GENERADOR DE HORARIOS

SEGUNDA ENTREGA

Proyectos de programación

2018-2019 Q1

David Bernal Soto david.antonio.bernal
Gerard Candón Arenas gerard.candon
Miguel Salamero Muñoz miguel.salamero

ÍNDICE

Descripción del algoritmo

Elección del algoritmo

Durante la búsqueda de algoritmos en la categoría de "*constraint satisfaction problems"* me llamó rápidamente la atención los algoritmos genéticos porque ofrecían un potencial mayor que el resto de sus competidores, sobre todo de cara a la búsqueda de una solución óptima, donde se valoran múltiples restricciones entrelazando la evaluación de Hard and Soft constraints.
Es por esto que indagué en este grupo de algoritmos en búsqueda de alguno que más se ajustase a nuestro problema.
Spea2 'Strong pareto evolutionary algorithm' era el candidato perfecto.

El problema que se nos presenta busca encontrar un horario válido a partir de los siguientes parámetros:

Conjunto de turnos: donde un turno es una instancia de una asignatura que se impartirá a una cantidad de alumnos que pertenecen a un grupo de la asignatura durante un periodo de tiempo.
Conjunto de aulas: donde cada aula tiene asignada una capacidad de asistentes que puede alojar.
Conjunto de timeslots: fechas y horas que forman el periodo lectivo del centro.

A todo esto se le añaden las siguientes restricciones de cara algoritmo:
Fundamentalmente cada turno deberá tener una hora y aula asignada. (HC1)
En un aula y a una hora solo podrá haber un turno. (HC2)
Las asignaturas tienen dependencias entre ellas ya sea porque unas son co-requisitos entre ellas o bien porque pertenecen al mismo nivel, para estos casos, no se podrán impartir turnos de asignaturas que tengan este tipo de dependencias entre ellas en la misma hora. (HC3)
Como tampoco puede darse que en una clase haya un turno con una cantidad de asistentes mayor que la capacidad máxima de asistentes de esta clase. (HC4)
Y finalmente, cada turno puede ser impartido durante una o dos horas, para este último caso
estas horas han de estar consecutivas. (HC5)

Representación de una solución

Consideremos una sesión como una combinación de instancias de turno, timeslot y aula.

Supongamos X una variable del problema en cuestión, que tiene un dominio D.
En nuestro caso X viene a ser una sesión y puesto que los diferentes valores que pueden tomar los elementos de los que está compuesto forman su dominio.

Esto es, identificador de la sesión que guarda una correspondencia con el turno que se imparte, y cuyos valores pueden ser cualquiera de los turnos del conjunto de turnos del problema.
Un identificador de aula que guarda la correspondencia con una aula del conjunto de aulas.
Y un identificador de timeSlot que guarda la correspondencia con un timeSlot del conjunto.

Entonces tenemos X = {tr,a,ts} donde:
tr ∈ [0,nt), nt = tamaño del conjunto de turnos
a ∈ [0,na) na = tamaño del conjunto de aulas
ts ∈ [0,ns) ns = tamaño del conjunto de turnos

Un conjunto de estas variables donde todos los elementos de turno tienen asociada una aula y hora representaría una solución de nuestro problema.
Entonces una solución sería un conjunto de nt elementos X

Esto supondría, puesto que los turnos pueden ser de hasta 2 horas, que una solución válida podría requerir hasta 2*nt elementos X.

Sin embargo esta representación tiene varios inconvenientes como por ejemplo que ha de haber un control que garantice que se encuentran todos los turnos representados.

Tras múltiples pruebas la representación resultante y actualmente implementada en el algoritmo es:

Un conjunto de nt elementos X, donde cada elemento es X = {a1,ts1,a2,ts2}
a1,a2 ∈ [0,na) na = tamaño del conjunto de aulas
ts1,ts2 ∈ [0,ns) ns = tamaño del conjunto de turnos

Se elimina el elemento tr y se utiliza como identificador la posición del elemento X en la solución, que se corresponde con el Turno de la misma posición del conjunto de turnos.
Y puesto que cada sesión puede tener hasta 2 horas de duración se asignan unas aulas a1 y a2 a unos timeslots ts1 y ts2 respectivamente. Y en el caso de que la sesión tenga una duración de 1 hora solo se tendrá en cuenta la primera pareja a1,ts1.
De esta forma garantizamos por una parte (HC1) y por otra reducimos el tamaño de la solución.

Funcionamiento del algoritmo

Este algoritmo como cualquier otro algoritmo genético se compone de una población de individuos(soluciones) de tamaño N, cada individuo es un cromosoma, una cadena de genes.
Donde cada gen es un elemento X.

En primer lugar se genera una población inicial de tamaño N donde cada individuo tiene genes con valores randoms siempre dentro del dominio D.

Cada individuo de la población será evaluado bajo un conjunto de objetivos que determinan cuán bien cumplen las restricciones del problema, entonces tras este cálculo cada cromosoma tendrá un conjunto de objetivos. Más información sobre cuáles son estos objetivos más adelante, por ahora será suficiente con indicar que un individuo con un valor de alguno de sus objetivos menor que el de otro individuo en el mismo objetivo indica que el primer individuo es mejor que el segundo cumpliendo dicho objetivo.

Inicio del bucle.

Se crea un nuevo conjunto poblacion union archivo(este concepto se introducirá seguidamente, por ahora solo es necesario saber que se inicializa vacía y por lo tanto este nuevo conjunto será solo la población inicial)

Y a éste se le hace un cálculo de fitness a cada individuo que cualifica a un individuo dentro de la población, utilizando los objetivos calculados. En Spea2 esto es la suma de raw fitness y la densidad de las soluciones, la primera se hace a partir de la dominación.
Esto es, se calcula cada individuo la cantidad de otros individuos que dominan. Un individuo domina a otro si todos sus objetivos son menor o iguales que el segundo individuo.
Una vez obtenido esto, el raw fitness de un individuo se calcula de la siguiente forma:
Por todos los elementos de la población que dominen a este individuo se obtiene la cantidad de individuos que dominan y se realiza una suma, ésta será el valor del raw fitness.
Esto permite clasificar a todas las soluciones que rinden mejor en los objetivos del problema.
Por otro lado el segundo componente del fitness es se hace a partir del cálculo de la densidad de las soluciones en base a sus objetivos, que permite distinguir soluciones distintas, es decir en conjunción con el raw fitness nos permite evaluar las soluciones no solo por lo bien que cumplan sus objetivos sino también por lo distintas que son entre ellas.

El archivo es un subconjunto de la población con los mejores individuos este conjunto ha de tener un tamaño inferior al de la población generalmente N/2

Tras haber calculado el fitness de la población se llena el archivo con los N/2(o tamaño del archivo previamente definido) individuos de la población.

Ahora se comprueba si algún elemento de la población tiene todos sus objetivos a cero. Indicando que es una solución válida del problema para finalizar la ejecución.

En caso contrario se crea un nuevo conjunto llamado conjunto de apareamiento(mantingpool) que se llena hasta alcanzar el tamaño de la población mediante un torneo binario sobre el archivo.
Finalmente sobre este manting pool se realiza la reproducción que consiste en aplicar crossover entre sus individuos y mutación a los hijos generados

Estos hijos formarán la siguiente generación de población.
Ahora con una nueva población y un mating creado se vuelve al principio del bucle y se ejecutan tantas iteraciones como que se indiquen en sus parámetros de entrada.

Descripción de las clases

- Titulación: Organismo que tiene un nombre y contiene asignaturas y un número máximo de alumnos en cada grupo de teoría y laboratorio de sus asignaturas.
- Asignatura: Objeto que pertenece a una titulación, que contiene una cierta cantidad de alumnos matriculados, un número concreto de horas semanales tanto de teoría como de laboratorio, un curso y tantos turnos como sean necesarios para que participen sus alumnos matriculados.
- Aula: Lugar en el que se imparten clases, se identifican por un Id y pueden tener PC. Tienen una capacidad máxima.
- Turno: Clase que se imparte durante la semana, pertenece a una asignatura en concreto, su id indica el grupo que la hace y tiene un identificador único para todos los turnos del mismo grupo y asignatura.
- SesionAsignatura: Se define como un turno en una aula y en una fecha en concreto.
- GeneticAlgorithm: Clase que implementa un algoritmo para generar un horario con todos los datos del sistema.
- Horario: Conjunto de sesiones de asignaturas en los que no se produce ningún solapamiento, es decir, clases de la misma titulación y curso del mismo grupo no se imparten a la misma hora, así como turnos de dos asignaturas que son requisito la una de la otra.

Relación de clases

David Antonio Bernal Soto

GeneticAlgorithm.java
Population.java
Chromosome.java
Gene.java
vistaGenerarHorario.java
vistaHorario.java
vistaMenuHorario.java

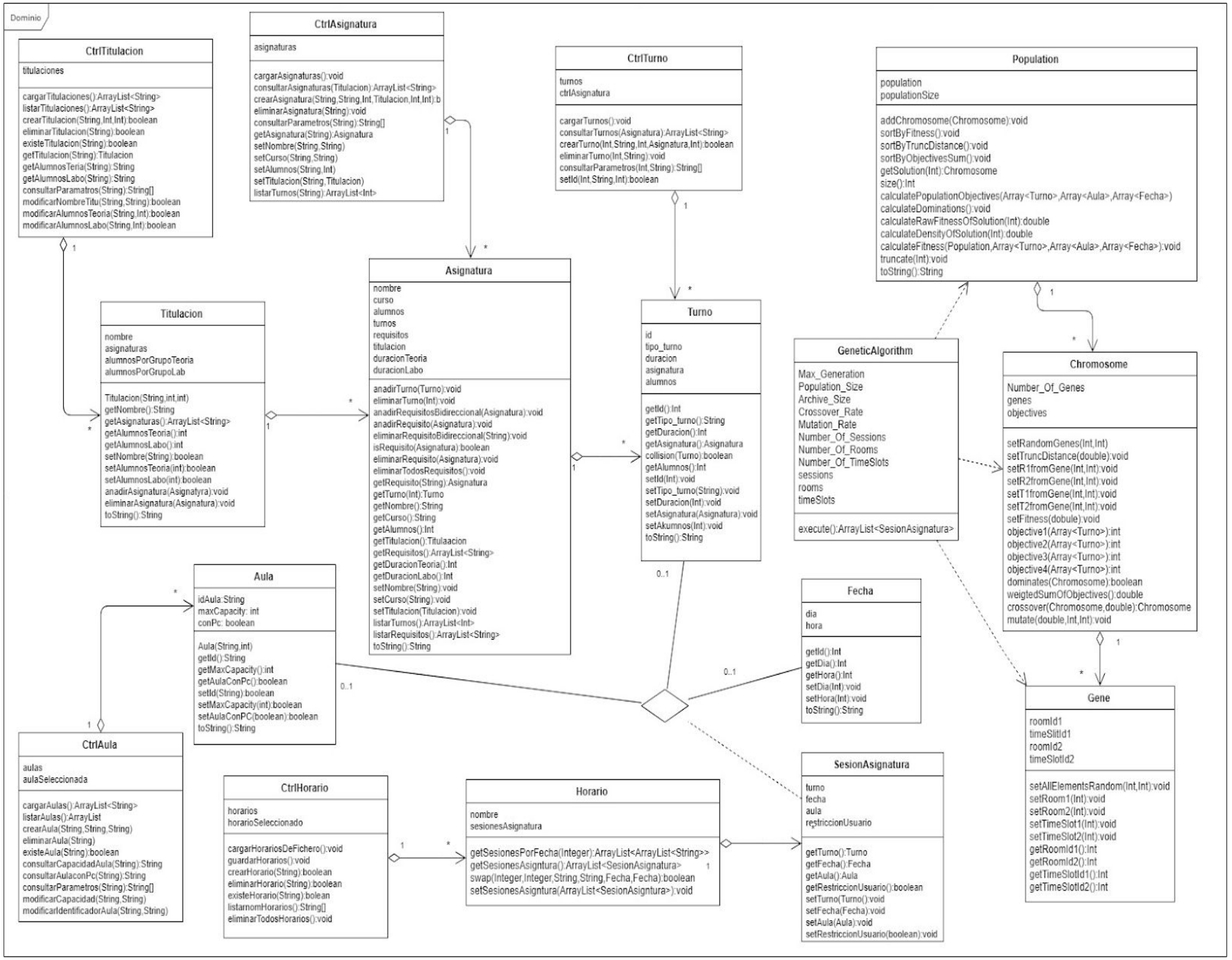
Gerard Candón Arenas

Asignatura.java
Turno.java
Fecha.java
SesionAsignatura.java
CtrlAsignatura.java
CtrlTurno.java
CtrlFecha.java
CtrlDominio.java
VistaAsignaturas.java
VistaAnyadirRequisito.java
VistaGestionAsignatura.java
VistaMenu.java
CtrlDatosAsignatura.java
CtrlDatosTurno.java
main.java

Miguel Salamero Muñoz

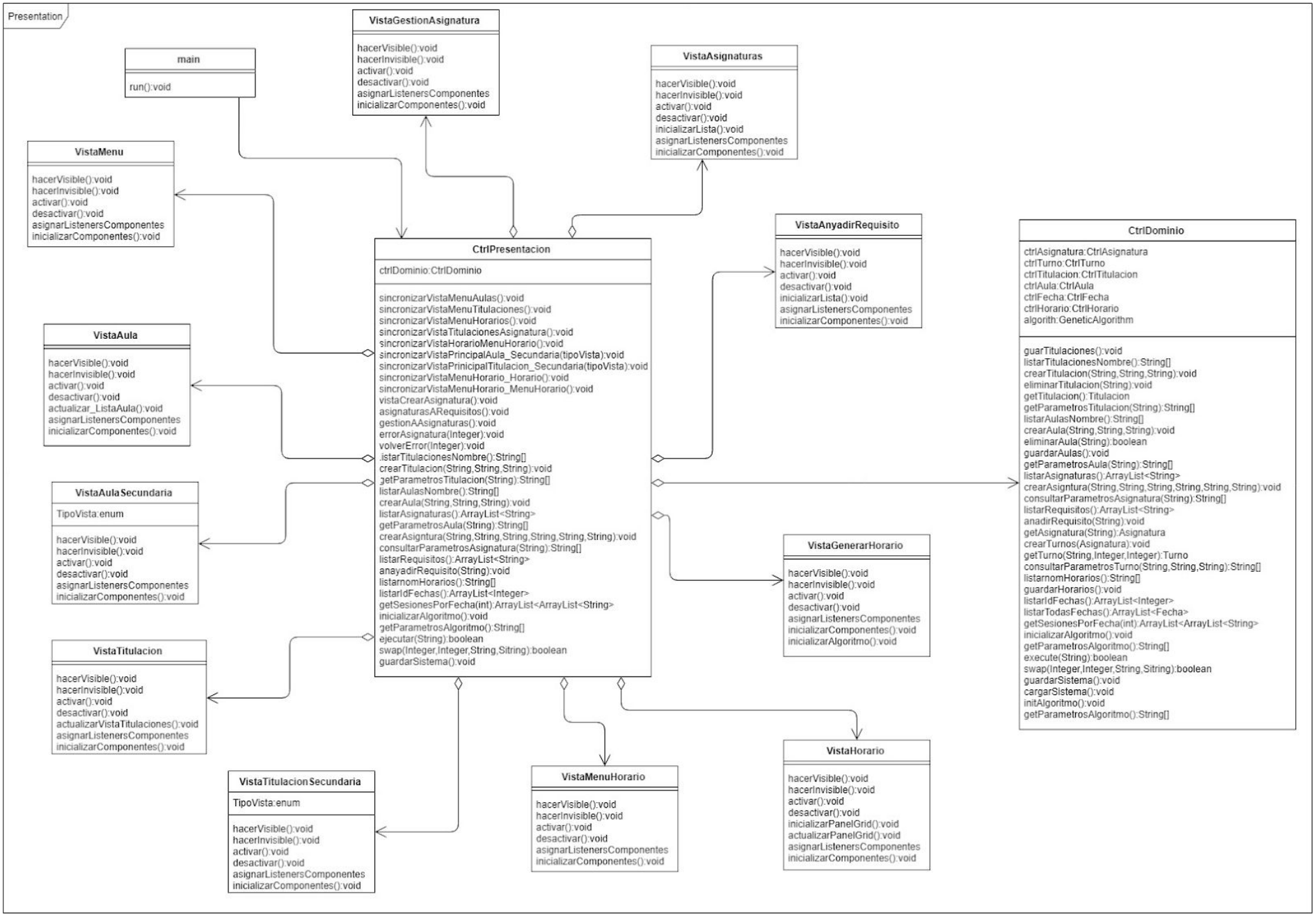
Aula.java
Titulacion.java
Horario.java
CtrlAula.java
CtrlTitulacion.java
CtrlHorario.java
vistaAula.java
vistaAulaSecundaria.java
vistaTitulacion.java
vistaTitulacionSecundaria.java
CtrlDatosAula.java
CtrlDatosHorario.java
CtrlDatosTitulacion.java

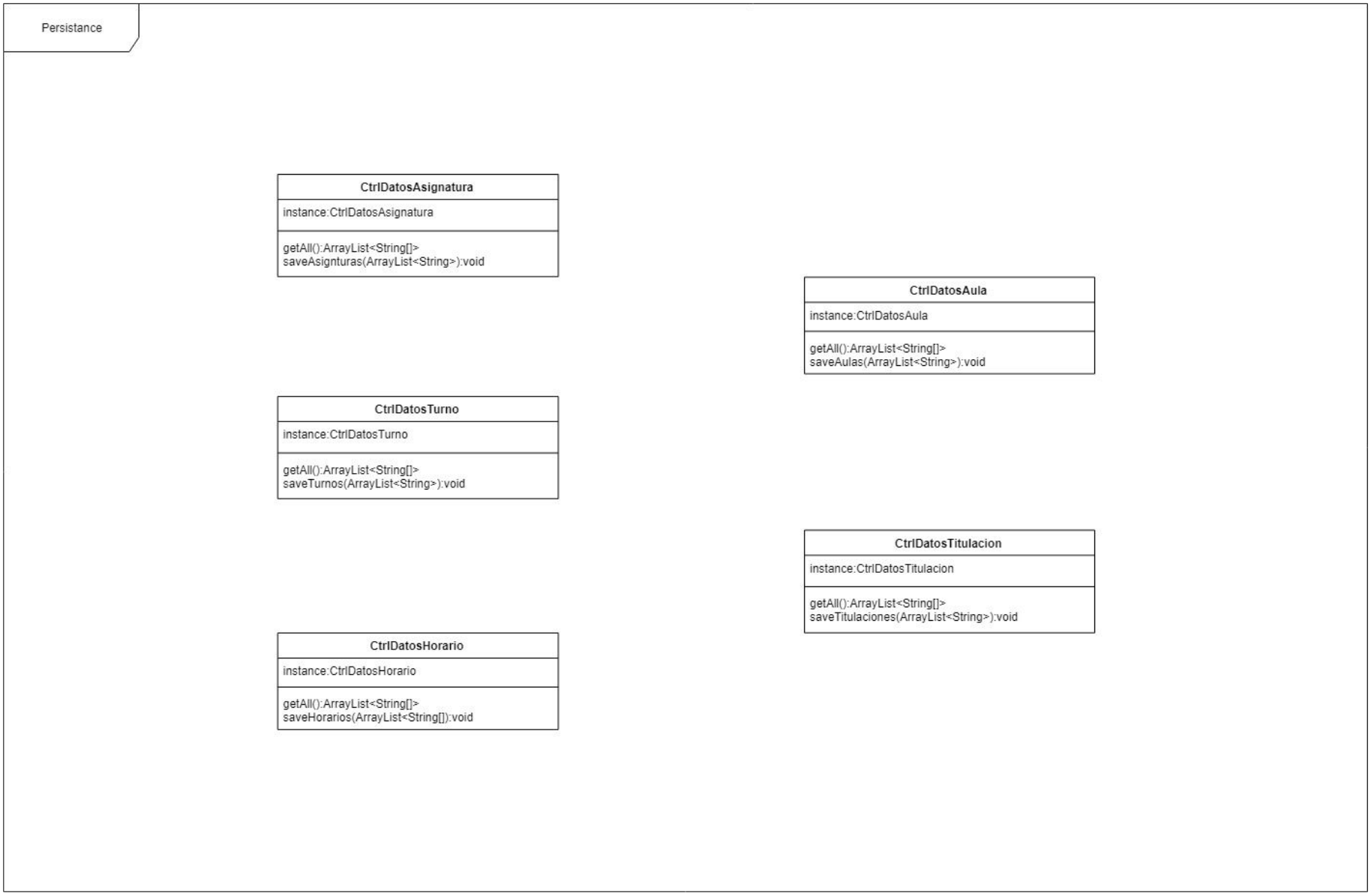
Conjunta entre los tres: CtrlPresentacion.java


Dominio
CtrlTitulacion
titulaciones
cargarTitulaciones():ArrayList<String>
listarTitulaciones():ArrayList<String>
crearTitulacion(String,Int,Int):boolean
eliminarTitulacion(String):boolean
existeTitulacion(String):boolean
getTitulacion(String):Titulacion
getAlumnosTeria(String):String
getAlumnosLabo(String):String
consultarParamatros(String):String[]
modificarNombreTitu(String,String):boolean
modificarAlumnosTeoria(String,Int):boolean
modificarAlumnosLabo(String,Int):boolean
CtrlAsignatura
asignaturas
cargarAsignaturas():void
consultarAsignaturas(Titulacion):ArrayList<String>
crearAsignatura(String,String,Int,Titulacion,Int,Int):b
eliminarAsignatura(String):void
consultarParametros(String):String[]
getAsignatura(String):Asignatura
setNombre(String,String)
setCurso(String,String)
setAlumnos(String,Int)
setTitulacion(String,Titulacion)
listarTurnos(String):ArrayList<Int>
CtrlTurno
turnos
ctrlAsignatura
cargarTurnos():void
consultarTurnos(Asignatura):ArrayList<String>
crearTurno(Int,String,Int,Asignatura,Int):boolean
eliminarTurno(Int,String):void
consultarParametros(Int,String):String[]
setId(Int,String,Int):boolean
Population
population
populationSize
addChromosome(Chromosome):void
sortByFitness():void
sortByTruncDistance():void
sortByObjectivesSum():void
getSolution(Int):Chromosome
size():Int
calculatePopulationObjectives(Array<Turno>,Array<Aula>,Array<Fecha>)
calculateDominations():void
calculateRawFitnessOfSolution(Int):double
calculateDensityOfSolution(Int):double
calculateFitness(Population,Array<Turno>,Array<Aula>,Array<Fecha>):void
truncate(Int):void
toString():String
Titulacion
nombre
asignaturas
alumnosPorGrupoTeoria
alumnosPorGrupoLab
Titulacion(String,int,int)
getNombre():String
getAsignaturas():ArrayList<String>
getAlumnosTeoria():int
getAlumnosLabo():int
setNombre(String):boolean
setAlumnosTeoria(int):boolean
setAlumnosLabo(int):boolean
anadirAsignatura(Asignatyra):void
eliminarAsignatura(Asignatura):void
toString():String
Asignatura
nombre
curso
alumnos
turnos
requisitos
titulacion
duracionTeoria
duracionLabo
anadirTurno(Turno):void
eliminarTurno(Int):void
anadirRequisitosBidireccional(Asignatura):void
anadirRequisito(Asignatura):void
eliminarRequisitoBidireccional(String):void
isRequisito(Asignatura):boolean
eliminarRequisito(Asignatura):void
eliminarTodosRequisitos():void
getRequisito(String):Asignatura
getTurno(Int):Turno
getNombre():String
getCurso():String
getAlumnos():Int
getTitulacion():Titulaacion
getRequisitos():ArrayList<String>
getDuracionTeoria():Int
getDuracionLabo():Int
setNombre(String):void
setCurso(String):void
setTitulacion(Titulacion):void
listarTurnos():ArrayList<Int>
listarRequisitos():ArrayList<String>
toString():String
Turno
id
tipo_turno
duracion
asignatura
alumnos
getId():Int
getTipo_turno():String
getDuracion():Int
getAsignatura():Asignatura
collision(Turno):boolean
getAlumnos():Int
setId(Int):void
setTipo_turno(String):void
setDuracion(Int):void
setAsignatura(Asignatura):void
setAkumnos(Int):void
toString():String
GeneticAlgorithm
Max_Generation
Population_Size
Archive_Size
Crossover_Rate
Mutation_Rate
Number_Of_Sessions
Number_Of_Rooms
Number_Of_TimeSlots
sessions
rooms
timeSlots
execute():ArrayList<SesionAsignatura>
Chromosome
Number_Of_Genes
genes
objectives
setRandomGenes(Int,Int)
setTruncDistance(double):void
setR1fromGene(Int,Int):void
setR2fromGene(Int,Int):void
setT1fromGene(Int,Int):void
setT2fromGene(Int,Int):void
setFitness(dobule):void
objective1(Array<Turno>):int
objective2(Array<Turno>):int
objective3(Array<Turno>):int
objective4(Array<Turno>):int
dominates(Chromosome):boolean
weigtedSumOfObjectives():double
crossover(Chromosome,double):Chromosome
mutate(double,Int,Int):void
Aula
idAula:String
maxCapacity: int
conPc: boolean
Aula(String,int)
getId():String
getMaxCapacity():int
getAulaConPc():boolean
setId(String):boolean
setMaxCapacity(int):boolean
setAulaConPC(boolean):boolean
toString():String
Fecha
dia
hora
getId():Int
getDia():Int
getHora():Int
setDia(Int):void
setHora(Int):void
toString():String
Gene
roomId1
timeSlitId1
roomId2
timeSlotId2
setAllElementsRandom(Int,Int):void
setRoom1(Int):void
setRoom2(Int):void
setTimeSlot1(Int):void
setTimeSlot2(Int):void
getRoomId1():Int
getRoomId2():Int
getTimeSlotId1():Int
getTimeSlotId2():Int
CtrlAula
aulas
aulaSeleccionada
cargarAulas():ArrayList<String>
listarAulas():ArrayList
crearAula(String,String,String)
eliminarAula(String)
existeAula(String):boolean
consultarCapacidadAula(String):String
consultarAulaconPc(String):String
consultarParametros(String):String[]
modificarCapacidad(String,String)
modificarIdentificadorAula(String,String)
CtrlHorario
horarios
horarioSeleccionado
cargarHorariosDeFichero():void
guardarHorarios():void
crearHorario(String):boolean
eliminarHorario(String):boolean
existeHorario(String):bolean
listarnomHorarios():String[]
eliminarTodosHorarios():void
Horario
nombre
sesionesAsignatura
getSesionesPorFecha(Integer):ArrayList<ArrayList<String>>
getSesionesAsigntura():ArrayList<SesionAsignatura>
swap(Integer,Integer,String,String,Fecha,Fecha):boolean
setSesionesAsigntura(ArrayList<SesionAsigntura>):void
SesionAsignatura
turno
fecha
aula
restriccionUsuario
getTurno():Turno
getFecha():Fecha
getAula():Aula
getRestriccionUsuario():boolean
setTurno(Turno):void
setFecha(Fecha):void
setAula(Aula):void
setRestriccionUsuario(boolean):void

Restricciones textuales:

1. No pueden haber dos titulaciones con el mismo nombre.
2. No pueden haber dos aulas con el mismo Id.
3. No pueden haber dos asignaturas con el mismo nombre.
4. No pueden haber dos turnos del mismo id y la misma asignatura con el mismo identificador.
5. No pueden haber más de 100 titulaciones.
6. Una titulación no puede tener más de 100 asignaturas.
7. No pueden haber dos horarios con el mismo nombre.
8. No pueden haber dos fechas con el mismo identificador.
9. No pueden haber dos sesiones de asignatura con la misma fecha, aula y turno.
10. Un turno no puede pertenecer a una sesión de asignatura que se hace en una aula con capacidad menor a sus alumnos.


Presentation
main
run():void
VistaGestionAsignatura
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
asignarListenersComponentes
inicializarComponentes():void
VistaAsignaturas
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
inicializarLista():void
asignarListenersComponentes
inicializarComponentes():void
VistaMenu
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
asignarListenersComponentes
inicializarComponentes():void
CtrlPresentacion
ctrlDominio:CtrlDominio
sincronizarVistaMenuAulas():void
sincronizarVistaMenuTitulaciones():void
sincronizarVistaMenuHorarios():void
sincronizarVistaTitulacionesAsignatura():void
sincronizarVistaHorarioMenuHorario():void
sincronizarVistaPrincipalAula_Secundaria(tipoVista):void
sincronizarVistaPrincipalTitulacion_Secundaria(tipoVista):void
sincronizarVistaMenuHorario_Horario():void
sincronizarVistaMenuHorario_MenuHorario():void
vistaCrearAsignatura():void
asignaturasARequisitos():void
gestionAAsignaturas():void
errorAsignatura(Integer):void
volverError(Integer):void
.istarTitulacionesNombre():String[]
crearTitulacion(String,String,String):void
getParametrosTitulacion(String):String[]
listarAulasNombre():String[]
crearAula(String,String,String):void
listarAsignaturas():ArrayList<String>
getParametrosAula(String):String[]
crearAsigntura(String,String,String,String,String,String):void
consultarParametrosAsignatura(String):String[]
listarRequisitos():ArrayList<String>
anayadirRequisito(String):void
listarnomHorarios():String[]
listarIdFechas():ArrayList<Integer>
getSesionesPorFecha(int):ArrayList<ArrayList<String>
inicializarAlgoritmo():void
getParametrosAlgoritmo():String[]
ejecutar(String):boolean
swap(Integer,Integer,String,Sitring):boolean
guardarSistema():void
VistaAnyadirRequisito
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
inicializarLista():void
asignarListenersComponentes
inicializarComponentes():void
CtrlDominio
ctrlAsignatura:CtrlAsignatura
ctrlTurno:CtrlTurno
ctrlTitulacion:CtrlTitulacion
ctrlAula:CtrlAula
ctrlFecha:CtrlFecha
ctrlHorario:CtrlHorario
algorith:GeneticAlgorithm
guarTitulaciones():void
listarTitulacionesNombre():String[]
crearTitulacion(String,String,String):void
eliminarTitulacion(String):void
getTitulacion():Titulacion
getParametrosTitulacion(String):String[]
listarAulasNombre():String[]
crearAula(String,String,String):void
eliminarAula(String):boolean
guardarAulas():void
getParametrosAula(String):String[]
listarAsignaturas():ArrayList<String>
crearAsigntura(String,String,String,String,String,String):void
consultarParametrosAsignatura(String):String[]
listarRequisitos():ArrayList<String>
anadirRequisito(String):void
getAsignatura(String):Asignatura
crearTurnos(Asignatura):void
getTurno(String,Integer,Integer):Turno
consultarParametrosTurno(String,String,String):String[]
listarnomHorarios():String[]
guardarHorarios():void
listarIdFechas():ArrayList<Integer>
listarTodasFechas():ArrayList<Fecha>
getSesionesPorFecha(int):ArrayList<ArrayList<String>
inicializarAlgoritmo():void
getParametrosAlgoritmo():String[]
execute(String):boolean
swap(Integer,Integer,String,Sitring):boolean
guardarSistema():void
cargarSistema():void
initAlgoritmo():void
getParametrosAlgoritmo():String[]
VistaAula
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
actualizar_ListaAula():void
asignarListenersComponentes
inicializarComponentes():void
VistaAulaSecundaria
TipoVista:enum
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
asignarListenersComponentes
inicializarComponentes():void
VistaGenerarHorario
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
asignarListenersComponentes
inicializarComponentes():void
inicializarAlgoritmo():void
VistaTitulacion
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
actualizarVistaTitulaciones():void
asignarListenersComponentes
inicializarComponentes():void
VistaTitulacionSecundaria
TipoVista:enum
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
asignarListenersComponentes
inicializarComponentes():void
VistaMenuHorario
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
asignarListenersComponentes
inicializarComponentes():void
VistaHorario
hacerVisible():void
hacerInvisible():void
activar():void
desactivar():void
inicializarPanelGrid():void
actualizarPanelGrid():void
asignarListenersComponentes
inicializarComponentes():void


Persistance
CtrlDatosAsignatura
instance:CtrlDatosAsignatura
getAll():ArrayList<String[]>
saveAsignturas(ArrayList<String>):void
CtrlDatosAula
instance:CtrlDatosAula
getAll():ArrayList<String[]>
saveAulas(ArrayList<String>):void
CtrlDatosTurno
instance:CtrlDatosTurno
getAll():ArrayList<String[]>
saveTurnos(ArrayList<String>):void
CtrlDatosTitulacion
instance:CtrlDatosTitulacion
getAll():ArrayList<String[]>
saveTitulaciones(ArrayList<String>):void
CtrlDatosHorario
instance:CtrlDatosHorario
getAll():ArrayList<String[]>
saveHorarios(ArrayList<String[]):void