Universidad Rey Juan Carlos

Grado: **CIENCIA, GESTIÓN E INGENIERÍA DE SERVICIOS**

Curso: **2022 / 2023**

Asignatura: **INTRODUCCIÓN A LA PROGRAMACIÓN**

Trabajo: **DESARROLLO DE UNA APLICACIÓN EN JAVA**

Alumnado:

URL Codeboard: https://codeboard.io/projects/366271

Índice

Objetivos

En el presente trabajo se aplicarán los conocimientos adquiridos hasta el momento en la asignatura de Introducción a la programación, perteneciente al Grado en Ciencia, Gestión e Ingeniería de Servicios por la Universidad Rey Juan Carlos.

Este proyecto consiste en crear una aplicación que permita planificar y ordenar toda la información referente al horario semanal de las asignaturas que tiene un curso concreto.

Con la creación de esta aplicación lo que esperamos conseguir es lo siguiente:

En primer lugar, buscamos la definición detallada de las asignaturas, esto debe incluir el nombre y código de la asignatura, así como las horas semanales y el máximo de horas al día. Además, queremos que almacene la siguiente información: día de la semana, hora de inicio y final de la clase, aula, y por supuesto, profesor que impartirá dicha clase.

El claro objetivo de nuestra aplicación es que mediante un sencillo proceso el cliente sea capaz de crear, modificar u eliminar un horario. Cabe destacar también que para verificar dicho horario existe la posibilidad de buscar mediante fecha, aula o incluso el profesor, sin necesidad de indicar la asignatura.

Introducción

El programa consta de varias clases, estas clases facilitan en gran cantidad a la elaboración del programa y ayudan a la optimización del mismo. Este programa está elaborado desde la herramienta *Codeboard* y la documentación del proyecto ha sido elaborada desde *Onedrive*.

Requisitos del sistema

Al hacer el código, debimos tener en mente a un hipotético cliente para así poder crearlo de la forma más eficiente posible.

En este caso, nuestro supuesto cliente es un centro educativo, ya que hemos creado una aplicación para que así pueda gestionar toda la información de las asignaturas de un curso, así como la disposición horaria semanal.

Lista de requisitos a cumplir:

- Las asignaturas que componen el horario han de tener como mínimo: código y nombre de la asignatura, total de horas semanales, máximo de horas al día.
- Las diferentes entradas del horario deben contener al menos: día de la semana, hora de inicio y fin, asignatura, aula y docente.
- Definición de las asignaturas del horario:
    - Inclusión de una nueva asignatura, su carga lectiva semanal y el máximo de horas que se pueden impartir en un mismo día.
    - Modificación de los datos de una asignatura.
    - Borrado de una asignatura.
    - Borrado de todas las asignaturas.
- Definición del horario:
    - Inclusión de una entrada en el horario.
    - Modificación de una entrada en el horario.
    - Borrado de una entrada del horario.
    - Borrado de todas las entradas del horario.
- Consulta del horario:
    - Mostrar todas las entradas del horario en orden cronológico (día de la semana y hora).
    - Mostrar la información relativa a una asignatura en particular: detalle y suma de horas.
    - Mostrar la información relativa a un aula en particular: detalle y suma de horas.
    - Mostrar la información relativa a un profesor en particular: detalle y suma de horas.

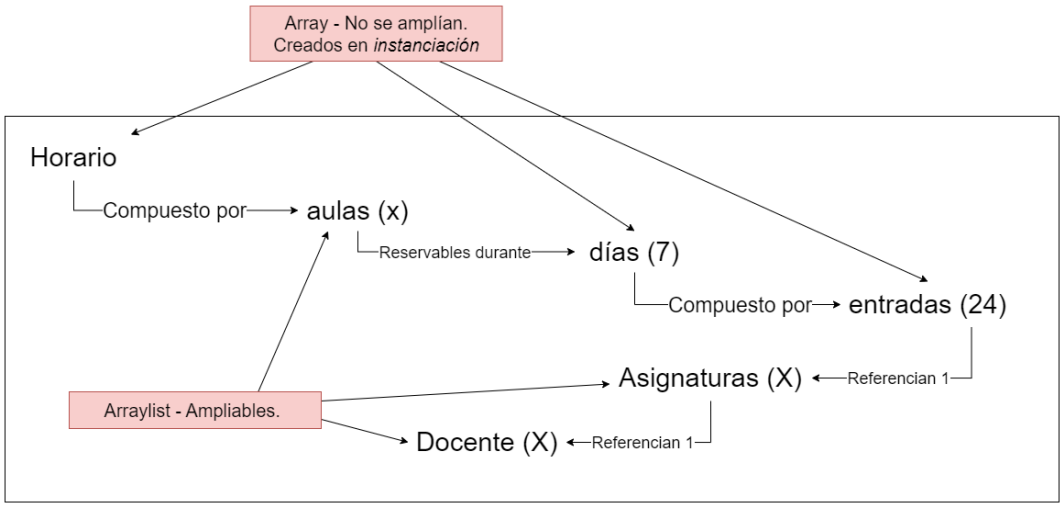
- Cuando comience la aplicación debe haberse precargado un horario.

Estos son los requisitos **mínimos** para la elaboración de la aplicación, nosotros los hemos ampliado, lo explicamos en el siguiente apartado (desarrollo de la aplicación).

Desarrollo de la aplicación

El desarrollo de la aplicación se ha llevado a cabo usando clases. Se crean objetos y para definir los objetos en Java usamos las clases, se usa para permitir mejoras sucesivas y que la cantidad de elementos sea menor y no suponga un problema más adelante. Estas clases componen el horario. Las clases que componen este horario son: horario, asignatura, aula, día, docente y hora. Para conseguir la información hemos hecho uso de "Scanner"; de esta forma, podemos pedir al usuario que nos de la información que necesitamos.

En este diagrama elaborado por nuestro grupo se encuentra la estructura:



Para comenzar el diseño del programa, se usa Scanner para solicitar la información necesaria al usuario, combinado con *println()*. De esta forma, el usuario puede leer lo que solicita el programa que introduzca dentro del Scanner.

De esta forma también le ponemos unas "reglas" al usuario y así conseguimos que el programa no termine de forma inesperada. Se ha tenido especial cuidado para la elección de tipos de datos.

Esto se puede observar cuando le explicamos al usuario como introducir los datos. Para menor confusión aún con el método anteriormente mencionado le introducimos en

pantalla al usuario las opciones que puede tomar. La metodología es sencilla. Por ejemplo, para la elección de la función a ejecutar desde el menú se usan números para evitar la corrupción en los datos y posibles errores durante la ejecución del programa debidos a datos mal introducidos por el usuario.

Clase Main

```java
import java.util.Scanner;

class Main {
    static Horario horario1 = new Horario("horario1");

    public static void main(String[] args) {
        // horario1.nueva_asignatura();
        carga_inicial(horario1);

        /*
         * horario1.listar_asignaturas();
         * horario1.get_asg_codes();
         * horario1.detalle_asignatura();
         * horario1.get_detalle_aula();
         * horario1.get_docente_listado();
         * horario1.editar_asignatura();
         */
        muestra_menu();
    }

    public static void muestra_menu() {

        int num = 0;
        while (num != 6) { // Este valor solo tomara 6 si se selecciona la opcion 6 del menu. De no ser
                           // asi, volvera a ejecutarse la funcion de forma continua.
            horario1.ClearScreen();
            System.out.println("\n\n\n\n\n\n ===============  MENU  ==============\n\n");
            System.out.println("Elija una de las siguientes opciones:");

            System.out.println("\n1. Gestion asignaturas:");

            System.out.println("    1.1. Agregar una asignatura");
            System.out.println("    1.2. Editar una asignatura");
            System.out.println("    1.3. Eliminar una asignatura");
            System.out.println("    1.4. Borrar TODAS las asiganturas");
            System.out.println("    1.5. Listar todas las asignaturas");
            System.out.println("    1.6. Ver los detalles de una asignatura");

            System.out.println("\n2. Definir horario:");
            System.out.println("    2.1. Agregar una entrada al horario");
            System.out.println("    2.2. Editar una entrada del horario");
            System.out.println("    2.3. Eliminar una entrada del horario");
            System.out.println("    2.4. Borrar todo el horario");
            System.out.println("    2.5. Restaurar horario/Cargar datos de muestra");

            System.out.println("\n3. Mostrar horario:");
            System.out.println("    3.1. Busqueda por asignatura");
            System.out.println("    3.2. Busqueda por aula");
            System.out.println("    3.3. Busqueda por profesor");
            System.out.println("    3.4. Mostrar todas las entradas");

            System.out.println("\n4. Gestion aulas:");
            System.out.println("    4.1. Agregar aula a la base de datos");
            System.out.println("    4.2. Editar aula ya creada");
            System.out.println("    4.3. Eliminar aula de la base de datos");
            System.out.println("    4.4. Eliminar TODAS las aulas.");
            System.out.println("    4.5. Listar aulas disponibles");
```

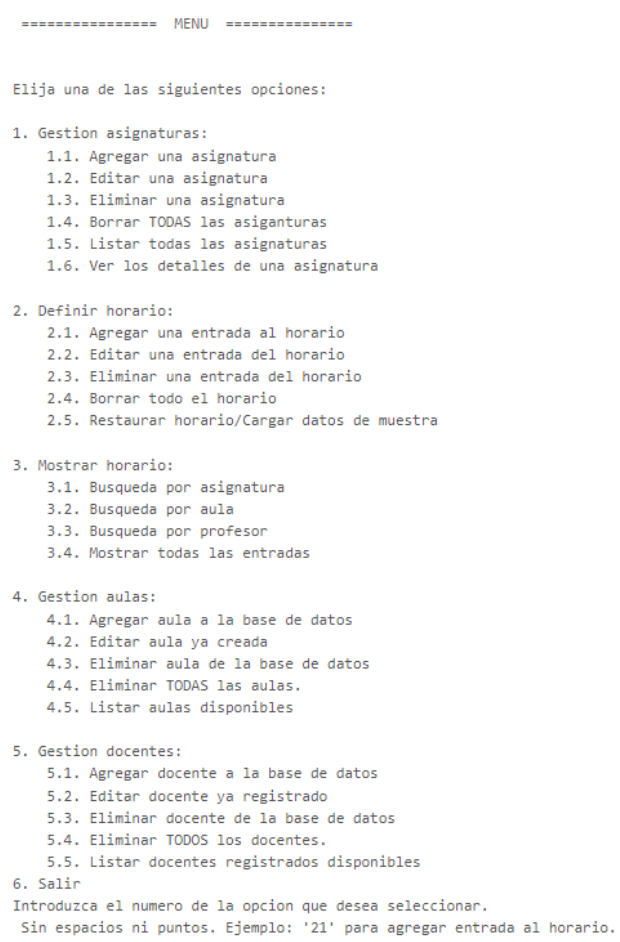
```
66          System.out.println("\n5. Gestion docentes:");
67          System.out.println("    5.1. Agregar docente a la base de datos");
68          System.out.println("    5.2. Editar docente ya registrado");
69          System.out.println("    5.3. Eliminar docente de la base de datos");
70          System.out.println("    5.4. Eliminar TODOS los docentes.");
71          System.out.println("    5.5. Listar docentes registrados disponibles");
72
73          System.out.println("6. Salir");
74
75          System.out.println(
76                  "Introduzca el numero de la opcion que desea seleccionar.\n Sin espacios ni puntos. Ejemplo: '21' para agregar entrada al horario.");
77
78          Scanner scanner = new Scanner(System.in);
79          num = scanner.nextInt();
80
81          switch (num) {
82              case 11:
83                  horario1.ClearScreen();
84                  horario1.nueva_asignatura();
85                  break;
86              case 12:
87                  horario1.ClearScreen();
88                  horario1.editar_asignatura();
89                  break;
90              case 13:
91                  horario1.ClearScreen();
92                  horario1.eliminar_asignatura();
93                  break;
94              case 14:
95                  horario1.ClearScreen();
96                  horario1.eliminar_asignaturas();
97                  break;
98              case 15:
99                  horario1.ClearScreen();
100                 horario1.listar_asignaturas();
101                 break;
102             case 16:
103                 horario1.ClearScreen();
104                 horario1.detalle_asignatura();
105                 break;
106             case 21:
107                 horario1.ClearScreen();
108                 horario1.add_entrada_horario();
109                 break;
110             case 22:
111                 horario1.ClearScreen();
112                 horario1.edit_entrada_horario();
113                 break;
114             case 23:
115                 horario1.ClearScreen();
116                 horario1.del_entrada_horario();
117                 break;
118             case 24:
119                 horario1.ClearScreen();
120                 // aqui puede ser interesante eliminar el objeto horario1 y volverlo a crear,
121                 // directamente. Sin annadir datos quiza
122                 horario1 = new Horario("horario1");
123                 System.out.println("Se han borrado todos los datos de la base de datos."); // Evidentemente no hay
124                                                                                           // base de datos, pero
125                                                                                           // asi queda mas guay
126                 horario1.waiter();
127                 break;
128             case 25:
129                 horario1.ClearScreen();
130                 horario1 = new Horario("horario1");
131                 carga_inicial(horario1);
132                 System.out.println("Se ha restaurado el horario a su estado inicial.");
133                 horario1.waiter();
134                 break;
135             case 31:
136                 horario1.ClearScreen();
137                 horario1.get_entradas_by_asg();
138                 break;
139             case 32:
140                 horario1.ClearScreen();
141                 horario1.get_entradas_by_aula();
142                 break;
143             case 33:
144                 horario1.ClearScreen();
145                 horario1.get_entradas_by_docente();
146                 break;
147             case 34:
148                 horario1.ClearScreen();
149                 horario1.get_entradas();
150                 break;
151             case 41:
152                 horario1.ClearScreen();
153                 horario1.aula_add();
154                 break;
155             case 42:
```

```
                case 42:
                    horario1.ClearScreen();
                    horario1.editar_aula();
                    break;
                case 43:
                    horario1.ClearScreen();
                    horario1.aula_delete();
                    break;
                case 44:
                    horario1.ClearScreen();
                    horario1.eliminar_aulas();
                    break;
                case 45:
                    horario1.ClearScreen();
                    horario1.get_detalle_aulas();
                    break;
                case 51:
                    horario1.ClearScreen();
                    horario1.add_docente();
                    break;
                case 52:
                    horario1.ClearScreen();
                    horario1.editar_docente();
                    break;
                case 53:
                    horario1.ClearScreen();
                    horario1.docente_delete();
                    break;
                case 54:
                    horario1.ClearScreen();
                    horario1.eliminar_docentes();
                    break;
                case 55:
                    horario1.ClearScreen();
                    horario1.get_docente_listado();
                    break;
                case 6:
                    horario1.ClearScreen();
                    System.out.println("Gracias por usar esta apliacion\nSaliendo...\n");
                    break;
                default:
                    System.out.println("El numero que ha introducido no es valido. Por favor, intruzca uno de nuevo.");
                    break;
            }

        }
    }

    public static void carga_inicial(Horario hor) {

        hor.add_docente_auto("12345678A", "Manuel", "Sanchez", "Diaz", 2000.0, "Profesor Titular");
        hor.add_docente_auto("23456789J", "Cristina", "Gonzalez", "Pons", 3000.0, "Doctora");
        hor.add_docente_auto("34567901C", "Rodrigo", "Hidalgo", "Esteban", 1000.0, "Profesor asociado");
        hor.add_docente_auto("45678012D", "Lucia", "Jimenez", "Llorente", 3000.0, "Doctora");
        hor.add_docente_auto("20458599E", "Javier", "Airado", "Rodriguez", 2000.0, "Profesor Titular");
        hor.add_docente_auto("50369736Z", "Pablo", "Aguado", "Gonzalez", 2000.0, "Profesor Titular");

        hor.asignatura_auto("I2P", "Introduccion a la Programacion", 8, 2, "50369736Z");
        hor.asignatura_auto("MCS", "Matematicas para la Computacion y Servicios", 9, 4, "23456789J");
        hor.asignatura_auto("HFS", "Historia y Fundamentos de los Servicios", 4, 1, "34567901C");
        hor.asignatura_auto("TDS", "Teoria de Sistemas", 10, 3, "45678012D");
        hor.asignatura_auto("ADQ", "Arquitectura de Computadores", 8, 3, "45678012D");
        hor.asignatura_auto("SDS", "Sociologia de los Servicios", 9, 2, "45678012D");

```

```
        hor.add_aula_auto("G107", 30);
        hor.add_aula_auto("G108", 20);
        hor.add_aula_auto("G109", 25);
        hor.add_aula_auto("G110", 40);
        hor.add_aula_auto("G111", 35);

        hor.add_entrada_horario_auto("lunes", "G107", "TDS", 10);
        hor.add_entrada_horario_auto("martes", "G108", "TDS", 9);
        hor.add_entrada_horario_auto("miercoles", "G109", "TDS", 8);
        hor.add_entrada_horario_auto("jueves", "G110", "TDS", 12);
        hor.add_entrada_horario_auto("viernes", "G111", "TDS", 13);
        hor.add_entrada_horario_auto("sabado", "G107", "TDS", 10);
        hor.add_entrada_horario_auto("domingo", "G107", "TDS", 9);

        hor.add_entrada_horario_auto("lunes", "G107", "MCS", 12);
        hor.add_entrada_horario_auto("martes", "G108", "MCS", 11);
        hor.add_entrada_horario_auto("miercoles", "G109", "MCS", 10);
        hor.add_entrada_horario_auto("jueves", "G110", "MCS", 10);
        hor.add_entrada_horario_auto("viernes", "G111", "MCS", 8);
        hor.add_entrada_horario_auto("sabado", "G108", "MCS", 10);
        hor.add_entrada_horario_auto("domingo", "G107", "MCS", 8);

        hor.add_entrada_horario_auto("lunes", "G107", "I2P", 14);
        hor.add_entrada_horario_auto("martes", "G108", "I2P", 13);
        hor.add_entrada_horario_auto("miercoles", "G109", "I2P", 14);
        hor.add_entrada_horario_auto("jueves", "G110", "I2P", 8);
        hor.add_entrada_horario_auto("viernes", "G111", "I2P", 10);
        hor.add_entrada_horario_auto("sabado", "G109", "I2P", 10);
        hor.add_entrada_horario_auto("domingo", "G108", "I2P", 10);

        hor.add_entrada_horario_auto("lunes", "G111", "HFS", 10);
        hor.add_entrada_horario_auto("martes", "G110", "HFS", 8);
        hor.add_entrada_horario_auto("miercoles", "G109", "HFS", 9);
        hor.add_entrada_horario_auto("jueves", "G108", "HFS", 10);
        hor.add_entrada_horario_auto("viernes", "G107", "HFS", 14);
        hor.add_entrada_horario_auto("sabado", "G110", "HFS", 9);
        hor.add_entrada_horario_auto("domingo", "G108", "HFS", 8);

        hor.add_entrada_horario_auto("lunes", "G110", "ADQ", 10);
        hor.add_entrada_horario_auto("martes", "G111", "ADQ", 10);
        hor.add_entrada_horario_auto("miercoles", "G107", "ADQ", 11);
        hor.add_entrada_horario_auto("jueves", "G108", "ADQ", 8);
        hor.add_entrada_horario_auto("viernes", "G109", "ADQ", 9);
        hor.add_entrada_horario_auto("sabado", "G107", "ADQ", 14);
        hor.add_entrada_horario_auto("domingo", "G108", "ADQ", 13);

        hor.add_entrada_horario_auto("lunes", "G111", "SDS", 8);
        hor.add_entrada_horario_auto("martes", "G107", "SDS", 10);
        hor.add_entrada_horario_auto("miercoles", "G107", "SDS", 9);
        hor.add_entrada_horario_auto("jueves", "G108", "SDS", 14);
        hor.add_entrada_horario_auto("viernes", "G109", "SDS", 13);
        hor.add_entrada_horario_auto("sabado", "G110", "SDS", 8);
        hor.add_entrada_horario_auto("domingo", "G109", "SDS", 10);

        hor.asignatura_auto("TMH", "Test para comprobar el maximo de horas al dia", 2, 1, "50369736Z");
        hor.add_entrada_horario_auto("lunes", "G111", "TMH", 8);

    };
}
```

Lo que se puede observar en estas imágenes son los diferentes tipos de casos con los que el usuario puede interactuar. Esto tiene relación con el System.out.println, ya que seleccionando los números el programa hace una serie de instrucciones. Adicionalmente, se ha declarado el método *ClearScreen()* en la clase Horario. La función de este método es insertar por consola una serie de saltos de línea y de esta manera facilitar al usuario la lectura de la salida del programa.

Las instrucciones por pantalla se ven de la siguiente manera:



Nos parece digno de mención la última parte de esta imagen. Como se puede observar, le indicamos al usuario la forma en la que debe seleccionar la opción que desea explicando cómo no debe incluir ni puntos ni comas, incluyendo además un ejemplo de los números que debe teclear para agregar una entrada al horario.

Al principio de esta sección, comentamos como nuestro grupo decidió basarse en la utilización de clases para la creación del programa. Es por ello que ahora pasaremos a mencionar esas clases y recoger el código de las mismas. Para algunas de ellas, debido a su inmenso tamaño, se mostrarán aquellos aspectos que sean de mayor relevancia o aquellos que a nuestro grupo le ha parecido destacable.

Como explicamos al principio, nuestro programa consta de varias clases, estas clases son un total de 7 incluyendo la clase *main*. Se trata de las clases: Asignatura, Aula, Día, Docente, Hora y Horario.

Asignatura

```java
public class Asignatura {
    public String codigo;
    public String nombre;
    public int total_horas_semana;
    public int max_horas_dia;
    public Docente docente;
    private int horas_actual;
    private int[] asg_dia = new int[] { 0, 0, 0, 0, 0, 0, 0 };
    // 0 --> +1 (lunes)
    // 1 --> +1 (martes)
    // 2 --> +3 (miercoles)

    public Asignatura(String cod, String nom, int ths, int mhd, Docente doc) {
        this.codigo = cod;
        this.nombre = nom;
        this.total_horas_semana = ths;
        this.max_horas_dia = mhd;
        this.horas_actual = 0;
        this.docente = doc;
        int[] asg_dia = new int[7];
    }

    public void update_scheduled_by_day(String operacion, int dow) {
        if (operacion.equals("+")) {
            asg_dia[dow]++;
        } else if (operacion.equals("-")) {
            asg_dia[dow]--;
        } else {
            System.out.println("Error en la modificacion.");
        }
    }
```

```java
    public void get_details() {
        System.out.println("CODigo:              " + this.codigo);
        System.out.println("NOMbre:              " + this.nombre);
        System.out.println("TOTal Horas Semana:  " + this.total_horas_semana);
        System.out.println("MAXimo Horas Dia:    " + this.max_horas_dia);
    }

    // Este metodo sirve para comprobar si puedo annadir mas horas a nivel semanal a
    // esta asignatura.
    // Devuelve true si puedo, porque el numero de horas actual es menor que el
    // maximo
    // Devielve false si no puedo, porque ya he alcanzando el numero maximo.

    public boolean can_be_scheduled_more(int dow) {
        if (asg_dia[dow] < max_horas_dia) {
            return true;
        } else {
            return false;
        }

        // ANTES ESTABA ESTO, claramente habiamos entendido mal y estabamos mirando el
        // total de la semana.
        // if (horas_actual < total_horas_semana) {
        // return true;
        // } else {
        // return false;
        // }
    }

    public void show_id_and_name() {
        System.out.println(this.codigo + ": " + this.nombre);
    }
```

Consideramos relevante explicar el funcionamiento de *update_scheduled_by_day*. Se trata de un método para actualizar el número de clases para esa asignatura en un día

concreto. El día concreto es pasado al programa a través de un parámetro. Como parámetro adicional, recibe la operación (sumar o restar 1 al contador).

```
24      public void update_scheduled_by_day(String operacion, int dow){
25          if (operacion.equals("+")){
26              asg_dia[dow]++;
27          } else if (operacion.equals("-")){
28              asg_dia[dow]--;
29          } else {
30              System.out.println("Error en la modificacion.");
31          }
32      }
```

Aula

```
1  public class Aula {
2      public String id;
3      private int capacidad;
4      public Dia[] dia = new Dia[] { new Dia("lunes"), new Dia("martes"), new Dia("miercoles"),
5              new Dia("jueves"), new Dia("viernes"), new Dia("sabado"), new Dia("domingo") };
6      public int registros_anadidos;
7
8      public Aula(String p_id, int p_csp) {
9          id = p_id;
10         // Problema: se deben poder manejar tantos horarios como aulas tenga!
11         // Solucion: El aula tiene los dias.
12         capacidad = p_csp;
13         registros_anadidos = 0;
14     }
15
16     public void get_details() {
17         System.out.println("Codigo (id):         " + this.id);
18         System.out.println("Capacidad:           " + this.capacidad);
19         System.out.println("Registros anadidos:  " + this.registros_anadidos);
20     }
```

```
14     }
15
16     public void get_details() {
17         System.out.println("Codigo (id):         " + this.id);
18         System.out.println("Capacidad:           " + this.capacidad);
19         System.out.println("Registros anadidos:  " + this.registros_anadidos);
20     }
21
22     public void get_lessons() {
23         int found = 0;
24         for (Dia d : dia) {
25             System.out.println("    Dia: " + d.nombre);
26             int i = 0;
27             for (Hora h : d.horas) {
28                 if (h != null) {
29                     System.out.println("        Hora:          " + i + ":00 - " + (i + 1) + ":00");
30                     System.out.println("        Asignatura:    " + h.asignatura.codigo + ": " + h.asignatura.nombre
31                             + "\n          -");
32                     found++;
33                 }
34                 i++;
35             }
36             System.out.println("    ----------------");
37         }
38         System.out.println("Se han encontrado " + found + " resultados para este aula.");
39     }
40
41     public void update_capacidad(int p_nueva_capacdad) {
42         this.capacidad = p_nueva_capacdad;
43     }
44 }
45
```

Día

```java
public class Dia {
    public String nombre;
    public Hora[] horas = new Hora[24];

    // Constructor
    public Dia(String nom) {
        nombre = nom;
        Hora[] horas = new Hora[24];
    }

    public Dia() {

    }

    public Hora get_entrada(int num) {
        return this.horas[num];
    }
}

```

Docente

```java
public class Docente {
    public String dni;
    private String nombre;
    private String apellido1;
    private String apellido2;
    private double sueldo;
    private String titulo;

    // TODO: usar sueldo y titulo. Actualizar acorde metodos

    public Docente(String p_dni, String p_nom, String p_ape1, String p_ape2, Double p_sueld, String p_tit) {
        dni = p_dni;
        nombre = p_nom;
        apellido1 = p_ape1;
        apellido2 = p_ape2;
        sueldo = p_sueld;
        titulo = p_tit;
    }

    public String get_fullname() {
        return (this.nombre + " " + this.apellido1 + " " + this.apellido2);
    }

    // Este metodo no tiene mucho sentido pero se ha utilizado durante la
    // realizacion del trabajo para la ilustracion de diferencias
    // private/public en las propiedades de la clase.
    // Se mantiene por si fuera necesario usarlo posteriormente.

    public String get_dni() {
        return this.dni;
    }

    public String get_fullname_and_dni() {
        return (this.get_fullname() + ": " + this.dni);
    }

    public void get_details() {
        System.out.println("   DNI:          " + this.dni);
        System.out.println("      NOMbre:    " + this.nombre);
        System.out.println("      APE1:      " + this.apellido1);
        System.out.println("      APE2:      " + this.apellido2);
        System.out.println("      SUEldo:    " + this.sueldo);
        System.out.println("      TITulo:    " + this.titulo);
    }

    // Estos metodos sirven para ilustrar que son necesarios metodos para actualizar
    // propiedades privadas.
    public void update_nombre(String p_introducido) {
        this.nombre = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }

    public void update_ape1(String p_introducido) {
        this.apellido1 = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }

    public void update_ape2(String p_introducido) {
        this.apellido2 = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }

    public void update_sueldo(Double p_introducido) {
        this.sueldo = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }

    public void update_titulo(String p_introducido) {
        this.titulo = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }
}
```

Hora

```java
public class Hora {
    // public docente docente;
    public Asignatura asignatura;

    public Hora() {
    }

    public Hora(Asignatura asg) {
        asignatura = asg;
        asg.total_horas_semana++;
    }
}

```

Como podemos observar, dentro de estas clases hay datos que son públicos y otros que son privados. Los privados solo pueden ser accedidos desde la misma clase, es decir, en el caso de Asignatura, podemos encontrar que hay valores que son públicos y otros que son privados.

Entonces, los datos privados se computan desde dentro de la clase. En cambio, los datos públicos como puede ser el caso de la propiedad *dni* en la clase Docente. Esto quiere decir que el dato o método es accesible desde cualquier parte del programa.

```java
public class Docente {
    public String dni;
    private String nombre;
    private String apellido1;
    private String apellido2;
    private double sueldo;
    private String titulo;

```

```java
public class Asignatura {
    public String codigo;
    public String nombre;
    public int total_horas_semana;
    public int max_horas_dia;
    public Docente docente;
    private int horas_actual;
    private int[] asg_dia = new int[] { 0, 0, 0, 0, 0, 0, 0 };
    // 0 --> +1 (lunes)
    // 1 --> +1 (martes)
    // 2 --> +3 (miercoles)

```

Funcionalidades

Apartado 1: Gestión asignaturas

Vamos a empezar con el A*partado 1*, encargado de la gestión de las asignaturas.

Este apartado contiene 6 subapartados que gracias a ellos podemos: agregar una asignatura, editar una asignatura, eliminar una asignatura, borrar TODAS las asignaturas, listar todas las asignaturas y ver los detalles de una asignatura.

Cada subapartado utilizará las clases y métodos definidos para así poder conseguir la información necesaria.

```
1. Gestion asignaturas:
    1.1. Agregar una asignatura
    1.2. Editar una asignatura
    1.3. Eliminar una asignatura
    1.4. Borrar TODAS las asiganturas
    1.5. Listar todas las asignaturas
    1.6. Ver los detalles de una asignatura
```

El *Apartado 1.1* encargado de añadir una asignatura al horario reutiliza diferentes métodos ya creados a lo largo del programa. Destacando, a modo de ejemplo, *get_listado_docente_name_and_dni()*. Así como la llamada al método constructor de la *clase Asignatura*.

```
75      public void nueva_asignatura(){
76          System.out.println("Bienvenido al proceso de creacion de asignaturas");
77
78          System.out.println("Por favor, introduzca los siguientes datos:\nCodigo:");
79          Scanner scanner_add_asg_cod = new Scanner(System.in);
80          String add_asg_cod = scanner_add_asg_cod.nextLine();
81
82          System.out.println("Nombre de la asignatura");
83          Scanner scanner_add_asg_nom = new Scanner(System.in);
84          String add_asg_nom = scanner_add_asg_nom.nextLine();
85
86          System.out.println("Total de horas a la semana:");
87          Scanner scanner_add_asg_ths = new Scanner(System.in);
88          int add_asg_ths = scanner_add_asg_ths.nextInt();
89
90          System.out.println("Maximo horas al dia:");
91          Scanner scanner_add_asg_mhd = new Scanner(System.in);
92          int add_asg_mhd = scanner_add_asg_mhd.nextInt();
93
94          System.out.println("Docente que impartira la nueva asignatura");
95          System.out.println("  > Como referencia, se muestran a continuacion los DNIs de los docentes.");
96          get_listado_docente_name_and_dni();
97          Scanner scanner_add_asg_doc = new Scanner(System.in);
98          String add_asg_doc = scanner_add_asg_doc.nextLine();
99
100         asignaturas.add(new asignatura(add_asg_cod, add_asg_nom, add_asg_ths, add_asg_mhd, docente_by_dni(add_asg_doc)));
101         System.out.println("Se ha creado correctamente la asignatura.");
102         waiter();
103     }
```

El *Subapartado 1.2* nos permite editar una asignatura ya creada. Para llevar a cabo lo dicho anteriormente, en primer lugar, lista todas las asignaturas ya registradas en el programa (recorriendo el *arrayList* que las contiene). En segundo lugar, solicita al usuario introducir el código de la asignatura que desea editar. A continuación, se recoge cuál es la propiedad que se desea editar y cuál será el nuevo valor de esta. Debido a que las diferentes propiedades almacenan tipos de datos distintos (*String* o *int*), encontramos una cláusula *if* para llamar correctamente al *scanner* y evitar, de este modo, errores durante la ejecución.

```java
public void editar_asignatura(){
    System.out.println("Mostrando lista de asignaturas: ");
    get_asg_codes();
    System.out.println("Introduzca el codigo de la asignatura que desea editar: ");
    Scanner scanner_editar = new Scanner(System.in);
    String cod_a_buscar = scanner_editar.next();
    for (asignatura i: asignaturas) {
        if (i.codigo.equals(cod_a_buscar)) {
            System.out.println("Estos son los detalles de la asignatura solicitada:");
            i.get_details();
            System.out.println("Introduzca los tres primeros caracteres de la variable que desea cambiar: ");
            Scanner scanner_trescaracteres = new Scanner(System.in);
            String tres_primeros_caracteres = scanner_trescaracteres.next();
            System.out.println("Introduzca el nuevo valor de " + tres_primeros_caracteres);
            Scanner scannernuevo_valor = new Scanner(System.in);

            if (tres_primeros_caracteres.equals("COD") || tres_primeros_caracteres.equals("NOM")){
                String valor_a_cambiar = scannernuevo_valor.nextLine();
                if (tres_primeros_caracteres.equals("COD")) {
                    i.codigo = valor_a_cambiar;
                } else {
                    i.nombre = valor_a_cambiar;
                }
            } else if (tres_primeros_caracteres.equals("MAX") || tres_primeros_caracteres.equals("TOT")) {
                int valor_a_cambiar = scannernuevo_valor.nextInt();
                if (tres_primeros_caracteres.equals("MAX")) {
                    i.max_horas_dia = valor_a_cambiar;
                } else {
                    i.total_horas_semana = valor_a_cambiar;
                }
            } else {
                System.out.println("Error en la introduccion de datos.");
                waiter();
                break;
            }
        }
    }
}
```


```java
public void editar_asignatura(){
    System.out.println("Mostrando lista de asignaturas: ");
    get_asg_codes();
    System.out.println("Introduzca el codigo de la asignatura que desea editar: ");
    Scanner scanner_editar = new Scanner(System.in);
    String cod_a_buscar = scanner_editar.next();
    for (asignatura i: asignaturas) {
        if (i.codigo.equals(cod_a_buscar)) {
            System.out.println("Estos son los detalles de la asignatura solicitada:");
            i.get_details();
            System.out.println("Introduzca los tres primeros caracteres de la variable que desea cambiar: ");
            Scanner scanner_trescaracteres = new Scanner(System.in);
            String tres_primeros_caracteres = scanner_trescaracteres.next();
            System.out.println("Introduzca el nuevo valor de " + tres_primeros_caracteres);
            Scanner scannernuevo_valor = new Scanner(System.in);

            if (tres_primeros_caracteres.equals("COD") || tres_primeros_caracteres.equals("NOM")){
                String valor_a_cambiar = scannernuevo_valor.nextLine();
                if (tres_primeros_caracteres.equals("COD")) {
                    i.codigo = valor_a_cambiar;
                } else {
                    i.nombre = valor_a_cambiar;
                }
            } else if (tres_primeros_caracteres.equals("MAX") || tres_primeros_caracteres.equals("TOT")) {
                int valor_a_cambiar = scannernuevo_valor.nextInt();
                if (tres_primeros_caracteres.equals("MAX")) {
                    i.max_horas_dia = valor_a_cambiar;
                } else {
                    i.total_horas_semana = valor_a_cambiar;
                }
            } else {
                System.out.println("Error en la introduccion de datos.");
                waiter();
                break;
            }
        }
    }
}
```

El *Subapartado 1.3* permite la eliminación de asignaturas que ya hayan sido registradas previamente en el programa. Recorriendo el arrayList de asignaturas y comparando su código con el que el usuario introduce mediante Scanner y, en caso de coincidir, eliminar dicha entrada.

```
185     public void eliminar_asignatura(){
186         System.out.println("Mostrando lista de asignaturas: ");
187         get_asg_codes();
188         System.out.println("Introduzca el codigo de la asignatura que desea eliminar: ");
189         Scanner scanner = new Scanner(System.in);
190         String cod_a_buscar = scanner.next();
191         for (asignatura i: asignaturas) {
192             if (i.codigo.equals(cod_a_buscar)) {
193                 asignaturas.remove(i);
194                 waiter();
195                 break;
196             }
197         }
198     }
```

El *Subapartado 1.4* permite la eliminación de todas las asignaturas.

```
200     public void eliminar_asignaturas(){
201         if(confirm_harmfull()){
202             asignaturas.clear();
203             System.out.println("TODAS Las asignaturas se han eliminado correctamente.");
204         } else {
205             System.out.println("Se ha cancelado la operacion. Volviendo al menu.");
206         }
207         waiter();
208     }
```

Además, debido a que está acción es potencialmente peligrosa, mediante la ejecución del método *confirm_harmfull()* se solicita la reconfirmación al usuario. Con el objetivo de detener la ejecución en caso de ser esto necesario.

```
 !!!   ATENCION   !!!
 La accion que va a realizar es potencialmente peligrosa.
Introduzca 'S' para confirmar, o cualquier otra tecla para continuar.
```

El S*ubapartado 1.5* busca listar las asignaturas y se consigue recorriendo el *arrayList* que almacena asignaturas. Para cada una de ellas llama al método show_id_and_*name()*. Además, se implementa un contador de resultados.

```
106     public void listar_asignaturas(){
107         int n = 0;
108         System.out.println("\n Listando asignaturas...\n");
109         for (asignatura i : asignaturas) {
110             i.show_id_and_name();
111             n++;
112         }
113         System.out.println("  \nSe han encontrado " + n + " asignaturas.");
114         waiter();
115     }
```

Por último, el S*ubapartado 1.6* permite ver los detalles de la asignatura que prefiera el usuario. Para conseguirlo, se listan todos los códigos

```
116         public void get_asg_codes(){
117             System.out.println("Asignaturas registradas en este momento: ");
118             for (asignatura i : asignaturas) {
119                 System.out.println(i.codigo);
120             }
121             System.out.println("Escoja una para ver los detalles: ");
122         }
123         public void detalle_asignatura(){
124             get_asg_codes();
125             Scanner scanner = new Scanner(System.in);
126             String cod_a_buscar = scanner.next();
127             System.out.println("\n");
128             for (asignatura i: asignaturas) {
129                 if (i.codigo.equals(cod_a_buscar)) {
130                     i.get_details();
131                 }
132             }
133             waiter();
134         }
```

Apartado 2: Definición del horario

El *Apartado 2* desarrolla la definición del horario. Este está formado por un total de cinco subapartados mostrados a continuación en la captura adjunta. Cada apartado por separado y en su conjunto harán posible la definición correcta del horario.

```
2. Definir horario:
    2.1. Agregar una entrada al horario
    2.2. Editar una entrada del horario
    2.3. Eliminar una entrada del horario
    2.4. Borrar todo el horario
    2.5. Restaurar horario/Cargar datos de muestra
```

El S*ubapartado 2.1* nos permite añadir una entrada al horario. Va preguntando al usuario las características de la entrada horario (el aula, el día, la hora y la asignatura). Posteriormente, comprueba si todos los valores introducidos son correctos y, de serlo, añade la entrada.

```java
public void add_entrada_horario() {
    System.out.println("Te damos la bienvenida al asistente de creacion de entradas en el horario!");
    int dia_deseado = 0;

    System.out.println("    Introduzca el aula para el que desea agregar una entrada\n Aulas actualmente registradas: ");
    get_aula_codes();
    Scanner scanner_p_aula = new Scanner(System.in);
    String id_aula = scanner_p_aula.next();

    aula aula_deseada = aula_by_id(id_aula);
    if (aula_deseada == null) {
        System.out.println("\n   ATENCION: El parametro introducido para el codigo de aula no es valido. Revise este valor, por favor.");
        waiter();
        return;
    }

    System.out.println("    Introduzca la asignatura para la que desea agregar una entrada\n Asignaturas actualmente registradas: ");
    get_asg_codes();
    Scanner scanner_p_asg = new Scanner(System.in);
    String id_asignatura = scanner_p_asg.next();

    asignatura asignatura_deseada = asignatura_by_id(id_asignatura);
    if (asignatura_deseada == null) {
        System.out.println("El parametro introducido para el codigo de asignatura no es valido. Revise este valor, por favor");
        waiter();
        return;
    }

    System.out.println("    Introduzca el dia para el que desea agregar una entrada\n Valores validos: ");
    System.out.println("'lunes', 'martes', 'miercoles', 'jueves', 'viernes', 'sabado', 'domingo'");
    Scanner scanner_p_dia = new Scanner(System.in);
    String p_dia = scanner_p_dia.next();
```

El S*ubapartado 2.2* consiste en la opción de editar las entradas introducidas en el paso 2.1. La aplicación permitirá crear un apartado donde los alumnos puedan editar las entradas, de tal manera que si su horario ha sufrido algún cambio específico, estos puedan modificar el horario sin problemas.

Así se podrá editar una sola entrada sin que el resto sufran modificaciones. Gracias a esta parte del código: if (aula_deseada == null), si el parámetro introducido para el código de x (aula, asignatura, profesor, etc) no es válido, la aplicación lo detectará y enviará un mensaje al cliente pidiéndole que revise los valores introducidos para evitar errores en la elaboración del horario.

```java
public void edit_entrada_horario() {
    System.out.println("Te damos la bienvenida al asistente de edicion de asignaturas!");
    int dia_deseado = 0;

    System.out.println("    Introduzca el aula para el que desea editar una entrada\n Aulas actualmente registradas: ");
    get_aula_codes();
    Scanner scanner_p_aula = new Scanner(System.in);
    String id_aula = scanner_p_aula.next();

    aula aula_deseada = aula_by_id(id_aula);
    if (aula_deseada == null) {
        System.out.println("\n   ATENCION: El parametro introducido para el codigo de aula no es valido. Revise este valor, por favor.");
        waiter();
        return;
    }

    System.out.println("En el aula indicada anteriormente, " + id_aula + ", se encuentran agregadas las siguentes entradas:");
    aula_deseada.get_lessons();

    System.out.println("    Introduzca el dia para el cual desea editar una entrada\n Valores validos: ");
    System.out.println("      'lunes', 'martes', 'miercoles', 'jueves', 'viernes', 'sabado', 'domingo'");

    Scanner scanner_p_dia = new Scanner(System.in);
    String p_dia = scanner_p_dia.next();
```

Una vez el cliente haya decidido cual es la parte de su código que quiere editar, debe introducir la nueva asignatura por la que desea sustituir la anterior, aplicando un nuevo parámetro válido con el código de la nueva entrada.

```
382     System.out.println("Se va a proceder a editar la asignatura " + aula_deseada.dia[dia_deseado].horas[p_hora].asignatura.nombre);
383     System.out.println("   Es correcta la informacion? (S/N)");
384
385     Scanner scanner_conf_edit = new Scanner(System.in);
386     String conf_edit_param = scanner_conf_edit.next();
387     if (conf_edit_param.equals("S") || conf_edit_param.equals("s")) { // mayusculas o minisculas para facilitar la entrada de datos
388         System.out.println("Introduzca la nueva asignatura por la que desea sustituir " + aula_deseada.dia[dia_deseado].horas[p_hora].asignatura.nombre);
389         Scanner scanner_nueva_asig = new Scanner(System.in);
390         String id_asignatura = scanner_nueva_asig.next();
391         asignatura asignatura_deseada = asignatura_by_id(id_asignatura);
392         if (asignatura_deseada == null) {
393             System.out.println("El parametro introducido para el codigo de asignatura no es valido. Revise este valor, por favor");
394             waiter();
395             return;
396         }
397         if (asignatura_deseada.can_be_scheduled_more(dow_string_to_int(p_dia.toLowerCase()))) {
398             aula_deseada.dia[dia_deseado].horas[p_hora] = new hora(asignatura_deseada);
399             asignatura_deseada.update_scheduled_by_day("+", dow_string_to_int(p_dia.toLowerCase()));
400             System.out.println("Se ha creado correctamente la entrada indicada. Regresando al menu...");
401             waiter();
402             return;
403         } else {
404             System.out.println("No es posible crear una nueva entrada para " + id_asignatura
405                     + " porque se ha alcanzado su limite (" + asignatura_deseada.max_horas_dia + ")");
```

Como vemos el **subapartado 2.3** es bastante parecido a los anteriores, pero en este caso la función que se lleva a cabo no es la de editar si no la de eliminar una de las entradas.

El código comienza dando la bienvenida y utilizar el método scanner scanner para que el cliente pueda introducir el aula para el que desea eliminar una de las entradas que propiamente había introducido al iniciar la aplicación.

Como siempre se utilizar el if (aula_deseada == null), para escribir un mensaje al cliente de que revise el valor añadido en el caso de que el parametro introducido para dicha variable no sea válido.

```
416     public void del_entrada_horario() {
417         System.out.println("Te damos la bienvenida al asistente de eliminacion de entradas en el hoario!");
418         int dia_deseado = 0;
419
420         System.out.println("    Introduzca el aula para el que desea eliminar una entrada\n Aulas actualmente registradas: ");
421         get_aula_codes();
422         Scanner scanner_p_aula = new Scanner(System.in);
423         String id_aula = scanner_p_aula.next();
424
425         aula aula_deseada = aula_by_id(id_aula);
426         if (aula_deseada == null) {
427             System.out.println("\n   ATENCION: El parametro introducido para el codigo de aula no es valido. Revise este valor, por favor.");
428             waiter();
429             return;
430         }
431
432         System.out.println("Mostrando planificacion de aula indicada");
433         aula_deseada.get_lessons();
434
435         System.out.println("    Introduzca el dia para el que desea eliminar una entrada\n Valores validos: ");
436         System.out.println("'lunes', 'martes', 'miercoles', 'jueves', 'viernes', 'sabado', 'domingo'");
437         Scanner scanner_p_dia = new Scanner(System.in);
438         String p_dia = scanner_p_dia.next();
439
440         dia_deseado = dow_string_to_int(p_dia.toLowerCase());
441         // Si la funcion devuelve 99 significa que el parametro introducido no es
442         // valido.
443         if (dia_deseado == 99) {
444             System.out.println("\n   ATENCION:El parametro introducido para el dia de la semana no es valido. Revise este valor, por favor.");
445             waiter();
446             return;
456         }
457         // AQUI se produce la eliminacion de la entrada
458         asignatura asignatura_deseada = aula_deseada.dia[dia_deseado].horas[p_hora].asignatura;
459         asignatura_deseada.update_scheduled_by_day("-", dow_string_to_int(p_dia.toLowerCase()));
460         aula_deseada.dia[dia_deseado].horas[p_hora] = null;
461         System.out.println("Se ha eliminado correctamente la entrada indicada");
462         waiter();
463     }
```

Para eliminar todo el horario **(subapartado 2.4)**. Sera necesario seleccionar todas las entradas, de la misma manera que si se quiere restaurar todo el horario **(subapartado 2.5)**. Se deben de utilizar los códigos nombrados previamente, una vez se haya eliminado el horario anterior, se restaurará uno nuevo utilizando los códigos de "agregar nueva entrada al horario".

```
655
656     public void eliminar_aulas(){
657         if(confirm_harmfull()){
658             aulas.clear();
659             System.out.println("TODAS los AULAS se han eliminado correctamente.");
660         } else {
661             System.out.println("Se ha cancelado la operacion. Volviendo al menu.");
662         }
663         waiter();
664     }
```

```
667     public void editar_aula(){
668         get_aula_codes();
669         System.out.println("\n Introduzca el ID del aula que desea editar:");
670         Scanner scanner_id_aula_a_editar = new Scanner(System.in);
671         String id_aula_a_editar = scanner_id_aula_a_editar.next();
672         if (aula_by_id(id_aula_a_editar) == null) {
673             System.out.println("El ID introducido no se corresponde con ninguno registrado en la base de datos");
674         } else {
675             aula aula_a_editar = aula_by_id(id_aula_a_editar);
676             aula_a_editar.get_details();
677             System.out.println("Desea cambiar el ID o la capacidad? [ID/CAP]");
678             Scanner scanner_cosa_a_cambiar = new Scanner(System.in);
679             String cosa_a_cambiar = scanner_cosa_a_cambiar.next();
680             switch (cosa_a_cambiar){
681                 case "ID": // no se pone brak para que haga de "OR" __ +info https://logfetch.com/java-use-or-switch-ca
682                 case "id":
683                     System.out.println("Introduzca el nuevo ID que desea asignar al aula:");
684                     Scanner scanner_valor_nuevo_id = new Scanner(System.in);
685                     String nuevo_id = scanner_valor_nuevo_id.next();
686                     aula_a_editar.id = nuevo_id;
687                     break;
688                 case "CAP":
689                 case "cap":
690                     System.out.println("Introduzca la nueva capacidad que desea asignar al aula:");
691                     Scanner scanner_valor_nueva_cap = new Scanner(System.in);
692                     int valor_nueva_cap = scanner_valor_nueva_cap.nextInt();
693                     aula_a_editar.update_capacidad(valor_nueva_cap);
694             }
695             System.out.println("Se ha actualizado correctamente. ");
696         }
697         waiter();
```

Apartado 3: Consulta de horario

En este apartado se mostrarán todas las entradas del horario. Además, se facilitará al cliente la búsqueda del horario mediante la asignatura, el aula o el profesor.

Esto permitirá que sea mucho más sencillo encontrar lo que se busca dentro de la aplicación, de tal manera que el cliente se ahorre mucho tiempo en la búsqueda de la información acerca del horario.

Este apartado está compuesto por 4 subapartados que son: búsqueda por asignatura, búsqueda por aula, búsqueda por profesor y mostrar todas las entradas.

```
3. Mostrar horario:
    3.1. Busqueda por asignatura
    3.2. Busqueda por aula
    3.3. Busqueda por profesor
    3.4. Mostrar todas las entradas
```

Los tres primeros apartados utilizan el mismo código. Lo único que los diferencia son los valores que el cliente debe introducir en cada parte del código, pues para el primero deberá introducir el nombre de la asignatura, en el segundo el número del aula y en el tercero el nombre del profesor.

Para poder acceder a la información el alumno deberá introducir su DNI, para autentificar que es alumno de esa asignatura. En un momento determinado la aplicación le pedirá que ingrese dicho número del DNI y el propio código identificará mediante scanner Scanner que este es correcto, por el contrario, se utilizara un if *(docente_by_dni(dni_docente_a_listar) == null)* para que en el caso de que el DNI de dicho alumno no sea encontrado en las bases de datos, la aplicación informe al cliente de que el DNI introducido no corresponde son ningún alumno, si esto ocurre la aplicación volverá al menú de forma inmediata.

```
466     public void get_entradas(){
467         for (aula a: aulas){
468             System.out.println("Para el aula codigo " + a.id + ":");
469             a.get_lessons();
470             System.out.println("\n================\n");
471         }
472         waiter();
473     }
474 
475     public void get_entradas_by_aula(){
476         get_aula_codes();
477         System.out.println("Introduzca el codigo del aula a listar:");
478         Scanner scanner_aula_a_listar = new Scanner(System.in);
479         String aula_a_listar = scanner_aula_a_listar.next();
480         if (aula_by_id(aula_a_listar) == null){
481             // Si entra en null significa que no hay ningun aula con ese ID dada de alta. Por tanto, no hay nada que listar.
482             System.out.println("El ID introducido no se corresponde con ningun aula registrada. Volviendo al menu");
483             waiter();
484             return;
485         } else {
486             aula au_a_listar = aula_by_id(aula_a_listar);
487             for (aula a: aulas) {
488                 if (a == au_a_listar){
489                     a.get_lessons();
490                 }
491             }
492         }
493         waiter();
```

```java
    public void get_entradas_by_asg() {
        System.out.println("Mostrando asignaturas...");
        listar_asignaturas();
        System.out.println("Introduzca el codigo de la asignatura a listar:");
        Scanner scanner_asg_a_listar = new Scanner(System.in);
        String asg_a_listar = scanner_asg_a_listar.next();
        if (asignatura_by_id(asg_a_listar) == null) {
            // Si entra en null significa que no hay ninguna asignatura con ese ID dada de
            // alta. Por tanto, no hay nada que listar.
            System.out.println("El ID introducido no se corresponde con ninguna asignatura. Volviendo al menu");
            waiter();
            return;
        } else {
            int found = 0;
            asignatura asignatura_para_listar = asignatura_by_id(asg_a_listar);
            //                  G109   miercoles    9:00 - 10:00
            System.out.println("    AULA |   DIA    |  HORA");
            for (aula a : aulas) {
                for (dia d : a.dia) {
                    int i = 0;
                    for (hora h: d.horas) {
                        if (h != null) {
                            if (h.asignatura == asignatura_para_listar) {
                                System.out.println(
                                        "     " + a.id + "    " + d.nombre + "     " + i + ":00 - " + (i + 1) + ":00");
                                found++;
                            }
                        }
                        i++;
                    }
                }
            }
            if (found == 0) {
                System.out.println("\n  --- No se han encontrado resultados ---");
            } else {
                System.out.println("\n   Se han encontrado " + found + " resultados. ");
            }
            waiter();
        }
    }

    public void get_entradas_by_docente(){
        get_docente_listado();
        System.out.println("\nIntroduzca DNI del docente para el que desea ver las clases:");
        Scanner scanner_dni_docente_a_listar = new Scanner(System.in);
        String dni_docente_a_listar = scanner_dni_docente_a_listar.next();
        if (docente_by_dni(dni_docente_a_listar) == null) {
            // Si entra en null significa que no hay ningun docente con ese DNI dada de alta. Por tanto, no hay nada que listar.
            System.out.println("El DNI introducido no se corresponde con ningun docente registrado. Volviendo al menu");
            return;
        } else {
            int found = 0;
            docente docente_a_listar = docente_by_dni(dni_docente_a_listar);
            for (aula a: aulas) {
                for (dia d: a.dia){
                    int i = 0;
                    for (hora h: d.horas){
                        if (h != null) {
                            if (h.asignatura.docente == docente_a_listar) {
                                //                              aula          nombre_dia                  hora                    asignatura
                                System.out.println("    " + a.id + "   " + d.nombre + "    " + i + ":00 - " + (i + 1) + ":00    " + h.asignatura.codigo);
                                found++;
                            }
                        }
                    }
                }
            }
            if (found == 0) {
                System.out.println("\n  --- No se han encontrado resultados ---");
            } else {
                System.out.println("\n   Se han encontrado " + found + " resultados. ");
            }
        }
        waiter();
    }
```

De igual manera sucederá con la introducción del aula para la búsqueda dentro del horario. Si los datos introducidos no corresponden con ID indicado, la aplicación regresara al apartado de menú de forma inmediata.

```
592     public void get_detalle_aula(){
593         get_aula_codes();
594         System.out.println("Introzca el codigo del aula para la que desea ver los detalles: ");
595         // TODO: Check if requested exists. Maybe return an array?
596         Scanner scanner = new Scanner(System.in);
597         String cod_a_buscar = scanner.next();
598         aula aula_a_ver_detalles = aula_by_id(cod_a_buscar);
599         if (aula_a_ver_detalles == null) {
600             System.out.println("No se encuetra ningun aula con el ID indicado. Volviendo al menu...");
601         } else {
602             aula_a_ver_detalles.get_details();
603         }
604         waiter();
605     }
606 
607     public void get_detalle_aulas(){
608         System.out.println("Listado de aulas registradas en este momento: ");
609         for (aula i: aulas) {
610                 i.get_details();
611                 System.out.println("");
612             }
613         waiter();
614         }
615 
616     public aula aula_by_id(String p_id){
617         // System.out.println("Entro en method");
618         for (aula a: aulas) {
619             // System.out.println("Recorriendo...");
620             if (p_id.equals(a.id)){
621                 // System.out.println("Encontrado");
622                 // a.get_details();
623                 return a;
```

Apartado 4: Gestión de aulas

Ahora vamos a explicar el apartado 4, es el de la gestión de aulas. Consta de 5 subapartados, estos son: agregar un aula a la base de datos, editar un aula ya creada, eliminar un aula de la base de datos, eliminar TODAS las aulas y listar las aulas disponibles.

```
4. Gestion aulas:
    4.1. Agregar aula a la base de datos
    4.2. Editar aula ya creada
    4.3. Eliminar aula de la base de datos
    4.4. Eliminar TODAS las aulas.
    4.5. Listar aulas disponibles
```

El **subapartado 4.1** consiste en agregar un aula a la base de datos. Dicho código lo encontramos en la clase “horario” y gracias al scanner el usuario puede interactuar primero escribiendo el nuevo aula ejemplo: G398 y luego escribiendo la capacidad.

```java
public void aula_add(){
    System.out.println("Bienvenido al asistente de adicion de aulas.");
    System.out.println("A continuacion se muestran las aulas ya existentes:");
    get_aula_codes();
    System.out.println("Introduzca el ID de la nueva aula a crear:");
    Scanner scannercreacionaula = new Scanner(System.in);
    String creacion_aula = scannercreacionaula.next();
    System.out.println("Indique la capacidad para aula " + creacion_aula + ":");
    Scanner scannercapacidad = new Scanner(System.in);
    int capacidad_aula_nueva = scannercapacidad.nextInt();
    aulas.add(new aula(creacion_aula, capacidad_aula_nueva));
    System.out.println("Su aula ha sido creada con exito.");
    waiter();
}
```

En el **subapartado 4.2** queremos editar un aula ya creada, para empezar, introducimos el aula que deseamos editar tienen que ser estas G107 G108 G109 G110 G111 ya que son las clases existentes, si por ejemplo escribimos H221 gracias a la función “if” el programa nos devolvería “el ID introducido no se corresponde con ninguno registrado en la base de datos”. Por otra parte, si escribimos G107 gracias a la función “else” el programa nos devuelve "¿Desea cambiar el ID o la capacidad? [ID/CAP]"

```java
public void editar_aula(){
    get_aula_codes();
    System.out.println("\n Introduzca el ID del aula que desea editar:");
    Scanner scanner_id_aula_a_editar = new Scanner(System.in);
    String id_aula_a_editar = scanner_id_aula_a_editar.next();
    if (aula_by_id(id_aula_a_editar) == null) {
        System.out.println("El ID introducido no se corresponde con ninguno registrado en la base de datos");
    } else {
        aula aula_a_editar = aula_by_id(id_aula_a_editar);
        aula_a_editar.get_details();
        System.out.println("Desea cambiar el ID o la capacidad? [ID/CAP]");
        Scanner scanner_cosa_a_cambiar = new Scanner(System.in);
        String cosa_a_cambiar = scanner_cosa_a_cambiar.next();
```

Entonces el usuario dependiendo de que escriba ID o CAP el programa devuelve una frase u otra, y al final si todo ha ido correctamente aparecerá: "Se ha actualizado correctamente."

```java
        switch (cosa_a_cambiar){
            case "ID":
            case "id":
                System.out.println("Introduzca el nuevo ID que desea asignar al aula:");
                Scanner scanner_valor_nuevo_id = new Scanner(System.in);
                String nuevo_id = scanner_valor_nuevo_id.next();
                aula_a_editar.id = nuevo_id;
                break;
            case "CAP":
            case "cap":
                System.out.println("Introduzca la nueva capacidad que desea asignar al aula:");
                Scanner scanner_valor_nueva_cap = new Scanner(System.in);
                int valor_nueva_cap = scanner_valor_nueva_cap.nextInt();
                aula_a_editar.update_capacidad(valor_nueva_cap);
        }
        System.out.println("Se ha actualizado correctamente. ");
    }
    waiter();
}
```

El **subapartado 4.3**, trata de eliminar un aula de la base de datos, mediante scanner escribes el aula que deseas eliminar y a continuación aparecerá "El aula ha sido eliminada correctamente"

```
628     public void aula_delete(){
629         System.out.println("Por favor, introduzca el identificador del aula que desea eliminar:");
630 
631         Scanner scanner = new Scanner(System.in);
632         String id_aula_pasado_por_scanner = scanner.next();
633         aula aula_a_eliminar = aula_by_id(id_aula_pasado_por_scanner);
634         aulas.remove(aula_a_eliminar);
635 
636         System.out.println("El aula ha sido eliminada correctamente.");
637         waiter();
638     }
```

En el **subapartado 4.4** se busca eliminar todas las aulas, el código es igual que en el subapartado 1.4 solo cambia asignaturas por aulas.

```
656     public void eliminar_aulas(){
657         if(confirm_harmfull()){
658             aulas.clear();
659             System.out.println("TODAS los AULAS se han eliminado correctamente.");
660         } else {
661             System.out.println("Se ha cancelado la operacion. Volviendo al menu.");
662         }
663         waiter();
664     }
```

En **el subapartado 4.5**, que es el último, queremos un listado de las aulas. Y lo conseguimos gracias a los códigos que están en las clases:

"horario"

```
607     public void get_detalle_aulas(){
608         System.out.println("Listado de aulas registradas en este momento: ");
609         for (aula i: aulas) {
610                 i.get_details();
611                 System.out.println("");
612             }
613         waiter();
614         }
```

"aula"

```
16     public void get_details(){
17         System.out.println("Codigo (id):         " + this.id);
18         System.out.println("Capacidad:          " + this.capacidad);
19         System.out.println("Registros anadidos:  " + this.registros_anadidos);
20     }
```

Apartado 5: Gestión de docentes

El siguiente apartado es el apartado 5, este apartado es el de la gestión de docentes. Este apartado consta de 5 subapartados, estos son: agregar un docente a la base de datos, editar un docente ya registrado, eliminar un docente de la base de datos, eliminar TODOS los docentes y listar a los docentes registrados disponibles.

Cada apartado hará uso de las clases para conseguir la información, ya que hay información que el programa necesita que se encuentra en public y private.

```
5. Gestion docentes:
    5.1. Agregar docente a la base de datos
    5.2. Editar docente ya registrado
    5.3. Eliminar docente de la base de datos
    5.4. Eliminar TODOS los docentes.
    5.5. Listar docentes registrados disponibles
```

El **subapartado 5.1** trata de agregar un docente a la base de datos. La programación para elaborar este apartado lo podemos encontrar en la clase horario, en esta clase, entre la línea 759 y la línea 793 se encuentra el código para agregar un nuevo docente a la base de datos, se hace uso de un scanner para que el usuario pueda interactuar con el programa y de esta forma proporcionar los datos necesarios al programa para la correcta agregación del docente.

Además, hemos querido darle una opción al usuario en caso de que introduzca un DNI de un docente el cual se encuentra ya en la base de datos, esto lo hemos conseguido haciendo uso del método if, else.

```
759     public void add_docente(){
760         System.out.println("\nTe damos la bienvenida al asistente de adicion del equipo docente!");
761         System.out.println("Introduzca, por favor, el DNI del nuevo docente que desea dar de alta");
762         Scanner scanner_add_doce_dni = new Scanner(System.in);
763         String add_doce_dni = scanner_add_doce_dni.next();
764         if (docente_by_dni(add_doce_dni) == null){
765             // Si entra en null significa que no hay ningun docente dado de alta. Por tanto, podemos annadirlo.
766             System.out.println("Introduzca, por favor, los datos que se le solicitan. \n   NOMBRE:");
767             Scanner scanner_add_doce_nombre = new Scanner(System.in);
768             String add_doce_nombre = scanner_add_doce_nombre.nextLine();
769
```

```
                System.out.println("   Primer Apellido: ");
                Scanner scanner_add_doce_ape1 = new Scanner(System.in);
                String add_doce_ape1 = scanner_add_doce_ape1.nextLine();

                System.out.println("   Segundo Apellido: ");
                Scanner scanner_add_doce_ape2 = new Scanner(System.in);
                String add_doce_ape2 = scanner_add_doce_ape2.nextLine();

                System.out.println("   Sueldo (formato 1000.0): ");
                Scanner scanner_add_doce_sueldo = new Scanner(System.in);
                Double add_doce_sueldo = scanner_add_doce_sueldo.nextDouble();

                System.out.println("   Titulo: ");
                Scanner scanner_add_doce_tit = new Scanner(System.in);

                String add_doce_tit = scanner_add_doce_tit.nextLine();

                // Seria interesante hacer aqui comprobaciones con hasNextXXXX en los Scanners para evitar que java devuelva errores feos...

                docentes.add(new docente(add_doce_dni, add_doce_nombre, add_doce_ape1, add_doce_ape2, add_doce_sueldo, add_doce_tit));

                System.out.println("Se ha annadido correctamente el docente");
                waiter();

            } else {
                // si entra aqui significa que ya hay un/una docente con ese DNI. por tanto, no podemos annadirlo.
                // Mostramos tambien el nombre del registro ya annadido para ayudar al usuario.
                System.out.println("ERROR: ya existe un registro para ese DNI: " + docente_by_dni(add_doce_dni).get_fullname());
                System.out.println("Puede, si asi lo desea, eliminar el registro o modificarlo desde las diferentes opciones del menu.");
                waiter();
            }

        }
```

En el **subapartado 5.2** buscamos editar un docente ya registrado, ya que durante el proceso de agregación de dicho docente puede ocurrir un error. En este apartado hacemos uso del DNI del docente, de nuevo hacemos uso del método if,else en caso de que el DNI no exista, aparecerá por pantalla null (if), en caso de que si exista el DNI (else) se ejecutará la segunda parte del programa. De nuevo, se hace uso del scanner para que el usuario pueda interactuar con el programa. Estas líneas de código de 815-863.

```
    public void editar_docente(){
        System.out.println("Mostrando lista de docentes: ");
        get_listado_docente_name_and_dni();
        System.out.println("Introduzca el DNI del docente que desea editar: ");
        Scanner scanner = new Scanner(System.in);
        String doce_a_buscar = scanner.next();
        if (docente_by_dni(doce_a_buscar) == null){
            // Si es null el dni introducido no existe
            System.out.println("El DNI introducido no se corresponde con ninguno registrado en la base de datos");
        } else {
            docente docente_a_modificar = docente_by_dni(doce_a_buscar);
            System.out.print("Los datos actuales del docente son:");
            docente_a_modificar.get_details();
            System.out.println("Introduzca la clave del campo a variar\nLa claves estan formadas por las 3-4 primeras letras mayusculas mostradas en el listado");
            Scanner scanner_codigo_a_cambiar = new Scanner(System.in);
            String codigo_a_cambiar = scanner_codigo_a_cambiar.next();
            System.out.println("\nIntroduzca el nuevo contenido que desea actualizar en el campo indicado: ");
            Scanner scanner_valor_a_cambiar = new Scanner(System.in);
            switch (codigo_a_cambiar) {
                case "DNI":
                    String valor_a_cambiar = scanner_valor_a_cambiar.next();
                    docente_a_modificar.dni = valor_a_cambiar;
                    break;
                case "NOM":
                    String valor_a_cambiar_n = scanner_valor_a_cambiar.next(); // Se cambia el nombre porque si no chilla...
                    docente_a_modificar.update_nombre(valor_a_cambiar_n);
                    break;
                case "APE1":
                    String valor_a_cambiar_a1 = scanner_valor_a_cambiar.next();
                    docente_a_modificar.update_ape1(valor_a_cambiar_a1);
                    break;
                case "APE2":
                    String valor_a_cambiar_a2 = scanner_valor_a_cambiar.next();
                    docente_a_modificar.update_ape2(valor_a_cambiar_a2);
                    break;
                case "SUE":
                    Double valor_a_cambiar_s = scanner_valor_a_cambiar.nextDouble();
                    docente_a_modificar.update_sueldo(valor_a_cambiar_s);
                    break;
                case "TIT":
                    String valor_a_cambiar_t = scanner_valor_a_cambiar.next();
                    docente_a_modificar.update_titulo(valor_a_cambiar_t);
                    break;
                default:
                    System.out.println("El valor introducido no se corresponde con ninguno de los permitodos.\nVolviendo al menu.");
                    // Aqui no se llama a waiter porque esta al final.
            }
        }
        waiter();
```

En el **subapartado 5.3**, se trata de eliminar un docente de la base de datos, en este caso al igual que los dos anteriores apartados, hacemos uso de los if, else, en el cual si el usuario introduce el DNI del docente y no se encuentra directamente te devuelve al menú indicando cuál ha sido el problema y esperando que pulse cualquier tecla para continuar(if), en caso de que se encuentre, el docente será eliminado y se imprimirá por pantalla “El docente se ha dado de baja correctamente” (else).

```
734     public void docente_delete(){
735         System.out.println("Por favor, introduzca el DNI del docente que desee dar de baja:");
736         Scanner scanner = new Scanner(System.in);
737         String docente_de_baja = scanner.next();
738         docente docente_a_eliminar = docente_by_dni(docente_de_baja);
739         if (docente_a_eliminar == null) {
740             System.out.println("El docente que ha introducido no se encuentra. Volviendo al menu.");
741             waiter();
742         } else {
743             docentes.remove(docente_a_eliminar);
744             System.out.println("El docente se ha dado de baja correctamente.");
745             waiter();
746         }
747     }
```

En el **subapartado 5.4** se busca eliminar todos los docentes, esta línea de código se encuentra entre la línea 805 y 812. Se usa igual que en los anteriores apartados el método if,else. Si es true (el usuario ha confirmado que quiere continuar) seguirá la ejecución.

Si no confirm_harmfull retornará el boolean false y entrará en el else. Se usa así porque esta acción es peligrosa. Se llama al método en todas las acciones especialmente dañinas. Se puede restaurar la "base de datos" con una opción del menú (cabe destacar en este punto que la aplicación no realiza ninguna conexión con una base de datos al uso, y no es más que una forma de referiste a la estructura de datos en la que se almacena el horario).

```
805     public void eliminar_docentes(){
806         if(confirm_harmfull()){
807             docentes.clear();
808             System.out.println("TODOS los DOCENTES se han eliminado correctamente.");
809         } else {
810             System.out.println("Se ha cancelado la operacion. Volviendo al menu.");
811         }
812         waiter();
813     }
```

Para acabar con la clase docentes, tenemos el **subapartado 5.5** usamos un bloque for para reproducir los listados de docentes y DNI.

```
707     public void get_docente_dnis(){
708         System.out.println("DNIs registrados en este momento: ");
709         for (docente i: docentes) {
710             System.out.println(i.get_dni());
711         }
712         waiter();
713     }
714 
715     public void get_docente_listado(){
716         System.out.println("DNIs registrados en este momento: ");
717         for (docente i: docentes) {
718             i.get_details();
719             System.out.println("");
720         }
721         waiter();
722     }
723 
724     public void get_listado_docente_name_and_dni(){
725         for (docente d: docentes){
726             System.out.println(d.get_fullname_and_dni());
727         }
728     }
```

Apartado 6: Cierre de la aplicación

Anexo I. Código fuente

Nótese que el código aquí mostrado ha podido sufrir modificaciones para adaptarlo al formato del documento y facilitar su lectura. En caso de duda, consúltese la dirección de *Codeboard* proporcionada.

Asignatura.java

```
public class Asignatura {
    public String codigo;
    public String nombre;
    public int total_horas_semana;
    public int max_horas_dia;
    public Docente docente;
    private int horas_actual;
    private int[] asg_dia = new int[] { 0, 0, 0, 0, 0, 0, 0 };

    public Asignatura(String cod, String nom, int ths, int mhd, Docente doc)
{
        this.codigo = cod;
        this.nombre = nom;
        this.total_horas_semana = ths;
        this.max_horas_dia = mhd;
        this.horas_actual = 0;
        this.docente = doc;
        int[] asg_dia = new int[7];
    }

    public void update_scheduled_by_day(String operacion, int dow) {
        if (operacion.equals("+")) {
            asg_dia[dow]++;
        } else if (operacion.equals("-")) {
            asg_dia[dow]--;
        } else {
            System.out.println("Error en la modificacion.");
        }
    }

    public void get_details() {
        System.out.println("CODigo:              " + this.codigo);
        System.out.println("NOMbre:              " + this.nombre);
        System.out.println("TOTal Horas Semana:  " +
this.total_horas_semana);
        System.out.println("MAXimo Horas Dia:    " + this.max_horas_dia);
    }
    // Este metodo sirve para comprobar si puedo annadir mas horas a nivel
    // semanal a esta asignatura.
    // Devuelve true si puedo, porque el numero de horas actual es menor que
el
    // maximo
    // Devielve false si no puedo, porque ya he alcanzando el numero maximo.

    public boolean can_be_scheduled_more(int dow) {
        if (asg_dia[dow] < max_horas_dia) {
            return true;
        } else {
            return false;
        }
    }
```

```
    public void show_id_and_name() {
        System.out.println(this.codigo + ": " + this.nombre);
    }
}
```

Aula.java

```
public class Aula {
    public String id;
    private int capacidad;
    public Dia[] dia = new Dia[] { new Dia("lunes"), new Dia("martes"), new
Dia("miercoles"),
            new Dia("jueves"), new Dia("viernes"), new Dia("sabado"), new
Dia("domingo") };
    public int registros_anadidos;

    public Aula(String p_id, int p_csp) {
        id = p_id;
        // Problema: se deben poder manejar tantos horarios como aulas tenga!
        // Solucion: El aula tiene los dias.
        capacidad = p_csp;
        registros_anadidos = 0;
    }

    public void get_details() {
        System.out.println("Codigo (id):          " + this.id);
        System.out.println("Capacidad:            " + this.capacidad);
        System.out.println("Registros anadidos:   " + this.registros_anadidos);
    }

    public void get_lessons() {
        int found = 0;
        for (Dia d : dia) {
            System.out.println("    Dia: " + d.nombre);
            int i = 0;
            for (Hora h : d.horas) {
                if (h != null) {
                    System.out.println("        Hora:           " + i + ":00 - " +
                                       (i + 1) + ":00");

                   System.out.println("        Asignatura:     " +
                                      h.asignatura.codigo + ": " +
                                      h.asignatura.nombre
                                      + "\n           -");
                    found++;
                }
                i++;
            }
            System.out.println("    -----------------");
        }
        System.out.println("Se han encontrado " + found + " resultados para este
aula.");
    }

    public void update_capacidad(int p_nueva_capacdad) {
        this.capacidad = p_nueva_capacdad;
    }
}
```

Dia.java

```
public class Dia {
    public String nombre;
    public Hora[] horas = new Hora[24];

    // Constructor
    public Dia(String nom) {
        nombre = nom;
        Hora[] horas = new Hora[24];
    }

    public Dia() {

    }

    public Hora get_entrada(int num) {
        return this.horas[num];
    }
}
```

Docente.java

```
public class Docente {
    public String dni;
    private String nombre;
    private String apellido1;
    private String apellido2;
    private double sueldo;
    private String titulo;

    public Docente(String p_dni, String p_nom, String p_ape1,
                   String p_ape2, Double p_sueld, String p_tit) {
        dni = p_dni;
        nombre = p_nom;
        apellido1 = p_ape1;
        apellido2 = p_ape2;
        sueldo = p_sueld;
        titulo = p_tit;
    }

    public String get_fullname() {
        return (this.nombre + " " + this.apellido1 + " " + this.apellido2);
    }


    // Este metodo no tiene mucho sentido pero se ha utilizado durante la
    // realizacion del trabajo para la ilustracion de diferencias
    // private/public en las propiedades de la clase.
    // Se mantiene por si fuera necesario usarlo posteriormente.

    public String get_dni() {
        return this.dni;
    }
```

```
    public String get_fullname_and_dni() {
        return (this.get_fullname() + ": " + this.dni);
    }


    public void get_details() {
        System.out.println("   DNI:          " + this.dni);
        System.out.println("      NOMbre:    " + this.nombre);
        System.out.println("      APE1:      " + this.apellido1);
        System.out.println("      APE2:      " + this.apellido2);
        System.out.println("      SUEldo:    " + this.sueldo);
        System.out.println("      TITulo:    " + this.titulo);
    }


    // Estos metodos sirven para ilustrar que son necesarios metodos para
    // actualizar
    // propiedades privadas.


    public void update_nombre(String p_introducido) {
        this.nombre = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }


    public void update_ape1(String p_introducido) {
        this.apellido1 = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }


    public void update_ape2(String p_introducido) {
        this.apellido2 = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }


    public void update_sueldo(Double p_introducido) {
        this.sueldo = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }


    public void update_titulo(String p_introducido) {
        this.titulo = p_introducido;
        System.out.println("Se ha actualizado correctamente");
    }

}
```

Hora.java

```
public class Hora {
    // public docente docente;
    public Asignatura asignatura;

    public Hora() {
    }

    public Hora(Asignatura asg) {
        asignatura = asg;
        asg.total_horas_semana++;
    }
}
```

Horario.java

```
import java.util.ArrayList;
import java.util.Scanner;
import java.util.List;

public class Horario {
    List<Asignatura> asignaturas = new ArrayList<Asignatura>();
    List<Aula> aulas = new ArrayList<Aula>();
    List<Docente> docentes = new ArrayList<Docente>();
    String nombre;
    int length_semana = 0;

    public Horario(String nomhorario) {
        nombre = nomhorario;
    }


    // Convierte dias de la semana, dow, desde strings hacia int

    public int dow_string_to_int(String p_string) {
        switch (p_string) {
            case "lunes":
                return 0;
            case "martes":
                return 1;
            case "miercoles":
                return 2;
            case "jueves":
                return 3;
            case "viernes":
                return 4;
            case "sabado":
                return 5;
            case "domingo":
                return 6;
            default:
                return 99;
        }
    }
```

```
    // Esto sirve unicamente para esperar hasta que el usuario pulse enter.
    // Como se utiliza en varias ocasiones, se crea como metodo y se llama a
    // este segun sea necesario

    public void waiter() {
        System.out.println("\n >>> Pulse enter para continuar... <<< \n");
        Scanner scanner_enter = new Scanner(System.in);
        String dummy_scanner_content = scanner_enter.nextLine();
         // nexLine en lugar de 'next' para que admita el enter
         // sin letras ni espacios.
        return;
    }


    public boolean confirm_harmfull() {
        System.out.println("\n !!!   ATENCION   !!! \n La accion que va a realizar es
potencialmente peligrosa.");
        System.out.println("Introduzca 'S' para confirmar, o cualquier otra tecla para
continuar.");

        Scanner scanner_conf = new Scanner(System.in);
        String conf_edit_param = scanner_conf.next();
        if (conf_edit_param.equals("S") || conf_edit_param.equals("s")) {
          // mayusculas o minisculas para facilitar
la
          // entrada de datos
            return true;
        } else {
            return false;
        }
    }

    public void ClearScreen() {
        System.out.print("");
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }
    // https://www.delftstack.com/es/howto/java/java-clear-console/

    /*
     * ***********************************************
     * ***********************************************
     *              Clases ASIGNATURA
     * ***********************************************
     * ***********************************************
     */

    public void nueva_asignatura() {
        System.out.println("Bienvenido al proceso de creacion de asignaturas");

        System.out.println("Por favor, introduzca los siguientes
                           datos:\nCodigo:");
        Scanner scanner_add_asg_cod = new Scanner(System.in);
        String add_asg_cod = scanner_add_asg_cod.nextLine();

        System.out.println("Nombre de la asignatura");
        Scanner scanner_add_asg_nom = new Scanner(System.in);
        String add_asg_nom = scanner_add_asg_nom.nextLine();

        System.out.println("Total de horas a la semana:");
        Scanner scanner_add_asg_ths = new Scanner(System.in);
        int add_asg_ths = scanner_add_asg_ths.nextInt();

        System.out.println("Maximo horas al dia:");
        Scanner scanner_add_asg_mhd = new Scanner(System.in);
```

```
        int add_asg_mhd = scanner_add_asg_mhd.nextInt();

        System.out.println("Docente que impartira la nueva asignatura");
        System.out.println("  > Como referencia, se muestran a continuacion
                           los DNIs de los docentes.");
        get_listado_docente_name_and_dni();
        Scanner scanner_add_asg_doc = new Scanner(System.in);
        String add_asg_doc = scanner_add_asg_doc.nextLine();

        asignaturas.add(new Asignatura(add_asg_cod, add_asg_nom, add_asg_ths,
                                add_asg_mhd, docente_by_dni(add_asg_doc)));

        System.out.println("Se ha creado correctamente la asignatura.");
        waiter();
    }

    public void asignatura_auto(String cod, String nom, int ths, int mhd,
                                String dni_doc) {

         asignaturas.add(new Asignatura(cod, nom, ths, mhd,
                         docente_by_dni(dni_doc)));
    }

    public void listar_asignaturas() {
        int n = 0;
        System.out.println("\n Listando asignaturas...\n");
        for (Asignatura i : asignaturas) {
            i.show_id_and_name();
            n++;
        }
        System.out.println("  \nSe han encontrado " + n + " asignaturas.");
        waiter();
    }

    public void get_asg_codes() {
        System.out.println("Asignaturas registradas en este momento: ");
        for (Asignatura i : asignaturas) {
            System.out.println(i.codigo);
        }
    }

    public void detalle_asignatura() {
        get_asg_codes();
        Scanner scanner = new Scanner(System.in);
        String cod_a_buscar = scanner.next();
        System.out.println("\n");
        for (Asignatura i : asignaturas) {
            if (i.codigo.equals(cod_a_buscar)) {
                i.get_details();
            }
        }
        waiter();
    }



    public void editar_asignatura() {
        System.out.println("Mostrando lista de asignaturas: ");
        get_asg_codes();
        System.out.println("Introduzca el codigo de la asignatura que desea
                           editar: ");
        Scanner scanner_editar = new Scanner(System.in);
        String cod_a_buscar = scanner_editar.next();
```

```
        for (Asignatura i : asignaturas) {
            if (i.codigo.equals(cod_a_buscar)) {
                System.out.println("Estos son los detalles de la asignatura
                                    solicitada:");
                i.get_details();
                System.out.println("Introduzca los tres primeros caracteres de
                                    la variable que desea cambiar: ");
                Scanner scanner_trescaracteres = new Scanner(System.in);
                String tres_primeros_caracteres =
                                                scanner_trescaracteres.next();
                System.out.println("Introduzca el nuevo valor de " +
                                   tres_primeros_caracteres);
                Scanner scannernuevo_valor = new Scanner(System.in);

                if (tres_primeros_caracteres.equals("COD") ||
                    tres_primeros_caracteres.equals("NOM")) {
                    String valor_a_cambiar = scannernuevo_valor.nextLine();
                    if (tres_primeros_caracteres.equals("COD")) {
                        i.codigo = valor_a_cambiar;
                    } else {
                        i.nombre = valor_a_cambiar;
                    }
                } else if (tres_primeros_caracteres.equals("MAX") ||
                           tres_primeros_caracteres.equals("TOT")) {
                    int valor_a_cambiar = scannernuevo_valor.nextInt();
                    if (tres_primeros_caracteres.equals("MAX")) {
                        i.max_horas_dia = valor_a_cambiar;
                    } else {
                        i.total_horas_semana = valor_a_cambiar;
                    }
                } else {
                    System.out.println("Error en la introduccion de datos.");
                    waiter();
                    break;
                }
            }
        }
    }


    public Asignatura asignatura_by_id(String p_id) {
        for (Asignatura a : asignaturas) {
            if (p_id.equals(a.codigo)) {
                return a;
            }
        }
        return null;
    }


    public void eliminar_asignatura() {
        System.out.println("Mostrando lista de asignaturas: ");
        get_asg_codes();
        System.out.println("Introduzca el codigo de la asignatura que desea
                            eliminar: ");
        Scanner scanner = new Scanner(System.in);
        String cod_a_buscar = scanner.next();
        for (Asignatura i : asignaturas) {
            if (i.codigo.equals(cod_a_buscar)) {
                asignaturas.remove(i);
                waiter();
                break;
            }
        }
    }

    public void eliminar_asignaturas() {
```

```
        if (confirm_harmfull()) {
            asignaturas.clear();
            System.out.println("TODAS Las asignaturas se han eliminado
                               correctamente.");
        } else {
            System.out.println("Se ha cancelado la operacion. Volviendo al
                               menu.");
        }
        waiter();
    }


    /*
     * ***********************************************
     * ***********************************************
     *           Clases HORAS / ENTRADAS HORARIO
     * ***********************************************
     * ***********************************************
     */

    // TODO: PENDING -- update value horas_actual at asignatura on ADD

    public void add_entrada_horario_auto(String p_dia, String id_aula,
                              String id_asignatura, int p_hora) {
        int dia_deseado = 0;
        Aula aula_deseada = aula_by_id(id_aula);
        if (aula_deseada == null) {
            System.out.println(
                  "\n   ATENCION: El parametro introducido para el
                  codigo de aula no es valido. Revise este valor, por favor.");
            return;
        }
        Asignatura asignatura_deseada = asignatura_by_id(id_asignatura);
        if (asignatura_deseada == null) {
            System.out.println(
                    "El parametro introducido para el codigo
                     de asignatura no es valido. Revise este valor,
                     por favor");
            return;
        }

        dia_deseado = dow_string_to_int(p_dia.toLowerCase());
        // Si la funcion devuelve 99 significa que el parametro introducido no
        // es válido.
        if (dia_deseado == 99) {
            System.out.println(
                    "\n   ATENCION:El parametro introducido para el dia de la semana no
                       es valido. Revise este valor, por favor.");
            return;
        }
        // Si la asignatura no ha alcanzado su limite de 'ocupacion', permitira la
        // creacion de la hora. En caso contrario, else, dara mensaje de error y
        // retornara.
        if
(asignatura_deseada.can_be_scheduled_more(dow_string_to_int(p_dia.toLowerCase()))) {
            aula_deseada.dia[dia_deseado].horas[p_hora] = new Hora(asignatura_deseada);
            asignatura_deseada.update_scheduled_by_day("+",
            dow_string_to_int(p_dia.toLowerCase()));

        } else {



            System.out.println("No es posible crear una nueva entrada para " +
                               id_asignatura
                               + " porque se ha alcanzado su limite (" + a
```

```
                                signatura_deseada.max_horas_dia + ")");
            return;
            // return sirve para volver al menu, o a lo que ha llamado a la funcion
            // inicialmente
        }
        aula_deseada.registros_anadidos++;
    }

    public void add_entrada_horario() {
        System.out.println("Te damos la bienvenida al asistente de creacion de
                           entradas en el horario!");
        int dia_deseado = 0;

        System.out.println(
                "    Introduzca el aula para el que desea agregar una entrada\n
                 Aulas actualmente registradas: ");
        get_aula_codes();
        Scanner scanner_p_aula = new Scanner(System.in);
        String id_aula = scanner_p_aula.next();

        Aula aula_deseada = aula_by_id(id_aula);
        if (aula_deseada == null) {
            System.out.println(
                    "\n   ATENCION: El parametro introducido para el codigo de aula
                       no es valido. Revise este valor, por favor.");
            waiter();
            return;
        }

        System.out.println(
                "    Introduzca la asignatura para la que desea agregar una entrada\n
                 Asignaturas actualmente registradas: ");
        get_asg_codes();
        Scanner scanner_p_asg = new Scanner(System.in);
        String id_asignatura = scanner_p_asg.next();

        Asignatura asignatura_deseada = asignatura_by_id(id_asignatura);
        if (asignatura_deseada == null) {
            System.out.println(
                    "El parametro introducido para el codigo de asignatura no es
                     valido. Revise este valor, por favor");
            waiter();
            return;
        }

        System.out.println("    Introduzca el dia para el que desea agregar una
                           entrada\n Valores validos: ");
        System.out.println("'lunes', 'martes', 'miercoles', 'jueves', 'viernes',
                           'sabado', 'domingo'");
        Scanner scanner_p_dia = new Scanner(System.in);
        String p_dia = scanner_p_dia.next();

        dia_deseado = dow_string_to_int(p_dia.toLowerCase());

        // Si la funcion devuelve 99 significa que el parametro introducido no es
        // valido.
        if (dia_deseado == 99) {
            System.out.println(
                    "\n   ATENCION:El parametro introducido para el dia de la semana
                      no es valido. Revise este valor, por favor.");
            waiter();
            return;
        }

        System.out.println(
                "    Introduzca la hora para la que desea agregar una entrada\n En
```

```
                formato 0 - 24. Horas enteras unicamente. ");
        Scanner scanner_p_hora = new Scanner(System.in);
        int p_hora = scanner_p_hora.nextInt();

        if (p_hora < 0 || p_hora > 24) {
            System.out.println(
                    "\n  ATENCION: El parametro introducido para la hora del dia no
                      es valido. Revise este valor, por favor.");
            waiter();
            return;
        }

       // Hasta aqui no llegara si no alguno de los parametros introducidos
       // anteriormente han dado error.

       // Si la asignatura no ha alcanzado su limite de 'ocupacion', permitira la
       // creacion de la hora. En caso contrario, else, dara mensaje de error y
       // retornara.
       // System.out.println(" " +
       // asignatura_deseada.can_be_scheduled_more(dow_string_to_int(p_dia.toLowerCase())));
       if (asignatura_deseada.can_be_scheduled_more(dow_string_to_int(p_dia.toLowerCase()))) {
           // Si es posibe annadir una nueva entradas, ahora llega el momento de comprobar
           // si en esa hora ya existe una entrada.
           // Una posible mejora seria permitir la eliminacion directamente aqui...

           // aula_deseada.dia[dia_deseado].horas[p_hora] = new Hora(asignatura_deseada);
           // asignatura_deseada.update_scheduled_by_day("+",
           // dow_string_to_int(p_dia.toLowerCase()));
           // aula_deseada.registros_anadidos++;

           // SOLVED: revisar esto. Puede que el problema no sea tanto porque .asignatura
           // sea null sino que algo en la ruta si lo sea. Por tanto, no llegara hasta ahi.
           // Queda pdte comprobarlo.
           // como info, el problema es que estabamos comprobando si la asignatura era
           // null. Y claro, lo null es la posicion del array de horas del dia.

              if (aula_deseada.dia[dia_deseado].horas[p_hora] == null) {
                  aula_deseada.dia[dia_deseado].horas[p_hora] = new
                                                          Hora(asignatura_deseada);
                  asignatura_deseada.update_scheduled_by_day("+",

                  dow_string_to_int(p_dia.toLowerCase()));

                  aula_deseada.registros_anadidos++;
              } else {
                  // Si NO es null, else, significa que en esa hora ya existe una
                  // entrada. Por tanto, no la annadimos.
                  System.out.println("No es posible crear una nueva entrada para " +
                                     id_asignatura
                             + " porque ya existe una clase programada a esa misma hora.");
                  waiter();
                  return;
              }
              System.out.println("Se ha agregado correctamente la entrada al horario.");
        } else {
            System.out.println("No es posible crear una nueva entrada para " +
                               id_asignatura
                               + "porque se ha alcanzado su limite (" +
                               asignatura_deseada.max_horas_dia + ")");
            waiter();
            return;
           // return sirve para volver al menu, o a lo que ha llamado a la funcion
           // inicialmente
        }
        waiter();
    }


    public void edit_entrada_horario() {
        System.out.println("Te damos la bienvenida al asistente de edicion de
                           asignaturas!");
        int dia_deseado = 0;
```

```
        System.out.println(
                "    Introduzca el aula para el que desea editar una entrada\n Aulas
                 actualmente registradas: ");

        get_aula_codes();

        Scanner scanner_p_aula = new Scanner(System.in);

        String id_aula = scanner_p_aula.next();

        Aula aula_deseada = aula_by_id(id_aula);

        if (aula_deseada == null) {
            System.out.println(
                    "\n   ATENCION: El parametro introducido para el codigo de aula
                      no es valido. Revise este valor, por favor.");
            waiter();
            return;
        }

        System.out.println(
                "En el aula indicada anteriormente, " + id_aula + ", se encuentran
                 agregadas las siguentes entradas:");
        aula_deseada.get_lessons();

        System.out.println("    Introduzca el dia para el cual desea editar una
                           entrada\n Valores validos: ");
        System.out.println("      'lunes', 'martes', 'miercoles', 'jueves', 'viernes',
                           'sabado', 'domingo'");

        Scanner scanner_p_dia = new Scanner(System.in);
        String p_dia = scanner_p_dia.next();

        dia_deseado = dow_string_to_int(p_dia.toLowerCase());
        // Si la funcion devuelve 99 significa que el parametro introducido no es
        // valido.
        if (dia_deseado == 99) {
            System.out.println(
                    "\n   ATENCION:\n   ATENCION:El parametro introducido para el día
                       de la semana no es valido. Revise este valor, por favor.");
            waiter();
            return;
        }

        System.out.println(
                "    Introduzca la hora para la que desea editar una entrada\n En
                 formato 0 - 24. Horas enteras unicamente. ");
        Scanner scanner_p_hora = new Scanner(System.in);
        int p_hora = scanner_p_hora.nextInt();
        if (p_hora < 0 || p_hora > 24) {
            System.out.println(
                    "El parametro introducido para la hora del dia no es valido.
                     Revise este valor, por favor");
            waiter();
            return;
        }



        System.out.println("Se va a proceder a editar la asignatura "
                + aula_deseada.dia[dia_deseado].horas[p_hora].asignatura.nombre);

        System.out.println("   Es correcta la informacion? (S/N)");

        Scanner scanner_conf_edit = new Scanner(System.in);
```

```
        String conf_edit_param = scanner_conf_edit.next();
        if (conf_edit_param.equals("S") || conf_edit_param.equals("s")) {
        // mayusculas o minisculas para facilitar la
        // entrada de datos
            System.out.println("Introduzca la nueva asignatura por la que desea
                               sustituir "
                    + aula_deseada.dia[dia_deseado].horas[p_hora].asignatura.nombre);

            Scanner scanner_nueva_asig = new Scanner(System.in);

            String id_asignatura = scanner_nueva_asig.next();

            Asignatura asignatura_deseada = asignatura_by_id(id_asignatura);

            if (asignatura_deseada == null) {
                System.out.println(
                        "El parametro introducido para el codigo de asignatura no es
                         valido. Revise este valor, por favor");
                waiter();
                return;
            }
            if
(asignatura_deseada.can_be_scheduled_more(dow_string_to_int(p_dia.toLowerCase()))) {
                aula_deseada.dia[dia_deseado].horas[p_hora] = new
                                                              Hora(asignatura_deseada);
                asignatura_deseada.update_scheduled_by_day("+",
                                                   dow_string_to_int(p_dia.toLowerCase()));
                System.out.println("Se ha creado correctamente la entrada indicada.
                                    Regresando al menu...");
                waiter();
                return;

            } else {
                System.out.println("No es posible crear una nueva entrada para " +
                                   id_asignatura
                        + " porque se ha alcanzado su limite (" +
                            asignatura_deseada.max_horas_dia + ")");
                waiter();
                return;
               // return sirve para volver al menu, o a lo que ha llamado a la funcion
               // inicialmente
            }
        } else {
            System.out.println("Ha cancelado el proceso o ha indicado una letra no
                               admitida. Volviendo al menu");
            waiter();
            return;
        }
    }

    public void del_entrada_horario() {
        System.out.println("Te damos la bienvenida al asistente de eliminacion de
                           entradas en el hoario!");
        int dia_deseado = 0;

        System.out.println(
                "    Introduzca el aula para el que desea eliminar una entrada\n
                 Aulas actualmente registradas: ");


        get_aula_codes();

        Scanner scanner_p_aula = new Scanner(System.in);

        String id_aula = scanner_p_aula.next();

        Aula aula_deseada = aula_by_id(id_aula);
```

```
        if (aula_deseada == null) {
            System.out.println(
                    "\n   ATENCION: El parametro introducido para el codigo de
                      aula no es valido. Revise este valor, por favor.");
            waiter();
            return;
        }

        System.out.println("Mostrando planificacion de aula indicada");
        aula_deseada.get_lessons();

        System.out.println("    Introduzca el dia para el que desea eliminar
                           una entrada\n Valores validos: ");
        System.out.println("'lunes', 'martes', 'miercoles', 'jueves', 'viernes',
                           'sabado', 'domingo'");
        Scanner scanner_p_dia = new Scanner(System.in);
        String p_dia = scanner_p_dia.next();

        dia_deseado = dow_string_to_int(p_dia.toLowerCase());
        // Si la funcion devuelve 99 significa que el parametro introducido no es
        // valido.
        if (dia_deseado == 99) {
            System.out.println(
                    "\n   ATENCION:El parametro introducido para el dia de la semana
                       no es valido. Revise este valor, por favor.");
            waiter();
            return;
        }

        System.out.println(
                "    Introduzca la hora para la que desea eliminar una entrada\n En
                 formato 0 - 24. Horas enteras unicamente. ");
        Scanner scanner_p_hora = new Scanner(System.in);
        int p_hora = scanner_p_hora.nextInt();
        if (p_hora < 0 || p_hora > 24) {
            System.out.println(
                    "El parametro introducido para la hora del dia no es valido.
                     Revise este valor, por favor");
            waiter();
            return;
        }
        // AQUI se produce la eliminacion de la entrada

        Asignatura asignatura_deseada =
                                aula_deseada.dia[dia_deseado].horas[p_hora].asignatura;

        asignatura_deseada.update_scheduled_by_day("-",
                                              dow_string_to_int(p_dia.toLowerCase()));

        aula_deseada.dia[dia_deseado].horas[p_hora] = null;

        System.out.println("Se ha eliminado correctamente la entrada indicada");

        waiter();
    }



    public void get_entradas() {
        for (Aula a : aulas) {
            System.out.println("Para el aula codigo " + a.id + ":");
            a.get_lessons();
            System.out.println("\n================\n");
        }
        waiter();
    }
```

```
public void get_entradas_by_aula() {
    get_aula_codes();
    System.out.println("Introduzca el codigo del aula a listar:");
    Scanner scanner_aula_a_listar = new Scanner(System.in);
    String aula_a_listar = scanner_aula_a_listar.next();
    if (aula_by_id(aula_a_listar) == null) {
        // Si entra en null significa que no hay ningun aula con ese ID dada de
        // alta.
        // Por tanto, no hay nada que listar.

        System.out.println("El ID introducido no se corresponde con ningun aula
                           registrada. Volviendo al menu");
        waiter();
        return;
    } else {
        Aula au_a_listar = aula_by_id(aula_a_listar);
        for (Aula a : aulas) {
            if (a == au_a_listar) {
                a.get_lessons();
            }
        }
    }
    waiter();
}

public void get_entradas_by_asg() {
    System.out.println("Mostrando asignaturas...");
    listar_asignaturas();
    System.out.println("Introduzca el codigo de la asignatura a listar:");
    Scanner scanner_asg_a_listar = new Scanner(System.in);
    String asg_a_listar = scanner_asg_a_listar.next();

    if (asignatura_by_id(asg_a_listar) == null) {
        // Si entra en null significa que no hay ninguna asignatura con ese ID dada
        // de alta. Por tanto, no hay nada que listar.
        System.out.println("El ID introducido no se corresponde con ninguna
                           asignatura. Volviendo al menu");
        waiter();
        return;
    } else {
        int found = 0;
        Asignatura asignatura_para_listar = asignatura_by_id(asg_a_listar);
        System.out.println("    AULA |   DIA    |  HORA");
        for (Aula a : aulas) {
            for (Dia d : a.dia) {
                int i = 0;
                for (Hora h : d.horas) {
                    if (h != null) {
                        if (h.asignatura == asignatura_para_listar) {
                            System.out.println(
                                    "    " + a.id + "   " + d.nombre + "    " + i +
                                    ":00 - " + (i + 1) + ":00");
                            found++;
                        }
                    }
                    i++;
                }
            }
        }
        if (found == 0) {
            System.out.println("\n  --- No se han encontrado resultados ---");
        } else {
            System.out.println("\n   Se han encontrado " + found + " resultados.
                               ");
        }
```

```
            waiter();
        }
    }


    public void get_entradas_by_docente() {
        get_docente_listado();

        System.out.println("\nIntroduzca DNI del docente para el que desea ver las
                           clases:");

        Scanner scanner_dni_docente_a_listar = new Scanner(System.in);

        String dni_docente_a_listar = scanner_dni_docente_a_listar.next();

        if (docente_by_dni(dni_docente_a_listar) == null) {
            // Si entra en null significa que no hay ningun docente con ese DNI dada de
            // alta. Por tanto, no hay nada que listar.

            System.out.println("El DNI introducido no se corresponde con ningun docente
                               registrado. Volviendo al menu");
            return;
        } else {
            int found = 0;

            Docente docente_a_listar = docente_by_dni(dni_docente_a_listar);

            for (Aula a : aulas) {
                for (Dia d : a.dia) {
                    int i = 0;
                    for (Hora h : d.horas) {
                        if (h != null) {

                            if (h.asignatura.docente == docente_a_listar) {
                                // aula nombre_dia hora asignatura
                                System.out.println("    " + a.id + "   " + d.nombre + "
                                                   " + i + ":00 - " + (i + 1)
                                        + ":00    " + h.asignatura.codigo);
                                found++;
                            }
                        }
                        i++;
                    }
                }
            }
            if (found == 0) {
                System.out.println("\n  --- No se han encontrado resultados ---");
            } else {
                System.out.println("\n   Se han encontrado " + found + " resultados.
                                   ");
            }
        }

        waiter();
    }



    /*
     * ********************************************
     * ********************************************
     *                  Clases AULA
     * ********************************************
     * ********************************************
     */
```

```
public void add_aula_auto(String p_id, int p_csp) {
    aulas.add(new Aula(p_id, p_csp));
}


public void get_aula_codes() {
    System.out.println("Aulas registradas en este momento: ");
    for (Aula i : aulas) {

        System.out.println(i.id);

    }
}



public void get_detalle_aula() {

    get_aula_codes();

    System.out.println("Introzca el codigo del aula para la que desea ver los
                        detalles: ");
    // TODO: Check if requested exists. Maybe return an array?
    Scanner scanner = new Scanner(System.in);

    String cod_a_buscar = scanner.next();

    Aula aula_a_ver_detalles = aula_by_id(cod_a_buscar);

    if (aula_a_ver_detalles == null) {
        System.out.println("No se encuetra ningun aula con el ID indicado.
                            Volviendo al menu...");
    } else {

        aula_a_ver_detalles.get_details();
    }

    waiter();
}



public void get_detalle_aulas() {
    System.out.println("Listado de aulas registradas en este momento: ");

    for (Aula i : aulas) {
        i.get_details();
        System.out.println("");

    }
    waiter();
}




public Aula aula_by_id(String p_id) {
    // System.out.println("Entro en method");
    for (Aula a : aulas) {
        // System.out.println("Recorriendo...");
        if (p_id.equals(a.id)) {
```

```
                // System.out.println("Encontrado");
                // a.get_details();
                return a;
            }
        }
        return null;
    }

    public void aula_delete() {
        System.out.println("Por favor, introduzca el identificador del aula que
                           desea eliminar:");

        Scanner scanner = new Scanner(System.in);
        String id_aula_pasado_por_scanner = scanner.next();

        Aula aula_a_eliminar = aula_by_id(id_aula_pasado_por_scanner);
        aulas.remove(aula_a_eliminar);

        System.out.println("La aula ha sido eliminada correctamente.");

        waiter();
    }

    public void aula_add() {
        System.out.println("Bienvenido al asistente de adicion de aulas.");

        System.out.println("A continuacion se muestran las aulas ya existentes:");
        get_aula_codes();

        System.out.println("Introduzca el ID de la nueva aula a crear:");
        Scanner scannercreacionaula = new Scanner(System.in);
        String creacion_aula = scannercreacionaula.next();

        System.out.println("Indique la capacidad para aula " + creacion_aula + ":");
        Scanner scannercapacidad = new Scanner(System.in);
        int capacidad_aula_nueva = scannercapacidad.nextInt();

        aulas.add(new Aula(creacion_aula, capacidad_aula_nueva));

        System.out.println("Su aula ha sido creada con exito.");
        waiter();
    }


    public void eliminar_aulas() {

        if (confirm_harmfull()) {

            aulas.clear();
            System.out.println("TODAS los AULAS se han eliminado correctamente.");

        } else {

            System.out.println("Se ha cancelado la operacion. Volviendo al menu.");
        }

        waiter();
    }


    // EJEMPLO: este es un buen ejemplo de como modificar/acceder a private vs
    // public properties

    public void editar_aula() {
        get_aula_codes();
```

```
        System.out.println("\n Introduzca el ID del aula que desea editar:");

        Scanner scanner_id_aula_a_editar = new Scanner(System.in);
        String id_aula_a_editar = scanner_id_aula_a_editar.next();

        if (aula_by_id(id_aula_a_editar) == null) {
            System.out.println("El ID introducido no se corresponde con ninguno
                               registrado en la base de datos");

        } else {

            Aula aula_a_editar = aula_by_id(id_aula_a_editar);

            aula_a_editar.get_details();
            System.out.println("Desea cambiar el ID o la capacidad? [ID/CAP]");

            Scanner scanner_cosa_a_cambiar = new Scanner(System.in);
            String cosa_a_cambiar = scanner_cosa_a_cambiar.next();

            switch (cosa_a_cambiar) {

                case "ID": // no se pone brak para que haga de "OR" __ +info
                           // https://logfetch.com/java-use-or-switch-case-
                           //statement/#use-or-operator-by-removing-break

                case "id":
                    System.out.println("Introduzca el nuevo ID que desea asignar al
                                       aula:");
                    Scanner scanner_valor_nuevo_id = new Scanner(System.in);
                    String nuevo_id = scanner_valor_nuevo_id.next();
                    aula_a_editar.id = nuevo_id;

                    break;

                case "CAP":

                case "cap":

                    System.out.println("Introduzca la nueva capacidad que desea asignar
                                       al aula:");
                    Scanner scanner_valor_nueva_cap = new Scanner(System.in);

                    int valor_nueva_cap = scanner_valor_nueva_cap.nextInt();

                    aula_a_editar.update_capacidad(valor_nueva_cap);
            }

            System.out.println("Se ha actualizado correctamente. ");
        }
        waiter();
    }









    /*
     * ********************************************
     * ********************************************
     *               Clases DOCENTE
     * ********************************************
```

```
     * ******************************************
     */

    public void get_docente_dnis() {
        System.out.println("DNIs registrados en este momento: ");
        for (Docente i : docentes) {
            System.out.println(i.get_dni());
        }
        waiter();
    }

    public void get_docente_listado() {
        System.out.println("DNIs registrados en este momento: ");
        for (Docente i : docentes) {
            i.get_details();
            System.out.println("");
        }
        waiter();
    }


    public void get_listado_docente_name_and_dni() {
        for (Docente d : docentes) {
            System.out.println(d.get_fullname_and_dni());
        }
    }

    public void add_docente_auto(String dni, String nombre, String apellido1,
                                 String apellido2, Double p_sueldo, String p_tit) {

        docentes.add(new Docente(dni, nombre, apellido1, apellido2, p_sueldo, p_tit));

    }


    public void docente_delete() {

        System.out.println("Por favor, introduzca el DNI del docente que desee dar
                           de baja:");

        Scanner scanner = new Scanner(System.in);
        String docente_de_baja = scanner.next();

        Docente docente_a_eliminar = docente_by_dni(docente_de_baja);
        if (docente_a_eliminar == null) {
            System.out.println("El docente que ha introducido no se encuentra.
                               Volviendo al menu.");
            waiter();

        } else {

            docentes.remove(docente_a_eliminar);

            System.out.println("El docente se ha dado de baja correctamente.");
            waiter();
        }
    }


    public Docente docente_by_dni(String p_id) {
        for (Docente d : docentes) {
            // System.out.println("Recorriendo...");
            if (p_id.equals(d.dni)) {
                return d;
            }
```

```
        }
        return null;
    }

    public void add_docente() {
        System.out.println("\nTe damos la bienvenida al asistente de adicion del
                           equipo docente!");
        System.out.println("Introduzca, por favor, el DNI del nuevo docente que
                           desea dar de alta");
        Scanner scanner_add_doce_dni = new Scanner(System.in);

        String add_doce_dni = scanner_add_doce_dni.next();

        if (docente_by_dni(add_doce_dni) == null) {
            // Si entra en null significa que no hay ningun docente dado de alta. Por
            // tanto, podemos annadirlo.
            System.out.println("Introduzca, por favor, los datos que se le solicitan.
                               \n   NOMBRE:");
            Scanner scanner_add_doce_nombre = new Scanner(System.in);
            String add_doce_nombre = scanner_add_doce_nombre.nextLine();

            System.out.println("   Primer Apellido: ");
            Scanner scanner_add_doce_ape1 = new Scanner(System.in);
            String add_doce_ape1 = scanner_add_doce_ape1.nextLine();

            System.out.println("   Segundo Apellido: ");
            Scanner scanner_add_doce_ape2 = new Scanner(System.in);
            String add_doce_ape2 = scanner_add_doce_ape2.nextLine();

            System.out.println("   Sueldo (formato 1000.0): ");
            Scanner scanner_add_doce_sueldo = new Scanner(System.in);
            Double add_doce_sueldo = scanner_add_doce_sueldo.nextDouble();

            System.out.println("   Titulo: ");
            Scanner scanner_add_doce_tit = new Scanner(System.in);

            String add_doce_tit = scanner_add_doce_tit.nextLine();

           // Seria interesante hacer aqui comprobaciones con hasNextXXXX en los Scanners
           // para evitar que java devuelva errores feos...

            docentes.add(new Docente(add_doce_dni, add_doce_nombre, add_doce_ape1,
                                     add_doce_ape2, add_doce_sueldo,
                    add_doce_tit));

            System.out.println("Se ha annadido correctamente el docente");
            waiter();

        } else {

            // si entra aqui significa que ya hay un/una docente con ese DNI. por
            // tanto, no podemos annadirlo.
            // Mostramos tambien el nombre del registro ya annadido para ayudar al
            // usuario.
            System.out.println(
                    "ERROR: ya existe un registro para ese DNI: " +
                     docente_by_dni(add_doce_dni).get_fullname());



            System.out.println(
                    "Puede, si asi lo desea, eliminar el registro o modificarlo desde
                        las diferentes opciones del menu.");
            waiter();
```

```
        }

    }

    public void eliminar_docentes() {
        if (confirm_harmfull()) {
            docentes.clear();
            System.out.println("TODOS los DOCENTES se han eliminado correctamente.");
        } else {
            System.out.println("Se ha cancelado la operacion. Volviendo al menu.");
        }
        waiter();
    }

    public void editar_docente() {
        System.out.println("Mostrando lista de docentes: ");
        get_listado_docente_name_and_dni();
        System.out.println("Introduzca el DNI del docente que desea editar: ");
        Scanner scanner = new Scanner(System.in);
        String doce_a_buscar = scanner.next();
        if (docente_by_dni(doce_a_buscar) == null) {
            // Si es null el dni introducido no existe
            System.out.println("El DNI introducido no se corresponde con ninguno
                               registrado en la base de datos");
        } else {
            Docente docente_a_modificar = docente_by_dni(doce_a_buscar);
            System.out.print("Los datos actuales del docente son:");

            docente_a_modificar.get_details();
            System.out.println(
                    "Introduzca la clave del campo a variar\nLa claves estan formadas
                     por las 3-4 primeras letras mayusculas mostradas en el listado");

            Scanner scanner_codigo_a_cambiar = new Scanner(System.in);
            String codigo_a_cambiar = scanner_codigo_a_cambiar.next();

            System.out.println("\nIntroduzca el nuevo contenido que desea actualizar en
                               el campo indicado: ");
            Scanner scanner_valor_a_cambiar = new Scanner(System.in);
            switch (codigo_a_cambiar) {
                case "DNI":
                    String valor_a_cambiar = scanner_valor_a_cambiar.next();
                    docente_a_modificar.dni = valor_a_cambiar;
                    break;

                case "NOM":
                    String valor_a_cambiar_n = scanner_valor_a_cambiar.next();
                    // Se cambia el nombre porque si no
                    // chilla...
                    docente_a_modificar.update_nombre(valor_a_cambiar_n);
                    break;

                case "APE1":
                    String valor_a_cambiar_a1 = scanner_valor_a_cambiar.next();
                    docente_a_modificar.update_ape1(valor_a_cambiar_a1);
                    break;

                case "APE2":
                    String valor_a_cambiar_a2 = scanner_valor_a_cambiar.next();
                    docente_a_modificar.update_ape2(valor_a_cambiar_a2);
                    break;
                case "SUE":
                    Double valor_a_cambiar_s = scanner_valor_a_cambiar.nextDouble();
                    docente_a_modificar.update_sueldo(valor_a_cambiar_s);
                    break;
```

```
                case "TIT":
                    String valor_a_cambiar_t = scanner_valor_a_cambiar.next();
                    docente_a_modificar.update_titulo(valor_a_cambiar_t);
                    break;

                default:
                    System.out.println(
                            "El valor introducido no se corresponde con ninguno de los
                             permitodos.\nVolviendo al menu.");
                    // Aqui no se llama a waiter porque esta al final.
            }
        }
        waiter();
    }
}
```

Main.java

```
/**
 * main class of Java Program
 *
 * Practica Final
 *
 */
import java.util.Scanner;

class Main {
    static Horario horario1 = new Horario("horario1");

    public static void main(String[] args) {
        // horario1.nueva_asignatura();
        carga_inicial(horario1);

        /*
         * horario1.listar_asignaturas();
         * horario1.get_asg_codes();
         * horario1.detalle_asignatura();
         * horario1.get_detalle_aula();
         * horario1.get_docente_listado();
         * horario1.editar_asignatura();
         */
        muestra_menu();
    }




    public static void muestra_menu() {

        int num = 0;
        while (num != 6) {
        // Este valor solo tomara 6 si se selecciona la opcion 6 del menu.
        // De no ser asi, volvera a ejecutarse la funcion de forma continua.

            horario1.ClearScreen();

            System.out.println("\n\n\n\n\n ================  MENU
                                              ================\n\n");
            System.out.println("Elija una de las siguientes opciones:");

            System.out.println("\n1. Gestion asignaturas:");
```

```
            System.out.println("    1.1. Agregar una asignatura");
            System.out.println("    1.2. Editar una asignatura");
            System.out.println("    1.3. Eliminar una asignatura");
            System.out.println("    1.4. Borrar TODAS las asiganturas");
            System.out.println("    1.5. Listar todas las asignaturas");
            System.out.println("    1.6. Ver los detalles de una
asignatura");

            System.out.println("\n2. Definir horario:");
            System.out.println("    2.1. Agregar una entrada al horario");
            System.out.println("    2.2. Editar una entrada del horario");
            System.out.println("    2.3. Eliminar una entrada del horario");
            System.out.println("    2.4. Borrar todo el horario");
            System.out.println("    2.5. Restaurar horario/Cargar datos de
                                muestra");

            System.out.println("\n3. Mostrar horario:");
            System.out.println("    3.1. Busqueda por asignatura");
            System.out.println("    3.2. Busqueda por aula");
            System.out.println("    3.3. Busqueda por profesor");
            System.out.println("    3.4. Mostrar todas las entradas");

            System.out.println("\n4. Gestion aulas:");
            System.out.println("    4.1. Agregar aula a la base de datos");
            System.out.println("    4.2. Editar aula ya creada");
            System.out.println("    4.3. Eliminar aula de la base de
datos");
            System.out.println("    4.4. Eliminar TODAS las aulas.");
            System.out.println("    4.5. Listar aulas disponibles");

            System.out.println("\n5. Gestion docentes:");
            System.out.println("    5.1. Agregar docente a la base de
datos");
            System.out.println("    5.2. Editar docente ya registrado");
            System.out.println("    5.3. Eliminar docente de la base de
                                     datos");
            System.out.println("    5.4. Eliminar TODOS los docentes.");
            System.out.println("    5.5. Listar docentes registrados
                                     disponibles");

            System.out.println("6. Salir");

            System.out.println(
                    "Introduzca el numero de la opcion que desea
seleccionar.\n
                     Sin espacios ni puntos. Ejemplo: '21' para agregar
entrada
                     al horario.");


            Scanner scanner = new Scanner(System.in);

            num = scanner.nextInt();



         switch (num) {
                case 11:
                    horario1.ClearScreen();
                    horario1.nueva_asignatura();
                    break;
                case 12:
                    horario1.ClearScreen();
                    horario1.editar_asignatura();
                    break;
                case 13:
```

```
                    horario1.ClearScreen();
                    horario1.eliminar_asignatura();
                    break;
                case 14:
                    horario1.ClearScreen();
                    horario1.eliminar_asignaturas();
                    break;
                case 15:
                    horario1.ClearScreen();
                    horario1.listar_asignaturas();
                    break;
                case 16:
                    horario1.ClearScreen();
                    horario1.detalle_asignatura();
                    break;
                case 21:
                    horario1.ClearScreen();
                    horario1.add_entrada_horario();
                    break;
                case 22:
                    horario1.ClearScreen();
                    horario1.edit_entrada_horario();
                    break;
                case 23:
                    horario1.ClearScreen();
                    horario1.del_entrada_horario();
                    break;
                case 24:
                    horario1.ClearScreen();
                    horario1 = new Horario("horario1");
                    System.out.println("Se han borrado todos los datos de
                                        la base de datos.");
                           // Evidentemente no hay
                           // base de datos, pero
                           // asi queda mas guay
                    horario1.waiter();
                    break;
                case 25:
                    horario1.ClearScreen();
                    horario1 = new Horario("horario1");
                    carga_inicial(horario1);
                    System.out.println("Se ha restaurado el horario a su
                                        estado inicial.");
                    horario1.waiter();
                    break;
                case 31:
                    horario1.ClearScreen();
                    horario1.get_entradas_by_asg();
                    break;
                case 32:
                    horario1.ClearScreen();
                    horario1.get_entradas_by_aula();
                    break;
                case 33:
                    horario1.ClearScreen();
                    horario1.get_entradas_by_docente();
                    break;
                case 34:
                    horario1.ClearScreen();
                    horario1.get_entradas();
                    break;
                case 41:
                    horario1.ClearScreen();
                    horario1.aula_add();
                    break;
                case 42:
                    horario1.ClearScreen();
```

```
                    horario1.editar_aula();
                    break;
                case 43:
                    horario1.ClearScreen();
                    horario1.aula_delete();
                    break;
                case 44:
                    horario1.ClearScreen();
                    horario1.eliminar_aulas();
                    break;
                case 45:
                    horario1.ClearScreen();
                    horario1.get_detalle_aulas();
                    break;
                case 51:
                    horario1.ClearScreen();
                    horario1.add_docente();
                    break;
                case 52:
                    horario1.ClearScreen();
                    horario1.editar_docente();
                    break;
                case 53:
                    horario1.ClearScreen();
                    horario1.docente_delete();
                    break;
                case 54:
                    horario1.ClearScreen();
                    horario1.eliminar_docentes();
                    break;
                case 55:
                    horario1.ClearScreen();
                    horario1.get_docente_listado();
                    break;
                case 6:
                    horario1.ClearScreen();
                    System.out.println("Gracias por usar esta
                                        apliacion\nSaliendo...\n");
                    break;
                default:
                    System.out.println("El numero que ha introducido no
                                        es valido. Por favor, intruzca uno
de
                                        nuevo.");
                    break;
            }

        }
    }




    public static void carga_inicial(Horario hor) {

        hor.add_docente_auto("12345678A", "Manuel", "Sanchez", "Diaz",
2000.0,
                             "Profesor Titular");

        hor.add_docente_auto("23456789J", "Cristina", "Gonzalez", "Pons",
                             3000.0, "Doctora");

        hor.add_docente_auto("34567901C", "Rodrigo", "Hidalgo", "Esteban",
                             1000.0, "Profesor asociado");

        hor.add_docente_auto("45678012D", "Lucia", "Jimenez", "Llorente",
```

```
                                3000.0, "Doctora");

        hor.add_docente_auto("20458599E", "Javier", "Airado", "Rodriguez",
                                2000.0, "Profesor Titular");

        hor.add_docente_auto("50369736Z", "Pablo", "Aguado", "Gonzalez",
                                 2000.0, "Profesor Titular");



        hor.asignatura_auto("I2P", "Introduccion a la Programacion", 8, 2,
                               "50369736Z");

        hor.asignatura_auto("MCS", "Matematicas para la Computacion y
                                Servicios", 9, 4, "23456789J");

        hor.asignatura_auto("HFS", "Historia y Fundamentos de los
Servicios",
                                4, 1, "34567901C");

        hor.asignatura_auto("TDS", "Teoria de Sistemas", 10, 3,
"45678012D");

        hor.asignatura_auto("ADQ", "Arquitectura de Computadores", 8, 3,
                                "45678012D");

        hor.asignatura_auto("SDS", "Sociologia de los Servicios", 9, 2,
                                "45678012D");


        hor.add_aula_auto("G107", 30);

        hor.add_aula_auto("G108", 20);

        hor.add_aula_auto("G109", 25);

        hor.add_aula_auto("G110", 40);

        hor.add_aula_auto("G111", 35);


        hor.add_entrada_horario_auto("lunes", "G107", "TDS", 10);

        hor.add_entrada_horario_auto("martes", "G108", "TDS", 9);

        hor.add_entrada_horario_auto("miercoles", "G109", "TDS", 8);

        hor.add_entrada_horario_auto("jueves", "G110", "TDS", 12);

        hor.add_entrada_horario_auto("viernes", "G111", "TDS", 13);

        hor.add_entrada_horario_auto("sabado", "G107", "TDS", 10);

        hor.add_entrada_horario_auto("domingo", "G107", "TDS", 9);


        hor.add_entrada_horario_auto("lunes", "G107", "MCS", 12);
        hor.add_entrada_horario_auto("martes", "G108", "MCS", 11);
        hor.add_entrada_horario_auto("miercoles", "G109", "MCS", 10);
        hor.add_entrada_horario_auto("jueves", "G110", "MCS", 10);
        hor.add_entrada_horario_auto("viernes", "G111", "MCS", 8);
        hor.add_entrada_horario_auto("sabado", "G108", "MCS", 10);
        hor.add_entrada_horario_auto("domingo", "G107", "MCS", 8);

        hor.add_entrada_horario_auto("lunes", "G107", "I2P", 14);
        hor.add_entrada_horario_auto("martes", "G108", "I2P", 13);
        hor.add_entrada_horario_auto("miercoles", "G109", "I2P", 14);
```

```
        hor.add_entrada_horario_auto("jueves", "G110", "I2P", 8);
        hor.add_entrada_horario_auto("viernes", "G111", "I2P", 10);
        hor.add_entrada_horario_auto("sabado", "G109", "I2P", 10);
        hor.add_entrada_horario_auto("domingo", "G108", "I2P", 10);

        hor.add_entrada_horario_auto("lunes", "G111", "HFS", 10);
        hor.add_entrada_horario_auto("martes", "G110", "HFS", 8);
        hor.add_entrada_horario_auto("miercoles", "G109", "HFS", 9);
        hor.add_entrada_horario_auto("jueves", "G108", "HFS", 10);
        hor.add_entrada_horario_auto("viernes", "G107", "HFS", 14);
        hor.add_entrada_horario_auto("sabado", "G110", "HFS", 9);
        hor.add_entrada_horario_auto("domingo", "G108", "HFS", 8);

        hor.add_entrada_horario_auto("lunes", "G110", "ADQ", 10);
        hor.add_entrada_horario_auto("martes", "G111", "ADQ", 10);
        hor.add_entrada_horario_auto("miercoles", "G107", "ADQ", 11);
        hor.add_entrada_horario_auto("jueves", "G108", "ADQ", 8);
        hor.add_entrada_horario_auto("viernes", "G109", "ADQ", 9);
        hor.add_entrada_horario_auto("sabado", "G107", "ADQ", 14);
        hor.add_entrada_horario_auto("domingo", "G108", "ADQ", 13);

        hor.add_entrada_horario_auto("lunes", "G111", "SDS", 8);
        hor.add_entrada_horario_auto("martes", "G107", "SDS", 10);
        hor.add_entrada_horario_auto("miercoles", "G107", "SDS", 9);
        hor.add_entrada_horario_auto("jueves", "G108", "SDS", 14);
        hor.add_entrada_horario_auto("viernes", "G109", "SDS", 13);
        hor.add_entrada_horario_auto("sabado", "G110", "SDS", 8);
        hor.add_entrada_horario_auto("domingo", "G109", "SDS", 10);

        hor.asignatura_auto("TMH", "Test para comprobar el maximo de horas
                             al dia", 2, 1, "50369736Z");
        hor.add_entrada_horario_auto("lunes", "G111", "TMH", 8);

    };

}
```