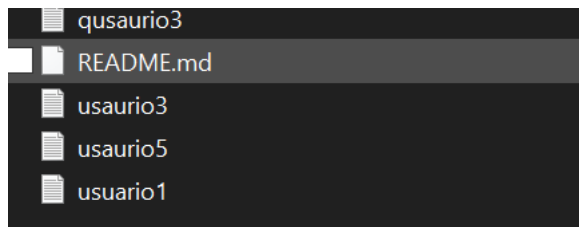
This screenshot shows the same 'Datos Conyugues' application window, but now displaying the list of registered data. The 'Fecha' field in the form is empty. A table is visible on the right side of the form, containing the following data:

Cedula	Nombre	Apellido	Direccion	Genero	Fecha
0100513589	Santiago	Cabrera	Manuel Gu...	masculino	Thu Jan 0...

The 'Actualizar', 'Registrar', and 'Listar' buttons are still present at the bottom of the form.

Al igual podremos actualizar los registros en el cual el usuario se haya confundido o necesite agregar alguna cosa.

Archivos creados a lo largo del programa.



4. Deberá generar un informe de la practica en formato PDF y en conjunto con el código se debe subir al GitHub personal.

Clase Abstracta

```
package ec.ups.edu.controlador;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author santi
 * @param <T>
 */
public abstract class ControladorAbstracto<T> {
```

```

    public abstract void create(T objeto);
    public abstract T read(String cedula);
    public abstract boolean update(T objeto);
    public abstract boolean delete(T objeto);
    public abstract List<T> lista();
}

```

Controlador Juez

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.controlador;

import ec.ups.edu.modelo.Persona;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 *
 * @author santi
 */
public class ControladorJuez extends ControladorAbstracto<Persona> {

    /*
    private String cedula; = 10
    private String nombre; = 15
    private String apellido; = 15
    private String direccion; = 50
    private String genero; = 10
    private Date fecha; = 30

    total registro = 142.
    */
    private RandomAccessFile archivo;
    private int salto;
    private int tamanoRegistro;

    private static ControladorJuez instancia;

    public static ControladorJuez getInstance() {

        if (instancia == null) {

            instancia = new ControladorJuez();

        }
        return instancia;
    }

    public ControladorJuez() {

        try {

```



```
        archivo = new RandomAccessFile("usaurio5.txt", "rw");

    } catch (IOException ex) {

        ex.printStackTrace();

    }

}

@Override
public void create(Persona objeto) {

    salto = 0;

    try {

        archivo.seek(salto);
        archivo.writeUTF(objeto.getCedula());
        archivo.writeUTF(objeto.getNombre());
        archivo.writeUTF(objeto.getApellido());
        archivo.writeUTF(objeto.getDireccion());
        archivo.writeUTF(objeto.getGenero());
        archivo.writeUTF(objeto.getFecha() + "");

    } catch (IOException ex) {

        System.out.println("Error de lectura y escritura");
        ex.printStackTrace();

    }

}

@Override
public Persona read(String cedula) {

    salto = 0;
    tamañoRegistro = 142;

    try {

        if (salto < archivo.length()) {

            archivo.seek(salto);
            String cedulaArchivo = archivo.readUTF();

            if (cedula.equals(cedulaArchivo)) {

                String nombre = archivo.readUTF().trim();
                String apellido = archivo.readUTF().trim();
                String direccion = archivo.readUTF().trim();
                String genero = archivo.readUTF().trim();
                String fecha = archivo.readUTF().trim();

                int dia = Integer.parseInt(fecha.substring(7,10));
                int anio = Integer.parseInt(fecha.substring(24,28));
                int mes = Integer.parseInt(fecha.substring(3,7));

                Date date = new Date(anio-1900,mes,dia);
```

```

        Persona persona = new Persona(cedula, nombre, apellido, direccion, genero, date);

        return persona;
    }

    salto += tamanoRegistro;
}

} catch (IOException ex) {

    System.out.println("Error de lectura y escritura");
    ex.printStackTrace();

}

return null;
}

@Override
public boolean update(Persona objeto) {

    salto = 0;
    tamanoRegistro = 142;

    try {

        if (salto < archivo.length()) {

            archivo.seek(salto);
            archivo.writeUTF(objeto.getCedula());
            archivo.writeUTF(objeto.getNombre());
            archivo.writeUTF(objeto.getApellido());
            archivo.writeUTF(objeto.getDireccion());
            archivo.writeUTF(objeto.getGenero());
            archivo.writeUTF(objeto.getFecha() + "");

        }

        salto += tamanoRegistro;

    } catch (IOException ex) {

        ex.printStackTrace();

    }

    return false;
}

@Override
public boolean delete(Persona objeto) {

    try {

        salto = 0;

```

```
tamanoRegistro = 142;
String cadena = "";

if (salto < archivo.length()) {

    archivo.seek(salto);
    archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
    archivo.writeUTF(String.format("%-" + 15 + "s", cadena));
    archivo.writeUTF(String.format("%-" + 15 + "s", cadena));
    archivo.writeUTF(String.format("%-" + 50 + "s", cadena));
    archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
    archivo.writeUTF(String.format("%-" + 30 + "s", cadena));

}

salto += tamanoRegistro;

} catch (IOException ex) {

    System.out.println("Error de lectura y escritura.");
    ex.printStackTrace();

}

return false;

}

@Override
public List<Persona> lista() {

    List<Persona> lista = new ArrayList<Persona>();
    int salto = 0;
    int registro = 142;
    try {
        while (salto < archivo.length()) {
            archivo.seek(salto);

            String cedula = archivo.readUTF().trim();
            if (!cedula.equals("")) {

                String nombre = archivo.readUTF().trim();
                String apellido = archivo.readUTF().trim();
                String direccion = archivo.readUTF().trim();
                String genero = archivo.readUTF().trim();
                String fecha = archivo.readUTF().trim();

                System.out.println(fecha);

                int dia = Integer.parseInt(fecha.substring(8,10));
                int anio = Integer.parseInt(fecha.substring(24,28));
                int mes = recuperarMes(fecha.substring(4,7));

                Date date = new Date(anio-1900,mes,dia);

                Persona persona = new Persona(cedula, nombre, apellido, direccion, genero,date);
                lista.add(persona);

            }

        }

    } catch (IOException ex) {
        System.out.println("Error de lectura y escritura.");
        ex.printStackTrace();
    }

    return lista;
}
```

```

        salto += registro;

    }

    } catch (IOException ex) {
        System.out.println("Error lectura escritura (UsuarioDao:Update)");
        ex.printStackTrace();
    }
    return lista;
}

public int recuperarMes(String mes){

    String meses[] =
{"Jan", "Feb", "Mar", "April", "May", "June", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

    for (int i = 0; i < 10; i++) {

        if (meses[i].equals(mes)) {

            return i;

        }

    }

    return 0;

}

}

```

Controlador Matrimonio

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.controlador;

import ec.ups.edu.modelo.Matrimonio;
import ec.ups.edu.modelo.Persona;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 *
 * @author santi
 */
public class ControladorMatrimonio extends ControladorAbstracto<Matrimonio> {

    /*
    private String codigo; = 5

```

```
private Date fecha; = 30
private String lugarDeCelebracion; =30
private Persona contribuyente1; = 10
private Persona contribuyente2; = 10
private Persona testigo1; = 10
private Persona testigo2; = 10
private Persona juez; = 10

total = 131

*/
private static ControladorMatrimonio instancia;

public static ControladorMatrimonio getInstance() {

    if (instancia == null) {

        instancia = new ControladorMatrimonio();

    }
    return instancia;
}

private RandomAccessFile archivo;
private int salto;
private int tamanoRegistro;

public ControladorMatrimonio() {

    try {

        archivo = new RandomAccessFile("usaurio4.txt", "rw");

    } catch (IOException ex) {

        ex.printStackTrace();

    }

}

@Override
public void create(Matrimonio objeto) {

    salto = 0;

    try {

        archivo.seek(salto);
        archivo.writeUTF(objeto.getCodigo());
        archivo.writeUTF(objeto.getFecha() + "");
        archivo.writeUTF(objeto.getLugarDeCelebracion());
        archivo.writeUTF(objeto.getContribuyente1().getCedula());
        archivo.writeUTF(objeto.getContribuyente2().getCedula());
        archivo.writeUTF(objeto.getTestigo1().getCedula());
        archivo.writeUTF(objeto.getTestigo2().getCedula());
        archivo.writeUTF(objeto.getJuez().getCedula());

    } catch (IOException ex) {

        System.out.println("Error de lectura y escritura");
    }
}
```

```

        ex.printStackTrace();

    }

}

public Persona contribuyente(String codigo) {

    ControladorPersona controladorPersona = ControladorPersona.getInstance();

    Persona persona = controladorPersona.read(codigo);

    return persona;

}

public Persona Testigo(String cedula) {

    ControladorTestigos controladorTestigo = ControladorTestigos.getInstance();

    Persona testigo = controladorTestigo.read(cedula);

    return testigo;

}

public Persona Juezz(String cedula) {

    ControladorJuez controladorJuez = ControladorJuez.getInstance();

    Persona juez = controladorJuez.read(cedula);

    return juez;

}

@Override
public Matrimonio read(String codigo) {

    salto = 0;
    tamañoRegistro = 131;

    try {

        if (salto < archivo.length()) {

            archivo.seek(salto);

            String codigoArchivo = archivo.readUTF();

            if (codigo.equals(codigoArchivo)) {

                String fecha = archivo.readUTF();
                String lugarDeCeremonia = archivo.readUTF();
                String contribuyente1 = archivo.readUTF();
                String contribuyente2 = archivo.readUTF();
                String testigo1 = archivo.readUTF();
                String testigo2 = archivo.readUTF();
            }
        }
    }
}

```

```
String juez = archivo.readUTF();

int dia = Integer.parseInt(archivo.substring(7, 10));
int anio = Integer.parseInt(archivo.substring(24, 28));
int mes = Integer.parseInt(archivo.substring(3, 7));

Date date = new Date(anio - 1900, mes, dia);

Matrimonio matrimonio = new matrimonio(codigo, date, lugarDeCele-
remonia,
                                contribuyente(contribuyente1), contribuyente(contribu-
yente2),
                                Testigo(testigo1), Testigo(testigo2), Juez(juez));

return matrimonio;

}

salto += tamañoRegistro;

}

} catch (IOException ex) {

    System.out.println("Error de lectura y escritura");
    ex.printStackTrace();

}

return null;

}

@Override
public boolean update(Matrimonio objeto) {

    salto = 0;
    tamañoRegistro = 131;

    try {

        if (salto < archivo.length()) {

            archivo.seek(salto);
            archivo.writeUTF(objeto.getFecha() + "");
            archivo.writeUTF(objeto.getLugarDeCelebracion());
            archivo.writeUTF(objeto.getContribuyente1() + "");
            archivo.writeUTF(objeto.getContribuyente2() + "");
            archivo.writeUTF(objeto.getJuez() + "");
            archivo.writeUTF(objeto.getTestigo1() + "");
            archivo.writeUTF(objeto.getTestigo2() + "");

        }

        salto += tamañoRegistro;

    } catch (IOException ex) {

        ex.printStackTrace();

    }

}
```

```

        return false;

    }

    @Override
    public boolean delete(Matrimonio objeto) {

        try {

            salto = 0;
            tamanoRegistro = 131;
            String cadena = "";

            if (salto < archivo.length()) {

                archivo.seek(salto);
                archivo.writeUTF(String.format("%-" + 5 + "s", cadena));
                archivo.writeUTF(String.format("%-" + 30 + "s", cadena));
                archivo.writeUTF(String.format("%-" + 30 + "s", cadena));
                archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
                archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
                archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
                archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
                archivo.writeUTF(String.format("%-" + 10 + "s", cadena));

            }

            salto += tamanoRegistro;

        } catch (IOException ex) {

            System.out.println("Error de lectura y escritura.");
            ex.printStackTrace();

        }

        return false;

    }

    @Override
    public List<Matrimonio> lista() {

        List<Persona> lista = new ArrayList<Persona>();
        int salto = 0;
        int registro = 131;
        try {
            while (salto < archivo.length()) {
                archivo.seek(salto);

                String cedula = archivo.readUTF().trim();
                if (!cedula.equals("")) {

                    String fecha = archivo.readUTF().trim();
                    String lugarCelebracion = archivo.readUTF().trim();
                    String contribuyente1 = archivo.readUTF().trim();
                    String contribuyente2 = archivo.readUTF().trim();
                    String testigo1 = archivo.readUTF().trim();
                    String testigo2 = archivo.readUTF().trim();
                    String juez = archivo.readUTF().trim();

```



```

        int dia = Integer.parseInt(fecha.substring(7,10));
        int anio = Integer.parseInt(fecha.substring(24,28));
        int mes = Integer.parseInt(fecha.substring(3,7));

        Date date = new Date(anio-1900,mes,dia);

        Matrimonio matrimonio = new Matrimonio(cedula, date, lugarCele-
bracion,
                                contribuyente(contribuyente1), contribuyente(contribu-
yente2), Testigo(testigo1),
                                Testigo(testigo2), Juezz(juez));

    }

    salto += registro;

}

} catch (IOException ex) {
    System.out.println("Error lectura escritura (UsuarioDao:Update)");
    ex.printStackTrace();
}

return null;
}

public int recuperarMes(String mes){

    String meses[] =
{"Jan","Feb","Mar","April","May","June","Jul","Aug","Sep","Oct","Nov","Dec"};

    for (int i = 0; i < 10; i++) {

        if (meses[i].equals(mes)) {

            return i;

        }

    }

    return 0;

}

}

```

Controlador Persona

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.controlador;

```

```

import ec.upse.edu.modelo.Persona;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 *
 * @author santi
 */
public class ControladorPersona extends ControladorAbstracto<Persona> {

    /*
    private String cedula; = 10
    private String nombre; = 15
    private String apellido; = 15
    private String direccion; = 50
    private String genero; = 10
    private Date fecha; = 30

    total registro = 142.
    */

    public int recuperarMes(String mes){

        String meses[] =
{"Jan", "Feb", "Mar", "April", "May", "June", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

        for (int i = 0; i < 10; i++) {

            if (meses[i].equals(mes)) {

                return i;

            }

        }

        return 0;

    }

    private static ControladorPersona instancia;

    public static ControladorPersona getInstance() {

        if (instancia == null) {

            instancia = new ControladorPersona();

        }

        return instancia;

    }

    private RandomAccessFile archivo;
    private int salto;

```

```
private int tamanoRegistro;

public ControladorPersona() {

    try {

        archivo = new RandomAccessFile("qusaurio3.txt", "rw");

    } catch (IOException ex) {

        ex.printStackTrace();

    }

}

@Override
public void create(Persona objeto) {

    try {

        archivo.seek(salto);
        archivo.writeUTF(objeto.getCedula());
        archivo.writeUTF(objeto.getNombre());
        archivo.writeUTF(objeto.getApellido());
        archivo.writeUTF(objeto.getDireccion());
        archivo.writeUTF(objeto.getGenero());
        archivo.writeUTF(objeto.getFecha() + "");

    } catch (IOException ex) {

        System.out.println("Error de lectura y escritura");
        ex.printStackTrace();

    }

}

@Override
public Persona read(String cedula) {

    salto = 0;
    tamanoRegistro = 142;

    try {

        if (salto < archivo.length()) {

            archivo.seek(salto);
            String cedulaArchivo = archivo.readUTF();

            if (cedula.equals(cedulaArchivo)) {

                String nombre = archivo.readUTF().trim();
                String apellido = archivo.readUTF().trim();
                String direccion = archivo.readUTF().trim();
                String genero = archivo.readUTF().trim();
                String fecha = archivo.readUTF().trim();

                int dia = Integer.parseInt(fecha.substring(8,10));
                int anio = Integer.parseInt(fecha.substring(24,28));
```

```

        int mes = recuperarMes(fecha.substring(4,7));

        Date date = new Date(anio-1900,mes,dia);

        Persona persona = new Persona(cedula, nombre, apellido, direccion, genero,date);

        return persona;
    }

    salto += tamanoRegistro;

}

} catch (IOException ex) {

    System.out.println("Error de lectura y escritura");
    ex.printStackTrace();

}

return null;

}

@Override
public boolean update(Persona objeto) {

    salto = 0;
    tamanoRegistro = 142;

    try {

        if (salto < archivo.length()) {

            archivo.seek(salto);
            archivo.writeUTF(objeto.getCedula());
            archivo.writeUTF(objeto.getNombre());
            archivo.writeUTF(objeto.getApellido());
            archivo.writeUTF(objeto.getDireccion());
            archivo.writeUTF(objeto.getGenero());
            archivo.writeUTF(objeto.getFecha() + "");

        }

        salto += tamanoRegistro;

    } catch (IOException ex) {

        ex.printStackTrace();

    }

    return false;

}

@Override
public boolean delete(Persona objeto) {

```

```

try {

    salto = 0;
    tamañoRegistro = 142;
    String cadena = "";

    if (salto < archivo.length()) {

        archivo.seek(salto);
        archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
        archivo.writeUTF(String.format("%-" + 15 + "s", cadena));
        archivo.writeUTF(String.format("%-" + 15 + "s", cadena));
        archivo.writeUTF(String.format("%-" + 50 + "s", cadena));
        archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
        archivo.writeUTF(String.format("%-" + 30 + "s", cadena));

    }

    salto += tamañoRegistro;

} catch (IOException ex) {

    System.out.println("Error de lectura y escritura.");
    ex.printStackTrace();

}

return false;

}

@Override
public List<Persona> lista() {

    List<Persona> lista = new ArrayList<Persona>();
    int salto = 0;
    int registro = 142;
    try {
        while (salto < archivo.length()) {
            archivo.seek(salto);

            String cedula = archivo.readUTF().trim();
            if (!cedula.equals("")) {

                String nombre = archivo.readUTF().trim();
                String apellido = archivo.readUTF().trim();
                String direccion = archivo.readUTF().trim();
                String genero = archivo.readUTF().trim();
                String fecha = archivo.readUTF().trim();

                int dia = Integer.parseInt(fecha.substring(8,10));
                int anio = Integer.parseInt(fecha.substring(24,28));
                int mes = recuperarMes(fecha.substring(4,7));

                Date date = new Date(anio-1900,mes,dia);

                Persona persona = new Persona(cedula, nombre, apellido, direc-
cion, genero,date);
                lista.add(persona);

            }
        }
    }
}

```

```

        salto += registro;

    }

    } catch (IOException ex) {
        System.out.println("Error lectura escritura (UsuarioDao:Update)");
        ex.printStackTrace();
    }
    return lista;
}

}

```

Controlador Testigos

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.controlador;

import ec.ups.edu.modelo.Persona;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 *
 * @author santi
 */
public class ControladorTestigos extends ControladorAbstracto<Persona> {

    /*
    private String cedula; = 10
    private String nombre; = 15
    private String apellido; = 15
    private String direccion; = 50
    private String genero; = 10
    private Date fecha; = 30

    total registro = 142.
    */

    public int recuperarMes(String mes){

        String meses[] =
{"Jan", "Feb", "Mar", "April", "May", "June", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

        for (int i = 0; i < 10; i++) {

            if (meses[i].equals(mes)) {

                return i;

            }

        }
    }
}

```

```
}

return 0;

}

private static ControladorTestigos instancia;

public static ControladorTestigos getInstance() {

    if (instancia == null) {

        instancia = new ControladorTestigos();

    }
    return instancia;
}

private RandomAccessFile archivo;
private int salto;
private int tamanoRegistro;

public ControladorTestigos() {

    try {

        archivo = new RandomAccessFile("usaurio2.txt", "rw");

    } catch (IOException ex) {

        ex.printStackTrace();

    }

}

@Override
public void create(Persona objeto) {

    salto = 0;

    try {

        archivo.seek(salto);
        archivo.writeUTF(objeto.getCedula());
        archivo.writeUTF(objeto.getNombre());
        archivo.writeUTF(objeto.getApellido());
        archivo.writeUTF(objeto.getDireccion());
        archivo.writeUTF(objeto.getGenero());
        archivo.writeUTF(objeto.getFecha() + "");

    } catch (IOException ex) {

        System.out.println("Error de lectura y escritura");
        ex.printStackTrace();

    }

}

@Override
```

```

public Persona read(String cedula) {

    salto = 0;
    tamanoRegistro = 142;

    try {

        if (salto < archivo.length()) {

            archivo.seek(salto);
            String cedulaArchivo = archivo.readUTF();

            if (cedula.equals(cedulaArchivo)) {

                String nombre = archivo.readUTF().trim();
                String apellido = archivo.readUTF().trim();
                String direccion = archivo.readUTF().trim();
                String genero = archivo.readUTF().trim();
                String fecha = archivo.readUTF().trim();

                int dia = Integer.parseInt(fecha.substring(8,10));
                int anio = Integer.parseInt(fecha.substring(24,28));
                int mes = recuperarMes(fecha.substring(4,7));

                Date date = new Date(anio-1900,mes,dia);

                Persona persona = new Persona(cedula, nombre, apellido, direccion, genero, date);

                return persona;

            }

            salto += tamanoRegistro;

        }

    } catch (IOException ex) {

        System.out.println("Error de lectura y escritura");
        ex.printStackTrace();

    }

    return null;

}

@Override
public boolean update(Persona objeto) {

    salto = 0;
    tamanoRegistro = 142;

    try {

        if (salto < archivo.length()) {

            archivo.seek(salto);
            archivo.writeUTF(objeto.getCedula());

```



```
        archivo.writeUTF(objeto.getNombre());
        archivo.writeUTF(objeto.getApellido());
        archivo.writeUTF(objeto.getDireccion());
        archivo.writeUTF(objeto.getGenero());
        archivo.writeUTF(objeto.getFecha() + "");

    }

    salto += tamañoRegistro;

} catch (IOException ex) {

    ex.printStackTrace();

}

return false;

}

@Override
public boolean delete(Persona objeto) {

    try {

        salto = 0;
        tamañoRegistro = 142;
        String cadena = "";

        if (salto < archivo.length()) {

            archivo.seek(salto);
            archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
            archivo.writeUTF(String.format("%-" + 15 + "s", cadena));
            archivo.writeUTF(String.format("%-" + 15 + "s", cadena));
            archivo.writeUTF(String.format("%-" + 50 + "s", cadena));
            archivo.writeUTF(String.format("%-" + 10 + "s", cadena));
            archivo.writeUTF(String.format("%-" + 30 + "s", cadena));

        }

        salto += tamañoRegistro;

    } catch (IOException ex) {

        System.out.println("Error de lectura y escritura.");
        ex.printStackTrace();

    }

    return false;

}

@Override
public List<Persona> lista() {

    List<Persona> lista = new ArrayList<Persona>();
    int salto = 0;
```

```

        int registro = 142;
        try {
            while (salto < archivo.length()) {
                archivo.seek(salto);

                String cedula = archivo.readUTF().trim();
                if (!cedula.equals("")) {

                    String nombre = archivo.readUTF().trim();
                    String apellido = archivo.readUTF().trim();
                    String direccion = archivo.readUTF().trim();
                    String genero = archivo.readUTF().trim();
                    String fecha = archivo.readUTF().trim();

                    int dia = Integer.parseInt(fecha.substring(8,10));
                    int anio = Integer.parseInt(fecha.substring(24,28));
                    int mes = recuperarMes(fecha.substring(4,7));

                    Date date = new Date(anio-1900,mes,dia);

                    Persona persona = new Persona(cedula, nombre, apellido, direccion, genero, date);
                    lista.add(persona);
                }

                salto += registro;
            }

        } catch (IOException ex) {
            System.out.println("Error lectura escritura (UsuarioDao:Update)");
            ex.printStackTrace();
        }
        return lista;
    }
}

```

Controlador Usuarios

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.controlador;

import ec.ups.edu.modelo.Usuario;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author santi
 */
public class ControladorUsuario extends ControladorAbstracto<Usuario> {

```

```
//Aplicando Singleton.
private static ControladorUsuario instancia;

public static ControladorUsuario getInstance() {

    if (instancia == null) {

        instancia = new ControladorUsuario();

    }
    return instancia;
}

private RandomAccessFile archivo;
int salto;

public ControladorUsuario() {

    try {
        archivo = new RandomAccessFile("usuario1.txt", "rw");
    } catch (FileNotFoundException ex) {
        Logger.getLogger(ControladorUsuario.class.getName()).log(Level.SEVERE,
null, ex);
    }

}

@Override
public void create(Usuario objeto) {

    salto = 0;

    try {

        archivo.seek(salto);
        archivo.writeUTF(objeto.getUsuario());
        archivo.writeUTF(objeto.getContraseña());

    } catch (IOException ex) {

        System.out.println("Error de lectura y escritura");
        ex.printStackTrace();

    }

}

@Override
public Usuario read(String cedula) {
    throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
}

@Override
public boolean update(Usuario objeto) {
    throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
}

@Override
public boolean delete(Usuario objeto) {
```

```

        throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
    }

    @Override
    public List<Usuario> lista() {
        throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
    }

    public Usuario login(String correo, String contraseña) {

        int salto = 0;
        int registro = 20;
        try {
            while (salto < archivo.length()) {
                archivo.seek(salto);
                String correoArchivo = archivo.readUTF();
                String contraseñaArchivo = archivo.readUTF();

                if (correo.equals(correoArchivo.trim()) && contraseña.equals(con-
traseñaArchivo.trim())) {
                    archivo.seek(salto);
                    Usuario usuario = new Usuario(correoArchivo, contraseñaArchivo);
                    return usuario;
                }
                salto += registro;
            }
        } catch (IOException ex) {
            System.out.println("Error lectura escritura (UsuarioDao:login)");
            ex.printStackTrace();
        }
        return null;
    }
}

```

- Paquete Modelo

Clase Matrimonio

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.modelo;

import java.util.Date;
import java.util.Objects;

/**
 *
 * @author santi
 */
public class Matrimonio {

    private String codigo;
    private Date fecha;
    private String lugarDeCelebracion;

```

```
private Persona contribuyente1;
private Persona contribuyente2;
private Persona testigo1;
private Persona testigo2;
private Persona juez;

public Matrimonio(String codigo, Date fecha, String lugarDeCelebracion,
    Persona contribuyente1, Persona contribuyente2, Persona testigo1,
    Persona testigo2, Persona juez) {

    this.codigo = codigo;
    this.fecha = fecha;
    this.lugarDeCelebracion = lugarDeCelebracion;
    this.contribuyente1 = contribuyente1;
    this.contribuyente2 = contribuyente2;
    this.testigo1 = testigo1;
    this.testigo2 = testigo2;
    this.juez = juez;
}

public String validarEspacios(String cadena, int numero) {
    if (cadena.length() == numero) {
        return cadena;
    } else {
        if (cadena.length() > numero) {
            cadena = cadena.substring(0, numero);
            return cadena;
        } else {
            for (int i = cadena.length(); i < numero; i++) {
                cadena += " ";
            }
            return cadena;
        }
    }
}

/*
private String codigo; = 5
private Date fecha; = 30
private String lugarDeCelebracion; =30
private Persona contribuyente1; = 10
private Persona contribuyente2; = 10
private Persona testigo1; = 10
private Persona testigo2; = 10
private Persona juez; = 10

total = 131
*/

public String getCodigo() {
    return codigo;
}

public void setCodigo(String codigo) {

    this.codigo = validarEspacios(codigo, 5);
}
```

```
}

public Date getFecha() {
    return fecha;
}

public void setFecha(Date fecha) {
    this.fecha = fecha;
}

public String getLugarDeCelebracion() {
    return lugarDeCelebracion;
}

public void setLugarDeCelebracion(String lugarDeCelebracion) {

    this.lugarDeCelebracion = validarEspacios(lugarDeCelebracion, 5);

}

public Persona getContribuyente1() {
    return contribuyente1;
}

public void setContribuyente1(Persona contribuyente1) {
    this.contribuyente1 = contribuyente1;
}

public Persona getContribuyente2() {
    return contribuyente2;
}

public void setContribuyente2(Persona contribuyente2) {
    this.contribuyente2 = contribuyente2;
}

public Persona getTestigo1() {
    return testigo1;
}

public void setTestigo1(Persona testigo1) {
    this.testigo1 = testigo1;
}

public Persona getTestigo2() {
    return testigo2;
}

public void setTestigo2(Persona testigo2) {
    this.testigo2 = testigo2;
}

public Persona getJuez() {
    return juez;
}

public void setJuez(Persona juez) {
    this.juez = juez;
}
```

```
@Override
public int hashCode() {
    int hash = 3;
    hash = 37 * hash + Objects.hashCode(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 Matrimonio other = (Matrimonio) obj;
    if (!Objects.equals(this.codigo, other.codigo)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Matrimonio{" + "codigo=" + codigo + ", fecha=" + fecha + ", lugarDe-
Celebracion=" + lugarDeCelebracion + ", contribuyente1=" + contribuyente1 + ", con-
tribuyente2=" + contribuyente2 + ", testigo1=" + testigo1 + ", testigo2=" + testigo2
+ ", juez=" + juez + '}';
}
}
```

Clase Persona

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.modelo;

import java.util.Date;
import java.util.Objects;

/**
 *
 * @author santi
 */
public class Persona {

    private String cedula;
    private String nombre;
    private String apellido;
    private String direccion;
    private String genero;
    private Date fecha;
```

```

public Persona(String cedula) {

    this.cedula = cedula;
    fecha = new Date();

}

public Persona(String cedula, String nombre, String apellido, String direccion,
    String genero, Date fecha) {

    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.direccion = direccion;
    this.genero = genero;
    this.fecha = fecha;

}

/*
private String cedula; = 10
private String nombre; = 15
private String apellido; = 15
private String direccion; = 50
private String genero; = 10
private Date fecha; = 30

total registro = 142.
*/

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {

    this.cedula = validarEspacios(cedula, 10);

}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {

    this.nombre = validarEspacios(nombre, 15);

}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {

    this.apellido = validarEspacios(apellido, 15);

}

```



```
public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {

    this.direccion = validarEspacios(direccion, 50);

}

public String getGenero() {
    return genero;
}

public void setGenero(String genero) {

    this.genero = validarEspacios(genero, 10);

}

public Date getFecha() {
    return fecha;
}

public void setFecha(Date fecha) {
    this.fecha = fecha;
}

public String validarEspacios(String cadena, int numero) {
    if (cadena.length() == numero) {
        return cadena;
    } else {
        if (cadena.length() > numero) {
            cadena = cadena.substring(0, numero);
            return cadena;
        } else {
            for (int i = cadena.length(); i < numero; i++) {
                cadena += " ";
            }
            return cadena;
        }
    }
}

@Override
public int hashCode() {
    int hash = 7;
    hash = 79 * 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 Persona other = (Persona) obj;
    if (!Objects.equals(this.cedula, other.cedula)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Persona{" + "cedula=" + cedula + ", nombre=" + nombre + ", ape-
llido=" + apellido + ", direccion=" + direccion + ", genero=" + genero + '\'';
}
}

```

Clase usuario

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.upse.edu.modelo;

import java.util.Objects;

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

    private String usuario;
    private String contrasena;

    public Usuario() {
    }

    public Usuario(String usuario, String contrasena) {
        this.usuario = usuario;
        this.contrasena = contrasena;
    }

    public String getUsuario() {
        return usuario;
    }

    public void setUsuario(String usuario) {

        this.usuario = validarEspacios(usuario, 10);

    }

    public String getContrasena() {
        return contrasena;
    }
}

```

```
}

public void setContraseña(String contraseña) {

    this.contraseña = validarEspacios(contraseña, 10);

}

public String validarEspacios(String cadena, int numero) {
    if (cadena.length() == numero) {
        return cadena;
    } else {
        if (cadena.length() > numero) {
            cadena = cadena.substring(0, numero);
            return cadena;
        } else {
            for (int i = cadena.length(); i < numero; i++) {
                cadena += " ";
            }
            return cadena;
        }
    }
}

@Override
public int hashCode() {
    int hash = 5;
    hash = 71 * hash + Objects.hashCode(this.usuario);
    hash = 71 * hash + Objects.hashCode(this.contraseña);
    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.usuario, other.usuario)) {
        return false;
    }
    if (!Objects.equals(this.contraseña, other.contraseña)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Usuario{" + "correo=" + usuario + ", contraseña=" + contraseña +
    '}';
}
```

}

}

RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones gráficas en sistemas.
- Los estudiantes están en la capacidad de implementar la persistencia en archivos.

RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la prueba.**

Nombre de estudiante: Jorge Santiago Cabrera Arias.

Firma de estudiante:



Escaneado con CamScanner