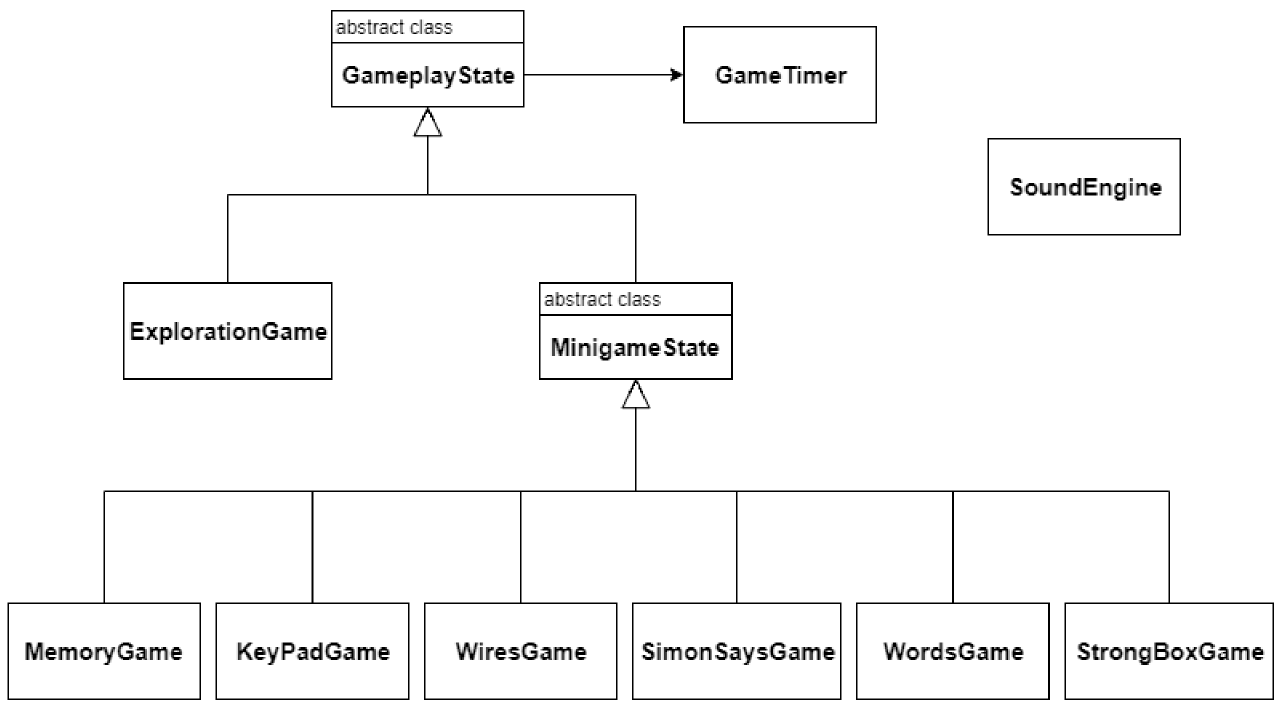
Videogame structure

This document describes the structure of the videogame code. In particular, it explains:

- The structure of the states of the game, the sound and the timer.
- The management of the elements in the map.

States game, timer and sound



All the concrete gameplay state classes represent a game part. They extend the abstract class **GameplayState** wich implements the timer and game difficult management, common to all states.

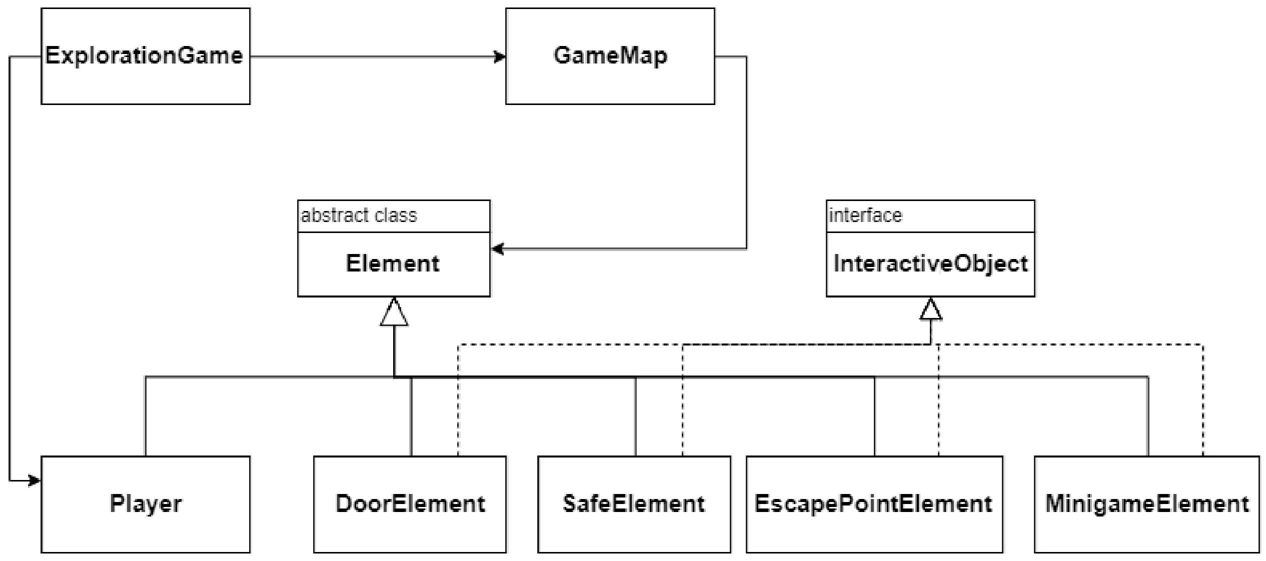
ExplorationGame class manage the main game state, where the player moves the character into the map and interacts with the ambient to access to minigames.

For each minigame there is a concrete class that extends the abstract class **MinigameState,** which provides a common interface for minigame management (ex. minigame completed, error done…) and implements data exchange between ExplorationGame and MinigameState concrete classes.

GameTimer class handles the timer. In order to provide a shared instance of the timer to all GameplayState concrete classes, GameTimer was made following the Singleton Pattern.

As the timer, the game sounds are handled by **SoundEngine** class following Singleton Pattern.

Management of map elements



GameMap class represents the map of each level, contains a set of objects of interest and provides to ExplorationGame class all the methods in order to manage the collision with them.
Each object of interest is an instance of a concrete class that extends **Element** abstract class.
One of the main Element concrete classes is **Player** class. This class represents the character controlled by player.
A subset of Element is composed of all the objects of interest that provide a reaction to a collision. This concrete classes extend Element and implement the interface **InteractiveObject** which requires to implement a particular interaction based on the type of object.