RPG

Objects

Exercise 1: Creating the RPG World

Create a `gameWorld` object that has the properties `players`, `enemies`, `items`, and `npcs` (Non-Player Characters). These properties should be empty arrays that will later hold the entities in the game world.

Also, within `gameWorld`, implement a `createEntity` function that accepts `name`, `health`, `position`, and `inventory` as parameters. This function should return a new entity object with these properties.

Exercise 2: Adding and Removing Entities

Within the `gameWorld` object, implement `addPlayer`, `removePlayer`, `addEnemy`, `removeEnemy`, `addItem`, `removeItem`, `addNPC`, and `removeNPC` methods. These methods should add and remove the respective entities to and from their respective arrays in the `gameWorld` object.

Exercise 3: Movement in the RPG World

Update the `createPlayer` function to add a `move` method to the player entities. The `move` method should accept a `newPosition` and update the `position` property of the player.

Exercise 4: Encounters and Combat

Implement a `checkForEncounters` function that loops over the enemies in the `gameWorld` and checks if any of them have the same `position` as the player. If an enemy is at the same position, the function should initiate combat.

Create a `combat` function that simulates a fight between a player and an enemy. The function should update the health of the player and enemy as they take turns dealing damage to each other. If the health of either the player or enemy reaches 0, they should be removed from the game world.

Exercise 5: Item Interaction

Update the `createPlayer` function to add a `pickUpItem` and `useItem` method to the player entities. `pickUpItem` should add an item to the player's inventory if the player and item are at the same position. `useItem` should remove an item from the player's inventory and apply its effect to the player.

Also, implement a `createItem` function within the `gameWorld` that accepts `name` and `effect` as parameters and returns a new item object.

Exercise 6: Adding Abilities

Implement a `createAbility` function that accepts `name` and `effect` as parameters and returns a new ability object.

Update the `createPlayer` and `createEnemy` functions to accept `abilities` as a parameter and add the `abilities` property to the player and enemy objects.

Exercise 7: Advanced Combat

Enhance the `combat` function to make use of player and enemy abilities. Consider adding random critical hits that deal extra damage.

Exercise 8: Leveling System

Update the `createPlayer` function to add `level` and `xp` (experience points) properties to the player. Also, add a `levelUp` method that increases the player's level by 1 for every 100 experience points, increases the player's health, and reduces the player's xp by the amount used to level up.

Exercise 9: Class System

Update the `createPlayer` function to accept a `playerClass` parameter and add the `playerClass` property to the player object.

Exercise 10: NPC Interaction

Implement a `createNPC` function within the `gameWorld` that accepts `name`, `health`, `position`, `inventory`, and `dialog` as parameters and returns a new NPC object.

Update the `gameWorld` object to include a `talkToNPC` method. This method should allow a player to interact with an NPC if they are at the same position, and print the NPC's dialog.

After each exercise, test your code by creating entities, items, abilities, moving the player, picking up and using items, leveling up, checking for encounters, combat, and NPC interaction.

Ensure all your functions are working as expected.