

Advanced RPG System using ES6 Classes & OOP Principles

Objective: To meticulously design and develop a comprehensive RPG system that encapsulates the intricacies of a vast universe with characters ranging from knights and mages to goblins and dragons, using the principles of OOP in ES6.

1. Abstraction:

- **Class:** Character
 - **Properties:**
 - `name` (String): Represents the character's name.
 - `#health` (Number, Private): Represents the character's health points. Create a getter and a setter for the health private property. In the setter, if the health property is less than zero, set the character's health to zero.
 - `strength` (Number): Quantifies the character's physical power.
 - `defense` (Number): Amount of damage the character can resist.
 - **Methods:**
 - `attack(target: Character)` : Engages another character in combat. Reduces target's health based on the attacker's strength and target's defense.
 - `receiveDamage(damage: Number)` : Implement the `receiveDamage(damage)` method in the `Character` class to reduce the character's health by the given `damage` value, ensuring the health never drops below 0.
 - `displayStats()` : Provides a log of the character's current stats, including name, health, strength, and defense.
 - `characterType()` : Returns a general statement: "This is a basic character."

2. Encapsulation:

- Safeguard the `#health` property by making it private. It can only be changed through the `receiveDamage()` method.
- Implement the `attack()` method following these instructions:
 1. **Method Signature:** Your method should be named `attack` and accept a single parameter, `target`, representing the character being attacked.
 2. **Validation:** - First, ensure that the target is an instance of the `Character` class using the `instanceof` keyword. - Next, validate that the target isn't the same as the character initiating the attack by comparing `target` to `this`.
If either of these conditions isn't met, the method should do nothing and exit.
 3. **Calculate Damage:** - Calculate the damage dealt by subtracting the target's defense value from the attacking character's strength value. - If the calculated damage is less than 0, set it to 0 to ensure that we never deal negative damage.
 4. **Apply Damage:** - Call the `receiveDamage` method on the `target` object, passing the calculated damage as an argument.

3. Inheritance:

- **Class:** `Knight` (Inherits from `Character`)
 - **Additional Properties:**
 - `armor` (Number): Additional defense points or damage (if used offensively).
 - **Methods:**
 - `shieldAttack(target: Character)`: Implement the `shieldAttack(target)` method following these steps:
 1. Ensure the `target` is an instance of the `Character` class and is not the current instance (`this`).
 2. Calculate the damage by adding the `Knight`'s strength to their armor value, then subtracting the target's defense from the sum.
 3. If the resulting damage is negative, set it to 0 to avoid healing the target.

4. Finally, call the `receiveDamage(damage)` method on the target, passing the computed damage value as an argument.

- **Polymorphism:**

- Override `displayStats()` to show armor points.
- Override `characterType()` to return: "This is a knight."

- **Class:** Mage (Inherits from Character)

- **Additional Properties:**

- `mana` (Number): Represents magical energy. Used to cast spells.

- **Methods:**

- `castSpell(target: Character)` : Implement the `castSpell` method following these steps:

- **Parameter:** A single parameter, `target` , which represents the character being attacked. The target should be an instance of the `Character` class.

1. Before casting the spell, ensure the following conditions are met:

a. The `target` should be an instance of the `Character` class.

b. The `target` should not be the same as the mage casting the spell (i.e., a mage should not be able to cast a spell on themselves).

c. The mage should have more than 10 mana points.

2. If the above conditions are satisfied:

a. Calculate the damage as the sum of the mage's `strength` and an additional 10 points for the power of the spell.

b. Reduce the mage's mana by 10 points.

c. Invoke the `receiveDamage` method on the target character with the calculated damage.

3. If any of the conditions are not met, the method should do nothing.

- **Polymorphism:**

- Override the `displayStats()` method in the `Mage` class to display the base character stats using `super.displayStats()` and then output the `Mage`'s mana value.
 - Override `characterType()` to indicate: "This is a mage."
-

Additional Challenges:

5. Associations & Composition:

- **Class:** `Quest`

- **Properties:**

- `name` (String): Title of the quest.
- `description` (String): Brief about the quest's objectives.
- `reward` (String/Number/Object): Could be gold, items, or abilities.
- `requiredEnemies` (Array): List of enemies to be defeated for quest completion.
- `completed` (Boolean): True if the quest has been completed, else false. Initial value `false`.

- **Methods:**

- `completeQuest()` : Marks a quest as completed.

- Players can have an array of `Quest` objects. Create methods to:

- Accept a new quest.
- Complete a quest, verifying all required enemies have been defeated.
- Receive quest rewards.

6. Advanced Polymorphism:

- Create terrain classes like `Forest`, `Desert`, and `Castle`. Each terrain may have methods that alter character attributes or enhance/restrict abilities.
 - E.g., A Mage in a `Forest` terrain could have a `manaRegeneration()` method that boosts mana recovery rates.
-

Tasks:

1. **Construction:** Start by creating the `Character` class and its associated methods. Then, expand to the subclasses, ensuring proper inheritance and method implementation.
2. **Tests:** Run the tests in the tests directory to verify your code runs correctly.

Bonus:

1. **Arena:** Craft a `BattleArena` class where two characters can duel. The duel continues until one character's health drops to zero.
2. **Inventory System:** Develop an inventory mechanism. Allow characters to pick up items that can enhance their stats, utilize these items in battles, and drop them when not needed.