Descripción de estructuras de datos y algoritmos utilizado para implementar las funcionalidades principales (generación de Horarios)

Estructuras de datos

- ReturnSet: Se usa para devolver valores de diferente tipo en las funciones de generación de horarios, de manera que dependiendo de si se ha podido crear un horario válido o no devuelve un Boolean diciendo que no se ha conseguido o un Boolean diciendo que se ha conseguido y el resultado.
- LimitacionesHorario: Permite guardar i tener fácilmente accesibles las restricciones que afectan a todo el horario: la franja de trabajo y los días libres.
- clases: ArrayList de Clases que contiene todas las Clases que se deben colocar en Horario.
- franjas: ArrayList que contiene horaIni y horaFin de las franjas horarias en que se puede colocar las Clases de clases.
- Placed: Array que contiene Booleans que indican si cada una de las Clases de clases están colocadas en lo solución que se está explorando.

Algoritmo

El algoritmo empleado para generar los horarios es un algoritmo de backtracking que emplea forward checking para reducir en la mayor medida posible el nombre de posibles soluciones que se comprueba si son válidas basándose en 2 principios básicos:

- Se comprueba todas las restricciones lo antes posible, en cuanto la información necesaria para comprobar cada restricción está definida para las asignaciones que se va a añadir a las posibles soluciones.
- Sólo se añade una asignación a una solución si esta es compatible con las asignaciones acumuladas en esa solución hasta el momento.

El algoritmo empieza generando todas las posibles Clases que se pueden crear en el escenario que se ha cargado en el programa. Una vez se generan las clases se ordenan de forma ascendente según el tamaño de la franja horaria en la que se pueden asignar a causa de las restricciones por las que se ven afectadas y este orden será el orden en el que las vayamos probando para generar posibles soluciones (evitamos que asignaturas que son asignables a un rango mayor de horas no ocupen sin necesidad las franjas a las que se pueden asignar las que tienen un rango menor sin necesidad).

También creamos un ArrayList con las franjas de todas las clases, para evitar tener que recalcular continuamente estas mismas.

Para mantener constancia de que Clases se van asignando a la solución creamos un Array de Booleans llamado placed.

Una vez se elige una Clase para que forme parte de la Asignacion de una posible solución se prueba días para la Asignacion (descartando aquellos que son DiaLibre o que causan que la Clase elegida en el momento incumpla una Restriccion), empezando por domingo o lunes alternativamente (dependiendo de si el número de Asignaciones definidas hasta el momento es par o impar).

Por cada Clase y día definidos se prueban las horas que no salen de la franja horaria que se ha calculado para la Clase en función de las Restricciones y que añadidas a las 2 variables anteriores no generan una Asignación que incumpla Restricciones.

Después, cuando ya tenemos Clase, día i hora para el candidato a Asignacion añadimos Aulas compatibles con la Sesion de la Clase involucrada y el número de plazas.

Este proceso se repite añadiendo Asignaciones hasta que nos quedamos sin Clases o hasta que no se es capaz de generar una Asignación válida. En el primer caso se devuelve la solución obtenida, en el 2o se retrocede 1 nivel y se prueba otra combinación de Clase, día, hora y Aula siguiendo el proceso descrito anteriormente.