Deusto
Facultad de Ingeniería
Faculty of Engineering

Programación III & Programación de Aplicaciones

Curso 2025/2026

18 de noviembre de 2024

Examen parcial - 10 puntos (Duración: 2,5h)

El caso que vamos a trabajar está relacionado con un entorno hospitalario. El examen se ha dividido en 3 tareas independientes. El código está organizado en 3 paquetes cada uno con una clase con un `main()` para que puedas ver el resultado: `PlanoHospital`, `PlanoMaternidad` y `Ascensores`. Dentro de la carpeta `resources/doc` hay imágenes y vídeos que te ayudarán a ver con mayor detalle el objetivo de las tareas.

Tarea 1.1: Layout [2 puntos] [PlanoHospital] [70 líneas]

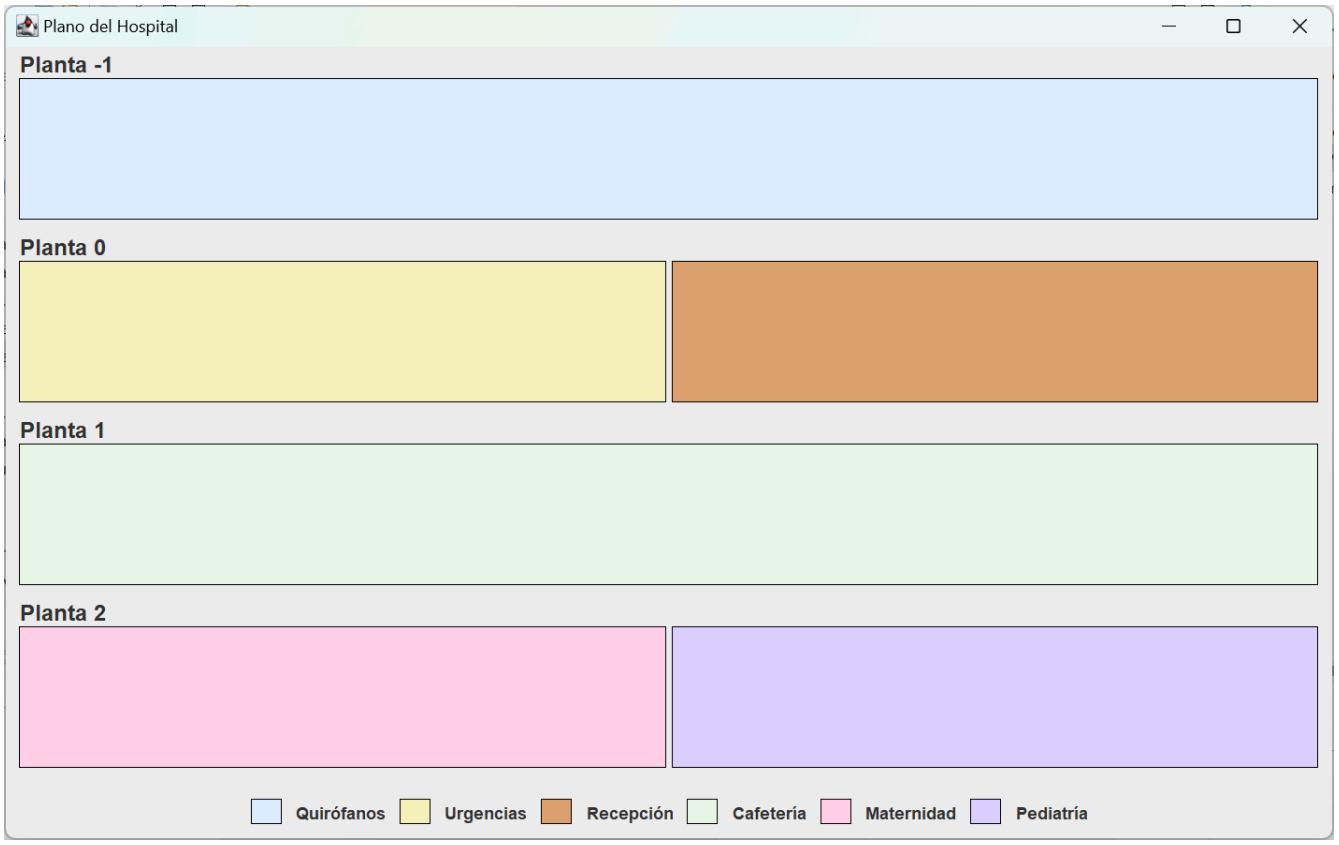
El objetivo de esta tarea es diseñar el plano de las zonas del hospital tal y como se muestra en la imagen de la página siguiente. Debes replicar el diseño en la clase `PlanoHospital`.

Para resolver esta tarea tienes que procesar el `SortedMap<Integer, List<Zona>>` que asocia la lista de zonas a cada una de las plantas del hospital. Usando el método `zonas.values()` obtienes una colección de `List<Zona>` que contiene las zonas de cada una de las plantas.

Para crear el diseño, sólo puedes utilizar los siguientes layouts: `FlowLayout`, `BorderLayout` y `GridLayout`. La clase `Zona` tiene un atributo de tipo `Color` para usar de relleno, cuando sea necesario.

Tarea 1.2: Evento de ratón [1 punto] [PlanoHospital] [15 líneas]

Implementa la gestión del evento de ratón para conseguir que al mover el ratón sobre cada una de las zonas se cambie el color de fondo y aparezca el nombre de la zona. Puedes ver el efecto en el vídeo `Envento raton.mp4`. Para el nuevo color de fondo de la zona (que es más oscuro que el original) utiliza el método `Color.darker()`. El color del texto para el nombre de la zona es el color de la clase `Zona`.



Tarea 2: Manejo y renderizado de JTable

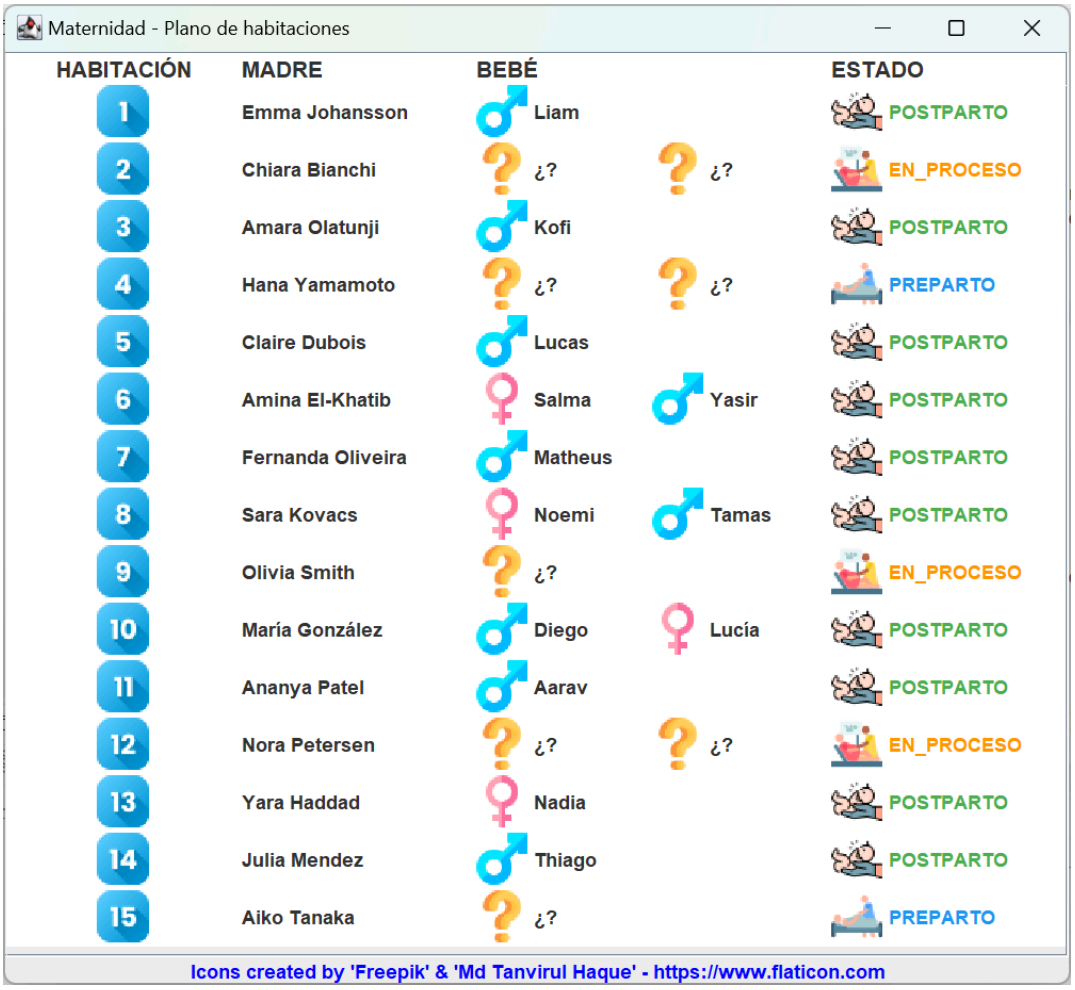
El objetivo de esta tarea es crear un diseño para la tabla de habitaciones de maternidad tal y como se muestra en la imagen de la página siguiente.

Tarea 2.1: Modelo de datos [1,5 puntos] [HabitacionModel] [50 líneas]

Crea la clase **`HabitacionModel`** que representa el modelo de datos personalizado para la tabla de habitaciones. Ten en cuenta las siguientes características:

- El constructor recibe como parámetro una lista de partos **`List<Parto>`**.
- Debes pensar bien el retorno del método **`getValueAt()`** para cada columna.
- Ninguna celda es editable.

Una vez creado el nuevo modelo de datos, debes utilizarlo para inicializar la tabla de habitaciones.



Tarea 2.2: Renderizado de JTable [1,5 puntos] [PlanoMaternidad] [60 líneas]

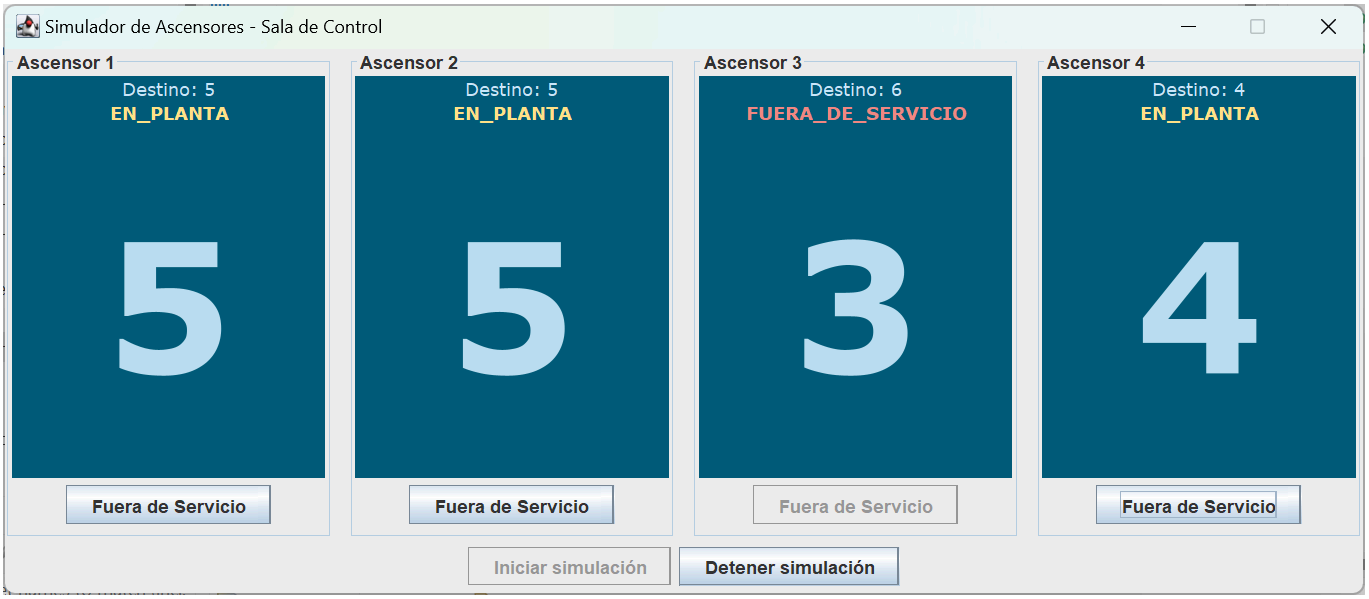
El objetivo de esta tarea es lograr el diseño de la tabla de habitaciones:

- Debes impedir que las columnas de la tabla puedan reorganizarse.
- Los textos de la cabecera están en negrita y tamaño 14pt.
- Ajusta de manera adecuada la alineación horizontal de las columnas.
- La tabla no tiene línea de cuadrícula.
- Las imágenes para los números de habitación están en **`resources/images/`**.
- Las imágenes para el sexo de los bebés están en la enumeración **`Sexo`** que está en la clase **`Bebe`**.
- Cada **`Estado`** de un parto tiene un atributo de tipo **`Color`**, además de su imagen correspondiente.
- La altura de las filas debe ajustarse para visualizar las imágenes completamente.
- La anchura de la columna bebé es de 150 píxeles.

Tarea 4: Hilos [4 puntos] [Ascensores] [60 líneas]

El objetivo de esta tarea es implementar un simulador del panel de control de los ascensores del hall principal del hospital. Para ello dispones de una interfaz gráfica con un botón para iniciar y detener la simulación completa y una serie de elementos para representar cada uno de los ascensores:

- Un display grande indicando la planta en la que se encuentra el ascensor.
- Una indicación del estado: **`PARADO`**, **`EN_MOVIMIENTO`**, **`EN_PLANTA`** y **`FUERA_DE_SERVICIO`**.
- Un display con la plata destino hacia la que se desplaza el ascensor.



La simulación para cada uno de los ascensores debe hacer lo siguiente (en un ciclo continuo):

1. Generar aleatoriamente la planta hacia la que se desplaza el ascensor a partir de **`int[] PLANTAS`**.
2. Actualizar la indicación visual del estado y la planta destino.
3. Simular el movimiento desde la planta actual hasta la planta destino. El tiempo para pasar de una planta a otra es de 500 mseg.
4. Una vez que el ascensor llegue al destino, se detiene en esa planta 2 seg. (y se modifica el estado).
5. Tras pasar los 2 seg. en la planta, se vuelve al punto 1.

En cualquier momento se puede detener un ascensor pulsando el botón **`btnDetener`**, en ese caso, se detiene el proceso del ascensor y se muestra el estado **`FUERA_DE_SERVICIO`**.

En la carpeta **`resources/doc`** tienes el vídeo **`Ascensores.mp4`** que muestra el objetivo a conseguir.