Il **Composite** è un pattern strutturale che permette di comporre oggetti in strutture ad albero per rappresentare gerarchie parte-tutto. Il Composite permette ai client di trattare oggetti singoli e composizioni di oggetti in modo uniforme.

Componenti del Composite

1. **Component (Componente)**: Un'interfaccia che dichiara operazioni comuni sia per oggetti compositi che per oggetti singoli.
2. **Leaf (Foglia)**: Un oggetto singolo che implementa le operazioni della componente.
3. **Composite (Composito)**: Un oggetto composto da foglie e altri oggetti composite, implementa le operazioni della componente e le delega ai suoi figli.

Esempio con Pokémon

Immagina di voler gestire squadre di Pokémon o gruppi di mosse.

Interfaccia Component

java Copia codice

```java
public interface ComponenteSquadra {
    void mostraDettagli();
}
```

Leaf

java Copia codice

```java
public class PokemonIndividuale implements ComponenteSquadra {
    private String nome;

    public PokemonIndividuale(String nome) {
        this.nome = nome;
    }

    @Override
    public void mostraDettagli() {
        System.out.println("Pokémon: " + nome);
    }
}
```

Composite

java Copia codice

```java
import java.util.ArrayList;
import java.util.List;

public class SquadraPokemon implements ComponenteSquadra {
    private List<ComponenteSquadra> membri = new ArrayList<>();

    @Override
    public void mostraDettagli() {
        for (ComponenteSquadra membro : membri) {
            membro.mostraDettagli();
        }
    }

    public void aggiungiMembro(ComponenteSquadra membro) {
        membri.add(membro);
    }

    public void rimuoviMembro(ComponenteSquadra membro) {
        membri.remove(membro);
    }
}
```

Applicazione Principale

java Copia codice

```java
public class MainApp {
    public static void main(String[] args) {
        ComponenteSquadra pikachu = new PokemonIndividuale("Pikachu");
        ComponenteSquadra charmander = new PokemonIndividuale("Charmander");

        SquadraPokemon squadra = new SquadraPokemon();
        squadra.aggiungiMembro(pikachu);
        squadra.aggiungiMembro(charmander);

        squadra.mostraDettagli();
        // Output:
        // Pokémon: Pikachu
        // Pokémon: Charmander
    }
}
```

Questi pattern di progettazione possono aiutarti a organizzare e gestire il codice in modo più efficace e manutenibile per il tuo progetto di simulazione di lotte Pokémon.