

UNIVERSIDAD TECNICA DE AMBATO



INTEGRANTES:

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

SEMESTRE:

TERCERO

FACULTAD:

INGENIERIA EN SISTEMAS, ELECTRONICA E
INDUSTRIAL

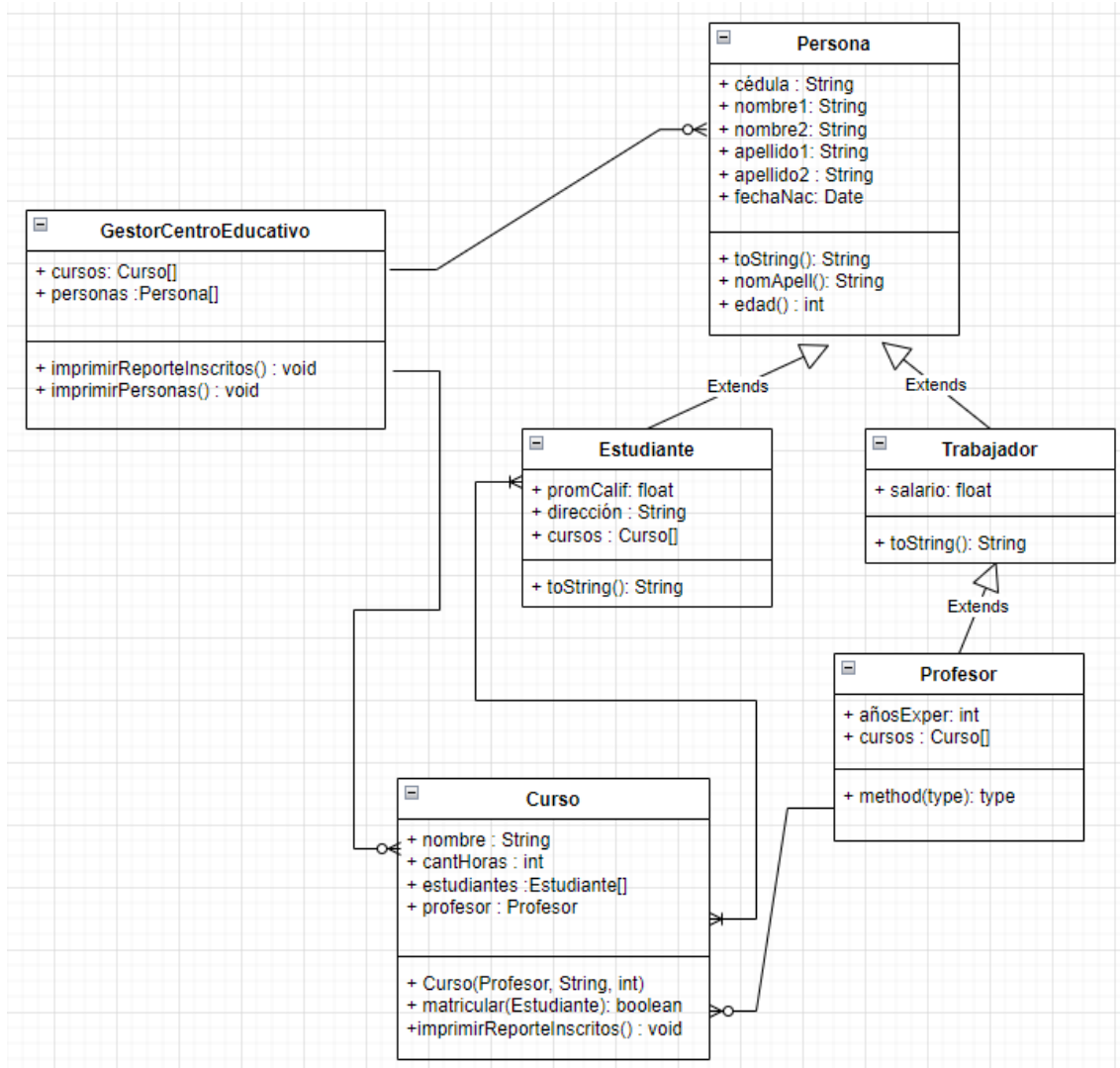
MATERIA:

Estructura de Datos

DOCENTE:

Ing. Fernández Peña Félix Oscar

DIAGRAMA DE CLASE



CLASES:

1. CURSO

```
2
3 class Curso implements Comparable<Curso> {
4
5     private String nombre;
6     private int cantHoras;
7     private Estudiante estudiantes[];
8     private Profesor profesor;
9     private int codigo;
10
11     public Curso(String nombre, int cantHoras, Profesor profesor, int codigo, int cupo) {
12         this.nombre = nombre;
13         this.cantHoras = cantHoras;
14         this.profesor = profesor;
15         this.codigo = codigo;
16         this.estudiantes = new Estudiante[cupo];
17     }
18
19     public int getCodigo() {
20         return this.codigo;
21     }
22
23     public int getCupo() {
24         return this.estudiantes.length;
25     }
26
27     public String getNombre() {
28         return this.nombre;
29     }
30
31     public void setNombre(String nombre) {
32         this.nombre = nombre;
33     }
34
35     @Override
36     public String toString() {
37         return "Codigo de Curso = " + this.codigo + "\n Nombre del curso = " + this.nombre + "\n Cantidad de horas = "
38             + this.cantHoras + "\n Profesor = " + this.profesor.nombreCompleto();
39     }
40
41     public int cantidadCuposDisponibles() {
42         int cupolibre = 0;
43         for (int i = 0; i < this.estudiantes.length; i++) {
44             if (this.estudiantes[i] == null) {
45                 cupolibre++;
46             }
47         }
48         return cupolibre;
49     }
50
51     @Override
52     public int compareTo(Curso c) {
53         if (c == null) {
54             return 1;
55         }
56         return c.escojerCupo(false) - this.escojerCupo(false);
57     }
58
59     public int escojerCupo(boolean op) {
60         int cupolibre = 0;
61         for (int i = 0; i < this.estudiantes.length; i++) {
62             if (this.estudiantes[i] == null) {
63                 cupolibre++;
64             }
65         }
66     }
67
68     public int getCantHoras() {
69         return this.cantHoras;
70     }
71
72     public void setCantHoras(int cantHoras) {
73         this.cantHoras = cantHoras;
74     }
75
76     public Profesor getProfesor() {
77         return this.profesor;
78     }
79
80     public void setProfesor(Profesor profesor) {
81         this.profesor = profesor;
82     }
83
84     public Estudiante[] getEstudiantes() {
85         return this.estudiantes;
86     }
87
88     public boolean matricularEstudianteCurso(Estudiante e) {
89         if (e != null) {
90             for (int i = 0; i < getEstudiantes().length; i++) {
91                 if (getEstudiantes()[i] == null) {
92                     getEstudiantes()[i] = e;
93                     return true;
94                 }
95             }
96         }
97         return false;
98     }
99 }
```

```

106         if (op) {
107             return cupolibre;
108         } else {
109             return this.estudiantes.length - cupolibre;
110         }
111     }
112 }
113

```

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

```

3  import java.time.LocalDate;
4
5  public class Estudiante extends Persona {
6
7      private float promedioCalificaciones;
8      private String direccion;
9      private Curso cursos[];
10     private final static int cantCursos = 5;
11
12     public Estudiante(float promedioCalificaciones, String direccion, String cedula, String nombre1, String nombre2,
13         String apellido1, String apellido2, LocalDate fechaNacimiento) {
14         super(cedula, nombre1, nombre2, apellido1, apellido2, fechaNacimiento);
15         this.promedioCalificaciones = promedioCalificaciones;
16         this.direccion = direccion;
17         this.cursos = new Curso[cantCursos];
18     }
19
20     public float getPromedioCalificaciones() {
21         return promedioCalificaciones;
22     }
23
24     public void setPromedioCalificaciones(float promedioCalificaciones) {
25         this.promedioCalificaciones = promedioCalificaciones;
26     }
27
28     public String getDireccion() {
29         return direccion;
30     }
31
32     public void setDireccion(String direccion) {
33         this.direccion = direccion;
34     }
35
36     public Curso[] getCursos() {
37         return cursos;
38     }
39
40     public void setCursos(Curso[] cursos) {
41         this.cursos = cursos;
42     }
43
44     public boolean asignarCursoEstudiante(Curso c) {
45         for (int j = 0; j < getCursos().length; j++) {
46             if (getCursos()[j] == null) {
47                 getCursos()[j] = c;
48                 return true;
49             }
50         }
51         return false;
52     }
53
54     @Override
55     public String toString() {
56         return super.toString() + "\nPromedio de Calificaciones = " + promedioCalificaciones + "\nDireccion = "
57             + direccion;
58     }
59 }

```

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

3. INSERTAR

```
3 import java.time.LocalDate;
4 import java.time.format.DateTimeFormatter;
5 import java.util.Scanner;
6
7 public class Insertar {
8
9     private DateTimeFormatter formatoFecha;
10    private Scanner tec;
11    private Control_Validacion cv;
12    private int cont;
13
14    public Insertar() {
15        this.formatoFecha = DateTimeFormatter.ofPattern("dd/MM/yyyy");
16        this.tec = new Scanner(System.in);
17        this.cv = new Control_Validacion();
18        this.cont = 0;
19    }
20
21    public Estudiante insertarDatosEstudiante(Estudiante estudiantes[]) {
22        System.out.println("Ingrese cedula del estudiante");
23        String cedula = tec.next();
24        if (this.cv.comprobarEstudiante(cedula, estudiantes)) {
25            System.out.println("Ingrese el primer nombre del Estudiante");
26            String nombrel = tec.next();
27            System.out.println("Ingrese el segundo nombre del Estudiante");
28            String nombre2 = tec.next();
29            System.out.println("Ingrese el primer apellido del Estudiante");
30            String apellidol = tec.next();
31            System.out.println("Ingrese el segundo apellido del Estudiante");
32            String apellido2 = tec.next();
33            System.out.println("Ingresar direccion");
34            String direccion = tec.next();
35
36            System.out.println("Ingrese la fecha de nacimiento (formato: 25/09/1990)");
37            String fechaNacimiento = tec.next();
38            LocalDate fecha = LocalDate.parse(fechaNacimiento, formatoFecha);
39            float promedioAcademico = 0.0f;
40            System.out.println("ESTUDIANTE REGISTRADO CON EXITO");
41            return new Estudiante(promedioAcademico, direccion, cedula, nombrel, nombre2, apellidol, apellido2, fecha);
42        } else {
43            return null;
44        }
45    }
46
47    public Curso insertarDatosCurso(GestorUnidadEducativa gue) {
48        if (cv.existeElemento(gue.getProfesores())) {
49            System.out.println("Ingrese nombre del curso");
50            String nombreCurso = tec.next();
51            System.out.println("Ingrese cantidad total de horas del curso");
52            int horasTotales = tec.nextInt();
53            System.out.println("Cuántos cupos disponibles existen");
54            int cupos = tec.nextInt();
55            System.out.println("Ingrese cedula del profesor a cargo del curso");
56            String cedulaProfesor = tec.next();
57            Profesor profesor = cv.validarProf(cedulaProfesor, gue.getProfesores());
58            if (profesor == null) {
59                System.out.println("HUBO UN ERROR, VERIFICA LOS DATOS");
60                return null;
61            } else {
62                Curso c = new Curso(nombreCurso, horasTotales, profesor, this.cont, cupos);
63                if (gue.asignarCursoProfesor(cedulaProfesor, c)) {
64                    this.cont++;
65                    System.out.println("CURSO CREADO CON EXITO");
66                    return c;
67                } else {
68                    return null;
69                }
70            } else {
71                System.out.println("NO EXISTEN PROFESORES PARA CREAR UN CURSO");
72            }
73            return null;
74        }
75    }
76
77    public Profesor insertarDatosProfesor(Profesor profesores[]) {
78        System.out.println("Ingrese su cedula");
79        String cedulaProfesor = tec.next();
80        if (this.cv.comprobarProfesor(cedulaProfesor, profesores)) {
81            System.out.println("Ingrese el primer nombre");
82            String nombrel = tec.next();
83            System.out.println("Ingresar el segundo nombre");
84            String nombre2 = tec.next();
85            System.out.println("Ingrese el primer apellido");
86            String apellidol = tec.next();
87            System.out.println("Ingrese el segundo apellido");
88            String apellido2 = tec.next();
89            System.out.println("Ingrese la fecha de nacimiento (formato: 25/09/1990)");
90            String fechaNacimiento = tec.next();
91            LocalDate fecha = LocalDate.parse(fechaNacimiento, formatoFecha);
92            System.out.println("Ingrese años de experiencia");
93            int añosExperiencia = tec.nextInt();
94            System.out.println("Ingrese el salario");
95            float salario = tec.nextFloat();
96            System.out.println("PROFESOR REGISTRADO CON EXITO");
97            return new Profesor(añosExperiencia, salario, cedulaProfesor, nombrel, nombre2, apellidol, apellido2,
98                fecha);
99        }
```

```

99         } else {
100             return null;
101         }
102     }
103
104     public void insertarDatosMatricEstu(GestorUnidadEducativa gue) {
105         if (cv.existeElemento(gue.getEstudiantes())) {
106             if (cv.existeElemento(gue.getCursos())) {
107                 int codCurso = gue.cursosDisponibles();
108                 for (Curso curso : gue.getCursos()) {
109                     if (curso != null && curso.getCodigo() == codCurso) {
110                         if (cv.existeCupos(curso)) {
111                             System.out.println("Ingrese la cedula del estudiante");
112                             String cedEstudiante = tec.next();
113                             Estudiante estudiante = cv.validarEstud(cedEstudiante, gue.getEstudiantes());
114                             if (estudiante != null && cv.comprobarEstudiante(cedEstudiante, curso.getEstudiantes())) {
115                                 if (gue.ingresoEstudiante(curso, estudiante)) {
116                                     System.out.println("MATRICULADO CON EXITO");
117                                 } else {
118                                     System.out.println("HUBO UN ERROR AL MATRICULAR");
119                                     System.out.println("VERIFICA QUE TENGAS CUPOS PARA MATRICULAR");
120                                 }
121                             }
122                         }
123                         break; // No es necesario seguir iterando después de encontrar el curso
124                     }
125                 }
126             } else {
127                 System.out.println("NO EXISTEN CURSOS DISPONIBLES");
128             }
129         } else {
130             System.out.println("NO EXISTEN ALUMNOS DISPONIBLES");
131         }
132     }
133 }
134

```

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

```

3  import java.util.Arrays;
4  import java.util.Comparator;
5  import java.util.Scanner;
6
7  public class GestorUnidadEducativa {
8
9      private static final int MAX_CANTIDAD = 100;
10     private Estudiante[] estudiantes;
11     private Profesor[] profesores;
12     private Curso[] cursos;
13     private Control_Validacion cv;
14
15     public GestorUnidadEducativa() {
16         this.estudiantes = new Estudiante[MAX_CANTIDAD];
17         this.profesores = new Profesor[MAX_CANTIDAD];
18         this.cursos = new Curso[MAX_CANTIDAD];
19         this.cv = new Control_Validacion();
20     }
21
22     public Profesor[] getProfesores() {
23         return this.profesores;
24     }
25
26     public Estudiante[] getEstudiantes() {
27         return this.estudiantes;
28     }
29
30     public Curso[] getCursos() {
31         return this.cursos;
32     }
33

```

```

34 public void mostrarEstudiantes() {
35     if (cv.existeElemento(this.estudiantes)) {
36         System.out.println("-----ESTUDIANTES-----");
37         for (Estudiante estudiante : this.estudiantes) {
38             if (estudiante != null) {
39                 System.out.println(estudiante.toString() + "\n");
40             }
41         }
42     } else {
43         System.out.println("NO EXISTEN ESTUDIANTES");
44     }
45 }
46
47 public boolean exisProfesor() {
48     for (int i = 0; i < this.profesores.length; i++) {
49         if (this.profesores[i] != null) {
50             return true;
51         }
52     }
53     return false;
54 }
55
56 public void mostrarProfesores() {
57     if (exisProfesor()) {
58         System.out.println("-----DOCENTES-----");
59         for (Profesor profesor : this.profesores) {
60             if (profesor != null) {
61                 System.out.println(profesor.toString() + "\n");
62             }
63         }
64     } else {
65         System.out.println("NO EXISTEN PROFESORES");

```

```

66     }
67 }
68
69 public void mostrarCursos() {
70     int posicion = this.cursosDisponibles();
71     if ((posicion) >= 0) {
72         if (this.cursos[posicion] != null) {
73             System.out.println(this.cursos[posicion].toString());
74             System.out.println(" Cupos disponibles: " + this.cursos[posicion].escojerCupo(true));
75             System.out.println("Estudiantes registrados:");
76             for (int i = 0; i < this.cursos[posicion].getEstudiantes().length; i++) {
77                 if (this.cursos[posicion].getEstudiantes()[i] != null) {
78                     System.out.println(i + ". " + this.cursos[posicion].getEstudiantes()[i].nombreCompleto());
79                 }
80             }
81         } else {
82             System.out.println("CURSO NO ENCONTRADO");
83         }
84     } else {
85         System.out.println("NO HAY CURSOS DISPONIBLES");
86     }
87 }
88
89 public void imprimirCursosOrdenados() {
90     Arrays.sort(this.cursos, Comparator.nullsLast(Comparator.naturalOrder()));
91     for (int i = 0; i < this.cursos.length; i++) {
92         if (this.cursos[i] != null) {
93             System.out.println(cursos[i].getNombre() + ": " + this.cursos[i].escojerCupo(false));
94         }
95     }
96 }
97

```

```

98 public boolean insertarObjeto(Object objeto, Object[] arreglo) {
99     for (int i = 0; i < arreglo.length; i++) {
100         if (arreglo[i] == null) {
101             arreglo[i] = objeto;
102             return true;
103         }
104     }
105     return false;
106 }
107
108 public int cursosDisponibles() {
109     Scanner tec = new Scanner(System.in);
110     int opcion = -1;
111     int conta = 0;
112     Arrays.sort(this.cursos, Comparator.nullsLast(Comparator.naturalOrder()));
113     for (int i = 0; i < this.cursos.length; i++) {
114         if (this.cursos[i] != null) {
115             conta++;
116         }
117     }
118     if (conta > 0) {
119         System.out.println("Esoja el codigo de los cursos disponibles");
120         for (int i = 0; i < this.cursos.length; i++) {
121             if (this.cursos[i] != null) {
122                 System.out.println(this.cursos[i].getCodigo() + ". " + this.cursos[i].getNombre());
123             }
124         }
125         opcion = tec.nextInt();
126     }
127     return opcion;
128 }
129

```

```

131 public boolean ingresoEstudiante(Curso c, Estudiante e) {
132     if (c != null && e != null) {
133         for (int i = 0; i < c.getEstudiantes().length; i++) {
134             if (c.getEstudiantes()[i] == null) {
135                 for (int j = 0; j < e.getCursos().length; j++) {
136                     if (e.getCursos()[j] == null) {
137                         c.matricularEstudianteCurso(e);
138                         e.asignarCursoEstudiante(c);
139                         return true;
140                     }
141                 }
142             }
143         }
144         return false;
145     }
146 }

147 public boolean asignarCursoProfesor(String ced, Curso c) {
148     for (int i = 0; i < this.profesores.length; i++) {
149         if (profesores[i] != null) {
150             if (profesores[i].getCedula().equalsIgnoreCase(ced)) {
151                 for (int j = 0; j < profesores[i].getCursos().length; j++) {
152                     if (profesores[i].getCursos()[j] == null) {
153                         profesores[i].asignarCurso(c);
154                         return true;
155                     } else {
156                         System.out.println("ESTE PROFESOR NO TIENE DISPONIBLE MAS CURSOS");
157                     }
158                 }
159             }
160         }
161     }
162     return false;
163 }

164 public void ordenarCursosporInscrito(Curso cursos[]) {
165     Arrays.sort(cursos);
166 }
167 }
168 }

```

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

```

1 package proyecto_2;
2
3 import java.time.LocalDate;
4
5 public class Trabajador extends Persona {
6
7     protected float salario;
8
9     public Trabajador(float salario, String cedula, String nombrel, String nombre2, String apellidol, String apellido2,
10         LocalDate fechaNacimiento) {
11         super(cedula, nombrel, nombre2, apellidol, apellido2, fechaNacimiento);
12         this.salario = salario;
13     }
14
15     @Override
16     public String toString() {
17         return String.format("%s\nSalario = %.2f\n", super.toString(), salario);
18     }
19
20 }
21

```

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

```
1 package proyecto_2;
2
3 import java.time.LocalDate;
4
5 class Profesor extends Trabajador {
6
7     private int añosExperiencia;
8     private Curso[] cursos;
9     private static final int CantidadMaxCursos = 4;
10
11     public Curso[] getCursos() {
12         return this.cursos;
13     }
14
15     public Profesor(int añosExperiencia, float salario, String cedula, String nombre1, String nombre2, String apellido1,
16                     String apellido2, LocalDate fechaNacimiento) {
17         super(salario, cedula, nombre1, nombre2, apellido1, apellido2, fechaNacimiento);
18         this.añosExperiencia = añosExperiencia;
19         this.cursos = new Curso[CantidadMaxCursos];
20     }
21
22     public boolean asignarCurso(Curso c) {
23         for (int j = 0; j < getCursos().length; j++) {
24             if (getCursos()[j] == null) {
25                 getCursos()[j] = c;
26                 c.setProfesor(this);
27                 return true;
28             }
29         }
30         return false;
31     }
32
33     @Override
34     public String toString() {
35         return String.format("%sAños de Experiencia: %d", super.toString(), añosExperiencia);
36     }
37 }
38
```

7. PERSONA

```
1 public class Persona {
2
3     protected String cedula, nombre1, nombre2, apellido1, apellido2;
4     protected LocalDate fechaNacimiento;
5
6     public Persona(String cedula, String nombre1, String nombre2, String apellido1, String apellido2,
7                     LocalDate fechaNacimiento) {
8         this.cedula = cedula;
9         this.nombre1 = nombre1;
10        this.nombre2 = nombre2;
11        this.apellido1 = apellido1;
12        this.apellido2 = apellido2;
13        this.fechaNacimiento = fechaNacimiento;
14    }
15
16    public String getCedula() {
17        return this.cedula;
18    }
19
20    public void setCedula(String cedula) {
21        this.cedula = cedula;
22    }
23
24    public LocalDate getFechaNacimiento() {
25        return this.fechaNacimiento;
26    }
27
28    public void setFechaNacimiento(LocalDate fechaNacimiento) {
29        this.fechaNacimiento = fechaNacimiento;
30    }
31
32    public String nombreCompleto() {
33        return String.format("%s %s %s %s", nombre1, nombre2, apellido1, apellido2);
34    }
35
36    public int edad() {
37        LocalDate fechaActual = LocalDate.now();
38        Period periodo = Period.between(fechaNacimiento, fechaActual);
39        return periodo.getYears();
40    }
41
42    @Override
43    public String toString() {
44        return String.format("Cedula = %s\nNombre: %s\nEdad = %d", cedula, nombreCompleto(), edad());
45    }
46
47 }
48
```

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

8. CONTROL_VALIDACION

```
2
3 public class Control_Validacion {
4
5     public Persona buscarPorCedula(String cedula, Persona[] personas) {
6         for (Persona persona : personas) {
7             if (persona != null && persona.getCedula().equalsIgnoreCase(cedula)) {
8                 return persona;
9             }
10        }
11        return null;
12    }
13
14    public Profesor validarProf(String cedula, Profesor[] profesores) {
15        return (Profesor) buscarPorCedula(cedula, profesores);
16    }
17
18    public Estudiante validarEstud(String cedula, Estudiante[] estudiantes) {
19        return (Estudiante) buscarPorCedula(cedula, estudiantes);
20    }
21
22    public boolean comprobarEstudiante(String ced, Estudiante[] estudiantes) {
23        for (int i = 0; i < estudiantes.length; i++) {
24            if (estudiantes[i] != null) {
25                if (estudiantes[i].getCedula().equalsIgnoreCase(ced)) {
26                    System.out.println("Ya existe un estudiante con esta cedula");
27                    return false;
28                }
29            }
30        }
31        return true;
32    }
33
34    public <T> boolean existeElemento(T[] elementos) {
35        for (T elemento : elementos) {
36            if (elemento != null) {
37                return true;
38            }
39        }
40        return false;
41    }
42
43    public boolean comprobarProfesor(String ced, Profesor[] profesores) {
44        for (int i = 0; i < profesores.length; i++) {
45            if (profesores[i] != null) {
46                if (profesores[i].getCedula().equalsIgnoreCase(ced)) {
47                    System.out.println("Ya existe un Profesor con esta cedula");
48                    return false;
49                }
50            }
51        }
52        return true;
53    }
54
55    public boolean existeCupos(Curso c) {
56        for (int i = 0; i < c.getEstudiantes().length; i++) {
57            if (c.getEstudiantes()[i] == null) {
58                return true;
59            }
60        }
61        return false;
62    }
63
64 }
65
```

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

9. MENU

```

3 import java.text.ParseException;
4 import java.util.Scanner;
5
6 public class Menu {
7
8     GestorUnidadEducativa gue;
9     Insertar insert;
10    int opcion = 0;
11    Scanner tecla = new Scanner(System.in);
12
13    public Menu() {
14        this.gue = new GestorUnidadEducativa();
15        this.insert = new Insertar();
16    }
17
18    public GestorUnidadEducativa getGestorUnidadEducativa() {
19        return this.gue;
20    }
21
22    public void Mostrar() throws ParseException {
23        caratula();
24        do {
25            System.out.println("\n\t\t\t\t\t MENU DE OPCIONES");
26            System.out.println("1. Registrar Estudiantes");
27            System.out.println("2. Inscribir estudiante a un curso");
28            System.out.println("3. Registrar Docente");
29            System.out.println("4. Crear un Curso");
30            System.out.println("5. Reporte de estudiantes y profesores del centro educativo");
31            System.out.println("6. Reporte de estudiantes de un curso");
32            System.out.println("7. Ordenar curso por numero de inscritos");
33            System.out.println("8. Salir");
34            opcion = tecla.nextInt();
35
36            switch (opcion) {
37                case 1:
38                    this.gue.insertarObjeto(this.insert.insertarDatosEstudiante(this.gue.getEstudiantes()),
39                                            this.gue.getEstudiantes());
40                    break;
41                case 2:
42                    this.insert.insertarDatosMatricEstu(gue);
43                    break;
44                case 3:
45                    this.gue.insertarObjeto(this.insert.insertarDatosProfesor(this.gue.getProfesores()),
46                                            this.gue.getProfesores());
47                    break;
48                case 4:
49                    this.gue.insertarObjeto(this.insert.insertarDatosCurso(this.gue), this.gue.getCursos());
50                    break;
51                case 5:
52                    this.gue.mostrarEstudiantes();
53                    this.gue.mostrarProfesores();
54                    break;
55                case 6:
56                    this.gue.mostrarCursos();
57                    break;
58                case 7:
59                    gue.imprimirCursosOrdenados();
60                    break;
61                case 8:
62                    System.out.println("GRACIAS POR USAR EL PROGRAMA");
63                    break;
64                default:
65                    System.out.println("Escoja una opcion valida");
66                    break;
67            }
68        } while (opcion != 8);
69    }
70
71    public void caratula() {
72        System.out.println(" ");
73        System.out.println("\t\t\t\t\t Centro Educativo Caritas Felices");
74        System.out.println("\t\t Sistema de Registro y Matriculacion de Personal Educativo");
75        System.out.println(" ");
76    }
77 }

```

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

```
1 package proyecto_2;
2
3 import java.text.ParseException;
4
5 public class Principal {
6
7     public static void main(String[] args) {
8         try {
9             new Menu().Mostrar();
10        } catch (ParseException e) {
11            System.err.println("Error al ejecutar el menú: " + e.getMessage());
12        }
13    }
14 }
15
```

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.