EXAMEN-mayo-2023.pdf

holiperri

Programación Orientada a Objetos

1º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

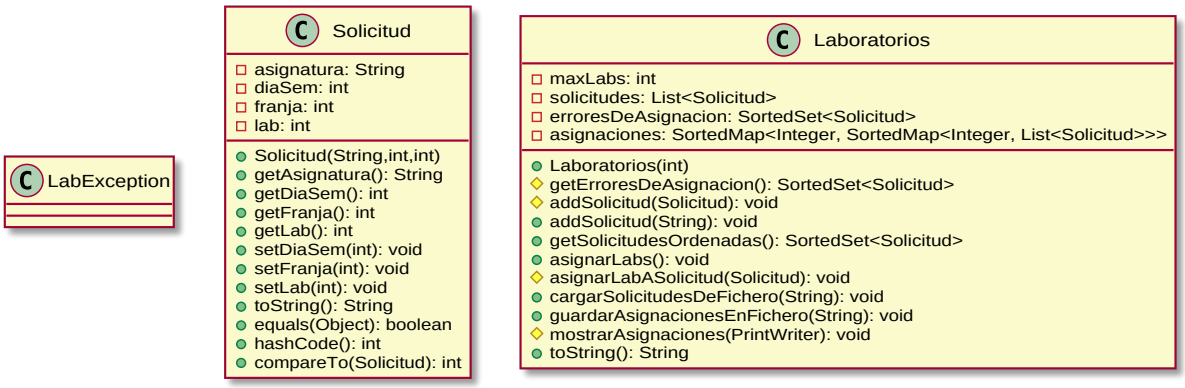
Examen de Programación Orientada a Objetos. Mayo de 2023

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- El alumnado debe haber comprobado si cumple los requisitos para acogerse a la modalidad de evaluación continua. De no ser así, además de la prueba práctica, deberá realizar un examen teórico adicional y, por lo tanto, ponerse en contacto con el profesorado.
- El alumnado deberá crear y trabajar en un **NUEVO espacio de trabajo**, denominado **pooex2305**, en la carpeta **Documentos**, en el entorno de desarrollo.
- Al inicio del contenido de cada fichero realizado deberá aparecer un comentario con los apellidos y nombre, titulación y grupo de la persona.
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no se sabe hacer, no debes pararte en él indefinidamente. Puedes abordar otros.
- Está permitido:
  - Consultar las presentaciones de clase, y la guía rápida de la API.
  - Añadir métodos privados a las clases.
- No está permitido:
  - Intercambiar documentación con otras personas.
  - Recibir ayuda de otras personas. Se debe realizar personal e individualmente la solución del ejercicio propuesto.
  - Añadir métodos no privados a las clases.
  - Añadir variables o constantes a las clases.
  - Modificar el nivel de acceso y visibilidad de las variables, constantes y métodos que aparecen en el diagrama UML.
  - Modificar el código suministrado.
- El código fuente en java entregado por el alumnado debe **compilar correctamente**, y será evaluado comprobando el funcionamiento de la **ejecución** de los métodos de las clases. **El código que no compile se debe poner entre comentarios.**
- Es muy **importante que se respeten los nombres** que aparecen en el enunciado, tanto los nombres de paquetes, de clases, de atributos y de métodos. Si se cambian los nombres, entonces el código no podrá ser compilado ni probado adecuadamente.
- Cuando termine el ejercicio, la persona debe subir a la tarea del campus virtual un archivo comprimido en formato **ZIP** de la carpeta **src** del proyecto (**src.zip**), conteniendo todo el código fuente en java desarrollado.

  Finalmente, la persona debe avisar al profesorado para cerrar la sesión del entorno seguro.
- La evaluación tendrá en cuenta la claridad de los algoritmos, del código y la correcta elección de las estructuras de datos, así como los criterios de diseño que favorezcan la reutilización.
- Se debe tener en cuenta que tanto el sistema como el entorno de desarrollo podrían **fallar**, y se podría **perder** todo el trabajo realizado, por lo que es importante **guardar frecuentemente** (**Ctrl+Shift+S**) el código fuente.

El diagrama de clases UML



Nota: todas las variables de instancia de todas las clases son **privadas**, y los métodos especificados en el diagrama UML son **públicos** o **protegidos**.

Se desea desarrollar una aplicación para la gestión de la asignación de laboratorios para realizar las prácticas de las asignaturas del departamento. Para ello, se deben desarrollar las clases especificadas a continuación.

El paquete ex2305

Se debe crear el paquete **ex2305** en un proyecto nuevo, denominado **prEx2305**. Es importante que el alumnado se asegure que el **nombre del paquete es correcto** según lo especificado en este enunciado.

La excepción `LabException` (0.25 pts.)

La clase `LabException` (del paquete `ex2305`) se define como una **excepción NO comprobada** utilizada para notificar las situaciones anómalas excepcionales que se produzcan.

La clase `Solicitud` (2.50 pts.)

La clase `Solicitud` (del paquete `ex2305`) contiene información sobre una determinada solicitud de laboratorio para realizar prácticas, tal como el *nombre de asignatura* (`asignatura: String`), el *día de la semana* (`diaSem: int`), la *franja horaria* (`franja: int`), y el *laboratorio asignado* (`lab: int`).

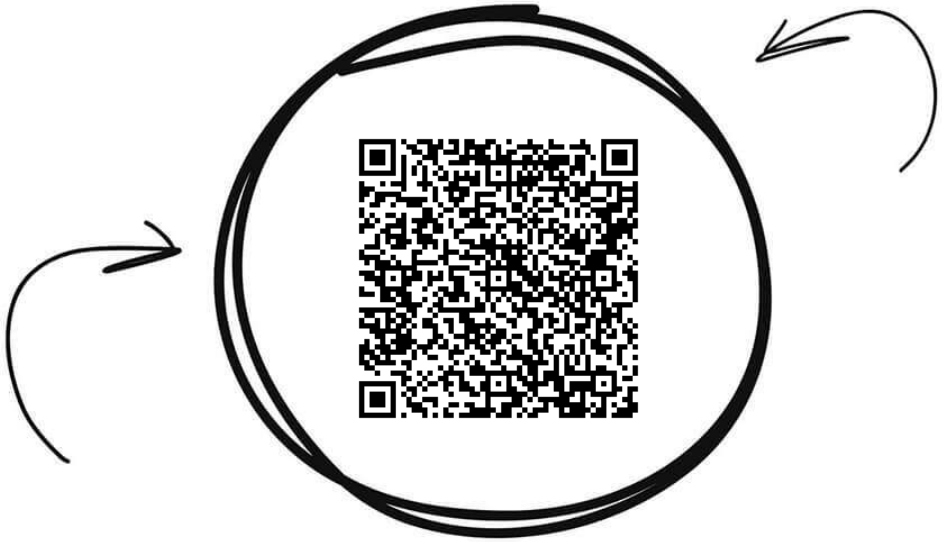
- El constructor recibe, en el siguiente orden, el *nombre de la asignatura*, el *día de la semana*, y la *franja horaria* de la solicitud. Si los parámetros son correctos, entonces los almacena, considerando que el *laboratorio asignado* tendrá un valor inicial de `-1` (que significa *sin asignar*). Sin embargo, si los parámetros son erróneos (el *día de la semana* correcto es un número entero entre 1 y 7, ambos inclusive, y la *franja horaria* correcta es un número entero entre 1 y 3, ambos inclusive), entonces lanza la excepción `LabException`, con el mensaje *"Argumentos erróneos"*.
- Los métodos, `getAsignatura()`, `getDiaSem()`, `getFranja()`, y `getLab()`, devuelven los valores almacenados en las variables de instancia correspondientes del objeto.
- El método `setDiaSem(int)`, si el *día de la semana* recibido como parámetro no es correcto (véase descripción del constructor), entonces lanza la excepción `LabException`, con el mensaje *"Argumentos erróneos"*.

  En otro caso, almacena en la variable de instancia correspondiente el *día de la semana* recibido como parámetro.
- El método `setFranja(int)`, si la *franja horaria* recibida como parámetro no es correcta (véase descripción del constructor), entonces lanza la excepción `LabException`, con el mensaje *"Argumentos erróneos"*.

  En otro caso, almacena en la variable de instancia correspondiente la *franja horaria* recibida como parámetro.
- El método `setLab(int)` almacena en la variable de instancia correspondiente el *laboratorio asignado* recibido como parámetro.


McDonald's
“SOY CREW DE McDONALD’S,
Y POR SUPUESTO QUE
MI TRABAJO Y MI PASIÓN
SON COMPATIBLES”
¿TE VIENES?
My CREW
Mi trabajo. Mi pasión. Mi gente.

Programación Orientada a Obj...



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

- El método `toString()` proporciona la representación textual del objeto, según el formato del ejemplo, que contiene todos los valores de las variables de instancia del objeto, en el mismo orden que la descripción anterior, todo entre *paréntesis*, utilizando la *coma* como separador entre los componentes:

  ```
  (POO-A, 2, 1, -1)
  ```

- Se considera que dos objetos de la clase `Solicitud` son **iguales** si el *nombre de la asignatura*, el *día de la semana* y la *franja horaria*, son iguales en ambos objetos, sin diferenciar mayúsculas de minúsculas en las comparaciones de *asignatura*.
- Los objetos de la clase `Solicitud` proporcionan el **orden natural** comparando el *día de la semana*, y en caso de igualdad, entonces se compara la *franja horaria*, y en caso de igualdad, entonces se compara el nombre de la *asignatura*, sin diferenciar mayúsculas de minúsculas en las comparaciones de *asignatura*. Todas las comparaciones serán de forma ascendente.

La aplicación `PruebaSolicitud` (0.50 pts.)

Desarrolle la clase distinguida `PruebaSolicitud` (en el paquete anónimo/por defecto) para probar la clase anterior. Para ello, la clase distinguida debe crear tres objetos de la clase `Solicitud`, con la siguiente información:

- obj1: *Asignatura*: `POO-B`, *DíaSem: 3*, *Franja: 2*
- obj2: *Asignatura*: `poo-b`, *DíaSem: 3*, *Franja: 2*
- obj3: *Asignatura*: `Web-A`, *DíaSem: 2*, *Franja: 3*

A continuación se mostrará la representación textual de los tres objetos.

A continuación se compara por **igualdad** el objeto 1 con el objeto 2, después se compara por igualdad el objeto 1 con el objeto 3, y finalmente se compara por igualdad el objeto 2 con el objeto 3, mostrando el mensaje `"Iguales"` o `"Distintos"` como resultado de cada comparación anterior.

Finalmente, se compara por **orden natural** el objeto 1 con el objeto 2, después se compara por orden natural el objeto 1 con el objeto 3, y finalmente se compara por orden natural el objeto 2 con el objeto 3, mostrando el valor resultado de cada comparación anterior.

```
(POO-B, 3, 2, -1)
(poo-b, 3, 2, -1)
(Web-A, 2, 3, -1)
Iguales
Distintos
Distintos
0
1
1
```

La clase `Laboratorios` (6.75 pts.)

La clase `Laboratorios` (del paquete `ex2305`) permite gestionar las solicitudes de laboratorios para prácticas, así como realizar la asignación de laboratorios a dichas solicitudes. Así, contiene el número máximo de laboratorios de los que se dispone (`maxLabs: int`). Además, contiene una lista con las solicitudes de laboratorios para prácticas (`solicitudes: List<Solicitud>`), un conjunto ordenado con las solicitudes que no han podido ser asignadas a ningún laboratorio (`erroresDeAsignacion: SortedSet<Solicitud>`), y las asignaciones de las solicitudes a los laboratorios (`asignaciones: SortedMap<Integer, SortedMap<Integer, List<Solicitud>>>`).

En la correspondencia de *asignaciones*, la primera clave corresponde con el *día de la semana*, la segunda clave corresponde con la *franja horaria*, y el valor asociado es una lista con las solicitudes que han sido asignadas a los laboratorios para ese día y franja, considerando que esta lista nunca debe almacenar más solicitudes del número máximo de laboratorios disponibles (`maxLabs`).

La figura anterior muestra un esquema de la correspondencia ***asignaciones****. Esta figura corresponde al ejemplo mostrado en la descripción de los métodos* `toString()` *y* `mostrarAsignaciones(PrintWriter)`*.*

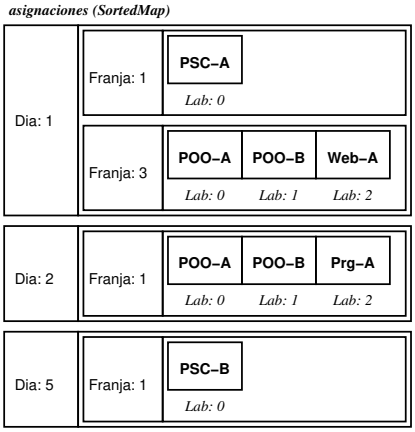


Figura 1: Correspondencia de asignaciones

- `Laboratorios(int)`

  Recibe como parámetro el *número máximo de laboratorios* de los que se dispone. Si el parámetro es correcto, entonces lo almacena, y construye las estructuras de datos, especificadas anteriormente, vacías. Sin embargo, si el parámetro es erróneo (*número máximo de laboratorios* menor que 1), entonces lanza la excepción `LabException`, con el mensaje *"Argumentos erróneos"*.

- `getErroresDeAsignacion(): SortedSet<Solicitud> // PROTEGIDO`

  Devuelve el conjunto ordenado de erroresDeAsignacion de la variable de instancia correspondiente.

- `addSolicitud(Solicitud): void // PROTEGIDO`

  Si la *lista de solicitudes* **no** contiene ya almacenada otra solicitud igual a la recibida como parámetro, entonces almacena la solicitud recibida como parámetro al final de la *lista de solicitudes* del objeto.

  Si embargo, si la *lista de solicitudes* **sí** contiene ya almacenada otra solicitud igual a la recibida como parámetro, entonces no se hace nada.

- `addSolicitud(String): void`

  Recibe como parámetro un `String` conteniendo los datos de una determinada solicitud, según el siguiente formato (delimitadores: `"\\s*[;]\\s*"`), considerando que el *laboratorio asignado* será el valor por defecto (-1 que significa *sin asignar*):

  ```
  asignatura; diaSem; franja
  ```

  Si los datos de la solicitud son correctos entonces se añadirá un nuevo objeto, instancia de `Solicitud`, a la *lista de solicitudes*, utilizando para ello el método protegido anterior.

  En caso de que se produzca algún error en este procesamiento, entonces lanza la excepción `LabException`, con el mensaje *"Argumentos erróneos"*.

- `getSolicitudesOrdenadas(): SortedSet<Solicitud>`

  Devuelve un conjunto ordenado con las *solicitudes* almacenadas. Este conjunto estará ordenado según un **orden alternativo** por el *nombre de la asignatura*, de forma ascendente, y sin diferenciar mayúsculas de minúsculas.

  Si lo considera necesario, **puede crear una nueva clase satélite** adicional que proporcione un **orden alternativo** que permita comparar objetos de la clase `Solicitud` de la forma especificada (nótese que esta nueva clase adicional no se muestra en el diagrama de clases).

- `asignarLabs(): void`

  Reinicia a vacío tanto el conjunto ordenado de *erroresDeAsignacion*, como la correspondencia de *asignaciones*. Posteriormente, para cada solicitud de la lista de *solicitudes*, invoca al siguiente método protegido

`asignarLabASolicitud`, para intenta asignar la solicitud a un laboratorio *disponible* (sin asignar).

- `asignarLabASolicitud(Solicitud): void` *// PROTEGIDO*

  Nota: *véase la figura que muestra un esquema de la correspondencia* ***asignaciones*** *al comienzo de la descripción de la clase Laboratorios.*

  Dada la solicitud recibida como parámetro, reinicia su *laboratorio asignado* al valor por defecto (-1 que significa *sin asignar*).

  Posteriormente, si hay algún laboratorio *disponible* (sin asignar), en la correspondencia de *asignaciones*, para el *día de la semana*, y la *franja horaria* de la solicitud (existe algún laboratorio *disponible* si la lista, correspondiente al día de la semana y franja horaria, tiene almacenadas una cantidad de solicitudes menor que el valor de `maxLabs`), entonces añade al final de la lista de solicitudes la solicitud recibida como parámetro, y le asigna a dicha solicitud el número de laboratorio correspondiente a la posición (índice) donde dicha solicitud ha sido almacenada en la lista.

  En otro caso, es decir, si no hay laboratorios disponibles para ese *día de la semana* y *franja horaria* (porque la lista correspondiente ya contiene almacenadas el número máximo de solicitudes `maxLabs`), entonces la solicitud recibida como parámetro se añade al conjunto ordenado de *erroresDeAsignacion*.

- `toString(): String` *// Redefinición*

  Devuelve la representación textual del objeto, según el formato del siguiente ejemplo, que contiene la *lista de solicitudes*, el conjunto ordenado de *erroresDeAsignacion*, y la correspondencia de *asignaciones*, entre *paréntesis*, separados por el símbolo *coma* (los saltos de línea han sido incluidos por legibilidad, y no es necesario que el alumno los incluya):

```
(Solicitudes: [(POO-A, 1, 3, 0), (POO-B, 1, 3, 1),
   (POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2),
   (Prg-B, 2, 1, -1), (PSC-A, 1, 1, 0), (PSC-B, 5, 1, 0),
   (IAx-A, 2, 1, -1), (IAx-B, 2, 1, -1), (Web-A, 1, 3, 2),
   (Web-B, 1, 3, -1), (Sec-A, 2, 1, -1), (Sec-B, 2, 1, -1)],
 ErroresDeAsignacion: [(Web-B, 1, 3, -1), (IAx-A, 2, 1, -1),
   (IAx-B, 2, 1, -1), (Prg-B, 2, 1, -1), (Sec-A, 2, 1, -1),
   (Sec-B, 2, 1, -1)],
 Asignaciones: {1={1=[(PSC-A, 1, 1, 0)],
                   3=[(POO-A, 1, 3, 0), (POO-B, 1, 3, 1), (Web-A, 1, 3, 2)]},
                2={1=[(POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2)]},
                5={1=[(PSC-B, 5, 1, 0)]}})
```

- `cargarSolicitudesDeFichero(String): void` *// lanza IOException*

  Lee del fichero, cuyo nombre se recibe como parámetro, los datos de las solicitudes, y las añade a la lista de solicitudes (utilizando para ello el método `addSolicitud(String)` definido anteriormente). En caso de error de lectura del fichero, propaga la excepción `IOException`.

  En el fichero, cada línea contiene los datos de una solicitud (*asignatura*; *diaSem*; *franja*), donde los componentes de cada solicitud se encuentran separados por los siguientes delimitadores (`"\\s*[;]\\s*"`), según el formato del siguiente ejemplo:

```
POO; 1; 3
POO; 1; 3
Prg; 2; 1
PSC; 1; 1
```

  Si existe algún error en el formato de los datos de entrada, se **desecharán** los datos de la línea involucrada, y se **continuará** con los datos de las siguientes líneas.

- `guardarAsignacionesEnFichero(String): void` *// lanza IOException*

  Guarda en el fichero, cuyo nombre se recibe como parámetro, todos los datos de las asignaciones realizadas, y de los erroresDeAsignacion de asignación, según el formato especificado en el método protegido `mostrarAsignaciones`. En caso de error al escribir en el fichero, propaga la excepción `IOException`.

- `mostrarAsignaciones(PrintWriter): void` *// PROTEGIDO*

  Muestra en el objeto `PrintWriter` que se recibe como parámetro, todos los datos de las asignaciones realizadas, y de los erroresDeAsignacion de asignación, según el formato mostrado en el siguiente ejemplo:

```
DiaSem: 1; Franja: 1
Lab: 0: (PSC-A, 1, 1, 0)

DiaSem: 1; Franja: 3
Lab: 0: (POO-A, 1, 3, 0)
Lab: 1: (POO-B, 1, 3, 1)
Lab: 2: (Web-A, 1, 3, 2)

DiaSem: 2; Franja: 1
Lab: 0: (POO-A, 2, 1, 0)
Lab: 1: (POO-B, 2, 1, 1)
Lab: 2: (Prg-A, 2, 1, 2)

DiaSem: 5; Franja: 1
Lab: 0: (PSC-B, 5, 1, 0)

ErroresDeAsignacion:
(Web-B, 1, 3, -1)
(IAx-A, 2, 1, -1)
(IAx-B, 2, 1, -1)
(Prg-B, 2, 1, -1)
(Sec-A, 2, 1, -1)
(Sec-B, 2, 1, -1)
```

La aplicación `PruebaLabs`

La aplicación `PruebaLabs` se proporciona ya desarrollada, y debe descargarse del campus virtual. Esta clase distinguida (en el paquete anónimo/por defecto) permite realizar una prueba de las clases anteriores.

Los ficheros de datos `solicitudes1.txt` y `solicitudes2.txt` se deben copiar a la carpeta raíz del proyecto.

La ejecución de este programa de prueba deberá mostrar exáctamente la siguiente salida (los saltos de línea han sido incluidos por legibilidad, y no es necesario que el alumno los incluya):

```
Prueba1:

Laboratorios:
(Solicitudes: [(POO-A, 1, 3, 0), (POO-B, 1, 3, 1),
   (POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2),
   (Prg-B, 2, 1, -1), (PSC-A, 1, 1, 0), (PSC-B, 5, 1, 0),
   (IAx-A, 2, 1, -1), (IAx-B, 2, 1, -1), (Web-A, 1, 3, 2),
   (Web-B, 1, 3, -1), (Sec-A, 2, 1, -1), (Sec-B, 2, 1, -1)],
 ErroresDeAsignacion: [(Web-B, 1, 3, -1), (IAx-A, 2, 1, -1),
   (IAx-B, 2, 1, -1), (Prg-B, 2, 1, -1), (Sec-A, 2, 1, -1),
   (Sec-B, 2, 1, -1)],
 Asignaciones: {1={1=[(PSC-A, 1, 1, 0)],
                   3=[(POO-A, 1, 3, 0), (POO-B, 1, 3, 1), (Web-A, 1, 3, 2)]},
                2={1=[(POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2)]},
                5={1=[(PSC-B, 5, 1, 0)]}})

Solicitudes Ordenadas:
[(IAx-A, 2, 1, -1), (IAx-B, 2, 1, -1), (POO-A, 1, 3, 0),
 (POO-B, 1, 3, 1), (Prg-A, 2, 1, 2), (Prg-B, 2, 1, -1),
 (PSC-A, 1, 1, 0), (PSC-B, 5, 1, 0), (Sec-A, 2, 1, -1),
 (Sec-B, 2, 1, -1), (Web-A, 1, 3, 2), (Web-B, 1, 3, -1)]

Prueba2:

Laboratorios:
(Solicitudes: [(POO-A, 1, 3, 0), (POO-B, 1, 3, 1),
   (POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2),
   (Prg-B, 2, 1, -1), (PSC-A, 1, 1, 0), (PSC-B, 5, 1, 0),
```

```
   (IAx-A, 2, 1, -1), (IAx-B, 2, 1, -1), (Web-A, 1, 3, 2),
   (Web-B, 1, 3, -1), (Sec-A, 2, 1, -1), (Sec-B, 2, 1, -1)],
 ErroresDeAsignacion: [(Web-B, 1, 3, -1), (IAx-A, 2, 1, -1),
   (IAx-B, 2, 1, -1), (Prg-B, 2, 1, -1), (Sec-A, 2, 1, -1),
   (Sec-B, 2, 1, -1)],
 Asignaciones: {1={1=[(PSC-A, 1, 1, 0)],
                   3=[(POO-A, 1, 3, 0), (POO-B, 1, 3, 1), (Web-A, 1, 3, 2)]},
                2={1=[(POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2)]},
                5={1=[(PSC-B, 5, 1, 0)]}})

Prueba3:

Laboratorios:
(Solicitudes: [(POO-A, 1, 3, 0), (POO-B, 1, 3, 1),
   (POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2),
   (Prg-B, 2, 1, -1), (PSC-A, 1, 1, 0), (PSC-B, 5, 1, 0),
   (IAx-A, 2, 1, -1), (IAx-B, 2, 1, -1), (Web-A, 1, 3, 2),
   (Web-B, 1, 3, -1), (Sec-A, 2, 1, -1), (Sec-B, 2, 1, -1)],
 ErroresDeAsignacion: [(Web-B, 1, 3, -1), (IAx-A, 2, 1, -1),
   (IAx-B, 2, 1, -1), (Prg-B, 2, 1, -1), (Sec-A, 2, 1, -1),
   (Sec-B, 2, 1, -1)],
 Asignaciones: {1={1=[(PSC-A, 1, 1, 0)],
                   3=[(POO-A, 1, 3, 0), (POO-B, 1, 3, 1), (Web-A, 1, 3, 2)]},
                2={1=[(POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2)]},
                5={1=[(PSC-B, 5, 1, 0)]}})

Prueba4:

Laboratorios:
(Solicitudes: [(POO-A, 1, 3, 0), (POO-B, 1, 3, 1),
   (POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2),
   (Prg-B, 2, 1, -1), (PSC-A, 1, 1, 0), (PSC-B, 5, 1, 0),
   (IAx-A, 2, 1, -1), (IAx-B, 2, 1, -1), (Web-A, 1, 3, 2),
   (Web-B, 1, 3, -1), (Sec-A, 2, 1, -1), (Sec-B, 2, 1, -1)],
 ErroresDeAsignacion: [(Web-B, 1, 3, -1), (IAx-A, 2, 1, -1),
   (IAx-B, 2, 1, -1), (Prg-B, 2, 1, -1), (Sec-A, 2, 1, -1),
   (Sec-B, 2, 1, -1)],
 Asignaciones: {1={1=[(PSC-A, 1, 1, 0)],
                   3=[(POO-A, 1, 3, 0), (POO-B, 1, 3, 1), (Web-A, 1, 3, 2)]},
                2={1=[(POO-A, 2, 1, 0), (POO-B, 2, 1, 1), (Prg-A, 2, 1, 2)]},
                5={1=[(PSC-B, 5, 1, 0)]}})
```

y el fichero `asignaciones.txt` debe contener la siguiente información:

```
DiaSem: 1; Franja: 1
Lab: 0: (PSC-A, 1, 1, 0)

DiaSem: 1; Franja: 3
Lab: 0: (POO-A, 1, 3, 0)
Lab: 1: (POO-B, 1, 3, 1)
Lab: 2: (Web-A, 1, 3, 2)

DiaSem: 2; Franja: 1
Lab: 0: (POO-A, 2, 1, 0)
Lab: 1: (POO-B, 2, 1, 1)
Lab: 2: (Prg-A, 2, 1, 2)

DiaSem: 5; Franja: 1
Lab: 0: (PSC-B, 5, 1, 0)

ErroresDeAsignacion:
(Web-B, 1, 3, -1)
```

```
(IAx-A, 2, 1, -1)
(IAx-B, 2, 1, -1)
(Prg-B, 2, 1, -1)
(Sec-A, 2, 1, -1)
(Sec-B, 2, 1, -1)
```