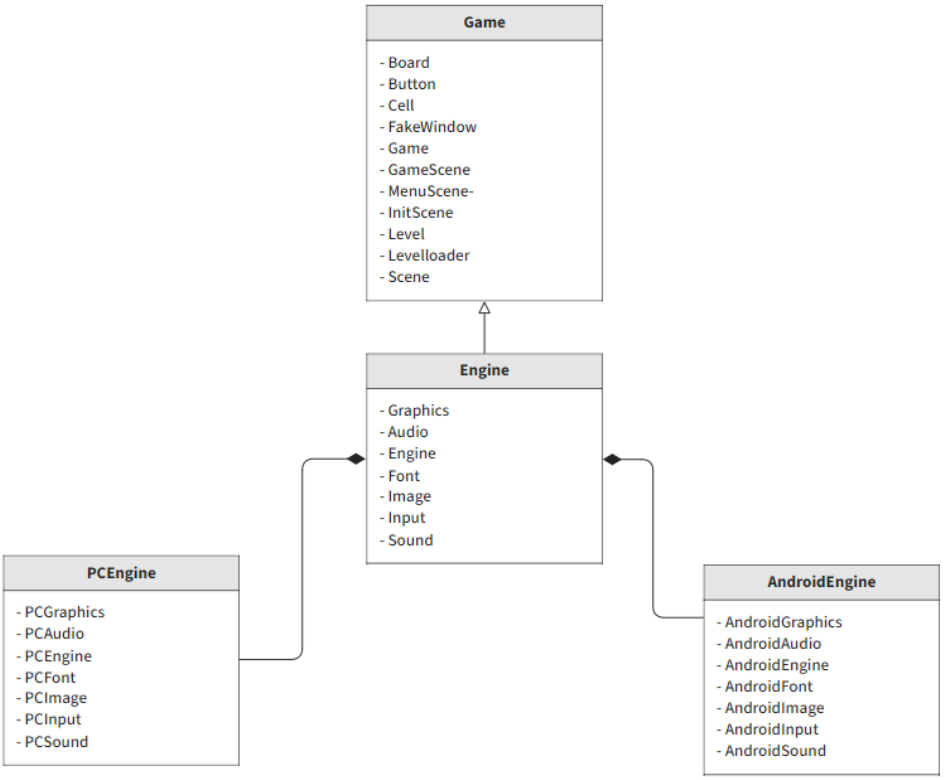
Práctica 1

1. Arquitectura del motor

Como se sugirió en clase, el módulo Engine sirve de interfaz de clases para los dos diferentes motores, y estos implementan sus métodos de manera correspondiente. El módulo Game contiene todas las clases que gestionan la Lógica del juego, que llamarán a métodos de las clases Interfaz, y consecuentemente, a los métodos que toquen según el sistema que ejecute la aplicación.



2. Clases lógicas

- **Board:** Clase que gestiona el juego, guarda la información de la partida y la solución del juego.
- **Button:** Clase que gestiona los botones del juego.
- **Cell:** Clase que define las casillas del tablero y el estado en el que se encuentran.
- **FakeWindow:** Clase que dibuja un rectángulo blanco sobre el que se basa el dibujado del resto de objetos del juego.
- **Game:** Clase que inicializa el juego y gestiona la ejecución de los estados.
- **GameScene:** Clase que contiene la escena jugable.
- **InitScene:** Clase que contiene la primera escena del juego.
- **MenuScene:** Clase que contiene la selección de nivel del juego.
- **Level:** Clase que contiene la información de un nivel.
- **LevelLoader:** Clase que crea los niveles de forma aleatoria o por archivo.
- **Scene:** Interfaz que se usa para la creación de escenas.

Tanto las escenas como el tablero, las celdas, los botones y la FakeWindow tienen métodos de **update**, **render** y **handleInput** para gestionar la colocación y dibujado de sus elementos.

Las escenas gestionan las funciones de sus botones con las funciones **buttonFunction**, que llaman a métodos del Game.

El tablero comprueba el estado de sus celdas con la función **checkResults**, que indica tanto si el resultado es correcto como la cantidad de casillas restantes y casillas erróneas que quedan.

3. Especificaciones de implementación

El renderizado de los elementos de las tres escenas se realiza en función de la posición y el tamaño de la **FakeWindow**, que simplemente es un rectángulo blanco que ajusta su tamaño para que sea siempre de relación ancho/alto 2:3.

Debido a ciertos problemas de implementación en la versión de móvil que no supimos resolver, la FakeWindow se dibuja un poco más alta de lo que debería pintarse para que los elementos no se saliesen por la parte de abajo de dicha

“ventana”. En la versión de PC el cambio no se nota en exceso, excepto cuando, si se modifica el tamaño de la ventana, se pasa de tener la ventana más ancha que alta a lo contrario.

En cuanto a los **assets**, tan sólo es necesario colocar los assets en la carpeta raíz data/assets, y en cuanto se inicialice la aplicación de Android, se copiarán a la carpeta que Android necesita. La carpeta en cuestión es:

AndroidLauncher/src/main/assets/data/assets/

Tal vez se note una pequeña redundancia en las carpetas, pero fue necesario hacerlo de esa manera para que la forma de poner las rutas en la obtención de assets en código fuera lo más cómoda posible.

Por último, sólo aclarar que los colores de las casillas no coinciden con los establecidos en el enunciado de la práctica, pero a decir verdad, no lo vimos en primera instancia y directamente pusimos los colores y formas de el juego original para móvil (por lo menos en iOS) **Nonogram.com**.