1. Encapsulation-

Encapsulation is the practice of **hiding internal data** of a class and allowing access only through **controlled methods**. This protects the object's state from unintended modification and ensures data integrity.

Implementation in the Project

In the Pokemon Battle Arena:

- Class properties such as **HP, attack stats, type, and status** are declared as private or protected.
- Access to these properties is provided through **public getter methods**.
- State changes are handled only through specific methods like takeDamage() or heal().

2. Inheritance-

Inheritance allows a class to **reuse properties and methods** from another class. It establishes a **parent-child relationship** between classes.

Implementation in the Project

- The Pokemon class **inherits** from the abstract class BattleEntity.
- Common properties such as name, hp, and battle-related methods are defined in BattleEntity.
- Pokemon-specific behavior (types, moves, STAB calculations) is added in the Pokemon class.

3. Polymorphism

Definition

Polymorphism allows different classes to be **treated as the same type** through a common interface, while still executing their own specific behavior.

Implementation in the Project

- A BattleAction interface defines a common method: execute().
- Different actions such as AttackAction, SwitchAction, and ItemAction implement this interface.
- The battle engine processes actions **without knowing their exact type**.

4. Abstraction-

Abstraction focuses on **what an object does** rather than **how it does it**. It hides complex implementation details and exposes only essential behavior.

Implementation in the Project

- The BattleEntity class is declared as **abstract**.
- It defines core structure and behavior shared by all battle participants.
- Abstract methods force child classes to implement required functionality.