

**LAPORAN PRAKTIKUM**  
**PEMROGRAMAN BERORIENTASI OBJEK**  
**“SETTER & GETTER METHOD dan STATIC KEYWORD”**



**Oleh:**  
**Benony Gabriel**  
**105222002**

**PROGRAM STUDI ILMU KOMPUTER**  
**FAKULTAS SAINS DAN ILMU KOMPUTER**  
**UNIVERSITAS PERTAMINA**  
**2024**

## 1. Pendahuluan

### 1.1 Setter dan Getter

- Setter: Metode yang digunakan untuk menetapkan nilai pada suatu variabel pribadi (private) dalam suatu kelas. Ini memungkinkan pengguna dari luar kelas untuk mengatur nilai variabel tersebut. Biasanya nama metode setter diawali dengan kata "set" diikuti dengan nama variabel yang diatur.
- Getter: Metode yang digunakan untuk mengambil nilai dari suatu variabel pribadi (private) dalam suatu kelas. Ini memungkinkan pengguna dari luar kelas untuk mendapatkan nilai variabel tersebut. Biasanya nama metode getter diawali dengan kata "get" diikuti dengan nama variabel yang diambil.

### 1.2 Static Keyword

- Static adalah kata kunci yang digunakan dalam Java untuk membuat suatu variabel atau metode bersifat statis.
- Variabel statis berbagi nilainya di antara semua instance dari kelas yang sama dan dapat diakses langsung menggunakan nama kelas, tanpa perlu menginstansiasi objek.
- Metode statis juga dapat diakses langsung menggunakan nama kelas, tanpa perlu menginstansiasi objek, dan biasanya digunakan untuk utilitas umum atau operasi yang berkaitan dengan kelas tersebut.

Singkatnya, setter dan getter digunakan untuk mengatur dan mengambil nilai variabel pribadi dalam suatu kelas, sementara kata kunci static digunakan untuk mendefinisikan variabel atau metode yang terkait dengan kelas itu sendiri, bukan dengan instance objek spesifik.

### 1.3 Studi Kasus

Buatlah game sederhana dimana memiliki 4 buah class dengan ketentuan sebagai berikut:

#### 1.3.1 class Player:

Player memiliki beberapa atribut yang semuanya memiliki access modifier private, yaitu name, baseHealth, baseAttack, incrementHealth, incrementAttack, level, totalDamage, dan isAlive. Player juga memiliki object member yaitu armor dari class Armor dan weapon dari class Weapon. Pada constructor Player memiliki 1 parameter saja, dimana isi constructornya adalah mengassign nilai name dengan parameter, baseHealth 100, baseAttack 100, level 1, incrementHealth 20, incrementAttack 20, dan isAlive true. Player juga menerapkan konsep enkapsulasi. Pada method Player, memiliki method display untuk menampilkan semua atribut dari Player, kemudian method attack, defence, dan maxHealth. Pada method attack terdapat aturan bahwasannya player yang diserang akan berkurang healthnya jika player yang diserang memiliki defence yang lebih rendah dari pada yang menyerang, jika tidak maka tidak ada damage yang didapatkan oleh yang diserang.

Setiap penyerangan maka player yang menyerang akan level up. Kemudian pada method defence memiliki pengkondisian dimana jika damage penyerang lebih besar daripada defence yang diserang, maka terdapat delta damage sebesar damage - defence, jika tidak maka delta demagenya 0. Kemudian pada method defence juga terdapat pengecekan apakah health si yang diserang kurang dari sama dengan 0 atau tidak, jika iya, maka atur variabel isAlivenya ke false dan atur totalDamage menjadi maxHealth. Method maxHealth berisikan  $\text{baseHealth} + \text{level} * \text{incrementHealth} + \text{armor.hetAddHealth}()$

### 1.3.2 class Weapon:

Pada Weapon terdapat 2 variabel atau atribut yang memiliki access modifier private, yaitu String name dan integer attack. Weapon juga menerapkan prinsip enkapsulasi. Constructor Weapon memiliki isi mengisi variabelnya sendiri dengan parameter (terdapat 2 parameter pada constructor).

### 1.3.3 class Armor:

Pada Armor terdapat 3 variabel atau atribut yang memiliki access modifier private, yaitu name, strength, dan health. Armor juga menerapkan prinsip enkapsulasi. Constructor Armor memiliki isi mengisi variabelnya sendiri dengan parameter (terdapat 3 parameter pada constructor). Pada Armor terdapat 3 variabel atau atribut yang memiliki access modifier private, yaitu name, strength, dan health. Armor juga menerapkan prinsip enkapsulasi. Constructor Armor memiliki isi mengisi variabelnya sendiri dengan parameter (terdapat 3 parameter pada constructor). Pada Armor terdapat method getAddHealth yang berisikan  $\text{strength} \times 10 + \text{health}$ . Kemudian terdapat method getDefencePower dimana isinya  $\text{strength} \times 2$ .

### 1.3.4 class Main:

Pada kelas utama buatlah sebuah objek player1, player2, armor1, armor2, weapon1, weapon2. Lakukan penyerangan player1 terhadap player2, dan player2 terhadap player1 kemudian player 2 terhadap player1.

Contoh tampilan:

Player : Marni  
Level : 1  
Health : 270/270  
Attack : 130  
Alive : true

Player : Issabela  
Level : 1  
Health : 170/170  
Attack : 160  
Alive : true

Marni is attacking Issabela with 130  
Issabela defence power = 2  
Damage earned = 128

Player : Issabela  
Level : 1  
Health : 42/170  
Attack : 160  
Alive : true

Issabela is attacking Marni with 180  
Marni defence power = 10  
Damage earned = 150

Player : Marni  
Level : 2  
Health : 140/290  
Attack : 150  
Alive : true

Issabela is attacking Marni with 180  
Marni defence power = 10  
Damage earned = 170

Player : Marni  
Level : 2  
Health : 0/290  
Attack : 150  
Alive : false

## 2. Variabel

Berikut ini adalah penjelasan singkat mengenai variabel-variabel yang dibuat di setiap kelas:

### 2.1 Class Player

No	Nama Variabel	Tipe data	Keterangan
1	name	String	Menyimpan nama player.
2	baseHealth	int	Menyimpan nilai health awal player.
3	baseAttack	int	Menyimpan nilai attack awal player.
4	incrementHealth	int	Menyimpan jumlah penambahan health setiap kali level naik.
5	inccrementAttack	int	Menyimpan jumlah penambahan attack setiap kali level naik.
	level	int	Menyimpan level player.
	totalDamage	int	Menyimpan total damage yang diterima player.
	isAlive	boolean	Menyimpan status hidup atau mati player.
	armor	Armor	Menyimpan objek armor player.
	weapon	Weapon	Menyimpan objek senjata player.

### 2.2 Class Weapoon

No	Nama Variabel	Tipe data	Keterangan
1	name	String	Menyimpan nama senjata.
2	attack	int	Menyimpan nilai attack pada senjata.

### 2.3 Class Armor

No	Nama Variabel	Tipe data	Fungsi
1	name	String	Menyimpan nama armor.
2	strenght	int	Menyimpan kekuatan armor.
3	health	int	Menyimpan kesehatan armor.

Setiap variabel memiliki tipe data yang sesuai dengan fungsinya dan digunakan untuk menyimpan informasi yang diperlukan dalam pemodelan sistem permainan yang diberikan.

## 3. Constructor dan Method

### 3.1 Class Player

No	Nama Metode	Jenis	Fungsi
1	Constructor Player(String name)	Constructor	Menginisialisasi objek Player dengan nilai awal seperti name, baseHealth, baseAttack, level, incrementHealth, incrementAttack, dan isAlive.
2	display()	Procedure	Menampilkan semua atribut Player.
3	attack(Player targetPlayer)	Procedure	Menyerang player lain dengan mengurangi healthnya berdasarkan attack dan defence.
4	defence(int damage)	Procedure	Mengurangi damage yang diterima dari serangan musuh berdasarkan defence power.
5	maxHealth()	Function	Menghitung dan mengembalikan nilai max health dari player.

### 3.2 Class Weapon

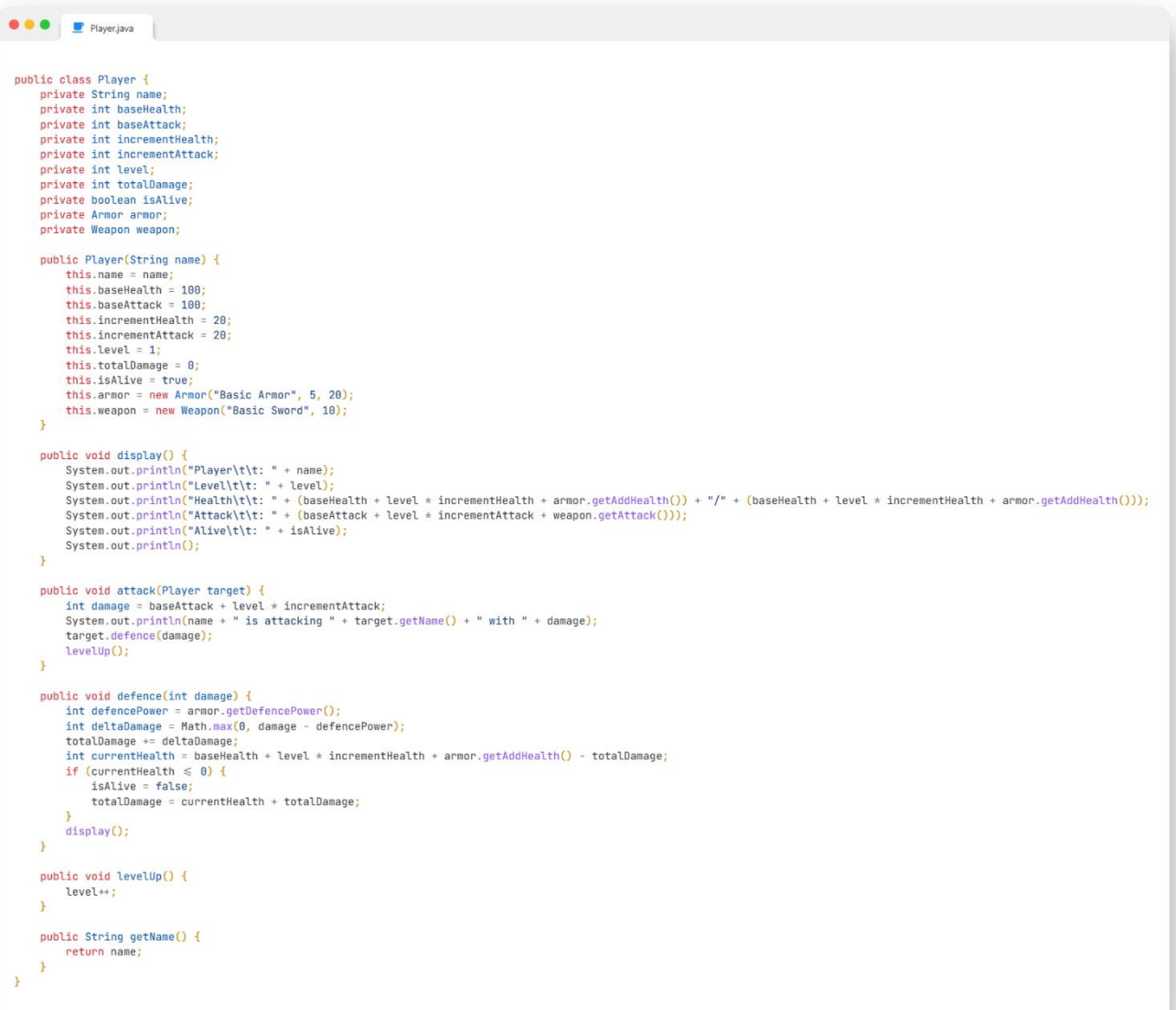
No	Nama Metode	Jenis	Fungsi
1	Constructor Weapon(String name, int attack):	Constructor	Menginisialisasi objek Weapon dengan nilai awal name dan attack.
2	getName():	Function	Mengembalikan nama weapon.
3	getAttack():	Function	Mengembalikan nilai attack weapon.

### 3.3 Class Armor

No	Nama Metode	Jenis	Fungsi
1	Constructor Armor(String name, int strength, int health):	Constructor	Menginisialisasi objek Armor dengan nilai awal name, strength, dan health.
2	getAddHealth():	Function	Menghitung dan mengembalikan nilai tambahan health dari armor.
3	getDefencePower():	Function	Menghitung dan mengembalikan nilai defence power dari armor.

## 5. Dokumentasi dan Pembahasan Code

### 5.1 Class Player



```
public class Player {
    private String name;
    private int baseHealth;
    private int baseAttack;
    private int incrementHealth;
    private int incrementAttack;
    private int level;
    private int totalDamage;
    private boolean isAlive;
    private Armor armor;
    private Weapon weapon;

    public Player(String name) {
        this.name = name;
        this.baseHealth = 100;
        this.baseAttack = 100;
        this.incrementHealth = 20;
        this.incrementAttack = 20;
        this.level = 1;
        this.totalDamage = 0;
        this.isAlive = true;
        this.armor = new Armor("Basic Armor", 5, 20);
        this.weapon = new Weapon("Basic Sword", 10);
    }

    public void display() {
        System.out.println("Player\t\t: " + name);
        System.out.println("Level\t\t: " + level);
        System.out.println("Health\t\t: " + (baseHealth + level * incrementHealth + armor.getAddHealth()) + "/" + (baseHealth + level * incrementHealth + armor.getAddHealth()));
        System.out.println("Attack\t\t: " + (baseAttack + level * incrementAttack + weapon.getAttack()));
        System.out.println("Alive\t\t: " + isAlive);
        System.out.println();
    }

    public void attack(Player target) {
        int damage = baseAttack + level * incrementAttack;
        System.out.println(name + " is attacking " + target.getName() + " with " + damage);
        target.defence(damage);
        levelUp();
    }

    public void defence(int damage) {
        int defencePower = armor.getDefencePower();
        int deltaDamage = Math.max(0, damage - defencePower);
        totalDamage += deltaDamage;
        int currentHealth = baseHealth + level * incrementHealth + armor.getAddHealth() - totalDamage;
        if (currentHealth <= 0) {
            isAlive = false;
            totalDamage = currentHealth + totalDamage;
        }
        display();
    }

    public void levelUp() {
        level++;
    }

    public String getName() {
        return name;
    }
}
```

Kodingan di atas adalah implementasi class `Player`. Berikut adalah penjelasan singkatnya:

#### 1. Variabel Kelas:

- `name`: Nama pemain.
- `baseHealth`: Kesehatan awal pemain.
- `baseAttack`: Serangan dasar pemain.
- `incrementHealth`: Peningkatan kesehatan setiap level.
- `incrementAttack`: Peningkatan serangan setiap level.
- `level`: Level pemain.
- `totalDamage`: Total kerusakan yang diterima oleh pemain.
- `isAlive`: Status hidup atau mati pemain.
- `armor`: Objek Armor yang digunakan oleh pemain.
- `weapon`: Objek Weapon yang digunakan oleh pemain.

2. Constructor `Player(String name)`:

- Menginisialisasi semua variabel kelas, termasuk objek Armor dan Weapon dengan nilai awal yang ditentukan.

3. Method `display()`:

- Menampilkan semua atribut pemain seperti nama, level, kesehatan, serangan, dan status hidup atau mati.

4. Method `attack(Player target)`:

- Mengeksekusi serangan pemain terhadap target dengan menghitung total damage yang dihasilkan dan memanggil method `defence()` dari target.

5. Method `defence(int damage)`:

- Mengurangi kerusakan yang diterima oleh pemain berdasarkan kekuatan pertahanan Armor.
- Memperbarui total damage yang diterima oleh pemain.
- Memperbarui status hidup atau mati pemain berdasarkan kesehatan saat ini.
- Memanggil method `display()` untuk menampilkan informasi pemain setelah pertahanan.

6. Method `levelUp()`:

- Meningkatkan level pemain.

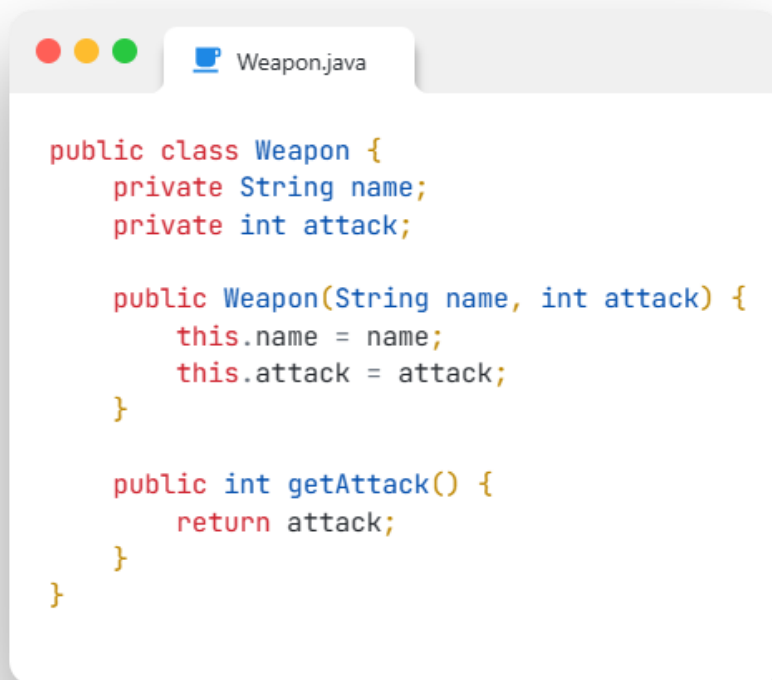
7. Getter `getName()`:

- Mengembalikan nama pemain.

Kelas `Player` ini mengimplementasikan prinsip enkapsulasi dengan menyembunyikan detail implementasi variabel kelas dan menyediakan method publik untuk mengakses dan memanipulasi data.



## 5.2 Class Weapon



```
public class Weapon {  
    private String name;  
    private int attack;  
  
    public Weapon(String name, int attack) {  
        this.name = name;  
        this.attack = attack;  
    }  
  
    public int getAttack() {  
        return attack;  
    }  
}
```

Kode di atas adalah implementasi class `Weapon`. Berikut adalah penjelasannya:

1. Variabel Kelas:

- `name`: Nama senjata.
- `attack`: Jumlah serangan yang dimiliki senjata.

2. Constructor `Weapon(String name, int attack)`:

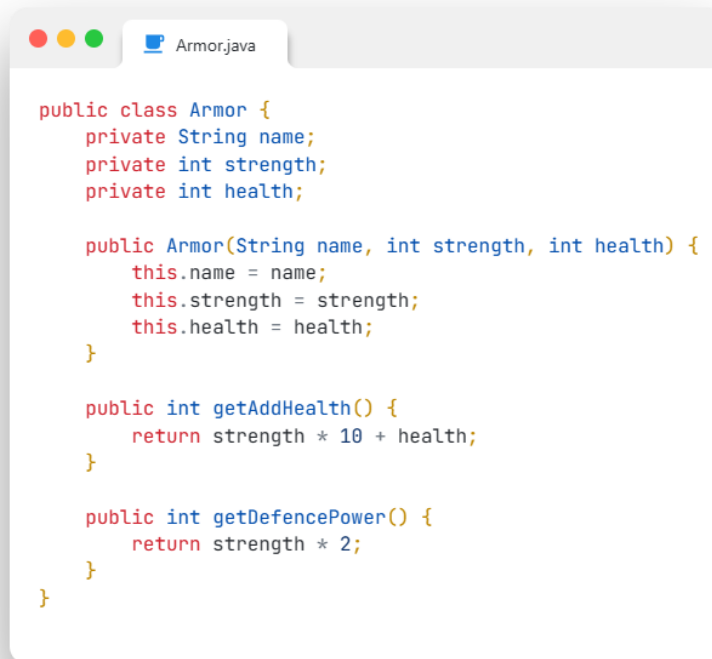
- Menerima dua parameter, yaitu nama senjata dan jumlah serangan.
- Menginisialisasi variabel kelas `name` dan `attack` dengan nilai yang diberikan.

3. Getter `getAttack()`:

- Mengembalikan nilai serangan (`attack`) dari senjata.

Kelas `Weapon` ini juga mengikuti prinsip enkapsulasi dengan menyembunyikan detail implementasi variabel kelas dan menyediakan getter publik untuk mengakses nilai serangan dari senjata.

### 5.3 Class Armor



```
public class Armor {  
    private String name;  
    private int strength;  
    private int health;  
  
    public Armor(String name, int strength, int health) {  
        this.name = name;  
        this.strength = strength;  
        this.health = health;  
    }  
  
    public int getAddHealth() {  
        return strength * 10 + health;  
    }  
  
    public int getDefencePower() {  
        return strength * 2;  
    }  
}
```

Kode di atas adalah implementasi class `Armor` Berikut adalah penjelasannya:

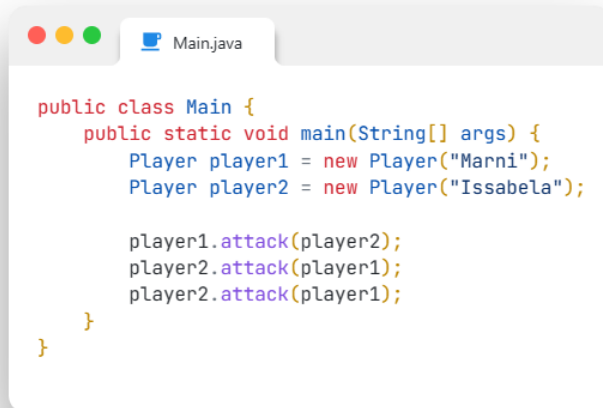
1. Variabel Kelas:
  - `name`: Nama armor.
  - `strength`: Kekuatan armor.
  - `health`: Kesehatan armor.
2. Constructor `Armor(String name, int strength, int health)`:
  - Menerima tiga parameter, yaitu nama armor, kekuatan armor, dan kesehatan armor.
  - Menginisialisasi variabel kelas `name`, `strength`, dan `health` dengan nilai yang diberikan.
3. Method `getAddHealth()`:
  - Mengembalikan nilai total kesehatan armor, dihitung sebagai `strength \* 10 + health`.
  - Menggabungkan kekuatan dan kesehatan armor untuk memberikan total kesehatan tambahan yang diberikan oleh armor.

#### 4. Method `getDefencePower()`:

- Mengembalikan nilai kekuatan pertahanan armor, dihitung sebagai `strength * 2`.
- Memberikan gambaran tentang seberapa kuat pertahanan yang dimiliki oleh armor.

Kelas `Armor` ini juga mengikuti prinsip enkapsulasi dengan menyembunyikan detail implementasi variabel kelas dan menyediakan method getter publik untuk mengakses nilai total kesehatan tambahan dan kekuatan pertahanan armor

## 5.4 Class Main



```
public class Main {  
    public static void main(String[] args) {  
        Player player1 = new Player("Marni");  
        Player player2 = new Player("Issabela");  
  
        player1.attack(player2);  
        player2.attack(player1);  
        player2.attack(player1);  
    }  
}
```

Kelas `Main` merupakan kelas utama dalam program. Berikut adalah penjelasan rinci dari kodenya:

### 1. Method `main(String[] args)`:

- Method utama yang dieksekusi ketika program dijalankan.
- Membuat objek `player1` dan `player2` dari kelas `Player` dengan nama "Marni" dan "Issabela" secara berturut-turut.
- Memanggil method `attack()` dari `player1` untuk menyerang `player2`.
- Memanggil method `attack()` dari `player2` untuk menyerang `player1`.
- Memanggil method `attack()` dari `player2` untuk menyerang `player1` lagi.

Kelas `Main` bertanggung jawab untuk membuat objek pemain, menjalankan serangkaian serangan antar pemain, dan menguji fungsionalitas kelas `Player` dalam simulasi permainan.

## 6. Kesimpulan

Dalam rangka pengembangan game sederhana dengan Java, beberapa konsep dan teknik pemrograman yang penting telah diimplementasikan:

1. OOP (Object-Oriented Programming): Program menggunakan paradigma pemrograman berorientasi objek, di mana entitas dalam permainan seperti pemain, senjata, dan armor diwakili sebagai objek dengan atribut dan perilaku yang terkait.
2. Enkapsulasi: Atribut pada kelas-kelas seperti `Player`, `Weapon`, dan `Armor` dideklarasikan sebagai variabel privat dan diakses melalui metode getter dan setter. Hal ini membantu dalam melindungi data dari modifikasi langsung dan memungkinkan kontrol atas cara akses dan manipulasi data.
3. Konstruktor: Setiap kelas memiliki konstruktor yang digunakan untuk menginisialisasi objek saat pembuatan instansinya. Konstruktor ini memungkinkan pengaturan nilai awal untuk atribut-atribut objek.
4. Method: Setiap kelas memiliki serangkaian metode yang digunakan untuk melakukan tugas-tugas tertentu. Contohnya, dalam kelas `Player`, terdapat metode untuk menyerang pemain lain, bertahan dari serangan, dan meningkatkan level pemain.

Dengan menggabungkan konsep-konsep ini, pengembang dapat membuat permainan yang lebih kompleks dengan logika dan interaksi yang lebih mendalam antara elemen-elemen dalam permainan.

