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

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

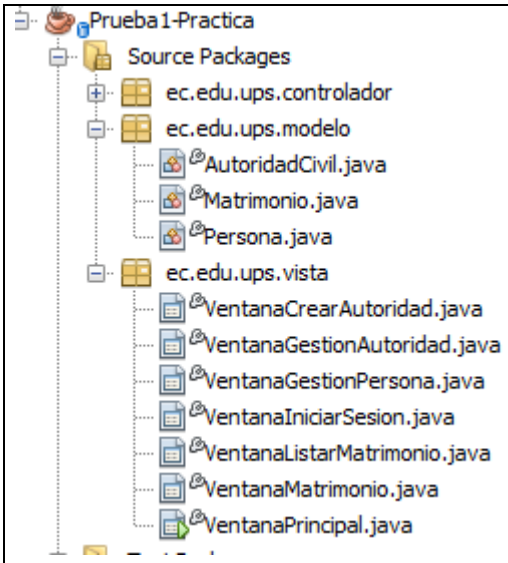
Igualmente se guardar los datos personales de los dos testigos y de la autoridad civil (juez o autoridad) que formalizan el acto. Además de gestionar la seguridad a través de un sistema de Usuarios y Autentificación.

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

2. Informe de Actividades:

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

Para la resolución del problema anteriormente planteado se ha creado el siguiente proyecto:



Paquete Controladores:

Controlador:

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

import java.io.*;
import java.util.List;
import java.util.Optional;


/**
 *
 * @author Usuario
 */
public abstract class Controlador<E>{
    private List<E> listaG;

    public Controlador() {
    }

    public Controlador(List<E> listaG) {
        this.listaG = listaG;
    }
    public List<E> getListaG() {
        return listaG;
    }

    public void setListaG(List<E> listaG) {
        this.listaG = listaG;
    }
```

```java
public boolean crear(E objeto) {

    if (validar(objeto) == true) {
        return listaG.add(objeto);
    }
    return false;

}
public abstract boolean validar(E objeto);

public abstract int generarId();

public Optional<E> buscar(E comparar) {
    return listaG.stream().filter(objeto -> objeto.equals(comparar)).findFirst();

}

public int posicion(E objetoC) {
    for (int i = 0; i < listaG.size() ; i++) {
        E objetoL = listaG.get(i);
        if (objetoL.equals(objetoC)) {
            return i;
        }
    }
    return -1;

}
```

```java
public boolean eliminar(E objeto) {
    Optional<E> buscar = buscar(objeto);
    E objetoE = buscar.get();
    if (objetoE != null) {
        System.out.println("Verdadero");
        return listaG.remove(objetoE);

    }
    System.out.println("Falso");
    return false;

}

public boolean actualizar(E objetoA) {
    int pos = posicion(objetoA);
    if (pos >=0) {
        listaG.set(pos, objetoA);
        System.out.println("True");
        return true;

    }
    System.out.println("false");
    return false;

}

public void cargarDatos(String ruta)throws FileNotFoundException, IOException, ClassNotFoundException{
    FileInputStream archivo = new FileInputStream(ruta);
    ObjectInputStream datos = new ObjectInputStream(archivo);
    listaG= (List<E>) datos.readObject();
}
public void guardarDatos(String ruta)throws FileNotFoundException, IOException, ClassNotFoundException{
    FileOutputStream archivo= new FileOutputStream(ruta);
    ObjectOutputStream datos = new ObjectOutputStream(archivo);
    datos.writeObject(listaG);
}
public List<E> getLista() {
    return listaG;
}

public void setLista(List<E> lista) {
    this.listaG= lista;
}
```

ControladorAutoridadCivil:

```java
public class ControladorAutoridadCivil extends Controlador<AutoridadCivil> {
  private AutoridadCivil autoridadCivil;
    @Override
    public boolean validar(AutoridadCivil objeto) {
        if(objeto.getTipo().equals("Autoridad")){
        return true;
        }
        return false;

    }

    @Override
    public int generarId() {
        return 0;

    }
     public boolean iniciarSesion(String correo, String contrasenia) {

        for (AutoridadCivil usuario : super.getListaG()) {
            AutoridadCivil u = (AutoridadCivil) usuario;
            if (u.getCorreo().equals(correo) && u.getContrasenia().equals(contrasenia)) {
                this.autoridadCivil = u;
                return true;
            }

        }
        return false;
    }

  public AutoridadCivil getAutoridad() {
      return autoridadCivil;
  }

  public void setAutoridad(AutoridadCivil autoridadCivil) {
      this.autoridadCivil = autoridadCivil;
  }
```

Controlador Matrimonio:

```java
public class ControladorMatrimonio extends Controlador<Matrimonio> {

    @Override
    public boolean validar(Matrimonio objeto) {
        if(objeto.getContrayente1().getTipo().equals("Contrayente: ")&&
                objeto.getContrayente2().getTipo().equals("Contrayente: ")){
            if(objeto.getContrayente1().getEstadoCivil()!="Casado"&&
                    objeto.getContrayente2().getEstadoCivil()!="Casado"){
            return true;
            }
        }
        return false;
    }

    @Override
    public int generarId() {
         List<Matrimonio> tm = new ArrayList();
        for ( Matrimonio matrimonio : super.getListaG()) {
            Matrimonio m = (Matrimonio) matrimonio;
            tm.add(m);

        }

        if (tm.size() > 0 && tm != null) {
            return tm.get(tm.size() - 1).getCodigo() + 1;
        } else {
            return 1;
        }
    }

}
```

ControladorPersona:

```java
@Override
public int generarId() {
    return 0;

}
public List<Persona> personas() {

    List<Persona> listaP = new ArrayList();
    Persona persona;
    Iterator i = super.getLista().iterator();
    while (i.hasNext()) {
        persona = (Persona) i.next();
        listaP.add(persona);

    }
    return listaP;

}

public class ControladorPersona extends Controlador<Persona>{

    @Override
    public boolean validar(Persona objeto) {
        int suma = 0;
        String id = objeto.getCedula();
        if (id.length() == 9) {
            return false;
        } else {
            int a[] = new int[id.length() / 2];
            int b[] = new int[(id.length() / 2)];
            int c = 0;
            int d = 1;
            for (int i = 0; i < id.length() / 2; i++) {
                a[i] = Integer.parseInt(String.valueOf(id.charAt(c)));
                c = c + 2;
                if (i < (id.length() / 2) - 1) {
                    b[i] = Integer.parseInt(String.valueOf(id.charAt(d)));
                    d = d + 2;
                }
            }
```

```
            for (int i = 0; i < a.length; i++) {
                a[i] = a[i] * 2;
                if (a[i] > 9) {
                    a[i] = a[i] - 9;
                }
                suma = suma + a[i] + b[i];
            }
            int aux = suma / 10;
            int dec = (aux + 1) * 10;
            if ((dec - suma) == Integer.parseInt(String.valueOf(id.charAt(id.length() - 1)))) {
                return true;
            } else if (suma % 10 == 0 && id.charAt(id.length() - 1) == '0') {
                return true;
            } else {
                return false;
            }
        }

    }
```

Paquete Modelo:

AutoridadCivil:

```
public class AutoridadCivil extends Persona{

    private String cargo;
    private String correo;
    private String contrasenia;

    public AutoridadCivil(String cargo, String correo, String contrasenia,
            String nombre, String apellido, String cedula, String direccion,
            String genero, Date fechaNacimiento, String tipo, String estadoCivil) {
        super(nombre, apellido, cedula, direccion, genero, fechaNacimiento, tipo, estadoCivil);
        this.cargo = cargo;
        this.correo = correo;
        this.contrasenia = contrasenia;
    }



    public String getCargo() {
        return cargo;
    }

    public void setCargo(String cargo) {
        this.cargo = cargo;
    }
```

```
  public String getCorreo() {
      return correo;
  }

  public void setCorreo(String correo) {
      this.correo = correo;
  }

  public String getContrasenia() {
      return contrasenia;
  }

  public void setContrasenia(String contrasenia) {
      this.contrasenia = contrasenia;
  }
```

Matrimonio:

```
public class Matrimonio{
    private int codigo;
    private Date fecha;
    private String lugar;
    private Persona contrayentel;
    private Persona contrayente2;
    private Persona testigol;
    private Persona testigo2;
    private Persona autoridad;

    public Matrimonio() {
    }

    public Matrimonio(int codigo, Date fecha, String lugar,
            Persona contrayentel, Persona contrayente2, Persona testigol,
            Persona testigo2, Persona autoridad) {
        this.codigo = codigo;
        this.fecha = fecha;
        this.lugar = lugar;
        this.contrayentel = contrayentel;
        this.contrayente2 = contrayente2;
        this.testigol = testigol;
        this.testigo2 = testigo2;
        this.autoridad = autoridad;
    }


    public int getCodigo() {
        return codigo;
    }
```

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

public Date getFecha() {
    return fecha;
}

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

public String getLugar() {
    return lugar;
}

public void setLugar(String lugar) {
    this.lugar = lugar;
}

public Persona getContrayente1() {
    return contrayente1;
}

public void setContrayente1(Persona contrayente1) {
    this.contrayente1 = contrayente1;
}

public Persona getContrayente2() {
    return contrayente2;
}
```

```java
public void setContrayente2(Persona contrayente2) {
    this.contrayente2 = contrayente2;
}

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 getAutoridad() {
    return autoridad;
}

public void setAutoridad(Persona autoridad) {
    this.autoridad = autoridad;
}
```

```java
@Override
public int hashCode() {
    int hash = 7;
    hash = 79 * 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 Matrimonio other = (Matrimonio) obj;
    if (this.codigo != other.codigo) {
        return false;
    }
    return true;
}
```

Persona:

```java
public class Persona {
    private String nombre;
    private String apellido;
    private String cedula;
    private String direccion;
    private String genero;
    private Date fechaNacimiento;
    private String tipo;
    private String estadoCivil;

    public Persona(String nombre, String apellido, String cedula,
            String direccion, String genero, Date fechaNacimiento,
            String tipo, String estadoCivil) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.cedula = cedula;
        this.direccion = direccion;
        this.genero = genero;
        this.fechaNacimiento = fechaNacimiento;
        this.tipo = tipo;
        this.estadoCivil = estadoCivil;
    }

    public Persona(){

    }

    public String getNombre() {
        return nombre;
    }
```

```java
public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public String getCedula() {
    return cedula;
}

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

public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getGenero() {
    return genero;
}
```

```java
public void setGenero(String genero) {
    this.genero = genero;
}

public Date getFechaNacimiento() {
    return fechaNacimiento;
}

public void setFechaNacimiento(Date fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
}

public String getTipo() {
    return tipo;
}

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

public String getEstadoCivil() {
    return estadoCivil;
}

public void setEstadoCivil(String estadoCivil) {
    this.estadoCivil = estadoCivil;
}
```

```java
@Override
public int hashCode() {
    int hash = 7;
    hash = 11 * 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{" + "nombre=" + nombre + ", apellido=" + apellido +
            ", cedula=" + cedula + ", direccion=" + direccion +
            ", genero=" + genero + ", fechaNacimiento=" + fechaNacimiento +
            ", tipo=" + tipo + ", estadoCivil=" + estadoCivil + '}';
}
```

Paquete Vista:

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 graficas 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*:** Nelson Paul Ortega Segarra

***Firma de estudiante*:**