Examen Programación III

- Tienes en ALUD ("Exámenes y Evaluación") los ficheros de partida en un archivo comprimido. Crea un proyecto en Eclipse y descomprime las carpetas src, test de ese fichero en el proyecto.
- Para entregar el examen en ALUD, comprime las carpetas src, test, en un fichero.zip o .rar. Asegúrate de que tiene los ficheros que has modificado.
- Tiempo de examen: 4 horas

Estos días has estado revisando el código preparatorio del programa que sirve de base para el examen. A ese código inicial se le ha añadido una entidad adicional *Examen*, que permite representar los resultados que un estudiante ha obtenido en una prueba para cada una de las *Competencias* evaluadas en la misma. Se ha creado también una tabla *Examen* en la BD que será utilizada en la tarea correspondiente de base de datos y se incluyen dos ficheros CSV de datos utilizados por la aplicación.

Tareas

T1. Base de datos [2 - C3: 10%] [GestorBD.java, 32 líneas de código]
En la clase *GestorBD* verás un método *insertarExamenes* que recibe una lista de exámenes que debes insertar en la tabla *Examen* de la base de datos. Ten en cuenta lo siguiente:

- Cada *Examen* tiene la fecha en la que se realizó como un objeto de tipo *LocalDate* que debe ser guardado en la BD como un *String* en formato ISO: "2021-01-10".
- Cada *Examen* contiene también una lista de resultados para cada una de las competencias evaluadas en el mismo, hasta un máximo de 5 resultados. Sin embargo, estos datos no son almacenados directamente en la BD. Debes almacenar la media de los resultados en la columna *media*.
- Por otro lado, debes crear un resumen en formato texto que deberás almacenar en la columna resumen de la tabla *Examen* que indique para cada competencia su nombre y la calificación obtenida en el examen, cada una en una línea, como el siguiente ejemplo:

  Swing: 6.3
  Colecciones: 5.0
  Persistencia: 8.3
  JUnit: 7.0
  Recursividad: 10.0

- Debes obtener los nombres de cada una de las competencias de la Asignatura a la que corresponde el examen recuperando la información de la tabla Competencia de la base de datos.
- **No puedes usar ningún otro método de la clase para solucionar la tarea** (es decir, codificar todo lo necesario en este método).

T2. JUnit [2 – C4:10%] [LeerExamenesTest.java, 31 líneas de código]
Encontrarás una clase de JUnit 4 llamada LeerExamenesTest, con un método que tienes que programar para probar el método *leerFicheros()* de la clase *LeerExamenes*, que debe comprobar que la carga de datos, ya implementada, desde los ficheros CSV se realiza correctamente. Este método devuelve un mapa que relaciona el NIA (clave numérica) del estudiante con los exámenes que ha realizado.

Para pasar la comprobación de forma correcta esta prueba de unidad debe:

- Llamar al método *leerFicheros* indicando los dos ficheros a leer y el código de asignatura 145315. Tienes un ejemplo de cómo realizar esto en el constructor de la *VentanaPrincipal*.
- Debes comprobar que:
  - Se han leído datos para 13 estudiantes distintos.
  - Todos los NIA están comprendidos entre 10000 y 20000.
  - Todos los estudiantes han realizado 2 exámenes.
  - Todos los exámenes realizados son de la asignatura 145315.
  - Todas las calificaciones están comprendidas entre 0 y 10 (ambas incluidas).
  - Comprobar que todos los estudiantes se han examinado de dos competencias en el primer examen y de 5 en el segundo.
  - Comprueba que se lanza una excepción IOException cuando se invoca el método *leerFicheros* con algún nombre de fichero que no existe.

T3. Recursividad [2 – C5:5%] [Recursividad.java, 20 líneas]
Queremos generar todos los grupos distintos de N estudiantes que puedan generarse a partir de una lista de estudiantes y que cumplan que la media de calificaciones del grupo (media de las medias de cada estudiante) sea >= 5. Para ello, tienes un método ya definido en la clase *Recursividad* llamado generarGrupos que recibe una lista de NIA's de estudiantes y una lista de sus calificaciones medias (existe una correspondencia entre ambas listas: la media del primer estudiante es el primer valor de la lista de medias, y así sucesivamente). Los grupos no pueden tener estudiantes repetidos, ni tampoco puede haber grupos con los mismos estudiantes, pero ordenados de forma distinta. Además, no es necesario que todos los estudiantes de un grupo tengan una calificación >=5, simplemente es necesario que la media del grupo sea >=5. Tienes un método denominado calcularMedia que te ayudará a calcular la media de una lista de valores de tipo Float.

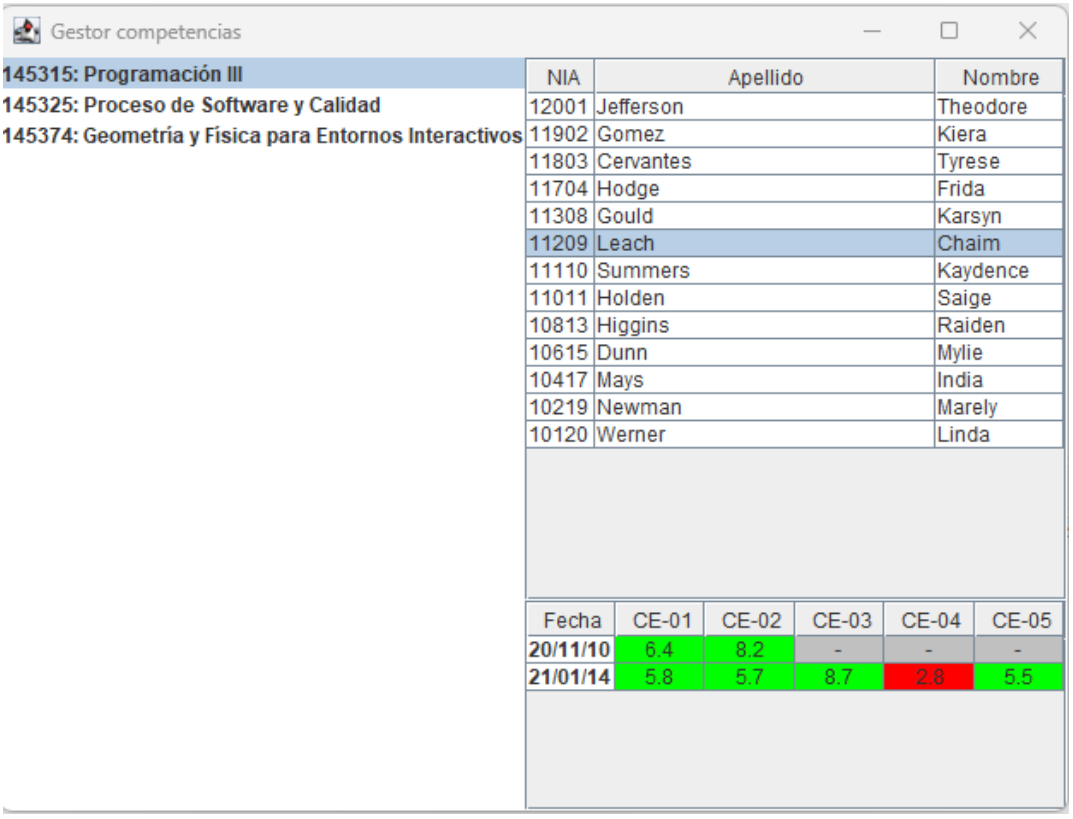
Debes implementar un método recursivo que proporcione la funcionalidad anterior para generar todos los grupos posibles de NIA's que cumplan la condición pedida y que llames a este método desde el método generarGrupos usando los parámetros extra que consideres necesarios. Los grupos obtenidos deben guardarse en la lista grupos (*List<List<Integer>>*).

Cuando hayas realizado la implementación, tienes un test unitario en la clase RecursividadTest que puedes ejecutar para comprobar si el método ha sido implementado correctamente. Fíjate que, para el juego de datos del test, con 9 estudiantes, el número total de grupos de 3 estudiantes que cumplen la condición es de 58 y el orden en que debieran generarse es el siguiente:

```
[12001, 12002, 12005]
[12001, 12002, 12009]
[12001, 12003, 12005]
…
[12006, 12007, 12009]
[12006, 12008, 12009]
[12007, 12008, 12009]
```

T4. Swing [2 - C1: 10%] [VentanaPrincipal.java (116 líneas)]
En el método `crearTablaResultados()` se crea una tabla vacía que servirá para visualizar los resultados de todos los exámenes realizados por un estudiante seleccionado en la tabla de estudiantes, que a su vez se carga a partir de la asignatura seleccionada. Se pide que:

- Definas un modelo de datos personalizado para esta tabla que a partir de la información contenida en *mapaExamenes*, que asocia el NIA de un estudiante con los exámenes que ha realizado:
  - Tiene 6 columnas nombradas: "*Fecha*", "*CE-01*", "*CE-02*","*CE-03*","*CE-04*","*CE-05*"
  - Debe tener tantas filas como exámenes realizados por el estudiante seleccionado en la tabla de estudiantes (*jTableEstudiantes*). Puedes obtener esta información consultando directamente el modelo de la tabla de estudiantes para obtener el de la primera columna de la fila seleccionada.
  - Las clases de las columnas deben ser de tipo LocalDate la primera, y de tipo Float las 5 restantes.
  - Si una competencia no ha sido evaluada en un examen se debe devolver NaN (*Float.NaN*).
  - Ten en cuenta que únicamente los alumnos de Programación III han realizado exámenes, el resto de asignaturas no tienen resultados.
- Definas y asignes un renderer para la columna de fecha que muestre la fecha con el formato: "yy/MM/dd" y en negrita.
- Definas y asignes un renderer para las columnas de competencias que:
  - Dibuje todos los valores centrados en las celdas.
  - Si los resultados son NaN para una competencia (no ha sido evaluada), saque un "-" y pinte el fondo de la celda de color gris claro.
  - Si una competencia ha sido superada (>= 5) pinta la celda de verde.
  - Si no ha sido superada (< 5) pinta la celda de rojo.



T5. Estructuras de Datos [2 - C2: 10%] [Estadisticas.java (25 líneas)]
Completa la clase *Estadisticas* para proporcionar funcionalidad a los dos métodos existentes:

- Para el método *obtenerNotasMedias* ten en cuenta que:
  - Debe devolver un mapa que asocie un estudiante (su NIA) con las notas medias obtenidas en cada uno de los exámenes realizados.
  - Debes generar la nueva estructura de datos utilizando el mapa recibido que relaciona un estudiante (su NIA) con la lista de exámenes realizados, teniendo en cuenta que cada examen contendrá el resultado de la evaluación de varias competencias. Por lo tanto, debes calcular la nota media del examen a partir de las notas obtenidas en cada competencia.
  - Las claves del mapa resultante deben estar ordenadas de mayor a menor número de NIA.
- Para el método *obtenerDiferenciasExamenes*, a partir del mapa de exámenes de cada estudiante (que contiene 2 exámenes) y la lista de estudiantes tienes que crear una lista que contenga las diferencias en la nota de las competencias evaluadas en los dos exámenes (como por ejemplo las C1 y C2 en nuestro examen parcial y final). Las diferencias entre dos exámenes se representan con la clase *DiferenciaExamenes* que contiene un Estudiante y una lista con la diferencia de los resultados para cada una de las competencias evaluadas en dos exámenes. Debes realizar lo siguiente:
  - Reorganizar todos los estudiantes de la lista en un mapa que asocie el NIA del estudiante con el objeto estudiante. Este mapa te ayudará a crear la lista final.
  - Obtener las diferencias entre el primer examen y el segundo examen para cada una de las dos competencias que han sido evaluadas en ambos (primera y segunda de los resultados de cada examen) y almacenar estos datos, junto con el objeto estudiante en una instancia de la clase *DiferenciaExamenes*.
  - Ordenar la lista de *DiferenciaExamenes* para que primero se encuentren aquellos objetos que contengan una mayor diferencia para la primera competencia. Si dos objetos *DiferenciaExamenes* tienen la misma diferencia para esta competencia, debes ordenar según el apellido del alumno en orden alfabético.