SOFTWARE ENGINEERING PROGRAMME
UNIVERSITY OF OXFORD
www.softeng.ox.ac.uk

ASSESSMENT

Student: 1049098

Course: Object-Oriented Programming

Date: 17th January 2022

Grade: 85

Stefano Gogioso

April 28, 2022

Assessment Report on OOP Assignment

You displayed excellence in object oriented design and implementation. You made use of all the tricks taught in the course, and added some new ones. Despite the slight lack of concision in some of your methods, this clearly rises to the level of a model answer.

Mark: 85/100

Design criteria

- **Low-level data structuring:** You structure your low-level data correctly, using immutable lightweight types (some of which are rather advanced, to your merit).
- **High-level component structuring:** Your high-level components are correctly structured via classes and interfaces, with a certain sophistication. You use private methods to implement internal functionality, and you prevent direct instantiation of certain classes by not exporting them directly.
- **Access control:** You make correct good use of private/readonly properties and getters/setters to maintain access control. This is almost always tight, with the exception of `Players.getOrder`, which exposes a mutable array. You missed plenty of opportunities to define property getters and setters, which would have provided a more natural way to design your classes.
- **Advanced type features:** You use several advanced types, including mapped types (with modifiers), generics (alone and combined with mapped types), type intersections, tagged unions, key-in/key-of type, type branding.

Implementation criteria

- **Implementation of specification:** Your implementation allows the game to be played programmatically, and game information to be accessed programmatically (although accessing the board information requires multiple queries). You implement an interesting staging system using the state pattern, ensuring the correct sequencing of game events.
- **Data validation:** You validate your low-level data by means of suitably designed types.
- **Object-oriented patterns:** You implement the factory pattern and the state pattern.
- **Reusable data structures:** You implement a reusable generic data structure for a stack.

Additional criteria

- **Documentation:** Your code is meticulously documented.
- **Code clarity and concision:** Your code is mostly clear, but not always concise. You could have improved the legibility of several of your methods by breaking down complex method logic into auxiliary methods and functions.
- **Idiomatic language use:** You use idiomatic language features, such as `for..of` loops, optional chaining (including for dictionary access), type casts, some array functional methods, arrow functions and generator functions.