1. OOP

* **Inheritance** 🡪 Hubungan kelas antara Parent Class dan Child Class, dimana Parent Class bersifat lebih general dan dapat mewariskan sifat yang indentik kepada Child Class.

Contoh:

**public class Tabung**

**{**

**int tinggi;**

**int jariJari;**

**public Tabung(int tinggi, int jariJari)**

**{**

**this.tinggi = tinggi;**

**this.jariJari = jariJari;**

**}**

**public float HitungVolume()**

**{**

**// t \* PI \* r^2**

**return tinggi \* Mathf.PI \* Mathf.Pow(jariJari, 2f);**

**}**

**}**

**public class Gelas : Tabung**

**{**

**string motif;**

**public Gelas(int tinggi, int jariJari, string motif) : base(tinggi, jariJari)**

**{**

**this.motif = motif;**

**}**

**public string GetMotif() => motif;**

**}**

Dalam kasus ini kelas Tabung adalah parent class dari kelas Gelas. Tabung dan Gelas dapat dibuatkan objek masing-masing. Gelas juga memiliki beberapa attribute yang sama dari tabung seperti tinggi dan jari-jari tabung dan volume juga dapat dihitung.

**Gelas gelas = new Gelas(6, 7, "Bunga");**

**Debug.Log(gelas.HitungVolume()); // Output: 923.6283**

**Debug.Log(gelas.GetMotif()); // Output: Bunga**

* **Polymorphism** 🡪 Sebuah interface untuk memfasilitasi kelas yang memiliki bentuk dan fungsi yang sama, walaupun bukan dari Parent Class-nya.

Contoh:

**public interface IGambar**

**{**

**string Gambarkan();**

**}**

**public class Artikel { … }**

**public class Komik : Artikel, IGambar**

**{**

**public string Gambarkan() => "Gambarkan Komik";**

**}**

**public class Peta : IGambar**

**{**

**public string Gambarkan() => "Gambarkan Peta";**

**}**

Dalam Kasus ini, Ada objek yang dapat menghasilkan gambar dengan menggunakan interface **IGambar**. Misalkan ingin membuat sebuah perusahaan media, terdapat kelas **Artikel** yang berisikan data konten, Artikel ini diturunkan menjadi sebuah cerita yang dapat dikonversikan ke bentuk **Komik**. Misalkan kelas **Peta** tidak ada hubungan-nya dengan **Artikel** tetapi dapat menghasilkan gambar.

**IGambar manga = new Komik();**

**IGambar peta = new Peta();**

**Debug.Log(manga.Gambarkan()); // Output: Gambarkan Komik**

**Debug.Log(peta.Gambarkan()); // Output: Gambarkan Peta**

* **Encapsulation** 🡪 Cara menutup data atau attribut yang dimiliki oleh suatu objek agar tidak sembarangan diakses atau diubah.

Contoh:

**public class KTP**

**{**

**private string NIK;**

**private string tanggalLahir;**

**private string alamat;**

**public KTP(string NIK, string tanggalLahir, string alamat)**

**{**

**this.NIK = NIK;**

**this.tanggalLahir = tanggalLahir;**

**this.alamat = alamat;**

**}**

**public string GetNIK() => NIK;**

**public string GetTglLahir() => tanggalLahir;**

**public void SetAlamat(string alamatBaru) => alamat = alamatBaru;**

**public string GetAlamat() => alamat;**

**}**

Dalam kasus ini, misalkan ingin membuat sebuah KTP yang terdirikan atas data NIK, tanggal lahir, dan alamat. Setelah dibuatkan sebuah objek, data privasi **NIK** dan **tanggalLahir** sudah menjadi data tetap dan tidak dapat diubah (Encapsule), hanya dapat akses mengambil data saja. Sementara Alamat masih bisa diubah.

**KTP ktp = new KTP("1234", "30 Maret 2001", "Jln. Kebon Jerok No.94");**

**ktp.NIK = "5678"; // ERROR karena NIK tidak dapat diakses secara langsung**

**Debug.Log(ktp.GetNIK());**

* **Abstraction** 🡪 Sama halnya dengan Inheritance, tetapi tidak dapat mempresentasikan sebuah objek, agar dapat dipresentasikan sebuah Abstract Class haruslah menurunkan sifatnya kepada Child Class.

Contoh:

**public abstract class Hewan**

**{**

**public abstract void KeluarkanSuara();**

**}**

**public class Kucing : Hewan**

**{**

**public override void KeluarkanSuara() => Debug.Log("Meowww");**

**}**

**public class Kuda : Hewan**

**{**

**public override void KeluarkanSuara() => Debug.Log("HIIIIIIIIIIIIIII HAAAAAAAAAAAAA");**

**}**

Dalam kasus ini contohnya kelas **Hewan**, dimana **Hewan** belum memiliki suara yang pasti seperti apa. **Hewan** tidak dapat dibuatkan sebuah objek karena kelas **Hewan** adalah abstrak yang artinya tidak memiliki bentuk yang pasti, hanyalah konsep.