Project Brief

Overview

Assignment is to build an object-oriented library to play the game of Carcassonne – see the PDF attached (CarcassonneRules.pdf). **Pages 8 to 16** are those relevant to this assignment – just the basic game, no expansions). Library should expose a class **Game** to its users, which can be instantiated to create a new game (with a given number of players, 2 to 6). The instance properties and methods of Game should allow the game to be played (according to the rules) and all necessary game information to be accessed at all times (in a structured, programmatic way). Code executing illegal game moves or other illegal actions, on the Game class or the components it exposes, **<u>should not compile.</u>**

Task

library with the functionality described above, using an object oriented approach in Typescript:

1. Low-level data used by the library should be structured by means of suitable types (try to avoid classes for lightweight data) and validated where necessary. As much data validation as possible should be delegated to the static type-checking
2. High-level components used by the library should be structured through interfaces and classes, with public methods providing access to external functionality and private/protected methods providing access to internal functionality
3. Design should make use of adequately chosen types, interfaces and access control to ensure that the library cannot be misused, while allowing individual components to be safely exposed to the users
4. Where relevant, use advanced type features (e.g. polymorphism, advanced types, type operators), idiomatic language features (e.g. destructuring, for..of loops, iterators, map/filter on arrays), and object-oriented patterns (e.g. factory, singleton, flyweight, facade, global object)
5. Where relevant, implement reusable generic data structures (e.g. trees, graphs, queues/stacks

Code should be clear and concise: long or complex method/function bodies should be avoided whenever possible (e.g. by delegating to helper methods/functions, or to other components). Where long or complex code is unavoidable, the individual steps should be concisely documented using single-line comments

Deliverables

1. TypeScript code, including the tsconfig.json file and a brief readme file (plain-text or markdown) explaining how to compile your code (it can be as simple as “Run tsc in this folder.”)
2. Clearly documented code to explain its design and implementation (code documentation should include notes on all classes, methods, functions, interfaces and types using TSDoc)