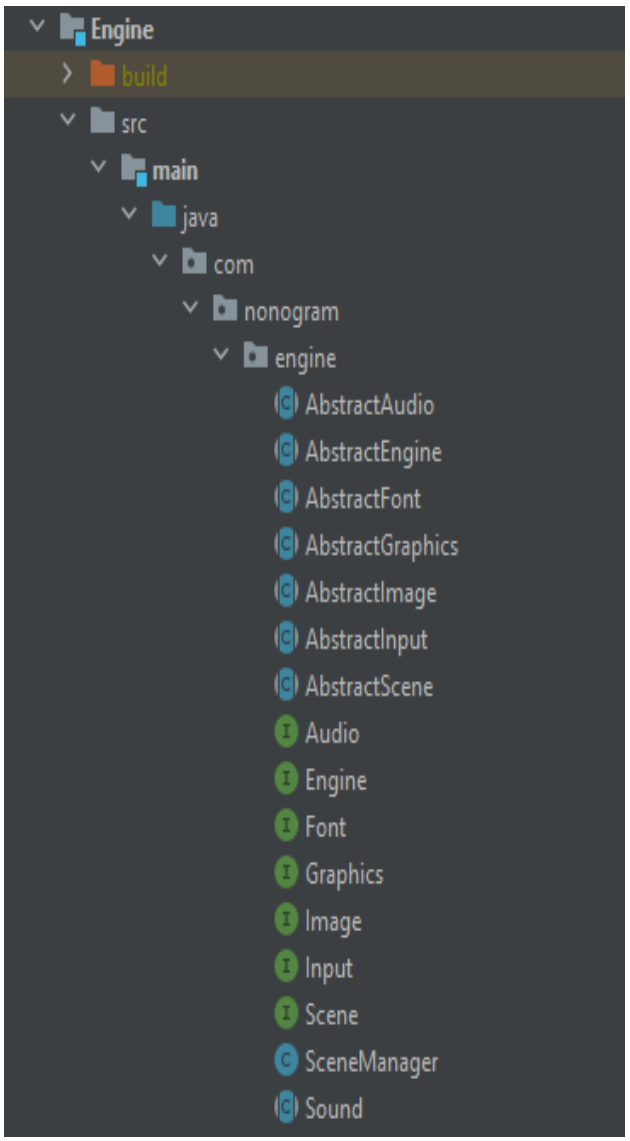
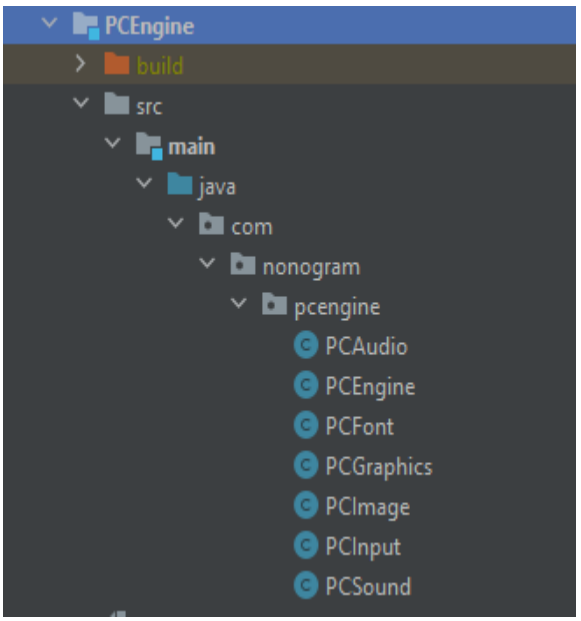
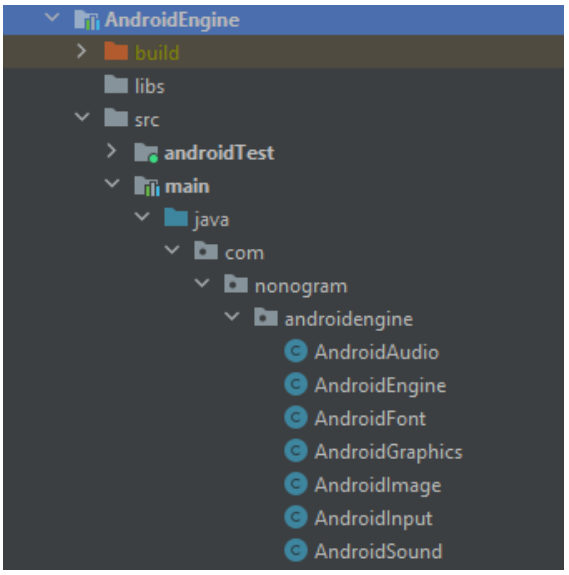
VDM – Práctica 1 – Grupo 04

1. Arquitectura de clases







En el módulo Engine se encuentran las interfaces y clases abstractas que contienen las funcionalidades compartidas por el motor de Android y de PC. Los módulos de PC y Android extenderán las clases abstractas con los métodos específicos para cada plataforma.

La clase más importante es Engine, que será la que inicialice y guarde las instancias de las diferentes clases que controlan los distintos aspectos de la funcionalidad: Input, Audio, Graphics, etc. La instancia de Engine tiene también un SceneManager, que tendrá un stack de escenas de juego. En el bucle de juego (que se encuentra en Engine), se llama a los métodos de actualización (update, render, input) de la escena actual. Estas escenas (clase AbstractScene) son las que se usarán de base para las escenas del módulo Logic, y será en ellas en las que se implementarán las funcionalidades que queramos que dependan del bucle de juego.

VDM – Práctica 1 – Grupo 04

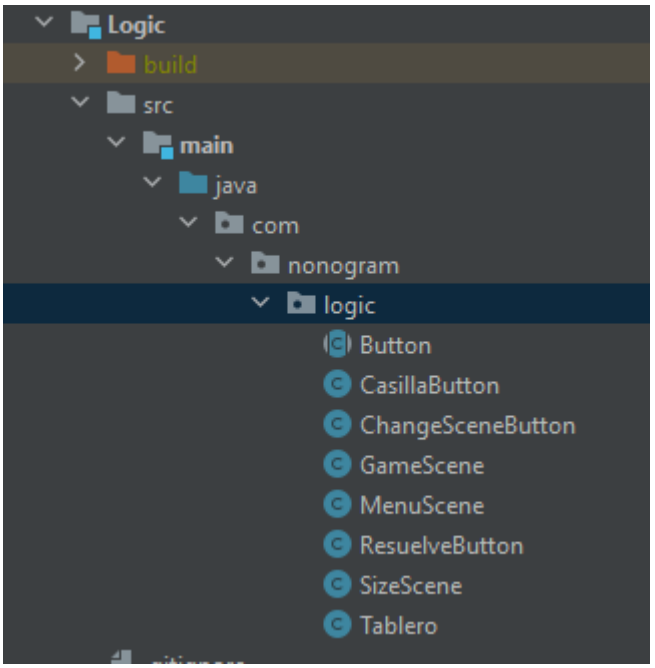
Para lanzar el juego, tendremos dos módulos diferentes, uno para lanzar el juego en PC y otro para lanzarlo desde Android.

```java
package com.nonogram.pclauncher;

import ...

public class PCLauncher {

    PCLauncher(){
        sceneinicial = new MenuScene( gameWidth: 450, gameHeight: 800);
        _engine = new PCEngine( windowName: "finestra", w: 450, h: 800,  fullScreen: false);
        //manejo de errores: si se crea mal algo, para antes de empezar.
        if(!_engine.init() || !_engine.getSceneManager().push(sceneinicial)){
            _engine.stop();
        }
    }

    public static void main(String[] args){
        PCLauncher pclauncher = new PCLauncher();
        pclauncher._engine.resume();
    }


    MenuScene sceneinicial;
    private final PCEngine _engine;
}
```

```java
public class AndroidLauncher extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MenuScene sceneinicial = new MenuScene( gameWidth: 450, gameHeight: 800);
        _myEngine = new AndroidEngine( context: this);
        //manejo de errores: si se crea mal algo, para antes de empezar.
        if(!_myEngine.init() || !_myEngine.getSceneManager().push(sceneinicial)){
            _myEngine.stop();
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        _myEngine.pause();
    }

    @Override
    protected void onResume() {
        // Avisamos a la vista (que es la encargada del active render)
        // de lo que está pasando.
        super.onResume();
        _myEngine.resume();
    }

    MenuScene sceneinicial;
    private AndroidEngine _myEngine;

}
```

La clase PCLauncher contiene un método main que inicia la hebra (el método resume() de la clase Engine) y la clase AndroidLauncher hereda de AppCompactActivity para que Android la detecte como una actividad de la cual queremos usar sus métodos OnPause,OnResume y OnCreate. Desde los archivos .gradle y manifiestos indicamos que ésta actividad será la que se lanzará de manera predeterminada, para que así comience con ella. Como las escenas las controlamos mediante la API del motor, no necesitaríamos más actividades.

En cuanto a la lógica, disponemos de tres escenas: MenuScene, SizeScene y GameScene. En los launchers, crearemos un MenuScene, que con su respectivo botón hará un push de una nueva escena SizeScene, en la que elegiremos el tamaño del tablero a jugar y creará una GameScene con el tamaño de tablero especificado.
El resto de casillas son clases auxiliares para el funcionamiento de la lógica.



VDM – Práctica 1 – Grupo 04

2. Opcionales y extras implementadas

En cuanto a las implementaciones opcionales, hemos realizado tableros con un tamaño rectangular y la opción de hacer la pantalla en PC a tamaño completo. Para lo segundo, se debe indicar mediante un booleano a la hora de crear la instancia de PCEngine en la constructora de PCLauncher.

A parte de eso, hemos añadido la opción de cambiar el nivel de dificultad de la partida. Lo que hace esto es cambiar la probabilidad de líneas resolubles generadas en el tablero, entendiendo líneas resolubles como una línea que puede ser resuelta por sí sola sin necesidad de saber el contexto de las línea de alrededor (ejemplos: una fila con valor 0 , una línea con valor igual al número máximo de línea, una línea que con los números indicados + espacios sólo tenga una solución).