RPG Game

Closures

1. **Exercise 1: Creating a Player Character**
    - Write a function `createPlayerCharacter` that takes the name (string) and initial level (number) of a player character as arguments. The function should return a closure with:
        - `name` property: This should store the character's name.
        - `level` property: This should store the character's level.
        - `xp` property: This should store the character's experience points.
        - `levelUp` method: This should increment the character's level by one and print a message indicating the character's new level.
        - `addXP` method: This should add the passed XP (number) to the character's current XP and print a message indicating the XP added.
2. **Exercise 2: Managing Inventory**
    - Write a function `createInventory` that doesn't take any arguments and returns a closure with:
        - `addItem` method: This should add the passed item (string) to the inventory and print a message indicating the addition.
        - `removeItem` method: This should remove the passed item (string) from the inventory if it exists, and print a message indicating the removal. If the item doesn't exist in the inventory, it should print a message stating so.
        - `getItems` method: This should return the list of items in the inventory.
3. **Exercise 3: Implementing Abilities**
    - Write a function `createAbility` that takes the name (string) and power (number) of an ability as arguments and returns a closure with:
        - `name` property: This should store the ability's name.
        - `power` property: This should store the ability's power.
        - `useAbility` method: This should print a message indicating the use of the ability and its power.
4. **Exercise 4: Handling Enemies**

- Write a function `createEnemy` that takes the name (string) and health (number) of an enemy as arguments and returns a closure with:
  - `name` property: This should store the enemy's name.
  - `health` property: This should store the enemy's health.
  - `attack` method: This should print a message indicating the enemy is attacking.
  - `reduceHealth` method: This should reduce the enemy's health by the passed damage (number), and print a message indicating the enemy's remaining health or if it has been defeated.
  - `isDefeated` method: This should return a boolean indicating if the enemy has been defeated (health <= 0).