UNIVERSIDAD TECNICA DE AMBATO

INTEGRANTES:

- Michael Chávez
- David Giler
- Aracelly Guangasi
- Steeven Loor

SEMESTRE:

TERCERO

FACULTAD:

INGENIERIA EN SISTEMAS, ELECTRONICA E INDUSTRIAL

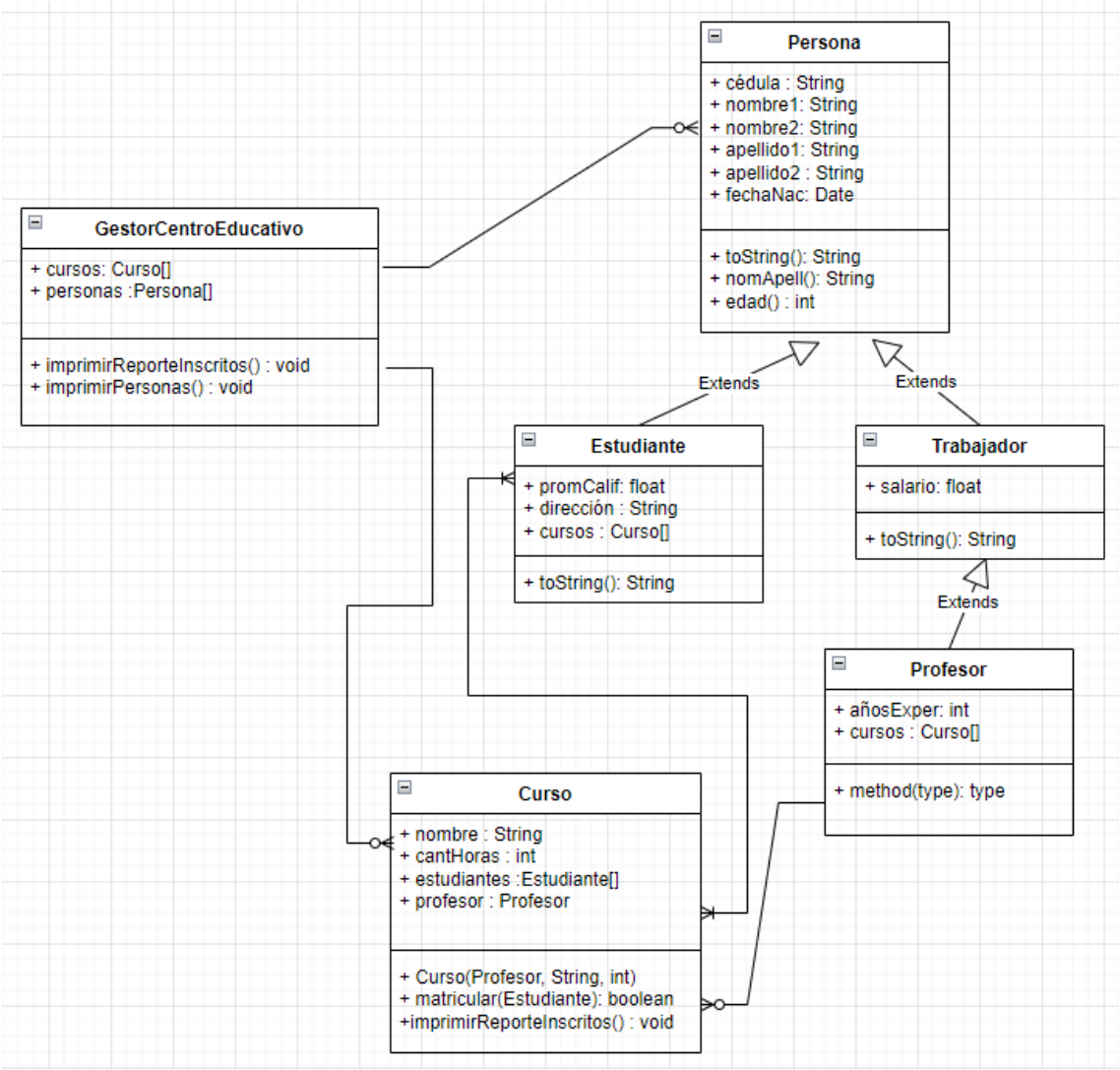
MATERIA:

Estructura de Datos

DOCENTE:

Ing. Fernández Peña Félix Oscar

DIAGRAMA DE CLASE



CLASES:

1. CURSO

```java
class Curso implements Comparable<Curso> {

    private String nombre;
    private int cantHoras;
    private Estudiante estudiantes[];
    private Profesor profesor;
    private int codigo;

    public Curso(String nombre, int cantHoras, Profesor profesor, int codigo, int cupo) {
        this.nombre = nombre;
        this.cantHoras = cantHoras;
        this.profesor = profesor;
        this.codigo = codigo;
        this.estudiantes = new Estudiante[cupo];
    }

    public int getCodigo() {
        return this.codigo;
    }

    public int getCupo() {
        return this.estudiantes.length;
    }

    public String getNombre() {
        return this.nombre;
    }

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

```java
    @Override
    public String toString() {
        return " Codigo de Curso = " + this.codigo + "\n Nombre del curso = " + this.nombre + "\n Cantidad de horas = "
                + this.cantHoras + "\n Profesor = " + this.profesor.nombreCompleto();
    }

    public int cantidadCuposDisponibles() {
        int cupolibre = 0;
        for (int i = 0; i < this.estudiantes.length; i++) {
            if (this.estudiantes[i] == null) {
                cupolibre++;
            }
        }
        return cupolibre;
    }

    @Override
    public int compareTo(Curso c) {
        if (c == null) {
            return 1;
        }
        return c.escojercupo(false) - this.escojercupo(false);
    }

    public int escojercupo(boolean op) {
        int cupolibre = 0;
        for (int i = 0; i < this.estudiantes.length; i++) {
            if (this.estudiantes[i] == null) {
                cupolibre++;
            }
        }
```

```java
    public int getCantHoras() {
        return this.cantHoras;
    }

    public void setCantHoras(int cantHoras) {
        this.cantHoras = cantHoras;
    }

    public Profesor getProfesor() {
        return this.profesor;
    }

    public void setProfesor(Profesor profesor) {
        this.profesor = profesor;
    }

    public Estudiante[] getEstudiantes() {
        return this.estudiantes;
    }

    public boolean matricularEstudianteCurso(Estudiante e) {
        if (e != null) {
            for (int i = 0; i < getEstudiantes().length; i++) {
                if (getEstudiantes()[i] == null) {
                    getEstudiantes()[i] = e;
                    return true;
                }
            }
        }
        return false;
    }
```

```
            if (op) {
                return cupolibre;
            } else {
                return this.estudiantes.length - cupolibre;
            }
        }
    }

```

La clase "**Curso**" es una de las clases principales ya que esta será usada para crear los distintos curso que un profesor puede dar a un determinado numero de estudiantes.

2. ESTUDIANTE

```
import java.time.LocalDate;

public class Estudiante extends Persona {

    private float promedioCalificaciones;
    private String direccion;
    private Curso cursos[];
    private final static int cantCursos = 5;

    public Estudiante(float promedioCalificaciones, String direccion, String cedula, String nombre1, String nombre2,
            String apellido1, String apellido2, LocalDate fechaNacimiento) {
        super(cedula, nombre1, nombre2, apellido1, apellido2, fechaNacimiento);
        this.promedioCalificaciones = promedioCalificaciones;
        this.direccion = direccion;
        this.cursos = new Curso[cantCursos];
    }

    public float getPromedioCalificaciones() {
        return promedioCalificaciones;
    }

    public void setPromedioCalificaciones(float promedioCalificaciones) {
        this.promedioCalificaciones = promedioCalificaciones;
    }

    public String getDireccion() {
        return direccion;
    }

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

    public Curso[] getCursos() {
        return cursos;
    }

    public void setCursos(Curso[] cursos) {
        this.cursos = cursos;
    }

    public boolean asignarCursoEstudiante(Curso c) {
        for (int j = 0; j < getCursos().length; j++) {
            if (getCursos()[j] == null) {
                getCursos()[j] = c;
                return true;
            }
        }
        return false;
    }

    @Override
    public String toString() {
        return super.toString() + "\nPromedio de Calificaciones = " + promedioCalificaciones + "\nDireccion = "
                + direccion;
    }
}
```

La clase "**Estudiante**" es la encargada de almacenar la información de los estudiantes que se irán implementando, además de sus cursos.

3. INSERTAR

```java
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class Insertar {

    private DateTimeFormatter formatoFecha;
    private Scanner tec;
    private Control_Validacion cv;
    private int cont;

    public Insertar() {
        this.formatoFecha = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        this.tec = new Scanner(System.in);
        this.cv = new Control_Validacion();
        this.cont = 0;
    }

    public Estudiante insertarDatosEstudiante(Estudiante estudiantes[]) {
        System.out.println("Ingrese cedula del estudiante");
        String cedula = tec.next();
        if (this.cv.comprobarEstudiante(cedula, estudiantes)) {
            System.out.println("Ingrese el primer nombre del Estudiante");
            String nombre1 = tec.next();
            System.out.println("Ingrese el segundo nombre del Estudiante");
            String nombre2 = tec.next();
            System.out.println("Ingrese el primer apellido del Estudiante");
            String apellido1 = tec.next();
            System.out.println("Ingrese el segundo apellido del Estudiante");
            String apellido2 = tec.next();
            System.out.println("Ingresa direccion");
            String direccion = tec.next();
            System.out.println("Ingrese la fecha de nacimiento (formato: 25/09/1990)");
            String fechaNacimiento = tec.next();
            LocalDate fecha = LocalDate.parse(fechaNacimiento, formatoFecha);
            float promedioAcademico = 0.0f;
            System.out.println("ESTUDIANTE REGISTRADO CON EXITO");
            return new Estudiante(promedioAcademico, direccion, cedula, nombre1, nombre2, apellido1, apellido2, fecha);
        } else {
            return null;
        }
    }

    public Curso insertarDatosCurso(GestorUnidadEducativa gue) {
        if (cv.existeElemento(gue.getProfesores())) {
            System.out.println("Ingrese nombre del curso");
            String nombreCurso = tec.next();
            System.out.println("Ingrese cantidad total de horas del curso");
            int horasTotales = tec.nextInt();
            System.out.println("Cuantos cupos disponibles existen");
            int cupos = tec.nextInt();
            System.out.println("Ingrese cedula del profesor a cargo del curso");
            String cedulaProfesor = tec.next();
            Profesor profesor = cv.validarProf(cedulaProfesor, gue.getProfesores());
            if (profesor == null) {
                System.out.println("HUBO UN ERROR, VERIFICA LOS DATOS");
                return null;
            } else {
                Curso c = new Curso(nombreCurso, horasTotales, profesor, this.cont, cupos);
                if (gue.asignarCursoProfesor(cedulaProfesor, c)) {
                    this.cont++;
                    System.out.println("CURSO CREADO CON EXITO");
                    return c;
                } else {
                    return null;
                }
            }
        } else {
            System.out.println("NO EXISTEN PROFESORES PARA CREAR UN CURSO");
        }
        return null;

    }

    public Profesor insertarDatosProfesor(Profesor profesores[]) {
        System.out.println("Ingrese su cedula");
        String cedulaProfesor = tec.next();
        if (this.cv.comprobarProfesor(cedulaProfesor, profesores)) {
            System.out.println("Ingrese el primer nombre");
            String nombre1 = tec.next();
            System.out.println("Ingresa el segundo nombre");
            String nombre2 = tec.next();
            System.out.println("Ingrese el primer apellido");
            String apellido1 = tec.next();
            System.out.println("Ingrese el segundo apellido");
            String apellido2 = tec.next();
            System.out.println("Ingrese la fecha de nacimiento (formato: 25/09/1990)");
            String fechaNacimiento = tec.next();
            LocalDate fecha = LocalDate.parse(fechaNacimiento, formatoFecha);
            System.out.println("Ingrese años de experiencia");
            int añosExperiencia = tec.nextInt();
            System.out.println("Ingrese el salario");
            float salario = tec.nextFloat();
            System.out.println("PROFESOR REGISTRADO CON EXITO");
            return new Profesor(añosExperiencia, salario, cedulaProfesor, nombre1, nombre2, apellido1, apellido2,
                    fecha);
```

```
        } else {
            return null;
        }
    }

    public void insertarDatosMatricEstu(GestorUnidadEducativa gue) {
        if (cv.existeElemento(gue.getEstudiantes())) {
            if (cv.existeElemento(gue.getCursos())) {
                int codCurso = gue.cursosDisponibles();
                for (Curso curso : gue.getCursos()) {
                    if (curso != null && curso.getCodigo() == codCurso) {
                        if (cv.existeCupos(curso)) {
                            System.out.println("Ingrese la cedula del estudiante");
                            String cedEstudiante = tec.next();
                            Estudiante estudiante = cv.validarEstud(cedEstudiante, gue.getEstudiantes());
                            if (estudiante != null && cv.comprobarEstudiante(cedEstudiante, curso.getEstudiantes())) {
                                if (gue.ingresoEstudiante(curso, estudiante)) {
                                    System.out.println("MATRICULADO CON EXITO");
                                } else {
                                    System.out.println("HUBO UN ERROR AL MATRICULAR");
                                    System.out.println("VERIFICA QUE TENGAS CUPOS PARA MATRICULAR");
                                }
                            }
                        }
                        break; // No es necesario seguir iterando después de encontrar el curso
                    }
                }
            } else {
                System.out.println("NO EXISTEN CURSOS DISPONIBLES");
            }
        } else {
            System.out.println("NO EXISTEN ALUMNOS DISPONIBLES");
        }
    }

}
```

La clase “**Insertar**” es una de las clases importantes, ya que aquí se crearán los métodos que servirán para insertar los datos necesarios para hacer funcionar el programa.

4. GESTOR UNIDAD EDUCATIVA

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

public class GestorUnidadEducativa {

    private static final int MAX_CANTIDAD = 100;
    private Estudiante[] estudiantes;
    private Profesor[] profesores;
    private Curso[] cursos;
    private Control_Validacion cv;

    public GestorUnidadEducativa() {
        this.estudiantes = new Estudiante[MAX_CANTIDAD];
        this.profesores = new Profesor[MAX_CANTIDAD];
        this.cursos = new Curso[MAX_CANTIDAD];
        this.cv = new Control_Validacion();
    }

    public Profesor[] getProfesores() {
        return this.profesores;
    }

    public Estudiante[] getEstudiantes() {
        return this.estudiantes;
    }

    public Curso[] getCursos() {
        return this.cursos;
    }

```

```java
    public void mostrarEstudiantes() {
        if (cv.existeElemento(this.estudiantes)) {
            System.out.println("-*-*-*-*-*-*-*-ESTUDIANTES-*-*-*-*-*-*-*-*-");
            for (Estudiante estudiante : this.estudiantes) {
                if (estudiante != null) {
                    System.out.println(estudiante.toString() + "\n");
                }
            }
        } else {
            System.out.println("NO EXISTEN ESTUDIANTES");
        }
    }

    public boolean exisProfesor() {
        for (int i = 0; i < this.profesores.length; i++) {
            if (this.profesores[i] != null) {
                return true;
            }
        }
        return false;
    }

    public void mostrarProfesores() {
        if (exisProfesor()) {
            System.out.println("-*-*-*-*-*-*-*-*-DOCENTES-*-*-*-*-*-*-*-*-");
            for (Profesor profesor : this.profesores) {
                if (profesor != null) {
                    System.out.println(profesor.toString() + "\n");
                }
            }
        } else {
            System.out.println("NO EXISTEN PROFESORES");
```

```java
        }
    }

    public void mostrarCursos() {
        int posicion = this.cursosDisponibles();
        if ((posicion) >= 0) {
            if (this.cursos[posicion] != null) {
                System.out.println(this.cursos[posicion].toString());
                System.out.println(" Cupos disponibles: " + this.cursos[posicion].escojercupo(true));
                System.out.println("Estudiantes registrados:");
                for (int i = 0; i < this.cursos[posicion].getEstudiantes().length; i++) {
                    if (this.cursos[posicion].getEstudiantes()[i] != null) {
                        System.out.println(i + ". " + this.cursos[posicion].getEstudiantes()[i].nombreCompleto());
                    }
                }
            } else {
                System.out.println("CURSO NO ENCONTRADO");
            }
        } else {
            System.out.println("NO HAY CURSOS DISPONIBLES");
        }
    }

    public void imprimirCursosOrdenados() {
        Arrays.sort(this.cursos, Comparator.nullsLast(Comparator.naturalOrder()));
        for (int i = 0; i < this.cursos.length; i++) {
            if (this.cursos[i] != null) {
                System.out.println(cursos[i].getNombre() + ": " + this.cursos[i].escojercupo(false));
            }
        }
    }

```

```java
    public boolean insertarObjeto(Object objeto, Object[] arreglo) {
        for (int i = 0; i < arreglo.length; i++) {
            if (arreglo[i] == null) {
                arreglo[i] = objeto;
                return true;
            }
        }
        return false;
    }

    public int cursosDisponibles() {
        Scanner tec = new Scanner(System.in);
        int opcion = -1;
        int conta = 0;
        Arrays.sort(this.cursos, Comparator.nullsLast(Comparator.naturalOrder()));
        for (int i = 0; i < this.cursos.length; i++) {
            if (this.cursos[i] != null) {
                conta++;
            }
        }
        if (conta > 0) {
            System.out.println("Escoja el codigo de los cursos disponibles");
            for (int i = 0; i < this.cursos.length; i++) {
                if (this.cursos[i] != null) {
                    System.out.println(this.cursos[i].getCodigo() + ". " + this.cursos[i].getNombre());
                }
            }
            opcion = tec.nextInt();
        }
        return opcion;
    }

```

```
    public boolean ingresoEstudiante(Curso c, Estudiante e) {
        if (c != null && e != null) {
            for (int i = 0; i < c.getEstudiantes().length; i++) {
                if (c.getEstudiantes()[i] == null) {
                    for (int j = 0; j < e.getCursos().length; j++) {
                        if (e.getCursos()[j] == null) {
                            c.matricularEstudianteCurso(e);
                            e.asignarCursoEstudiante(c);
                            return true;
                        }
                    }
                }
            }
        }
        return false;
    }

    public boolean asignarCursoProfesor(String ced, Curso c) {
        for (int i = 0; i < this.profesores.length; i++) {
            if (profesores[i] != null) {
                if (profesores[i].getCedula().equalsIgnoreCase(ced)) {
                    for (int j = 0; j < profesores[i].getCursos().length; j++) {
                        if (profesores[i].getCursos()[j] == null) {
                            profesores[i].asignarCurso(c);
                            return true;
                        } else {
                            System.out.println("ESTE PROFESOR NO TIENE DISPONIBLE MAS CURSOS");
                        }
                    }
                }
            }
        }
        return false;
    }

    public void ordenarCursosporInscrito(Curso cursos[]) {
        Arrays.sort(cursos);
    }
}
```

La clase "**GestorUnidadEducativa**" cuenta con los vectores que se necesitaran para poder gestionar todo el programa, entre ellos cuenta con el vector Estudiante, Profesor y Curso, también se implementaron métodos para realizar la impresión de los reportes.

5. TRABAJADOR

```
package proyecto_2;

import java.time.LocalDate;

public class Trabajador extends Persona {

    protected float salario;

    public Trabajador(float salario, String cedula, String nombre1, String nombre2, String apellido1, String apellido2,
            LocalDate fechaNacimiento) {
        super(cedula, nombre1, nombre2, apellido1, apellido2, fechaNacimiento);
        this.salario = salario;
    }

    @Override
    public String toString() {
        return String.format("%s\nSalario = %.2f\n", super.toString(), salario);
    }

}
```

La clase "**Trabajador**" cuenta con un atributo propio y 6 heredados de una clase padre llamada "**Persona**", en esta clase solo tendremos un constructor y un método toString para poder imprimir los datos necesarios.

6. PROFESOR

```java
package proyecto_2;

import java.time.LocalDate;

class Profesor extends Trabajador {

    private int añosExperiencia;
    private Curso cursos[];
    private static final int CantidadMaxCursos = 4;

    public Curso[] getCursos() {
        return this.cursos;
    }

    public Profesor(int añosExperiencia, float salario, String cedula, String nombre1, String nombre2, String apellido1,
            String apellido2, LocalDate fechaNacimiento) {
        super(salario, cedula, nombre1, nombre2, apellido1, apellido2, fechaNacimiento);
        this.añosExperiencia = añosExperiencia;
        this.cursos = new Curso[CantidadMaxCursos];
    }

    public boolean asignarCurso(Curso c) {
        for (int j = 0; j < getCursos().length; j++) {
            if (getCursos()[j] == null) {
                getCursos()[j] = c;
                c.setProfesor(this);
                return true;
            }
        }
        return false;
    }

    @Override
    public String toString() {
        return String.format("%sAños de Experiencia: %d", super.toString(), añosExperiencia);
    }
}
```

7. PERSONA

```java
public class Persona {

    protected String cedula, nombre1, nombre2, apellido1, apellido2;
    protected LocalDate fechaNacimiento;

    public Persona(String cedula, String nombre1, String nombre2, String apellido1, String apellido2,
            LocalDate fechaNacimiento) {
        this.cedula = cedula;
        this.nombre1 = nombre1;
        this.nombre2 = nombre2;
        this.apellido1 = apellido1;
        this.apellido2 = apellido2;
        this.fechaNacimiento = fechaNacimiento;
    }

    public String getCedula() {
        return this.cedula;
    }

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

    public LocalDate getFechaNacimiento() {
        return this.fechaNacimiento;
    }

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

    public String nombreCompleto() {
        return String.format("%s %s %s %s", nombre1, nombre2, apellido1, apellido2);
    }

    public int edad() {
        LocalDate fechaActual = LocalDate.now();
        Period periodo = Period.between(fechaNacimiento, fechaActual);
        return periodo.getYears();
    }

    @Override
    public String toString() {
        return String.format("Cedula = %s\nNombre: %s\nEdad = %d", cedula, nombreCompleto(), edad());
    }

}
```

La clase “**Persona”** es la clase padre que servirá para obtener los datos que una persona requiere.

8. CONTROL_VALIDACION

```java

public class Control_Validacion {

    public Persona buscarPorCedula(String cedula, Persona[] personas) {
        for (Persona persona : personas) {
            if (persona != null && persona.getCedula().equalsIgnoreCase(cedula)) {
                return persona;
            }
        }
        return null;
    }

    public Profesor validarProf(String cedula, Profesor[] profesores) {
        return (Profesor) buscarPorCedula(cedula, profesores);
    }

    public Estudiante validarEstud(String cedula, Estudiante[] estudiantes) {
        return (Estudiante) buscarPorCedula(cedula, estudiantes);
    }

    public boolean comprobarEstudiante(String ced, Estudiante[] estudents) {
        for (int i = 0; i < estudents.length; i++) {
            if (estudents[i] != null) {
                if (estudents[i].getCedula().equalsIgnoreCase(ced)) {
                    System.out.println("Ya existe un estudiante con esta cedula");
                    return false;
                }
            }
        }
        return true;
    }
```

```java
    public <T> boolean existeElemento(T[] elementos) {
        for (T elemento : elementos) {
            if (elemento != null) {
                return true;
            }
        }
        return false;
    }

    public boolean comprobarProfesor(String ced, Profesor[] profesores) {
        for (int i = 0; i < profesores.length; i++) {
            if (profesores[i] != null) {
                if (profesores[i].getCedula().equalsIgnoreCase(ced)) {
                    System.out.println("Ya existe un Profesor con esta cedula");
                    return false;
                }
            }
        }
        return true;
    }

    public boolean existeCupos(Curso c) {
        for (int i = 0; i < c.getEstudiantes().length; i++) {
            if (c.getEstudiantes()[i] == null) {
                return true;
            }
        }
        return false;
    }

}
```

La clase “**Control_Validacion**” nos ayuda a validar y a comprobar, los distintos estudiantes, profesores, además de saber si existen cupos disponibles.

9. MENU

```java
import java.text.ParseException;
import java.util.Scanner;

public class Menu {

    GestorUnidadEducativa gue;
    Insertar insert;
    int opcion = 0;
    Scanner tecla = new Scanner(System.in);

    public Menu() {
        this.gue = new GestorUnidadEducativa();
        this.insert = new Insertar();
    }

    public GestorUnidadEducativa getGestorUnidadEducativa() {
        return this.gue;
    }

    public void Mostrar() throws ParseException {
        caratula();
        do {
            System.out.println("\n\t\t             MENU DE OPCIONES");
            System.out.println("1. Registrar Estudiantes");
            System.out.println("2. Inscribir estudiante a un curso");
            System.out.println("3. Registrar Docente");
            System.out.println("4. Crear un Curso");
            System.out.println("5. Reporte de estudiantes y profesores del centro educativo");
            System.out.println("6. Reporte de estudiantes de un curso");
            System.out.println("7. Ordenar curso por numero de inscritos");
            System.out.println("8. Salir");
            opcion = tecla.nextInt();

            switch (opcion) {
                case 1:
                    this.gue.insertarObjeto(this.insert.insertarDatosEstudiante(this.gue.getEstudiantes()),
                            this.gue.getEstudiantes());
                    break;
                case 2:
                    this.insert.insertarDatosMatricEstu(gue);
                    break;
                case 3:
                    this.gue.insertarObjeto(this.insert.insertarDatosProfesor(this.gue.getProfesores()),
                            this.gue.getProfesores());
                    break;
                case 4:
                    this.gue.insertarObjeto(this.insert.insertarDatosCurso(this.gue), this.gue.getCursos());
                    break;
                case 5:
                    this.gue.mostrarEstudiantes();
                    this.gue.mostrarProfesores();
                    break;
                case 6:
                    this.gue.mostrarCursos();
                    break;
                case 7:
                    gue.imprimirCursosOrdenados();
                    break;
                case 8:
                    System.out.println("GRACIAS POR USAR EL PROGRAMA");
                    break;
                default:
                    System.out.println("Escoja una opcion valida");
                    break;
            }
        } while (opcion != 8);
    }

    public void caratula() {
        System.out.println("_____________________________________________________________________");
        System.out.println("\t\t           Centro Educativo Caritas Felices");
        System.out.println("\t Sistema de Registro y Matriculacion de Personal Educativo");
        System.out.println("_____________________________________________________________________");

    }
}
```

La clase “**Menu**” es toda la parte estética que se mostrara al momento de ejecutar el programa, cuenta con las instancias de otras clases para acceder a sus métodos que nos permitirán realizar las distintas opciones que cuenta el Menu.

10. PRINCIPAL

```java
package proyecto_2;

import java.text.ParseException;

public class Principal {

    public static void main(String[] args) {
        try {
            new Menu().Mostrar();
        } catch (ParseException e) {
            System.err.println("Error al ejecutar el menú: " + e.getMessage());
        }
    }
}
```

La clase "**Principal**" es la entrada principal de la aplicación. En su método "**main**" crea una instancia de la clase "**Menu**" e invoca al método "Mostrar" de esa instancia para presentar el menú interactivo y gestionar las operaciones. Cualquier excepción de tipo "**ParseException**" que pueda ocurrir durante la ejecución del menú se captura y se muestra como un mensaje de error en la consola.