Praktikum Lanjutan PBO — Monster Battle

Abstraction, Inheritance, Polymorphism, Encapsulation pada Studi Kasus yang Sama

Aturan Umum Praktikum

- Bahasa: Java.
- Setiap class dalam file terpisah, gunakan package bernama `game`.
- Gunakan akses modifier yang tepat (private, protected, public) dan getter/setter tervalidasi.

Bagian A — Refactor Dasar (Abstraction + Encapsulation)

1. Ubah Character menjadi abstract class Character dengan:

- private fields: name, health, attackPower.
- Constructor tervalidasi (health ≥ 0, attackPower ≥ 0).
  - Ada pengecekan jika health dan/atau attack power yang dimasukkan <0
- abstract method: void attack(Character target).
- final method: public final boolean isAlive() untuk mencegah override.

2. Tambahkan method: public final void performTurn(Character target) yang menjalankan urutan: cek isAlive(), panggil attack(target), log hasil.

Bagian B — Hierarki Musuh & Boss (Inheritance + Polymorphism)

3. Buat abstract class Enemy extends Character dengan field tambahan threatLevel (1–5) + validasi (ada pengecekan).
4. Ubah Monster menjadi class Monster extends Enemy. Implementasi attack boleh randomized damage berbasis attackPower.
5. Tambahkan class BossMonster extends Enemy dengan serangan spesial “Rage Strike” (damage ×2.0) jika health < 50% atau setiap 3 giliran (simpan state turn internal).

Bagian C — Interface untuk Skill & Efek (Abstraction + Polymorphism)

6. Buat interface Skill: String name(); void apply(Character self, Character target);
7. Buat Implementasi interface skill : HealSkill(int amount) untuk memulihkan self; PiercingStrike(double multiplier) untuk serangan yang mengabaikan 25% damage reduction (via hook onIncomingDamage).
8. Tambah ke Player: List<Skill> skills (immutable reference; modifikasi via addSkill). Pada attack, jika ada PiercingStrike gunakan salah satu secara acak.

Bagian D — Interface Tambahan: Status Effect / Buff

9. Buat interface StatusEffect: onTurnStart(self), onTurnEnd(self), isExpired().
10. Buat implementasinya, Contoh efek: Regen(perTurn, duration) dan Shield(flatReduce, duration); gunakan di Character.onIncomingDamage mengecek efek aktif.
11. Tambahkan manajemen efek di Character: List<StatusEffect> effects + addEffect(StatusEffect e); panggil onTurnStart/End pada performTurn.

Bagian E — Strategy untuk Perilaku Serang (Interface AttackStrategy)

12. Buat interface AttackStrategy: int computeDamage(Character self, Character target).
13. Implementasi interface: FixedStrategy() dan LevelScaledStrategy(int bonusPerLevel).
14. Modifikasi Player dan Enemy agar strategi dapat diaplikasikan pada kelas tersebut.

Bagian F — Simulasi Turn-Based & Kondisi Menang

15. Buat Battle dengan konstruktor menerima List<Character> teamA, teamB; method run() mensimulasikan giliran bergantian hingga salah satu tim kalah.
16. Auto-targeting: Enemy menarget Player dengan HP tertinggi; Player menarget Enemy dengan threatLevel tertinggi lalu HP terendah.
17. Cetak ringkas log tiap turn dan statistik akhir.

Bagian G — Uji & Validasi

18. HealSkill tidak melebihi maxHealth.
19. BossMonster memicu Rage Strike (HP < 50% atau giliran ke-3).
20. Shield mengurangi damage sesuai konfigurasi.
21. AttackStrategy berbeda menghasilkan damage berbeda.
22. Setter health melakukan pengecekan dan menolak nilai negatif.

Bagian H — Tugas Laporan

Teori:

23. UML Class Diagram (wajib): Character (abstract), Enemy (abstract), Player, Monster, BossMonster, Skill (interface), StatusEffect (interface), AttackStrategy (interface), Battle.

Praktikum:

24. Cuplikan Kode & Output: tangkapan layar eksekusi 3–5 turn yang melibatkan Player, Monster, BossMonster, serta penggunaan minimal 1 Skill dan 1 StatusEffect.

Ketentuan Penilaian (Rubrik)

1. Kebenaran Fungsional (35%): simulasi berjalan, kondisi menang/kalah jelas.

2. Penerapan OOP (35%): abstraction (abstract/interface), inheritance, polymorphism, encapsulation terlihat & benar.
3. Kualitas Desain & Kode (15%): struktur paket, akses modifier, dokumentasi ringkas.
4. ClassDiagram & Laporan (15%): diagram akurat & narasi padat.

Bonus (+10%): efek/strategi baru disertai pengujian.

Template Kode (Skeleton) — Bukan Solusi, Hanya Signature Penting

```
package game;


import java.util.*;


public abstract class Character {

    private final String name;

    protected final int maxHealth;

    private int health;

    private final int attackPower;


    private final List<StatusEffect> effects = new ArrayList<>();


    protected Character(String name, int health, int attackPower) {

        // validasi & inisialisasi

    }


    public final String getName() { return name; }

    public final int getAttackPower() { return attackPower; }

    public final int getHealth() { return health; }

    protected final void setHealth(int value) { /* clamp 0..maxHealth */ }
```

```
    protected int onIncomingDamage(int base) {
        // hook untuk efek seperti Shield; default return base
        return base;
    }

    public final boolean isAlive() { return health > 0; }

    public final void takeDamage(int dmg) { setHealth(getHealth() -
Math.max(0, dmg)); }

    public final void addEffect(StatusEffect e) { /* validasi & tambah */ }

    public final void performTurn(Character target) {
        // panggil effects.onTurnStart, attack(target), effects.onTurnEnd
    }

    public abstract void attack(Character target);
}
```

```
package game;

public abstract class Enemy extends Character {
    private int threatLevel; // 1..5
    protected AttackStrategy strategy;

    protected Enemy(String name, int hp, int ap, int threatLevel,
AttackStrategy strategy) {
```

```
        super(name, hp, ap);
        // validasi threatLevel & strategy
    }

    public final int getThreatLevel() { return threatLevel; }
    public final void setStrategy(AttackStrategy s) { /* non-null */ }
}
```

```
package game;

import java.util.*;

public class Player extends Character {
    private int level;
    private AttackStrategy strategy;
    private final List<Skill> skills = new ArrayList<>();

    public Player(String name, int hp, int ap, int level, AttackStrategy
strategy) {
        super(name, hp, ap);
        // validasi
    }

    public void addSkill(Skill s) { /* non-null */ }

    @Override
    public void attack(Character target) {
```

```
        // gunakan strategy computeDamage, cek apakah ada PiercingStrike di
skills lalu apply

    }

}
```

```
package game;



public class Monster extends Enemy {

    public Monster(String name, int hp, int ap, int threatLevel,
AttackStrategy strategy) {

        super(name, hp, ap, threatLevel, strategy);

    }



    @Override

    public void attack(Character target) {

        // random damage berbasis attackPower / strategy

    }

}
```

```
package game;



public class BossMonster extends Enemy {

    private int turnCounter = 0;



    public BossMonster(String name, int hp, int ap, int threatLevel,
AttackStrategy strategy) {

        super(name, hp, ap, threatLevel, strategy);
```

```
    }


    @Override
    public void attack(Character target) {
        // Rage Strike: 2x damage jika hp < 50% atau setiap 3 giliran
    }
}
```

```
package game;


public interface Skill {
    String name();
    void apply(Character self, Character target);
}
```

```
package game;


public interface StatusEffect {
    void onTurnStart(Character self);
    void onTurnEnd(Character self);
    boolean isExpired();
}
```

```
package game;
```

```
public interface AttackStrategy {

    int computeDamage(Character self, Character target);

}
```

```
package game;


import java.util.*;


public class Battle {

    private final List<Character> teamA;

    private final List<Character> teamB;


    public Battle(List<Character> teamA, List<Character> teamB) {

        this.teamA = teamA; this.teamB = teamB;

    }


    public void run() {

        // loop turn, auto-targeting, log hasil, berhenti saat salah satu tim
habis

    }

}
```

```
package game;


public class GameTest {

    public static void main(String[] args) {
```

```
        Player p = new Player("HeroVipkas", 120, 25, 5, new
LevelScaledStrategy(2));

        p.addSkill(new HealSkill(15));

        // tambah efek awal, musuh, boss, lalu jalankan Battle

    }

}
```

Contoh output :

```
=== SETUP ===
Team A:
  - Player(name=HeroVipkas, HP=120/120, AP=25, Lv=5, Strategy=LevelScaled(+2/level))
    Skills: [HealSkill(+15), PiercingStrike(x1.2)]
    Effects: [Shield(-10 dmg, 3 turns), Regen(+8 HP, 4 turns)]
Team B:
  - BossMonster(name=Drake, HP=150/150, AP=28, Threat=5)
  - Monster(name=Goblin, HP=80/80, AP=12, Threat=2)

Damage rules:
  - Player base damage = AP + Lv*2 = 25 + 10 = 35
  - PiercingStrike: 35 * 1.2 ≈ 42 (bypass sebagian reduksi)
  - Shield(flat -10) aktif 3 giliran, Regen(+8) aktif 4 giliran
  - Boss Rage Strike: 2x damage jika HP < 50% ATAU setiap turn ke-3

=== TURN 1 ===
[Start Effects] Player: Shield(active, 3), Regen(active, +8 at end)
[Team A] Player -> Boss (PiercingStrike): 42 dmg
  Boss HP: 150 -> 108
[Team B] Boss -> Player (Normal hit 28, Shield -10): 18 dmg
  Player HP: 120 -> 102
[Team B] Goblin -> Player (Normal 12, Shield -10): 2 dmg
  Player HP: 102 -> 100
[End Effects] Player Regen: +8 HP => 108 ; Shield remaining: 2 turns

=== TURN 2 ===
[Team A] Player -> Boss (Normal Strategy): 35 dmg
  Boss HP: 108 -> 73
[Team B] Boss -> Player (RAGE x2: 56, Shield -10): 46 dmg
  Player HP: 108 -> 62
[Team B] Goblin -> Player (12, Shield -10): 2 dmg
  Player HP: 62 -> 60
[End Effects] Player Regen: +8 HP => 68 ; Shield remaining: 1 turn
```

```
=== TURN 3 ===
[Team A] Player uses HealSkill(+15): 68 -> 83
[Team A] Player -> Boss (PiercingStrike): 42 dmg
  Boss HP: 73 -> 31
[Team B] Boss -> Player (RAGE by 3rd turn: 56, Shield -10): 46 dmg
  Player HP: 83 -> 37
[Team B] Goblin -> Player (12, Shield -10): 2 dmg
  Player HP: 37 -> 35
[End Effects] Player Regen: +8 HP => 43 ; Shield EXPIRES

=== TURN 4 ===
[Team A] Player -> Boss (Normal 35): 35 dmg
  Boss HP: 31 -> 0  (Boss defeated)
[Team B] Goblin -> Player (12, no shield): 12 dmg
  Player HP: 43 -> 31
[End Effects] Player Regen: +8 HP => 39

=== TURN 5 ===
[Team A] Player -> Goblin (Normal 35): 35 dmg
  Goblin HP: 80 -> 45
[Team B] Goblin -> Player (12): 12 dmg
  Player HP: 39 -> 27
[End Effects] Player Regen: +8 HP => 35  (Regen remaining: 1 turn)

=== TURN 6 ===
[Team A] Player -> Goblin (Normal 35): 35 dmg
  Goblin HP: 45 -> 10
[Team B] Goblin -> Player (12): 12 dmg
  Player HP: 35 -> 23
[End Effects] Player Regen: +8 HP => 31  (Regen EXPIRES)

=== TURN 7 ===
[Team A] Player -> Goblin (Normal 35): 35 dmg
  Goblin HP: 10 -> 0  (Goblin defeated)

=== RESULT ===
Team A menang!
Sisa HP:
  - Player(HeroVipkas): 31/120
  - BossMonster(Drake): 0/150
  - Monster(Goblin): 0/80
Stat ringkas:
  - Player total serangan: 42 + 35 + 42 + 35 + 35 + 35 + 35 = 259
  - Boss Rage terpicu: 2 kali (HP<50% dan turn ke-3)
  - Shield menyerap total: 10+10+10+10+10+10 = 60 dmg
  - Regen memulihkan total: 8 * 4 = 32 HP
```