Advanced Movie Rating System using ES6 Classes & OOP Principles

---

Objective: Design and develop a versatile Movie Rating System utilizing the core principles of OOP with ES6 classes. This system will cater to various types of movies, offering capabilities like rating, displaying details, and type differentiation.

---

1. Abstraction:

- **Class**: `Movie`
- **Properties**:
  - `title` (String): Title of the movie.
  - `releaseYear` (Number): The year the movie was released.
  - `genre` (String): The genre of the movie.
  - `#ratings` (Array of Numbers): Stores ratings (private property). Use methods for interaction.
- **Methods**:
  - `addRating(rating: Number)` : Validates and appends a rating to the `#ratings` array.
  - `getAverageRating()` : Computes the average of all ratings and returns it.
  - `displayDetails()` : Logs basic movie details like title, release year, and genre.
  - `typeOfMovie()` : Outputs a string "This is a standard movie.".

---

2. Encapsulation:

- Keep the `#ratings` array private. Externally, the ratings should only be modified using the `addRating` method.
- `addRating(rating: Number)` : Ensures the input rating lies between 1 and 5. If valid, the rating gets appended to the `#ratings` array. If not, an error message is displayed.

---

3. Inheritance:

Class: `Series`

- **Properties**:
  - Inherits properties from `Movie`.
  - `numberOfEpisodes` (Number): Total episodes in the series.
- **Methods**:
  - `displayDetails()`: Show details specific to the series.
  - `typeOfMovie()`: Returns "This is a series."

Class: `Documentary`

- **Properties**:
  - Inherits properties from `Movie`.
  - `topic` (String): Central theme or subject of the documentary.
- **Methods**:
  - `displayDetails()`: Show details specific to the documentary.
  - `typeOfMovie()`: Returns "This is a documentary."

---

4. Polymorphism:

- Override the `displayDetails()` and `typeOfMovie()` methods in each subclass (`Series` and `Documentary`). This ensures each subclass provides its interpretation tailored to its context.

Tasks:

1. **Implementation**:

   - Develop the `Movie` class, ensuring all methods are functional. Test a movie instance to validate its operations.
   - Create the `Series` subclass. Ensure properties and methods are inherited correctly and add specific methods/properties. Test with a series instance.
   - Design the `Documentary` subclass similarly to the `Series`. Again, test its functionality.

2. **Demonstration**:

- Generate a few movie, series, and documentary instances.
- Use the tests file to test your code.

Bonus:

Class: `StreamingPlatform`

- **Properties**:
  - `movies` (Array of Movie objects): Stores movies present on the platform.
- **Methods**:
  - `addMovie(movie: Movie)` : Validates if the given object is a `Movie` instance and adds it to the platform.
  - `findMovieByTitle(title: String)` : Searches and returns the movie object with a matching title.
  - `overallAverageRating()` : Computes the average rating of all movies on the platform.

1. **Implementation**:

   - Develop the `StreamingPlatform` class. Begin by adding movies and then implement search functionality using `findMovieByTitle` .
   - Calculates the overall average rating of all movies on the platform with the `overallAverageRating` method.

2. **Enhanced Encapsulation**:

   - In the `Movie` class, add validation checks for the `releaseYear` (e.g., it shouldn't be in the future or too far in the past).
   - Validate the `title` length, ensuring it's neither too short nor too long.

Note: Ensure to maintain code clarity, comment where necessary, and refactor any repetitive segments.