
	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 Autenticación.

Calificación:

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

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
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 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.

CARRERA: Computación

ASIGNATURA: Programación Aplicada

NRO. PRÁCTICA: 1

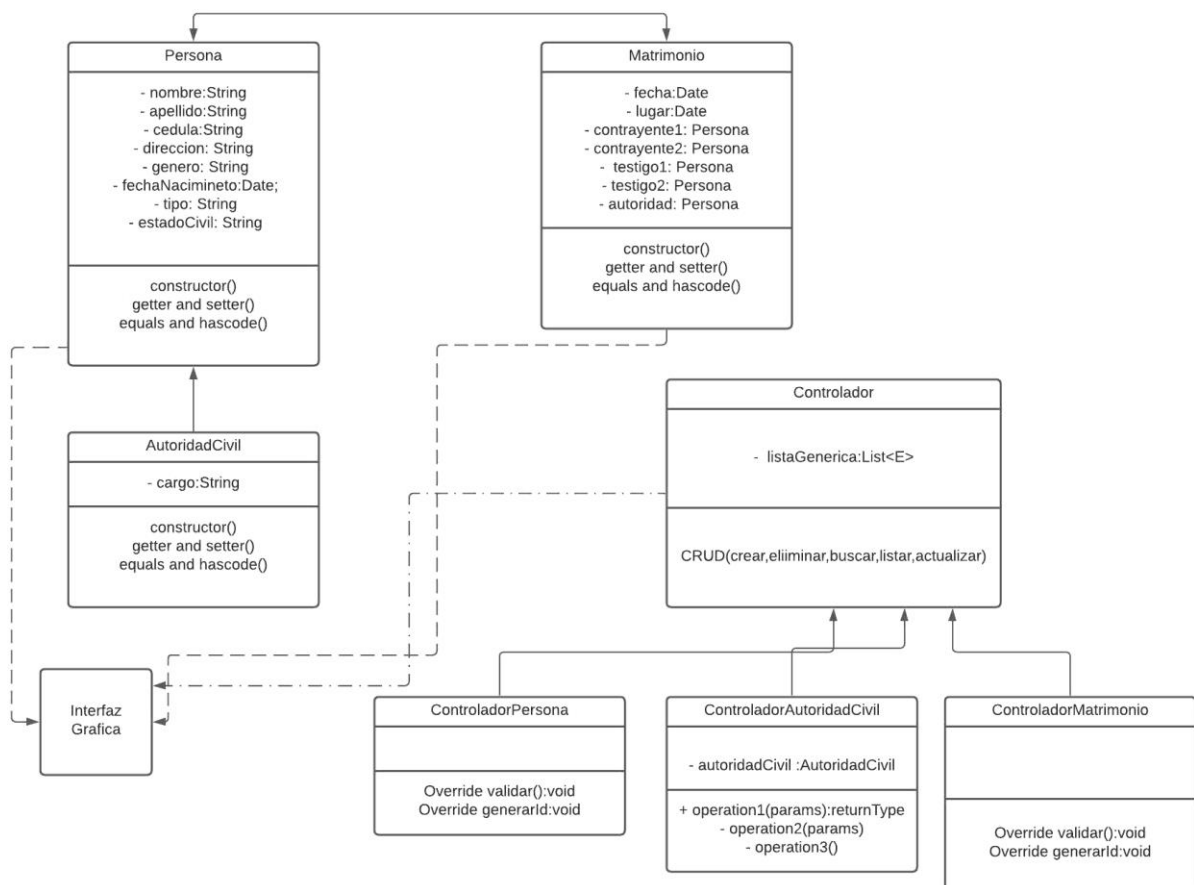
TÍTULO PRÁCTICA: Prueba Unidad 1

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. Realizar el diagrama de clase del sistema para gestionar los matrimonios de la ciudad de Cuenca



2. Programar el sistema para gestionar los matrimonios de la ciudad de Cuenca

Paquete `ec.edu.ups.modelo`

Clase Persona: La clase persona es la clase padre que va a heredar sus atributos y métodos a la clase Autoridad Civil, sin embargo posee un atributo “tipo” que nos permitirá definir si una persona va a ser contrayente o testigo en un matrimonio.

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

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

/**
 *
 * @author Anahi
 */
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) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.cedula = cedula;
        this.direccion = direccion;
        this.genero = genero;
        this.fechaNacimiento = fechaNacimiento;
    }

    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 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;
    }
}
```

```

}

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;
}

@Override
public int hashCode() {
    int hash = 7;
    hash = 31 * 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 + '}}';  
    }  
}
```

Clase AutoridadCivil

```
package ec.edu.upes.modelo;  
  
import java.util.Date;  
  
/**  
 *  
 * @author Anahi  
 */  
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) {  
        super(nombre, apellido, cedula, direccion, genero, fechaNacimiento);  
        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;  
    }  
}
```

Clase Matrimonio

```
package ec.edu.upse.modelo;

import java.util.Date;
import java.util.List;

/**
 *
 * @author Anahi
 */
public class Matrimonio{
    private int codigo;
    private Date fecha;
    private String lugar;
    private Persona contrayente1;
    private Persona contrayente2;
    private Persona testigo1;
    private Persona testigo2;
    private Persona autoridad;

    public Matrimonio() {
    }

    public int getCodigo() {
        return codigo;
    }

    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;
    }

    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;
    }

    @Override
    public int hashCode() {
        int hash = 5;
        hash = 59 * 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;
    }
}
```

Paquete ec.edu.ups.controlador

Clase abstracta Controlador

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

import java.io.*;
```

```

import java.util.List;
import java.util.Optional;

/**
 *
 * @author Anahi
 */
public abstract class Controlador<E>{
    private List<E> listaGenerica;

    public Controlador() {
    }

    public Controlador(List<E> listaGenerica) {
        this.listaGenerica = listaGenerica;
    }
    public List<E> getListaGenerica() {
        return listaGenerica;
    }

    public void setListaGenerica(List<E> listaGenerica) {
        this.listaGenerica = listaGenerica;
    }
    public boolean crear(E objeto) {

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

    public abstract boolean validar(E objeto);

    public abstract int generarId();

    public Optional<E> buscar(E comparar) {
        return listaGenerica.stream().filter(objeto -> objeto.equals(com-
parar)).findFirst();
    }

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

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

```

```
        return false;
    }

    public boolean actualizar(E objetoA) {
        int pos = posicion(objetoA);
        if (pos >= 0) {
            listaGenerica.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);
        listaGenerica = (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(listaGenerica);
    }
}
```

Clase ControladorPersona

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

import ec.edu.ups.modelo.Persona;

/**
 *
 * @author Anahi
 */
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;
    }
}

}

@Override
public int generarId() {
    return 0;
}

}

```

Clase ControladorAutoridadCivil

```

package ec.edu.ups.controlador;
import ec.edu.ups.modelo.AutoridadCivil;
/**
 *
 * @author Anahi
 */
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.getListGenerica()) {
            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;
}

}
```

Clase ControladorMatrimonio

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

import ec.edu.ups.modelo.Matrimonio;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Anahi
 */
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> temp = new ArrayList();
        for (Matrimonio matrimonio : super.getListGenerica()) {
            Matrimonio m = (Matrimonio) matrimonio;
            temp.add(m);
        }

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

}

}

Paquete ec.edu.ups.vista

Ventana Principal

Ventana Iniciar Sesión

Ventana Crear Autoridad

Ventana Crear Persona

Ventana Acta de Matrimonio

Ventana Gestión Persona

3. Patrón de Diseño aplicado

En la programación se aplicó el patrón de diseño Factory Method que nos permite crear un método estándar en una clase para crear objetos. Por lo que el método estándar utilizado fue "public abstract validar()" que fue creado en la clase abstracta Controlador para así validar varios ámbitos de la programación como es el del estado civil de una persona para que esta pueda contraer matrimonio.

RESULTADO(S) OBTENIDO(S):

Mediante un buen uso de los conocimientos adquiridos en clase el resultado del sistema para Matrimonios en el Registro Civil de Cuenca fue realizado con éxito y su funcionamiento es fácil y legible para los usuarios a los que va dirigido el sistema.

CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones gráficas en sistemas.

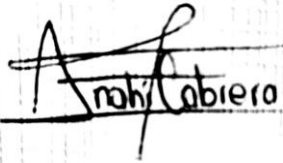
Los estudiantes están en la capacidad de implementar la persistencia en archivos.

RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.

Nombre de estudiante: Edith Anahí Cabrera Bermeo

Firma de estudiante:



Edith Cabrera