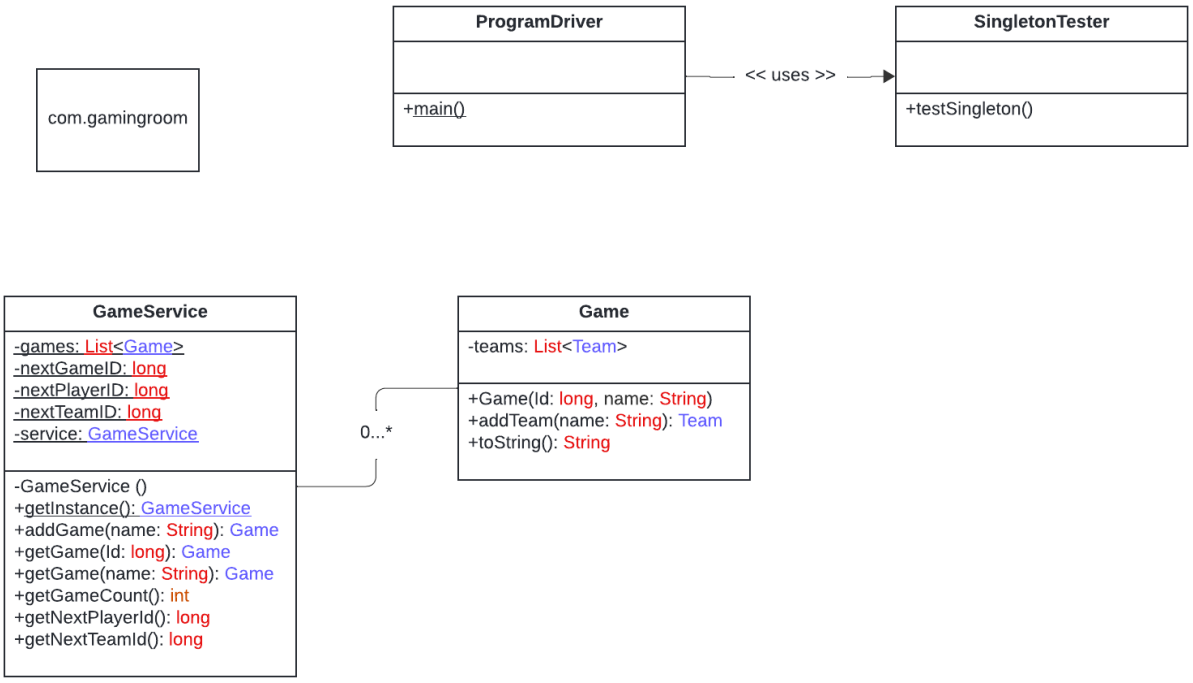

Iterator Pattern

Purpose: The Iterator pattern provides a method to sequentially access elements in a collection without revealing its underlying structure. It enables traversal through complex data structures, like lists or trees, without requiring the client to understand the specifics of how the collection is organized.

Characteristics: The Iterator pattern encapsulates the logic for traversing a collection, allowing for different traversal mechanisms without altering the underlying collection. This promotes separation of concerns by decoupling the iteration logic from the collection itself, leading to a cleaner design. It also supports the use of multiple iterators on the same collection, enabling simultaneous traversal in different ways. Additionally, the pattern is versatile and can be applied to a variety of data structures.

Singleton Pattern

Purpose: The Singleton pattern guarantees that a class has only one instance and offers a global access point to it. This is especially useful when a single object is required to manage actions across the system, such as in logging, configuration management, or connection pooling.

Characteristics: The Singleton class restricts instantiation to a single instance, providing controlled, global access to it. Often, this instance is created using lazy initialization, meaning it is only instantiated when first requested, which aids in resource management. In multi-threaded applications, thread safety must be ensured to avoid issues during the instance creation process. While the global access point simplifies management, it can also lead to potential drawbacks, such as tight coupling within the system.