



## COACHING PROGRAM Week 4

### OOP (*Object-Oriented-Programming*)

#### Bagian 1: Apa Itu OOP?

OOP adalah cara berpikir baru dalam pemrograman! Bayangkan kamu sedang membuat game RPG. Ada banyak objek di dalamnya: ksatria, penyihir, monster, pedang, ramuan. Dengan OOP, kamu bisa merepresentasikan setiap objek tersebut dalam kode dengan cara yang terstruktur dan mudah dikelola.

- **Dunia Nyata di Dalam Kode:** OOP membantumu "menerjemahkan" objek di dunia nyata ke dalam kode program.
- **Kode yang Lebih Terstruktur:** Dengan OOP, kodemu akan lebih terorganisir, mudah dibaca, dan mudah dikembangkan.
- **Reusable dan Efisien:** Kamu bisa menggunakan kembali kode yang sudah ada untuk membuat objek baru yang mirip, sehingga lebih efisien.



## COACHING PROGRAM Week 4

### OOP (*Object-Oriented-Programming*)

#### Bagian 2: Class: Blueprint untuk Objek

*Class* adalah cetakan atau *blueprint* untuk membuat objek. *Class* mendefinisikan karakteristik (atribut) dan perilaku (method) dari sebuah objek.

Contoh:

```
class Ksatria
{
    // Atribut
    public string nama;
    public int nyawa;
    public int kekuatan;

    // Method
    public void Serang()
    {
        // Kode untuk menyerang
    }

    public void TerimaDamage(int damage)
    {
        // Kode untuk mengurangi nyawa
    }
}
```



## COACHING PROGRAM Week 4

### OOP (*Object-Oriented-Programming*)

#### Bagian 3: Object: Wujud Nyata dari Class

*Object* adalah instansiasi dari *class*. Jika *class* adalah cetakan, maka *object* adalah kue yang jadi. Kamu bisa membuat banyak *object* dari satu *class*.

Contoh:

```
Ksatria ksatria1 = new Ksatria();
ksatria1.nama = "Arthur";
ksatria1.nyawa = 100;
ksatria1.kekuatan = 20;

Ksatria ksatria2 = new Ksatria();
ksatria2.nama = "Lancelot";
ksatria2.nyawa = 80;
ksatria2.kekuatan = 25;
```



## COACHING PROGRAM Week 4

### OOP (*Object-Oriented-Programming*)

#### Bagian 4: Inheritance: Mewarisi Sifat

*Inheritance* memungkinkan sebuah *class* (anak) untuk mewarisi atribut dan method dari *class* lain (induk). Ini sangat berguna untuk membuat kode yang efisien dan menghindari duplikasi.

Contoh:

```
class Musuh
{
    public int nyawa;
    public void TerimaDamage(int damage) { ... }

class Goblin : Musuh { ... }
class Orc : Musuh { ... }
```

Goblin dan Orc akan mewarisi atribut nyawa dan method TerimaDamage() dari *class* Musuh.



## COACHING PROGRAM Week 4

### OOP (*Object-Oriented-Programming*)

#### Bagian 5: Encapsulation: Melindungi Data

*Encapsulation* adalah konsep menyembunyikan data di dalam *class* dan mengontrol akses ke data tersebut. Ini membuat kode lebih aman dan mudah dipelihara.

Contoh:

```
class Pemain
{
    private int skor;

    public void TambahSkor(int poin)
    {
        if (poin > 0)
        {
            skor += poin;
        }
    }

    public int GetSkor()
    {
        return skor;
    }
}
```

Atribut skor dibuat private sehingga tidak bisa diakses langsung dari luar *class*. Akses ke skor diatur melalui method TambahSkor() dan GetSkor().



## COACHING PROGRAM Week 4

### OOP (*Object-Oriented-Programming*)

#### Tugas:

- Buatlah program C# yang mensimulasikan pertarungan antara dua ksatria menggunakan konsep OOP. Setiap ksatria memiliki nama, nyawa, dan kekuatan. Gunakan *inheritance* jika memungkinkan.

Semoga materi ini membuat kamu semakin *powerful* dalam menggunakan OOP di C# dan membangun game yang lebih keren! 🎮