Deusto
Facultad de Ingeniería
Faculty of Engineering

Programación III & Programación de Aplicaciones

Curso 2024/2025

Nombre y Apellidos _____

3 de febrero de 2025

Examen parcial - 4,5 puntos (Duración: 2,5h)

Antes de comenzar a resolver el examen debes seguir los pasos indicados en la hoja de instrucciones

Continuamos con el dominio relacionado con la gestión del gimnasio. Para esta convocatoria extraordinaria se proponen ejercicios para recuperar las competencias: CE1 - Swing, CE2 - Threads, CE3 - JDBC y CE4 - Recursividad. Recuerda que debes realizar todas las competencias que no superaste en los exámenes anteriores. En el caso de volver a realizar una competencia ya superada, se utilizará la nota más alta obtenida en cualquiera de los exámenes realizados.

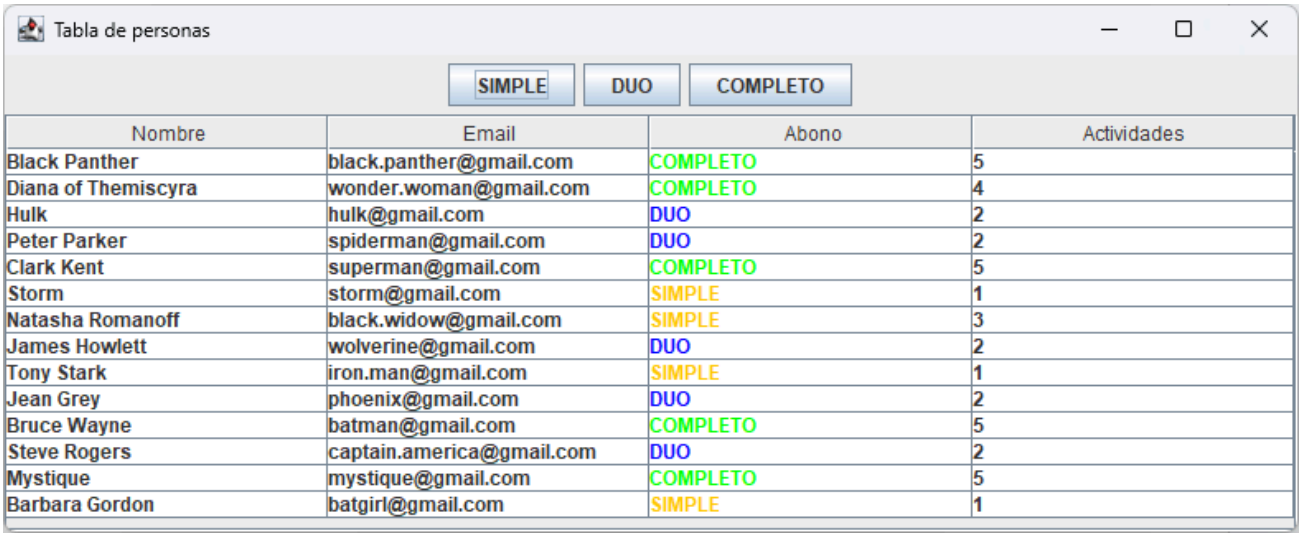
Para cada tarea, se indica su puntuación (sobre el total de 4,5 puntos del examen), la clase en la que debes añadir el código y una estimación del número de líneas de código necesarias para resolverlo. El código base proporcionado está organizado en varios paquetes, cada uno de ellos relacionado con una competencia, conteniendo cada uno de ellos una clase con un método `main()` para poder probar ese ejercicio.

Tarea 1: Swing [1,5 puntos] [TablaPersonasModelo, TablaPersonas] [50 líneas]

El objetivo de esta tarea es completar el código necesario para visualizar los datos de personas en la tabla actualmente en blanco y obtener el resultado de la imagen. Para ello, debes realizar los siguientes pasos:

1. Completa el modelo de datos parcial que se proporciona en la clase **`TablaPersonasModelo`** para visualizar la misma estructura de tabla que se muestra en la imagen .
2. Implementa un renderer para la tabla localizada en **`TablaPersonas`** de tal forma que, en la columna Abono, los abonos de tipo **`SIMPLE`**, **`DUO`** y **`COMPLETO`** aparezcan con las letras en colores naranja, azul y verde, respectivamente.
3. Añade 3 botones (uno por tipo de abono) en la parte superior de la ventana, para conseguir que, cuando el usuario pulse cualquiera de ellos, en la tabla se resalte con el fondo amarillo la celda de los abonos que se correspondan con el botón pulsado.

Puedes ver el resultado esperado en el vídeo contenido en la carpeta **`resources/doc`**.



Nombre	Email	Abono	Actividades
Black Panther	black.panther@gmail.com	COMPLETO	5
Diana of Themiscyra	wonder.woman@gmail.com	COMPLETO	4
Hulk	hulk@gmail.com	DUO	2
Peter Parker	spiderman@gmail.com	DUO	2
Clark Kent	superman@gmail.com	COMPLETO	5
Storm	storm@gmail.com	SIMPLE	1
Natasha Romanoff	black.widow@gmail.com	SIMPLE	3
James Howlett	wolverine@gmail.com	DUO	2
Tony Stark	iron.man@gmail.com	SIMPLE	1
Jean Grey	phoenix@gmail.com	DUO	2
Bruce Wayne	batman@gmail.com	COMPLETO	5
Steve Rogers	captain.america@gmail.com	DUO	2
Mystique	mystique@gmail.com	COMPLETO	5
Barbara Gordon	batgirl@gmail.com	SIMPLE	1

Tarea 2: Hilos [1 punto] [MainThreads] [30 líneas]

El objetivo de esta tarea es implementar una visualización que tenga el comportamiento que se muestra en el vídeo contenido en la carpeta **`resources/doc`**.

Se proporciona una ventana con una rejilla de paneles de Swing distribuidos en filas y columnas. Cuando el usuario pulsa el botón iniciar, cada uno de los paneles debe intercambiar su color de fondo periódicamente entre **`ROJO`** y **`GRIS CLARO (LIGHT_GRAY)`**. La visualización debe poder detenerse con el botón "Parar" y poder ser reiniciada de nuevo. El periodo (tiempo entre cambio de color) de cada panel de la rejilla se establece de manera aleatoria cada vez que el usuario pulsa el botón "Iniciar" y se mantiene fijo durante toda la visualización hasta que esta es detenida. Dispones de los métodos auxiliares **`periodoAleatorio()`** y **`cambiarEstado()`** para ayudarte a implementar la funcionalidad solicitada.

Tarea 3: JDBC [1 punto] [GestorBD] [20 líneas]

El objetivo de esta tarea es registrar en una nueva tabla ya creada en la BD las acciones de cambio de abono que se han realizado. La BBDD ya tiene tablas y datos precargados para las tablas de Actividades, Sesiones, Personas, y Persona_Actividad. Sin embargo, en este ejercicio nos vamos a centrar únicamente en la nueva tabla Auditoria y en la tabla Persona del modelo de datos. La interacción con la base de datos se realiza desde la ventana **`VentanaGestorAbonos`**, usando la BD que está en **`resources/db/gym.db`**.

`AUDITORIA`: • **`ID_PERSONA`**: Integer (FK) • **`ID`**: Integer autoincremental (PK) • **`FECHA`**: Text • **`ABONO_ANTERIOR:T`**ext • **`ABONO_NUEVO:`** Text	**`PERSONA`**: • **`ID`**: Integer autoincremental (PK) • **`NOMBRE`**: Text • **`EMAIL`**: Text • **`TIPO_ABONO`**: Text

Desde la **`VentanaGestorAbonos`**, al seleccionar una persona en la lista de la parte izquierda, se muestra en la parte derecha la información de la persona y las actividades en las que está apuntada. Si se modifica el abono, se actualizan las actividades y se habilita el botón "Actualizar abono y actividades". Cuando se pulsa ese botón, se guardan los cambios de Abono y Actividades en la BD de datos. Toda está funcionalidad ya se realizó en el examen ordinario.

Para completar la tarea de este examen, tienes que implementar el método **`insertRecord()`**. Este método debe insertar una nueva entrada en la tabla **`AUDITORIA`** que debe contener, a partir de la información del objeto **`Persona`** que recibe el método, qué persona ha realizado el cambio, la fecha en la que se ha realizado este cambio (el día actual) en formato "dd-MM-yyyy" y tanto el tipo de abono nuevo como el que tenía previamente. Recuerda, que el nuevo abono seleccionado en la UI es parte de la información del objeto **`Persona`** recibido en el método pero para obtener el abono anterior al cambio, deberás realizar una consulta a la tabla **`PERSONA`**.

Puedes validar el correcto funcionamiento de la implementación anterior interactuando con la ventana: (1) cambia el abono de una persona, (2) pulsa el botón de actualizar en la ventana, y (3) cierra la ventana. Al cerrar la ventana debes observar en la consola un volcado de todos los cambios de abono que has almacenado en la tabla, como en el siguiente ejemplo (se muestran 3 cambios de abono realizados):

```
Registros de auditoría:
Clark Kent: COMPLETO -> SIMPLE [2025-01-28]
Diana of Themiscyra: COMPLETO -> SIMPLE [2025-01-28]
Diana of Themiscyra: SIMPLE -> DUO [2025-01-28]
```

Tarea 4: Recursividad [1 punto] [MainRecursividad] [6 líneas]

El objetivo de esta tarea es implementar el método recursivo **`sumarDuracionRec()`** que recibe la lista de todos los ejercicios disponibles, un grupo muscular y un nivel concretos y, como resultado, devuelve la suma total del tiempo de todos los ejercicios que correspondan a ese grupo y nivel.

Para los datos de prueba, si implementas correctamente la función recursiva, deberás observar en consola los siguientes resultados:

```
Duración total Grupo: ESPALDA Nivel: MEDIO -> 30
Duración total Grupo: PIERNAS Nivel: AVANZADO -> 25
Duración total Grupo: CARDIO Nivel: INICIAL -> 3
```