Responsibility driven design

For this sprint we have to implement multiplayer mode (competitive/cooperative) and survival mode. For this to be possible we have to store the game mode somehow and we also need a better way to store the score, as the current method doesn't allow us to add a 2nd player in a nice way. We have to make a storage class of which only a single instance may be created at any time. Because of this we will use the singleton pattern.

Singleton requirements:

- The singleton storing the game mode will be called "GameInfo". It will store the mode of the game, the score(s), the name(s) of the player(s) and all other general game variables which are needed for future features.
- The singleton will have a static getter method, which returns the instance of the class (and create it if there are no instances of it yet).
- Public getters and setters for all variables which have to be read/written from/to.
- All getters may only return copies of objects, or primitive types to avoid external classes from changing them and to make synchronizing the class easier (if the need for synchronization ever occurs).
- Methods may only take primitive types, or enum instances as variables. This is to ensure the objects in the singleton will never contain invalid (or null) objects.
- The GameInfo class may not produce any inCode warnings.

UML of the "GameInfo" class: