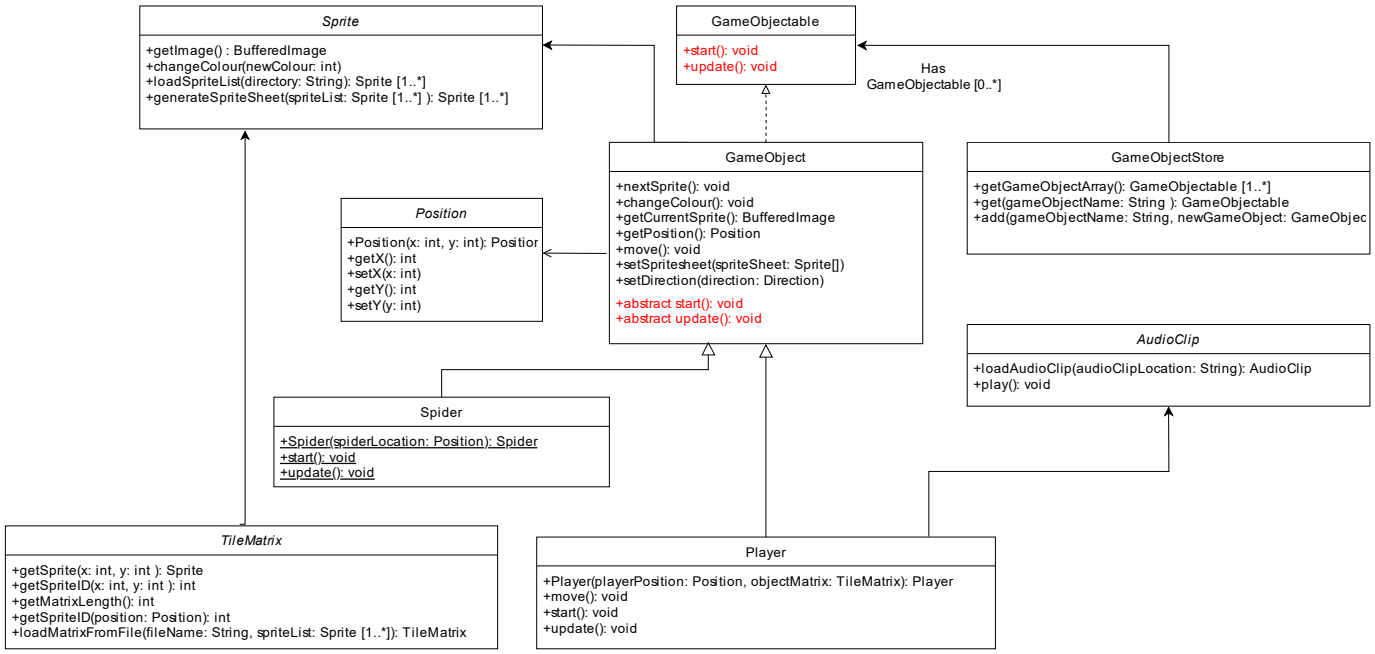
Declan Kelly - G00378925

UML diagram



To avoid clutter `Direction`, `GameView`, `Position`, `Renderer` and `Runner` are absent from this diagram, since they are not key to the design.

Design

1. Tile matrices have been moved into their own files found in `./resources/tilemaps/`, load a new tile matrix using `TileMatrix.loadMatrixFromFile`, this will avoid hard-coding matrices into the code.
2. The `Player` and `Spider` exhibit common functionality (both having a `Position`, `Direction` and `Sprite` associated with them), they now both inherit from the `GameObject`, this increases code reuse. (DRY)
3. To be a `GameObject` you must implement the `start` and `update` methods, start is called on initialisation and update called once per a frame, this pattern is consistent with game engines such as Unity.
4. `GameObjects` are stored in `GameObjectStore`, when the game starts, the `GameView` iterates through all objects in the store, calling the `start` method on each, allowing entity's to set themselves up.
5. On each frame the `update` method is called on each `GameObject`, and each `GameObject` is passed to the `Renderer` class, to be drawn to the screen.
6. To decoupled the `GameObject` from the `GameObjectStore`, I've introduced a `GameObjectable` interface, that `GameObject`'s must implement.
7. `GameObjectStore` is implemented as a singleton, because there only one instance needed for the game, it can be retrieved using `getInstance`, for example the `Player` could query this and check if there is a `Spider` nearby.
8. The `GameView.Builder` can be used to assist with the setup of the `GameView`, this is used by the `Runner`.

9. The `AudioClip` cannot be initialised directly, only with the use of the factory static method `AudioClip.loadAudioClip`.
10. Rendering specific code has been moved to the `Renderer` class, this include isometric calculations and code associated with `Graphics2D`. (note the class is absent from UML above, to having avoid too many classes in it); As described above, once every frame, the `GameObject`'s and `TileMatrix` are drawn using `Renderer.renderGameObject` and `Renderer.renderMatrices` methods.

Extras

1. The `AudioClip` class is used to implement the feature of playing sound effects in the game.
2. When you press the **X** key, the `move` method in the `Player` class is executed, playing the move sound.
3. The objective of the game is to locate the treasure chest. When the treasure chest is found, a celebratory sound will be played.
4. I produced the sounds myself using a tool (`https://raylibtech.itch.io/rfxgen`).