# Kuis Pemrograman Berbasis Objek

## Studi Kasus: Game Petualangan "Monster Battle"

### Deskripsi Singkat

Seorang developer game ingin membuat permainan sederhana bertema adventure battle  antara karakter pemain dan monster. Dalam game ini, setiap karakter memiliki atribut  umum (seperti nama, health, dan attack power), tetapi juga perilaku khusus tergantung  jenis karakternya. Anda diminta untuk mengimplementasikan konsep inheritance,  encapsulation, dan polymorphism agar kode tetap terstruktur dan mudah dikembangkan.

### Spesifikasi Kasus

1. Superclass: Character
 - Atribut: name, health, attackPower
 - Method: attack(Character target), takeDamage(int damage), getter & setter

2. Subclass: Player
 - Tambahan atribut: level
 - Override method attack() dengan tambahan bonus level

3. Subclass: Monster
 - Tambahan atribut: type (contoh: Dragon, Goblin)
 - Override method attack() dengan damage acak

### Contoh Kode (Java)

```
 // Superclass
 class Character {
  private String name;
  private int health;
  private int attackPower;

  public Character(String name, int health, int attackPower) {
  this.name = name;
  this.health = health;
  this.attackPower = attackPower;
  }

  public String getName() { return name; }
  public int getHealth() { return health; }
  public void setHealth(int health) { this.health = health; }

  public void takeDamage(int damage) {
```

```
 health -= damage;
 if (health < 0) health = 0;
 System.out.println(name + " took " + damage + " damage! Remaining health: " +
health);   }

 public abstract void attack(Character target);
}

// Subclass Player
class Player extends Character {
 private int level;

 public Player(String name, int health, int attackPower, int level) {
 super(name, health, attackPower);
 this.level = level;
 }

 @Override
 public void attack(Character target) {
 int damage = (int)(super.getHealth() * 0.1) + level; // bonus dari level
System.out.println(getName() + " attacks with sword! Damage: " + damage);
target.takeDamage(damage);
 }
}

// Subclass Monster
class Monster extends Character {
 private String type;

 public Monster(String name, int health, int attackPower, String type) {
super(name, health, attackPower);
 this.type = type;
 }

 @Override
 public void attack(Character target) {
 int damage = (int)(Math.random() * 10) + 5; // random damage
 System.out.println(getName() + " (" + type + ") attacks ferociously! Damage: " +
damage);
 target.takeDamage(damage);
 }
}

// Main class
public class GameTest {
 public static void main(String[] args) {
 Player p1 = new Player("HeroIrsyad", 100, 20, 5);
 Monster m1 = new Monster("Goblin", 80, 15, "Goblin");
 Monster m2 = new Monster("Dragon", 200, 30, "Dragon");
  Character[] characters = {p1, m1, m2};

  for (Character c : characters) {
  c.attack(p1); // semua karakter menyerang player
  }
  }
 }
```

## Kuis Praktikum

Implementasikan kode di atas (pisah tiap class menjadi masing-masing 1 file), lalu modifikasi agar:

1. Menambahkan subclass baru BossMonster dengan serangan spesial (damage 2x lipat).  2. Ganti nama objek player dengan nama anda.
3. Menambahkan fitur healing pada Player.
4. Buat simulasi pertarungan sampai salah satu karakter kalah.

Buat laporan berisi screenshot potongan kode dan/atau hasil eksekusi program

## Kuis Teori

1. Jelaskan bagaimana inheritance, polymorphism, dan encapsulation digunakan dalam program.
2. Buat class diagramnya
3. Kumpulkan dalam bentuk laporan pengerjaan