# Pràctica de PROP 2018-2019 Q1

# Generador d'Horaris

Juan Sebastian Brito
Jinson Pardo
Pere Vergés Boncompte

# INDEX

# Documentació de les classes

## Classe Assignatura

- **Descripció:** Assignatura d'un pla d'estudis i titulació.
- **Cardinalitat:** 1 **.. *,** Relació amb GrupAssignatura i Pla d'estudis i 1 amb escenari
- **Atributs:**
    - **nom:** String
    - **id:** String
    - **nº credits:** Int
    - **quadrimestre:** Int
    - **numGrups:** Int
    - **numSubgrupsDeGrups:** Int
    - **numSessionsSetmana:** Int
    - **numHoresSetmana:** Int
    - **grups:** ArrayList<GrupAssignatura>
    - **fase:** String
    - **correquisists:** ArrayList<Assignatura>
    - **inverseCorrequisists:** ArrayList<Assignatura>
- **Mètodes:**
    - **void print():** Fa un print de tots els seus atributs.
    - **getters i setters**

## Classe GrupAssignatura:

- **Descripció:** Grup pertanyent a una assignatura, és el que fem servir per associar un grup a l'horari.
- **Cardinalitat:** 0 **.. *,** Relació amb Assignatura, Aula, Restriccions i Posibilidad
- **Atributs:**
    - **numGrup:** Int
    - **numPlaces:** Int
    - **tipusAula:** TipusAula
    - **idAssignatura:** Strin
    - **aula:** Aula
    - **horari:** Horari
    - **posibilidades:** List<Posibilidad>
- **Mètodes:**
    - **void print():** Fa un print de tots els seus atributs.

- **void generarPosibilidades():** Genera totes les posibles combinaciones d'aula i horari

**- void isValid():** Fa una poda del domini segons el tipus i aforament de les aules

**- getters i setters**

## Classe Aula

- **Descripció:** Aula on esta assignat un GrupAssignatura
- **Cardinalitat: *,** Relació amb GrupAssignatura i 1 amb escenari
- **Atributs:**
  - **id:** String
  - **aforament:** Int
  - **tipus:** TipusAula

**- Mètodes:**

**- void print():** Fa un print de tots els seus atributs.

**- getters i setters**

## Classe Pla d'Estudi:

- **Descripció:** Pla d'estudi del centre Universitari
- **Cardinalitat:** 1..* amb Assignatura, 1 amb PeriodeLectiu
- **Atributs:**
  - **nom:** String
  - **id:** String
  - **assignaturas:** Assignaturas
  - **PeriodeLectiu:** PeriodeLectiu

**- Metodes:**

**- getters i setters**

## Classe PeriodeLectiu:

- **Descripció:** Hora començament i final de les classes, dia Inici i Final de la setmana lectius (ex: Ini: DILLUNS, Fi: DIVENDRES)
- **Cardinalitat: 1,** Relació amb Pla d'estudis i 1 amb escenari
- **Atributs:**
  - **nom:** String
  - **id:** String
  - **horaInici:** Int
  - **horaFinal:** Int

- **DiaInici:** DiaSetmana
- **DiaFinal:** DiaSetmana

**- Metodes:**

**- getters i setters**

## Classe User:

- **Descripció:** És l'usuari que farà servir l'aplicació, es distingeixen dos tipus: Admin o Alumne
- **Cardinalitat:** 1**..*,** Relació amb GrupAssignatura
- **Atributs:**
  - **nom:** String
  - **id:** String
  - **password:** password
  - **tipus:** TipusUser

**- Metodes:**

**- getters i setters**

## Classe Horari:

- **Descripció:** És l'horari que s'assigna a un GrupAssignatura com a horari de classe
- **Cardinalitat:** ***,** Relació amb Restriccions
- **Atributs:**
  - **horaInici:** String
  - **durada:** String
  - **dia:** DiaSetmana

**- Mètodes:**

**- void print():** Fa un print de tots els seus atributs.

**- List<Horari> generateAllPosibleHoraris():** Genera tots els horaris possibles en un periode lectiu.

**- solapa:** cert si dos grups solapen, altrament fals

**- getters i setters**

## Classe Posibilidad:

- **Descripció:** És una combinació d'aula i horari
- **Cardinalitat:** 1 amb Aula i 1 amb Horari
- **Atributs:**
    - **aula:** Aula
    - **horari:** String

- **Mètodes:**
    - **Aula getAula()**
    - **void setAula()**
    - **Horari getHorari()**
    - **voidsetHorari()**

## Classe Escenari:

- **Descripció:** És una representació d'un escenari, que inclou assignatures, aules, restriccions i un periode lectiu
- **Cardinalitat:** * Relació amb GrupAssignatura, Aula, Restriccions i 1 amb periodeLectiu
- **Atributs:**
    - **assignaturas:** Map<String, Assignatura>
    - **aulas:** Map<String, Aula>
    - **restriccions:** Map<String, Restriccions>
    - **periodeLectiu:** PeriodeLectiu

- **Mètodes:**
    - **void setEscenari():** Crea un escenari
    - **void addAssignatura()** Afegeix Assignatura
    - **void removeAssignatura()** Elimina Assignatura
    - **List<Assignatura> getAssignaturas()** Retorna les assignatures
    - **Assignatura getAssignatura()** Retorna una assignatura
    - **List<Aula> getAulas()** Retrona una llista d'aules
    - **Aula getAula()** Retorna una aula
    - **boolean existeixAula()** Mira si existeix una aula
    - **boolean existeixAssignatura()** Mira si existeix una assignatura
    - **void addAula()** afegeix una aula
    - **void removeAula()** elimina una aula
    - **PeriodeLectiu getPeriodeLectiu()** retorna un periodeLectiu
    - **boolean existeixRestriccio()** mira si existeix una restriccio

- **void addRestriccions()** afegiex una restriccio
- **void removeRestriccions()** elimina una restriccio
- **List<Restriccions> getRestriccionsAsList()** retorna una llista de restriccions

## Classe Restriccions:

- **Descripció:** Són les restriccions que ha de cumplir un horari per ser correcte.
- **Cardinalitat: *,** Relació amb Horari i GrupAssignatura i 1 amb escenari
- **Atributs:**
  - **tipus:** String
  - **id:** String
  - **ids:** List<String>

**- Mètodes:**

**- String getNom():** Obté el nom o tipus de la restricció

**- List<String> getIds():** Obté els ids de les assignatures que afecta la restricció.

**- String getId():** Obté l'id de la restricció

**- void propagarRestriccions():** A partir d'una assignació a un grup, propaga les restriccions a tota la resta dels grups.

**- void propagarTeoriaAndLabDifferentDay():** Poda totes les possibilitats que no compleixen la restricció en la que un grup de teoria i els seus subgrups han de fer classe en dies diferents.

**- void propagarGruposDiferenteAula():** Poda totes les possibilitats que no compleixen la restricció en la que dos grups no poden fer classe a la misma hora i aula.

**- void propagarNoSolapamentMateixaFase():** Poda totes les possibilitats que no compleixen la restricció en la que dos assignatures de la mateixa fase no se solapin.

**- void propagarRestriccionsPersonalizades():** Poda totes les possibilitats que no compleixen la restricció en la que dos assignatures del mateix subnivell no se solapin.

**- boolean horariDinsdePeriodeLectiu():** Cert si l'horari es troba dins del periode lectiu, altrament fals.

**- boolean grupFitsAula():** Cert si el grup cap a l'aula, altrament fals

**- boolean aulaCorrecta():** Cert si el grup te assignada una aula del tipus correcte, altrament fals

**- boolean teoriaAndLabDifferentDay():** Cert si el grup de teoria i laboratori d'un mateixa assignatura fan classe diferents dies, altrament fals

**- boolean correquisists():** Cert si no es fan classes el mateix dia i hora dins d'assignatures que son correquisits, altrament fals.

**- boolean gruposDiferenteAula():** Cert si no es fan classes el mateix dia i hora i mateixa aula, altrament fals.

**- boolean noSolapamentMateixaFase():** Cert si no es fan a la mateixa hora i dia classes de la mateixa fase i grup, altrament fals.

**- boolean RestriccionesPersonalizadaSubnivel():** Cert si no se solapa cap assignatura del mateix subnivell

**- boolean horaDinsInterval():** Cert si l'horari del grup es troba dins l'interval definit a les restricciones de l'escenari

**- boolean horariDinsInterval():** Cert si l'horari es troba dins l'interval definit a les restricciones de l'escenari

**- int getHoraIniRestriccioInterval():** Hora d'inici de la restricció d'interval

**- int getHoraFiRestriccioInterval():** Hora final de la restricció d'interval

**- String getIdAssigRestriccioInterval():** Id de l'assignatura de la restricció d'interval

## Classe AdminController:

- **Descripció:** És el controllador dels admins, permet generar horaris
- **Cardinalitat: *** amb Restriccions, **1** amb PeriodeLectiu, **1..*** amb Aules, 1..* amb Assignatures, 1..* amb Horaris, 1..* amb Grups
- **Atributs:**
  - **escenari: Escenari**

**- Mètodes:**

**- List<GrupAssignatura> generarHorari():** Genera un horari a partir d'unes assignatures, aules i un periode lectiu.

**- List<Horari> forwardChecking():** Algoritme de backtracking amb forward checking usat en el métode generarHorari(), rep una llista de grups sense aules ni horari, una altra llista de grups buida i retorna una llista de grups amb aules i horaris assignats a cada grup.

- **boolean isValid()**: Rep una llista de grups i comprova si compleix totes les restriccions.

**- List<GrupAssignatura> fallo():** Retorna una llista de grups que representan una solució no vàlida (l'últim grup té numGrup = -1).

**- boolean esFallo():** Rep una llista de grups i comprova si representa una solució no vàlida.

**- List<GrupAssignatura> crearGrupos():** Rep una llista d'assignaturas i retorna una llista amb tots els seus grups, sense aula ni horari assignats.

## Classe CtrlDominio:

- **Descripció:** És el controlador de domini.
- **Cardinalitat:** 1 amb CtrlDominioMantUser, 0..1 amb AdminController
- **Atributs:**
  - **ctrlDominioMantUser:** CtrlDominioMantUser
  - **escenari:** Escenari
  - **nomFitxersAAF:** List<String>
  - **horari:** List<GrupAssignatura>
  - **ctrlFitxerGrupAssig:** CtrlFitxerGrupAssig

- **Mètodes:**

**- List<GrupAssignatura> generarHorari():** Rep una llista amb el nom dels fitxers de les assignatures i les aules per generar l'horari, i el retorna en una llista de grups d'assignatura.

**- List<Horari> generateAllPosibleHoraris():** Genera tots els horaris possibles en un període lectiu.

**- List<Assignatura> cargarAssignaturas():** Rep el nom d'un fitxer i retorna la llista d'assignatures guardada a aquest fitxer.

**- List<Aula> cargarAules():** Rep el nom d'un fitxer i retorna la llista d'aules guardada a aquest fitxer.

**- List<PeriodeLectiu> cargarPeriodeLectiu():** Rep el nom d'un fitxer i retorna la llista de períodes lectius guardats a aquest fitxer.

**- List<GrupAssignatura> cargarHorari():** Rep el nom d'un fitxer i retorna l'horari guardat a aquest fitxer.

**- void guardarHorari():** Rep un horari (llista de grups), un nom de fitxer i crea un fitxer amb aquest horari.

**- void printHorari():** Printa l'horari rebut com a paràmetre.

**- void modificar():** Modifica un grup assignatura de aula i de horari si es possible, en cas que no sigui possible treu un missatge explicitant el motiu

**- void afegirAulaEscenari():** Afegeix una aula si es possible al escenari

**- void afegirAssignaturaEscenari():** Afegeix una assignatura si es possible al escenari

**- void afegirRestriccioEscenari():** Afegeix una restriccio si es possible al escenari.

**- void eliminarAulaEscenari():** Elimina una aula si es possible al escenari.

**- void eliminarAssignaturaEscenari():** Elimina una assignatura si existeix, del escenari.

**- void eliminarRestriccioEscenari():** Afegeix una restriccio si existeix, del escenari.

## Classe CtrlDominioMantUser:

- **Descripció:** És el controlador de domini dels usuaris.
- **Cardinalitat: *,** Relació amb Usuaris
- **Atributs:**
  - **usuaris:** TreeMap<String, User>
  - **usuariActiu:** User
  - **ctrlFitxerUser:** ctrlFitxerUser

- **Mètodes:**
  - **void carregarUsuaris():** Carrega els usuaris dins l'atribut usuaris.
  - **User getUsuari():** Retorna un User a partir d'un id.

## Enumeration

**DiaSetmana**

- **Descripció:** Son els dies de la setmana
- **Valors:**
  - DILLUNS
  - DIMARTS
  - DIMECRES
  - DIJOUS
  - DIVENDRES
  - DISSABTE
  - DIUMENGE

**TipusAula**

- **Descripció:** Son els tipus d'aula dins una facultat
- **Valors:**
  - TEORIA
  - LABORATORI
  - PROBLEMES

**TipusUser**

- **Descripció:** Son els tipus d'usuari
- **Valors:**
  - ADMIN
  - ALUMNE

# Descripció de l'estructura de dades i algorismes utilitzats

Les estructures de dades que hem fet servir bàsicament han sigut dues, llistes i maps, les llistes les hem fet servir perquè és una de les estructures més basiques i eficients, i hem fet servir el map per tal de que en el escenari no hi hagi elements duplicats i fer la cerca i eliminació d'elements sigui molt més fàcil de dur a terme.

Per implementar la funcionalitat de crear l'horari hem utilizat un algorisme de backtraking amb forward checking, que segons les restriccions establertes crearà l'horari. I abans de començar es fa la poda d'algunes de les restriccions unàries.

A partir d'un conjunt d'assignatures, aules, restriccions i un període lectiu, generarà totes les hores disponibles per fer classe, i tots els grups, de moment sense aula ni hora assignades, però amb una llista de totes les combinacions d'aula i horari possible segons les restriccions, tant implícites com explícites.

Amb aquests paràmetres, farà un backtracking amb forward checking en el qual anirà assignant a cada grup alguna de les possibles combinacions d'hora i aula. A cada assignació podarà de la resta de grups les possibilitats que a partir d'ara no compleixin les restriccions.

Si en algun moment algun dels grups es queda sense possibles combinacions d'hora i aula, l'algorisme farà un backtracking, desfent l'última assignació i intentant una diferent. En el moment que es generi un horari vàlid es mostrarà a l'usuari, si no es pot trobar cap horari que satisfaci totes les restriccions, ho notificará.

# Classes implementades per cada integrant del grup

| | |
|---|---|
| CtrlFitxerAssig | Pere Vergés |

| | |
|---|---|
| CtrlFitxerAules | Pere Vergés |
| CtrlFitxerGrupAssig | Pere Vergés |
| CtrlFitxerPeriodeLectiu | Pere Vergés |
| CtrlFixerUser | Pere Vergés |
| Admin | Jinson Pardo |
| Alumne | Jinson Pardo |
| Assignatura | Pere Vergés |
| Aula | Pere Vergés |
| GrupAssignatura | Juan Brito |
| Horari | Juan Brito |
| PeriodeLectiu | Juan Brito |
| PlaEstudis | Juan Brito |
| Restriccions | Juan Brito i Pere Vergés |
| User | Jinson Pardo |
| AdminCotroller | Juan Brito |
| CtrlDominio | Pere Vergés |
| CtrlDominioMantUser | Jinson Pardo |
| CtrlPresentacionVistaPrincipal | Jinson Pardo |
| CtrlPresentacio | Jinson Pardo |
| CtrlPresentacioVistaLogin | Jinson Pardo |
| CtrlPresentacioVistaUser | Jinson Pardo |
| VistaLogin | Jinson Pardo |
| VistaPrincipal | Jinson Pardo |
| VistaUser | Jinson Pardo |
| AdminControllerTest | Juan Brito |
| DriverAdmin | Pere Vergés |
| DriverAlumne | Pere Vergés |

| | |
|---|---|
| DriverAssignatura | Pere Vergés |
| DriverAula | Pere Vergés |
| DriverCtrlDominio | Pere Vergés |
| DriverGrupAssignatura | Pere Vergés |
| DriverHorari | Pere Vergés |
| DriverPeriodeLectiu | Pere Vergés |
| DriverUser | Pere Vergés |
| HorariTest | Juan Brito |
| RestriccionsTest | Juan Brito |
| CtrlDriver | Pere Vergés |
| DriverRestriccions | Pere Vergés |
| Escenari | Pere Vergés |
| Posibilidad | Juan Brito |
| CtrlFitxerRestriccions | Pere Vergés |
| ViewLogin | Jinson Pardo |
| ViewPrincipal | Jinson Pardo |
| LlistaGruaAssigYfilename | Pere Vergés |
| JocsDeProvaFinal | Pere Vergés |

Els fitxers dels jocs de proves han estat fets per Pere Vergés.
La documentació ha estat feta per Juan Brito i Pere Vergés

# Llibreries externes utilitzades

Hem fet servir la llibreria *json-simple* per tal de poder llegir i guardar les dades d'una manera més còmoda i rapida.

# Llista de funcionalitats implementadas

Les funcionalitats que hem implementat són las de carregar fitxer d'assignatures, carregar fitxer d'aules, carregar fitxer del període lectiu i carregar fitxer de restriccions, també per a cada fitxer mencionat també hem creat la seva respectiva funció de guardar el fitxer, per tal de guardar els canvis que puguin haver-se fet. Aquesta serien pel que fa a carregar i guardar fitxers de l'escenari, després també hem implementat guardar i carregar un horari. Un de les coses importants que hem aconseguit és que qualsevol horari guardat prèviament es pot editar tant l'horari en si (moure grups de hora i aula) i també editar el seu escenari.
Pel que fa a la edició de l'escenari el que permet fer és afegir o eliminar aules, assignatures i restriccions, a més a les restriccions permetem editar-les directament.
També com es demana hi ha la funcionalitat de modificar l'horari, és a dir, canviar un grup de aula i hora, evidentment quan es fa això es mira que totes les restriccions es compleixin.

Fins ara, aprat de que tot horari generat es pugui modificar tant l'escenari com les restriccions, serien les funcionalitats que venien especificades per l'enunciat de la pràctica, després nosaltres el que hem fet és afegir un parell més de restriccions que consisteixen:

- La primera restriccio consisteix en una llista d'assignatures que no poden solapar-se en les classes de la mateixa desena en un mateix dia i hora, nosaltres a aquesta restriccio la hem anomenada com a restriccio de subnivell, ja que es molt semblant a la restriccio de nivell que ja ve directament implementada amb el codi.
- La segona restriccio que hem decidit afegir és una restriccio que consisteix en limitar a una assignatura en un interval horari determinat, nosaltres la hem anomenada com restriccio de interval horari.

En la restriccio de subnivell permetem afegir o eliminar assignatures a una restriccio especifica, en canvi a la segona restriccio directament permentem editar l'interval horari.

# Documentació Jocs de Prova

Aquest primer apartat dels jocs de proves son els que varem fer per a la primera prova, els jocs de prova importants són els que venen a continuació d'aquests, ja que s'ensenya com

funciona el generador amb totes les funcionalitat de recuperació d'horaris antics per poder editar-los i com podem modificar, afegir i eliminar les restriccions que hem decidit afegir nosaltres, la de subnivell i la de interval horari.

## Jocs de proves antics

Hem creat diversos escenaris predeterminats amb fitxers d'Assignatures i Aules per tal de veure el correcte funcionament de l'algorisme de backtracking.
Els dos primers escenaris són trivials i és per veure en general que la presentació de l'horari funciona correctament i que les restriccions més basiques (unaries) funcionen de manera correcta.
Per tant, en aquests dos escenaris (Escenari 1: Arxius: Assignaturas.json i Aules.json, que es d'una mida mitjana i Escenari 2: Arxius: AssignaturasProvaGran.json i AulesProvaGran.json, que es d'una mida més gran) es poden veure clarament com els grups estan dins del període lectiu, un grup no té assignada una aula a la qual no hi cap, o un grup no pot estar en una aula que no és del seu tipus, una classe no pot acabar fora d'horari lectiu.
Després hem generat 4 escenaris més:
En primer usem els fitxers: Assignaturas.json i AulesPetita.json, per tal de veure que no és possible generar un horari amb moltes assignatures i molt poques aules, l'algoritme ens avisarà de que no s'ha pogut crear cap horari..
En el segon escenari usem els fitxers: AssignaturasMateixNivell.json i AulesPetita.json per tal de veure que dos assignatures amb el mateix nivell no es solapen en hora i dia.
En el tercer escenari usem els fitxers: AssignaturasAmbCorrequisits.json i AulesPetita.json per tal de veure que dos assignatures que son correquisits no es solapen en hora i dia.
A l'últim escenari usem els fitxers:
AssignaturesClasseGrupISubGrupNoPotSerElMateixDia.json i AulesPetita.json per tal de veure que no es pot fer classe de teoria i laboratori el mateix dia.

## Jocs de proves nous (2a Entrega)

**Prova 1**

Descripció: Escenari amb poques assignatures, i només la restricció de interval horari.
Entrada:

Assignatures: AssignaturesPetitaPerRestriccionsDeIntervalISubnivell
Aules: Aules
PeriodeLectiu: PeriodeLectius
Restriccions: RestriccionsIntervalHorariSenzill

Sortida: La sortida esperada es que s'apliqui la restricció interval en l'assignatura de LI i de PROP, que en aquest cas es que les classes de LI només poden estar en l'interval de 8 a 12 i les de PROP en l'interval 12 a 16.

**Prova 2**

Descripció: Escenari amb poques assignatures, i només la restricció de interval horari, fent que sigui massa restrictiu
Entrada:

Assignatures: AssignaturesPetitaPerRestriccionsDeIntervalISubnivell
Aules: AulesPetita
PeriodeLectiu: PeriodeLectius
Restriccions: RestriccionsIntervalHorariSenzillRestrictiu

Sortida: La sortida esperada es que no es pugui crear cap horari ja que la restricció de interval horari és massa restrictiva en aquest escenari.

**Prova 3**

Descripció: Escenari amb poques assignatures, i assignatures sense correquisits ni mateix nivell, i sense restriccions, el que farem és veure com podem modificar al principi l'horari pero a mesura que anem afegint restriccions de INTERVAL HORARI comencem a veure conflictes.
Entrada:

Assignatures: AssignaturesPetitaPerRestriccionsDeIntervalISubnivell
Aules: AulesPetita
PeriodeLectiu: PeriodeLectius
Restriccions: RestriccionsBuit
Modifiquiem: LI 10 Teoria A6202 de 14-16 del Dilluns, a l'aula A6201 de 8-10 del Dilluns. (Canvi Satisfactori).
Afegim restricció de interval horari de LI de 8-12, tornem a generar horari. Ara al generar l'horari les classes de LI només estan en el període de 8-12.
Modifiquiem: LI 10 Teoria A6202 de 8-10 del Dimarts, a l'aula A6202 de 16-18 del Dimarts. Error al modificar ja que la restricció de interval diu que ha d'estar entre les 8-12
Modifiquiem: LI 10 Teoria A6202 de 8-10 del Dimarts, a l'aula A6202 de 10-12 del Dimarts. Modificació correcte ja que està dins dels intervals

Sortida: La sortida esperada es que al principi es pugui modificar l'horari i després hi hagi o no conflictes segons si es poden dur a terme les restriccions.

**Prova 4**

Descripció: Escenari amb poques assignatures, i assignatures sense correquisits ni mateix nivell, i sense restriccions, el que farem és veure com podem modificar al principi l'horari pero a mesura que anem afegint restriccions SUBNIVELL comencem a veure conflictes.
Entrada:

Assignatures: AssignaturesPetitaPerRestriccionsDeIntervalISubnivell

Aules: AulesPetita

PeriodeLectiu: PeriodeLectius

Restriccions: RestriccionsBuit

Modifiquiem: LI 10 Teoria A6202 de 14-16 del Dilluns, a l'aula A6201 de 8-10 del Dilluns. (Canvi Satisfactori).

Afegim restricció de subnivell entre LI i PROP (les dues assignatures de l'arxiu), per tant el canvi que hem fet posteriorment no es podra fer, tornem a generar horari.

Modifiquiem: LI 10 Teoria A6202 de 8-10 del Dilluns, a l'aula A6201 de 8-10 del DIlluns. Error al modificar ja que la restricció de mateix subnivell fa que el grup 10 de LI i el grup 10 de prop no puguin fer classe en el mateix dia i hora.

Modifiquiem: LI 10 Teoria A6202 de 8-10 del Dilluns, a l'aula A6201 de 10-12 del DIlluns. Es modifica correctament ja que LI i PROP de grup 10 i grup 20 pot fer classe simultàneament.

Sortida: La sortida esperada es que al principi es pugui modificar l'horari i després hi hagi o no conflictes segons si es poden dur a terme les restriccions.

**Prova 5**

Descripció: Escenari amb poques assignatures, restricció de Interval i subnivell senzilles, aules petita. Simplement volem veure com l'horari es genera correctament amb aquestes restriccions per guardar-lo i poder editar-ho en el próxim joc de proves.
Entrada:

Assignatures: AssignaturesPetitaPerRestriccionsDeIntervalISubnivell

Aules: AulesPetita

PeriodeLectiu: PeriodeLectius

Restriccions: RestriccionsSenzillesIntervalHorariISubnivell

Sortida: La sortida esperada es que s'apliquin les dues restriccions i l'horari es guardi correctament.

Horari: jocDeProves6

**Prova 6**

Descripció: Partint de l'horari generat en la prova anterior el que volem primer de tot és poder canviar les restriccions per demostrar que es pot modificar un horari prèviament, posteriorment el que farem es eliminar completament alguna de les restriccions.

Entrada:

Horari: Joc de proves6.

Carreguem l'horari i veiem com surt correctament,

Modifiquem: La restriccio de interval de LI i ara li diem que és de 12-16 i veiem com l'horari a canviat per tal de complir la restriccio.

Eliminem: La restriccio de subnivell per complet, per comprovar que aquesta restriccio ja no existeix el que fem és modificar el grup 10 de LI i el posem amb un grup 10 de PROP i veiem com l'horari es pot crear perfectament ja que la restriccio de subnivell entre les dues assignatures ja no es existent.

**Prova 7**

Descripció: Escenari gran, amb un restriccio de interval horari senzilla.
Entrada:

Assignatures: AssignaturesProvaGran

Aules: Aules

PeriodeLectiu: PeriodeLectius

Restriccions: RestriccionsIntervalHorariSenzill

Veiem com es genera l'horari i compleix la restricció senzilla del horari.

Afegim més restriccions de interval horari a l'escenari per veure com evoluciona el temps d'execució i fins a quin moment es pot anar afegint restriccions, primer afegim la restriccio de M2 de 14-20, i l'horari es genera correctament, després n'afegim una de més restrictiva, XC de 8-10, l'horari es segueix poden generar, després afegim una de molt restricitva, IES de 8-10 i ara ja no es pot generar l'horari, així que decidim modificar IES de 8-10 per IES de 8-12 i l'horari es genera correctament una altra vegada. Així que demostrem que el nostre algoritme funciona bé per a data set més o menys grans. I que les nostres funcionalitats de afegir, modificar i eliminar restriccions també funciona ràpid i correctament per a data set de mida considerable.

**Prova 8**

Descripció: Escenari de mida normal, és a dir de assignatures i aules amb una mida normal, pero la part interessant d'aquest joc de proves es que el fitxer de restriccions es molt restricitu, per tant el volem fer en aquest joc de proves és com va responent el nostre generador d'horaris a mesura que anem eliminant restriccions
Entrada:

Assignatures: Assignatures

Aules: Aules

PeriodeLectiu: PeriodeLectius

Restriccions: joc8restriccions

Primer de tot tenim el fitxer de restriccions es molt restricitiu, per tant a la hora de generar el primer horari ens surt que no es pot generar, per tant el que fem es eliminar una de les assignatures de la restriccio de nivells (EC), tornem a generar l'horari pero se segueix sense poder generar, per tant el que fem es eliminar la restriccio de LI de interval horari (id 3), tornem a generar l'horari pero ens torna a dir que no es pot genera, així que el que fem és treure PRO2 de la restriccio de subnivell, i ara ja ens deixa generar l'horari.

Sortida: En aquest joc de proves el que volem veure és com el nostre generador d'horaris es va adaptant als canvis i eliminacions de restriccions, i com hem pogut observar funciona perfectament.

Tots aquest jocs de proves estan implementats en la classe JocsDeProvaFinal, per la qual cosa es poden provar per consola sense cap problema, està dissenyat de la mateixa manera que els drivers.

# Documentació dels Drivers

Tot driver té primer de tot les opcions d'entrada manual per tal de poder fer totes les proves que es vulguin, i sempre en l'última opció de cada driver hi ha el driver automatic que fa una serie de crides per comprobar el correcte funcionament de la classe en qüestió.

# DriverAdmin

Volem veure que la classe creadora i els getters i setters que venen heredats de la seva classe pare funcionen correctament (Opció DriverAutomatic).

## Prova 1

Primer comprovem que la creadora i els getters funcionen correctament.
**Entrada**: Admin a = new Admin("Joan", "1", "1234");
**Sortida**: ID = 1, nom = Joan, TipusUser = ADMIN, Password = 1234.
Resultat de la prova 1 OK.

## Prova 2

Usem els setters per canviar els valors de l'admin creat a la prova 1.
**Entrada**:
a.setId("2");
a.setNom("Maria");
a.setPassword("5678");
**Sortida**: ID = 2, nom = Maria, TipusUser = ADMIN, Password = 5678.
Resultat de la prova 2 OK.

# DriverAlumne

Volem veure que la classe creadora i els getters i setters que venen heredats de la seva classe pare funcionen correctament (Opció DriverAutomatic).

## Prova 1

Primer comprovem que la creadora i els getters funcionen correctament.
**Entrada**: Admin a = new Alumne("Joan", "1", "1234");
**Sortida**: ID = 1, nom = Joan, TipusUser = ALUMNE, Password = 1234.
Resultat de la prova 1 OK.

## Prova 2

Usem els setters per canviar els valors del alumne creat a la prova 1.
**Entrada**:
a.setId("2");
a.setNom("Maria");
a.setPassword("5678");
**Sortida**: ID = 2, nom = Maria, TipusUser = ALUMNE, Password = 5678.
Resultat de la prova 2 OK.

# DriverAssignatura

Volem veure que la classe creadora i els getters i setters de la seva classe funcionen correctament (Opció DriverAutomatic).

## Prova 1

Primer comprovem que la creadora i els getters funcionen correctament, i que un assignatura pot no tenir Correquisists i/o InversCorrequisists assignats.

**Entrada**: Assignatura a = new Assignatura("LI", "Logica en l'informatica", 6, 60, 1, 3, 4, 2, 4, new ArrayList<>(), new ArrayList<>(), "inicial" );

**Sortida**:

ID Assig: LI
Nom Assig: Logica en l'informatica
NumCredits: 6
NumPlaces: 60
NumGrups: 3
NumSubgrups: 4
NumSessions: 2
HoresSetmanals: 4
Quadrimestre: 1
Fase: inicial
Correquisits:
InverseCorrequisits:
Resultat de la prova 1 OK.

## Prova 2

Usem els setters per canviar els valors del Assignatura creat a la prova 1.

**Entrada**:

a.setId("LP");
a.setNom("Llengutages de programacio");
a.setNumCredits(4);
a.setNumPlaces(70);
a.setQuadrimestre(2);
a.setNumGrups(4);
a.setNumSubgrupsPerGrup(3);
a.setNumSessionsSetmanals(3);
a.setHoresSetmanals(6);
a.setCorrequisits([“PROP”]);
a.setInverseCorrequisits([“A”]);
a.setFase("Computacio");

**Sortida**:

ID Assig: LP
Nom Assig: Llengutages de programacio

NumCredits: 4
NumPlaces: 70
NumGrups: 4
NumSubgrups: 3
NumSessions: 3
HoresSetmanals: 6
Quadrimestre: 2
Fase: Computacio
Correquisits: PROP
InverseCorrequisits: A
Resultat de la prova 2 OK.

# DriverAula

Volem veure que la classe creadora, els getters i setters de la seva classe funcionen correctament (Opció DriverAutomatic).

## Prova 1

Primer comprovem que la creadora i els getters funcionen correctament.
**Entrada**: Aula a = new Aula("A5201", TipusAula.TEORIA, 60);
**Sortida:**
Aula: A5201
TipusAula: TEORIA
Aformanet: 60
Resultat de la prova 1 OK.

## Prova 2

Usem els setters per canviar els valors del Aula creat a la prova 1.
**Entrada**:
a.setNom("A6101");
a.setTipusAula(TipusAula.PROBLEMES);
a.setAforament(50);
**Sortida**:
Aula: A6101
TipusAula: PROBLEMES
Aformanet: 50
Resultat de la prova 2 OK.

## Prova 3

Usem els setters per canviar el valor de l'ultima enummeration que falta.
**Entrada**:
a.setTipusAula(TipusAula.LABORATORI);
**Sortida**:
Aula: A6101

TipusAula: LABORATORI
Aformanet: 50
Resultat de la prova 3 OK.

# DriverCtrlDominio

Volem veure que les funcions són correctes

### Prova 1

Carregar fitxer d'assignatures correctament.
**Entrada**: a.cargarAsignaturas("Assignaturas");
**Sortida**: Faig un print de les assignatures i surten correctamnet
Resultat de la prova 1 OK.

### Prova 2

Carregar fitxer d'aules correctament.
**Entrada**: a.cargarAules("Aules");
**Sortida**: Faig un print de les Aules i surten correctamnet
Resultat de la prova 2 OK.

### Prova 3

Carregar fitxer de PeriodesLectius correctament.
**Entrada**: a.cargarPeriodeLectiu("PeriodesLectius");
**Sortida**: Faig un print de les PeriodesLectius i surten correctamnet
Resultat de la prova 3 OK.

### Prova 4

Carregar fitxer de Horari correctament.
**Entrada**: aux = a.cargarHorari("/1/test1.json");
**Sortida**: Faig un print de de GrupAssignatures correctament
Resultat de la prova 4 OK.

### Prova 5

Imprim fitxer de Horari amb un format llegible.
**Entrada**: a.printHorari(aux);
**Sortida**: L'horari surt amb el format adequat.
Resultat de la prova 4 OK.

# DriverGrupAssignatura

Volem veure que la classe creadora, els getters i setters de la seva classe funcionen correctament (Opció DriverAutomatic).

### Prova 1

Primer comprovem que la creadora sense Aula i Horari  i els getters funcionen correctament.

**Entrada**: a = new GrupAssignatura(10, 20, TipusAula.TEORIA, "PROP", null, null);

**Sortida:**

Num Grup: 10
Num Places: 20
Assignatura: PROP
TipusAula: TEORIA
Resultat de la prova 1 OK.

## Prova 2

Usem els setters per canviar els valors del GrupAssignatura creat a la prova 1 i afegir-li horari i aula .

**Entrada**:

```
a.setNumGrup(11);
a.setNumPlaces(30);
a.setTipusAula(TipusAula.LABORATORI);
a.setIdAssignatura("IDI");
aula = new Aula("A5202", TipusAula.LABORATORI, 30);
a.setAula(aula);
horari = new Horari(8,2,DiaSetmana.DILLUNS);
a.setHorari(horari);
```

**Sortida**:

Num Grup: 11
Num Places: 30
Assignatura: IDI
TipusAula: LABORATORI
---
Aula: A5202
TipusAula: LABORATORI
Aformanet: 30
---
Hora inici: 8
Durada: 2
Dia: DILLUNS
Resultat de la prova 2 OK.

## Prova 3

Comparem dos GrupAssignatura que són diferents:

**Entrada**:

```
GrupAssignatura a = new GrupAssignatura(10, 20, TipusAula.TEORIA, "PROP", aula, horari);
GrupAssignatura b = new GrupAssignatura(10, 20, TipusAula.TEORIA, "PROP", aula, horari1);
a.equals(b);
```

**Sortida**:
False
Resultat de la prova 3 OK.

### Prova 4

Comparem dos GrupAssignatura que son iguals
**Entrada**:
GrupAssignatura a = new GrupAssignatura(10, 20, TipusAula.TEORIA, "PROP", aula, horari);
a.equals(a);
**Sortida**:
True
Resultat de la prova 4 OK.

## DriverPeriodeLectiu

Volem veure que la classe creadora, els getters i setters de la seva classe funcionen correctament (Opció DriverAutomatic).

### Prova 1

Primer comprovem que la creadora i els getters funcionen correctament.
**Entrada**: a = new PeriodeLectiu("Periode1", "P1", 8 , 20, DiaSetmana.DILLUNS, DiaSetmana.DIVENDRES);
**Sortida:**
Nom: Periode1
ID: P1
HoraIni: 8
HoraFi: 20
DiaIni: DILLUNS
DiaFinal: DIVENDRES
Resultat de la prova 1 OK.

### Prova 2

Usem els setters per canviar els valors del PeriodeLectiu creat a la prova 1.
**Entrada**:
a.setNom("Periode2");
a.setId("P2");
a.setHoraInici(9);
a.setHoraFinal(19);
a.setDataInici(DiaSetmana.DIMARTS);
a.setDataFinal(DiaSetmana.DISSABTE)
**Sortida**:
Nom: Periode2
ID: P2
HoraIni: 9

HoraFi: 19
DiaIni: DIMARTS
DiaFinal: DISSABTE
Resultat de la prova 2 OK.

# DriverUser

Volem veure que la classe creadora i els getters i setters funcionen correctament (Opció DriverAutomatic).

### Prova 1

Primer comprovem que la creadora i els getters funcionen correctament.
**Entrada**: a = new User("Joan", "1", TipusUser.ADMIN, "1234");
**Sortida**: ID = 1, nom = Joan, TipusUser = ADMIN, Password = 1234.
Resultat de la prova 1 OK.

### Prova 2

Usem els setters per canviar els valors del user(Admin) i fer-lo user(Alumne) creat a la prova1
**Entrada**:
a.setId("2");
a.setNom("Maria");
a.setTipusUser(TipusUser.ALUMNE);
a.setPassword("5678");
**Sortida**: ID = 2, nom = Maria, TipusUser = ALUMNE, Password = 5678.
Resultat de la prova 2 OK.

# DriverRestriccions

Volem veure com les restriccions funcionen per cada cas

### Prova 1.1

Comprobem horari de classe dins del periode lectiu.
**Entrada**:
PeriodeLectiu periodeLectiu = newPeriodeLectiu("Periode1","P1",8,20, DiaSetmana.DILLUNS ,DiaSetmana.DIVENDRES);
Horari HorariDins = new Horari(8, 2, DiaSetmana.DILLUNS);
a.horariDinsPeriodeLectiu(HorariDins, periodeLectiu)
**Sortida**: True
Resultat de la prova 1 OK.

### Prova 1.2

Comprobem horari de classe fora del periode lectiu.

**Entrada**:
Horari HorariFora1 = new Horari(7, 2, DiaSetmana.DILLUNS);
a.horariDinsPeriodeLectiu(HorariFora1, periodeLectiu)
**Sortida**: False
Resultat de la prova 2 OK.

### Prova 1.3

Comprobem horari de classe comeca dins pero fora del periode lectiu.
**Entrada**:
Horari HorariFora2 = new Horari(20, 2, DiaSetmana.DILLUNS);
a.horariDinsPeriodeLectiu(HorariFora2, periodeLectiu)
**Sortida**: False
Resultat de la prova 3 OK.

Volem veure que un grupAssignatura esta assignat a una aula amb aforament suficient

### Prova 2.1

Comprobem aula amb suficient aformanet.
**Entrada**:
GrupAssignatura g = new GrupAssignatura(10, 20, TipusAula.TEORIA, "PROP", null, null);
Aula aula1 = new Aula("A5202", TipusAula.TEORIA, 30);
a.grupFitsAula(grupAssignatura, aula1)
**Sortida**: True
Resultat de la prova 1 OK.

### Prova 2.2

Comprobem aula amb insuficient aformanet.
**Entrada**:
Aula aula2 = new Aula("A5202", TipusAula.TEORIA, 10);
a.grupFitsAula(grupAssignatura, aula1)
**Sortida**: False
Resultat de la prova 2 OK.

Volem veure que un grupAssignatura esta assignat a una aula de tipus correcte

### Prova 3.1

Comprobem aula tipus correcte
**Entrada**:
Aula aula3 = new Aula("A5202", TipusAula.LABORATORI, 30);
GrupAssignatura g1 = new GrupAssignatura(10, 20, TipusAula.TEORIA, "PROP", aula1, null);
a.aulaCorrecta(grupAssignatura1)
**Sortida**: True
Resultat de la prova 1 OK.

### Prova 3.2

Comprobem aula amb tipus incorrecte.

**Entrada**:
GrupAssignatura g2 = new GrupAssignatura(10, 20, TipusAula.TEORIA, "PROP", aula3, null);
a.aulaCorrecta(grupAssignatura2)
**Sortida**: False
Resultat de la prova 2 OK.

Volem comprobar que un grupAssignatura i un dels seus subGrups no tenen lab i teoria el mateix dia.

## Prova 4.1

**Entrada**: Entra un grupAssignatura i un subGrup de la mateixa asignatura amb laboratori i teoria al mateix dia
**Sortida**: False
Resultat de la prova 1 OK.

## Prova 4.2

**Entrada**: Entra un grupAssignatura i un subGrup de la mateixa asignatura amb laboratori i teoria en diferents dies
**Sortida**: True
Resultat de la prova 2 OK.

Volem comprobar que dos grups no poden fer classe el mateix dia, a la mateixa hora i a la mateixa aula.

## Prova 5.1

**Entrada**: Entra una llista de grupAssignatura i un grupAssignatura g en els quals un dels grups es solapa amb el grupAssignara g
**Sortida**: False
Resultat de la prova 1 OK.

## Prova 5.2

**Entrada**: Entra una llista de grupAssignatura i un grupAssignatura g els quals no es solapen en hora i dia i aula.
**Sortida**: True
Resultat de la prova 2 OK.