WEB3 Assignment 1

The Case

UNO is a card game invented by Merle Robbins in 1971. The game can be played by 2 or more players. The game is played as a series of *hands*. After each hand, the winning player is awarded points (in standard rules). The first to 500 points wins the game.

Each hand is played out as follows:

1. Every player is dealt 7 cards
2. A card is placed face up in the middle of the table – this forms the discard pile
3. The remainder of the cards are placed next to the discard pile – this is the draw pile
4. Every player in turn must play a card that matches the card on the top of the discard pile
5. A card is a match if it has the same colour (i.e. blue) or the same type (i.e. same number or same type of special card) as the other card
6. If a play can't match the card, they must draw a card instead.
7. The first player to play their last card wins the hand
8. The winner is awarded points based on the remaining cards in the other player's hands
9. UNO: Before a player plays their penultimate card, they must say "UNO". Failure to say "UNO" is liable to a 4-card draw penalty.

On top of that, there are several special cards with different effects. Also, there are a lot of details not covered above. The rules are uploaded to itslearning.

The Task

The task of assignment 1 is to implement the standard rules of UNO as described in the uploaded rule set. (**NOTE:** The special case of 2-player UNO is considered standard for this purpose.)

Requirements

Must have

The solution must be as object-oriented as possible.

1. Define a type, Card, representing UNO cards, including special cards but not blank cards. Define the type as precisely as you can (allowing all UNO cards, not allowing anything that isn't a UNO cards).
2. Define types for numbered cards, all coloured cards, and wild cards.
3. Define a type, Type, denoting the different kinds of UNO cards.
4. Define a utility type, TypedCard<Type>, representing UNO cards of the given type.
5. Define an interface or type, Deck, representing a full deck of UNO cards. You might also want to use this to represent any pile of cards. Create an implementation of Deck.
6. Define an interface or type to represent a player hand. Create an implementation of a player hand.
7. Define an interface or type to represent playing a round (or *hand* as it's known in the rule set). Create an implementation of playing a round according to the rules of UNO except the rules surrounding saying "UNO" (also known as rules for "going out").

Should have

1. Implement the rules for saying “UNO”.

Could have

1. Define an interface or type, Game, representing playing a full game of UNO consisting of several rounds complete with scoring. Implement Game.
2. Make as much of the test suite as relevant pass. The test suite is based on my own understanding of the UNO rules.
3. Define the memento types

NOTE: It is not a requirement that the tests pass. You may have a different understanding of the rules. Also, feel free to modify the tests as you please.

The Tests

The tests are unit tests written using Behaviour-Driven Development using Jest.

I have attempted to make the tests assume as little as possible about the internals of your code. (I’m still iterating on that one, so let me know when I fail.) It does make some of them a little complex.

To run the tests, you must implement the (relevant) test adapters. There are many ways to create an object in JavaScript/TypeScript, so I’ve left it open and use adapters instead.

Mementos

Mementos are objects that represent the internal state of an object, allowing you to save the state to a memento and restoring the state from a memento. They have several uses. Here (and int the Yahtzee project) they serve 3 purposes:

1. Saving and loading objects to a database
2. Sending objects over HTTP
3. Simplifying some of the complex tests

The memento objects in this case are plain JavaScript objects of the kind that can be stringified to and serialized from JSON (everything is public, no methods). That’s a bit of a simplification, but it will do.

Advice

- Try working on one test at a time instead of fixing everything at once
- Try to implement the mementos as soon as possible to avoid rework

Deadlines and Other Rules

- Deadline for the first 3 assignments is 19 October.
- The assignments are meant for groups of 2 or 3 students, but I’ll allow groups of 4 students. If you want to try to do it by yourself that’s okay but be aware that it’s a high workload.
- I cannot give feedback on all assignments, but I’m always available for questions.