JS-Intro-OOP-Exercises

Exercises

Objectives:

- Get comfortable creating a class with properties and methods
- Practice creating objects and methods that interact with one another
- Instance several objects that are contained within a special container object
- Encapsulate functionality within the proper class

Exercises:

The Cat

Write yourself a virtual cat - animals with a CLI are so much nicer than ones with fur.

- Create an object that represents a cat. It should have properties for `tiredness`, `hunger`, `lonliness` and `happiness`
- Next, write methods that increase and decrease those properties. Call them something that actually represents what would increase or decrease these things, like "feed", "sleep", or "pet".
- Last, write a method that prints out the cat's status in each area. (Be creative e.g. Paws is really hungry, Paws is VERY happy.)
- Bonus: Make the functions take arguments that increase or decrease arbitrary amounts
- Bonus: Make the functions as arbitrary as cats are - sometimes make it so the cat doesn't *want* to be petted.

The Reading List

An object-oriented book-list!

- Create a class `BookList`
- Create another class called `Book`
- **BookLists** should have the following properties:
    - Number of books marked as read
    - Number of books marked not read yet
    - A reference to the next book to read (book object)
    - A reference to the current book being read (book object)
    - A reference to the last book read (book object)
    - An array of all the Books
- Each **Book** should have several properties:
    - Title
    - Genre

    - Author
    - Read (true or false)
    - Read date, can be blank, otherwise needs to be a JS Date() object

- Every **Booklist** should have a few methods:

    - .add(book)
        - should add a **book** to the books list.
    - .finishCurrentBook()
        - should mark the **book** that is currently being read as "read"
        - Give it a read date of new Date(Date.now())
        - Change the last **book** read to be the book that just got finished
        - Change the current **book** to be the next book to be read
        - Change the next **book** to be read property to be the first unread book you find in the list of books

- **Booklists** and **Books** might need more methods than that. Try to think of more that might be useful.

The Game (advanced)

Pick one of three games: Chess, Poker, or Roshambo (rock, paper, scissors). Roshambo is the easiest, followed by Poker, then Chess.

- Your game should have a **Game** object shat should be responsible for keeping track of it's state
    - State depends on the game, all games have players, but not all games have pieces, cards, or moves. try to plan out what your state will be first
    - Your game should keep a reference to **players**, and it should tell them whether or not they have won or lost
    - Your game should be able to look at the state of the players and execute a **turn**- this is where you put code that looks at the state of each player and evaluates the results of what happens when that player changes it's state
    - Some games will have multiple turns that will change the state of the game, while others (like roshambo) only have one turn that determines a win or a loss.
- You should have **Players** for your game, which should be a class
    - Each player should keep track of how many wins and losses it has
    - Players should keep track of their pieces, cards, or hands
- You should have a class for each **Piece**, **Card** or **Move**

It's up to you to do the rest of the design for this program! Ensure two players can be created in the console as classes, join a game, execute methods with moves, and one player can win each game.