Program Write-Up Document

Student Name: Justin Akwuba

Date:3/26/2024
Course: CIDS 235

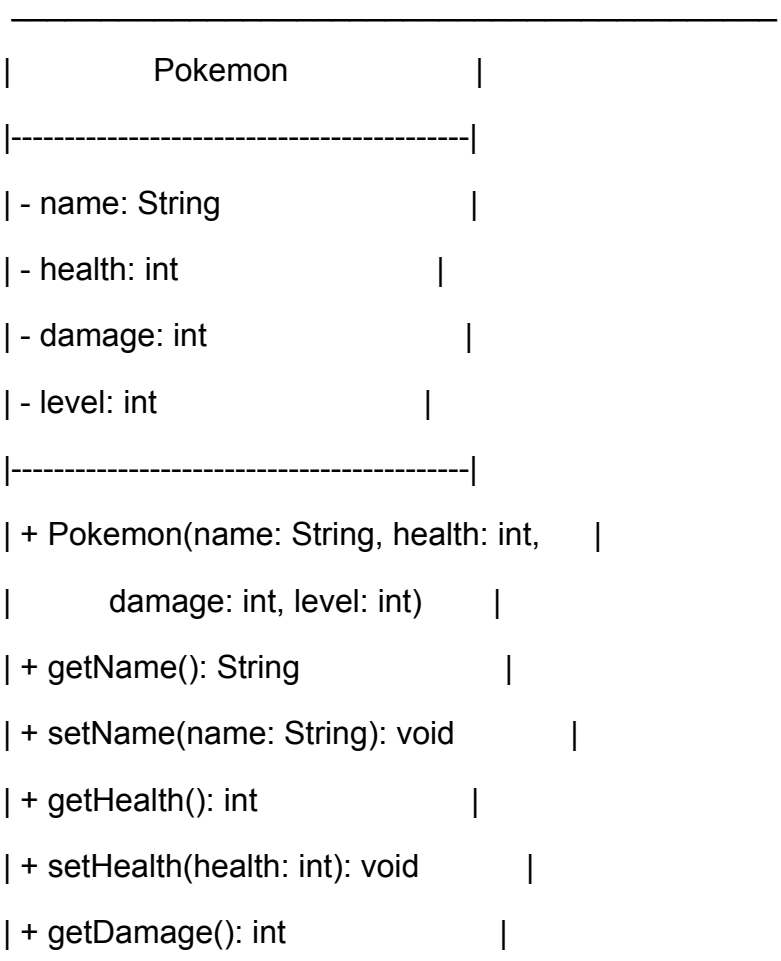
Assignment: Designing a Reusable Pokemon Class with Data Encapsulation in Java

Link to working code:
https://codehs.com/sandbox/id/java-main-WkwatX?filepath=Pokemon.java

1. **Introduction**

This document presents the design and implementation details of the Pokemon (Non-Player Character) class in Java. The objective is to create a versatile Pokemon class that adheres to principles of data encapsulation, allowing for reuse in different game scenarios.

2. **Class Diagrams**



| + setDamage(damage: int): void |

| + getLevel(): int |

| + setLevel(level: int): void |

| + generateRandomPokemon(): Pokemon |

---

3. **Implementation Details**

a) Pokemon Class Documentation

Class Summary:

The Pokemon class representsa battle simulator, storing its attributes such as name, health, damage, and level.

Field Summary:

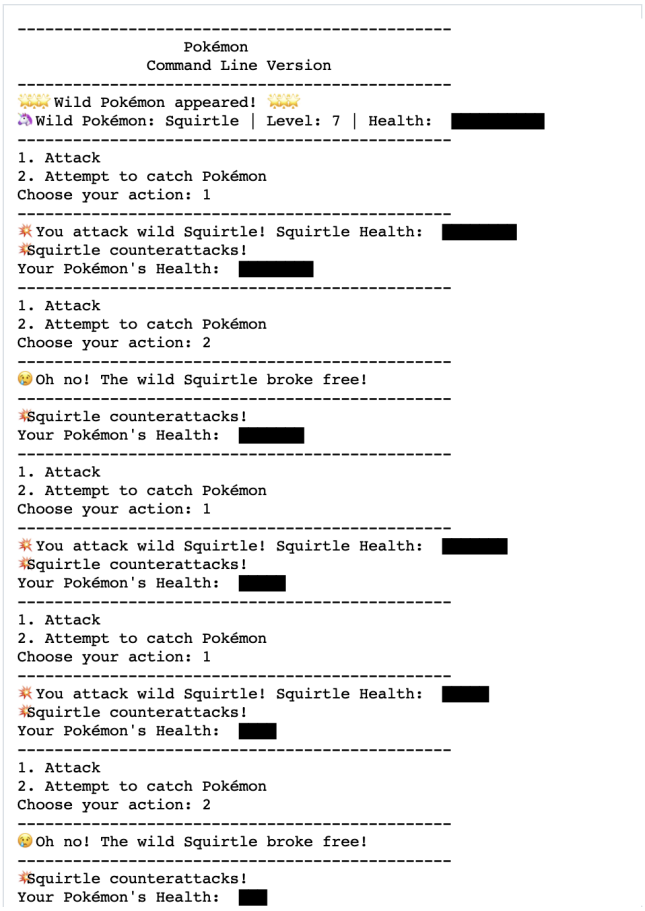
- name: Stores the Pokemon's name.
- health: Stores the Pokemon's health points.
- damage: Stores the Pokemon's damage points.
- level: Stores the Pokemon's level.

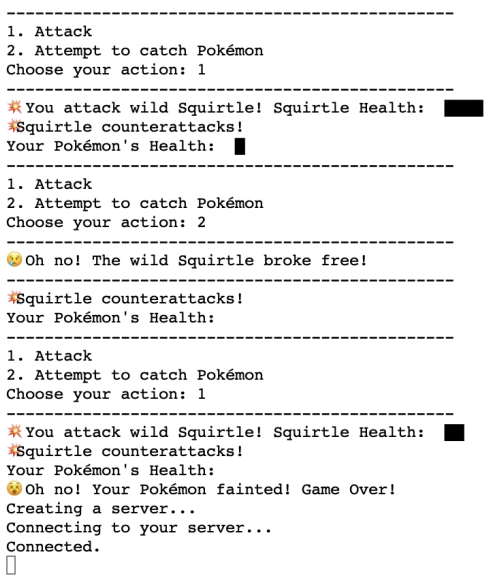
Method Summary:

- Pokemon(name, health, damage, level): Constructor for initializing a Pokemon object.
- getName(): gets the Pokemon's name.
- setName(name): Sets the Pokemon's name.
- getHealth(): gets the Pokemon's health points.
- setHealth(health): Sets the Pokemon's health points.
- getDamage(): gets the Pokemon's damage points.
- setDamage(damage): Sets the Pokemon's damage points.
- getLevel(): gets the Pokemon's level.
- setLevel(level): Sets the Pokemon's level.

- generateRandomPokemon(): Static method to generate a random Pokemon.

4. Screenshots





5. Extra

No extras

6. Discussion

Data encapsulation is like putting your data in a locked box. It means bundling data and the methods that work with it together, and allowing access to the data only through specified methods.

7. Conclusion

The code I worked on creates a Pokemon class in Java. It encapsulates the Pokemon's details like name, health, damage, and level. It provides methods to access and modify these details, ensuring data is controlled and protected. This encapsulation makes the code more organized and easier to use.

8. Appendix

Main.java

```
import java.util.Random;

import java.util.Scanner;


public class Main {


   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      Random random = new Random();


      Pokemon playerPokemon = new Pokemon("Pikachu", 100, 15, 12);


      System.out.println("-------------------------------------------------");

      System.out.println("                Pokémon          ");

      System.out.println("            Command Line Version  ");

      System.out.println("-------------------------------------------------");


      while (playerPokemon.getHealth() > 0) {

         System.out.println("🌟🌟🌟 Wild Pokémon appeared! 🌟🌟🌟");


         Pokemon wildPokemon = Pokemon.generateRandomPokemon();


         System.out.println("🦄 Wild Pokémon: " + wildPokemon.getName() +

               " | Level: " + wildPokemon.getLevel() + " | Health: "

               + healthBar(wildPokemon.getHealth()));


         while (wildPokemon.getHealth() > 0 && playerPokemon.getHealth() > 0) {

            System.out.println("-------------------------------------------------");

            System.out.println("1. Attack");

            System.out.println("2. Attempt to catch Pokémon");

            System.out.print("Choose your action: ");

            int choice = scanner.nextInt();

            scanner.nextLine(); // Consume newline


            switch (choice) {

               case 1:
```

```
            wildPokemon.setHealth(wildPokemon.getHealth() - playerPokemon.getDamage());
            System.out.println("-----------------------------------------------");
            System.out.println("💥 You attack wild " + wildPokemon.getName()
                    + "! " + wildPokemon.getName() + " Health: "
                    + healthBar(wildPokemon.getHealth()));
            break;
        case 2:
            if (random.nextDouble() * wildPokemon.getHealth() + wildPokemon.getLevel() < 10) {
                System.out.println("-----------------------------------------------");
                System.out.println("🎉 Congratulations! You caught the wild "
                        + wildPokemon.getName() + "!");
                System.out.println("-----------------------------------------------");
                return;
            } else {
                System.out.println("-----------------------------------------------");
                System.out.println("😢 Oh no! The wild " + wildPokemon.getName() +
                        " broke free!");
                System.out.println("-----------------------------------------------");
            }
            break;
        default:
            System.out.println("❌ Invalid choice. Please choose again.");
    }

    if (wildPokemon.getHealth() > 0) {
        playerPokemon.setHealth(playerPokemon.getHealth() - wildPokemon.getDamage());
        System.out.println("💥" + wildPokemon.getName() + " counterattacks!");
        System.out.println("Your Pokémon's Health: " + healthBar(playerPokemon.getHealth()));
    }
}

if (playerPokemon.getHealth() <= 0) {
    System.out.println("😵 Oh no! Your Pokémon fainted! Game Over!");
    break;
}

if (wildPokemon.getHealth() <= 0) {
    System.out.println("The wild Pokémon fainted!");
```

```
            System.out.println("Searching for wild Pokémon...");
        }
    }

    scanner.close();
  }

  private static String healthBar(int health) {
    int numBars = health / 10;
    StringBuilder bar = new StringBuilder(" ");
    for (int i = 0; i < numBars; i++) {
      bar.append("");
    }
    for (int i = numBars; i < 5; i++) {
      bar.append(" ");
    }
    return bar.toString();
  }
}
```

Pokemon.java

```
import java.util.Random;

public class Pokemon {
  private String name;
  private int health;
  private int damage;
  private int level;

  public Pokemon(String name, int health, int damage, int level) {
    this.name = name;
    this.health = health;
    this.damage = damage;
    this.level = level;
```

```
}

// Getter and setter methods for name
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

// Getter and setter methods for health
public int getHealth() {
    return health;
}

public void setHealth(int health) {
    this.health = health;
}

// Getter and setter methods for damage
public int getDamage() {
    return damage;
}

public void setDamage(int damage) {
    this.damage = damage;
}

// Getter and setter methods for level
public int getLevel() {
    return level;
}

public void setLevel(int level) {
    this.level = level;
}
```

```
    // Static method to generate random Pokemon
    public static Pokemon generateRandomPokemon() {
        String[] names = {"Charmander", "Bulbasaur", "Squirtle", "Pidgey", "Rattata", "Caterpie"};
        int randomIndex = new Random().nextInt(names.length);
        int level = new Random().nextInt(10) + 1;
        int health = 100;
        int damage = level * 2;
        return new Pokemon(names[randomIndex], health, damage, level);
    }
}
```

1. References