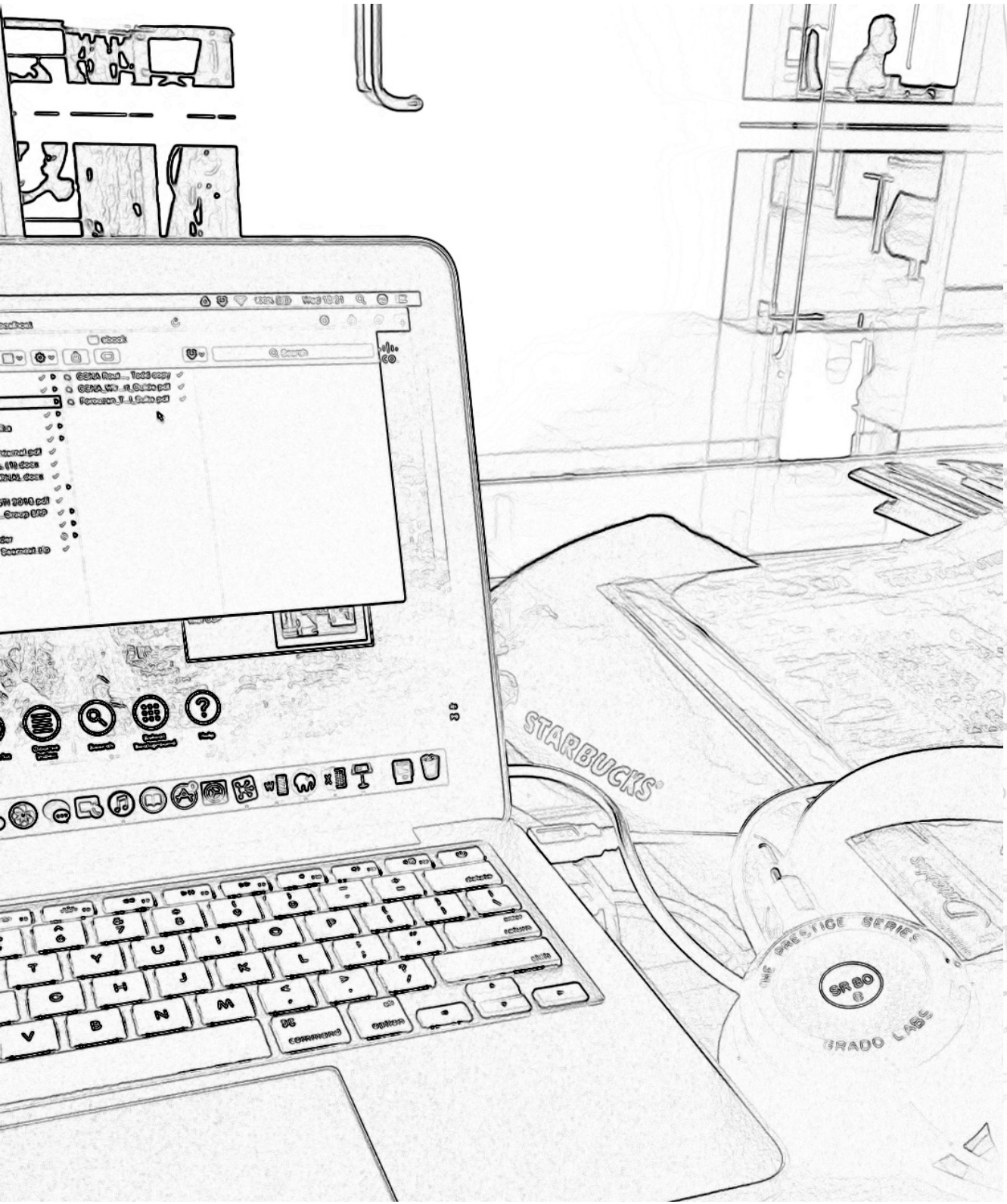



# Pemograman Berorientasi Objek

# C#

**Week 3**
**Abstrak dan Interface dalam suatu kelas**

UNIVERSITAS KOMPUTER INDONESIA
UNIKOM

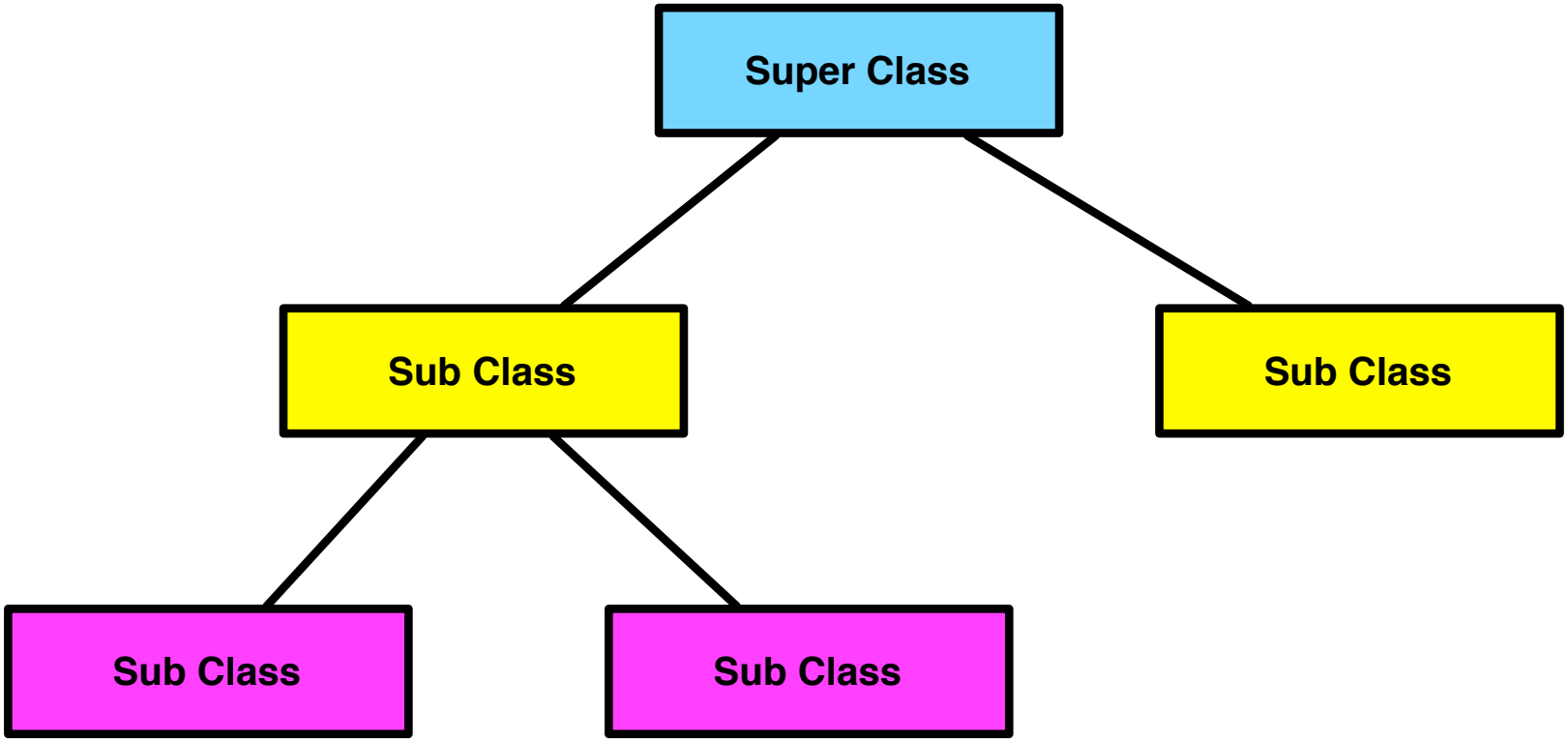
# Kompetensi Dasar

- Setelah mengikuti mata kuliah ini diharapkan mahasiswa memiliki kemampuan untuk:
    1. Menguraikan konsep dasar PBO
    2. Mengimplementasikan kelas (class) dan metode (method) untuk mewakili obyek-obyek dalam sistem
    3. Mengimplementasikan abstrak dan interface dalam suatu kelas
    4. Menggunakan initialization dan instance dalam class
    5. Menerapkan konsep orientasi objek : Inheritance dalam sistem
    6. Menerapkan konsep Polimorfisme
    7. Menguraikan dasar UML
    8. Mengindentifikasi Permasalahan menggunakan Use Case Diagram
    9. Mengindentifikasi Permasalahan menggunakan Activity Diagram
    10. Mengindentifikasi Permasalahan menggunakan Sequence Diagram
    11. Menyusun Class Diagram dari permasalahan
    12. Mewujudkan Object Oriented Design (OOD) dan Object Oriented Programing (OOP) menjadi sebuah aplikasi (Studi Kasus)

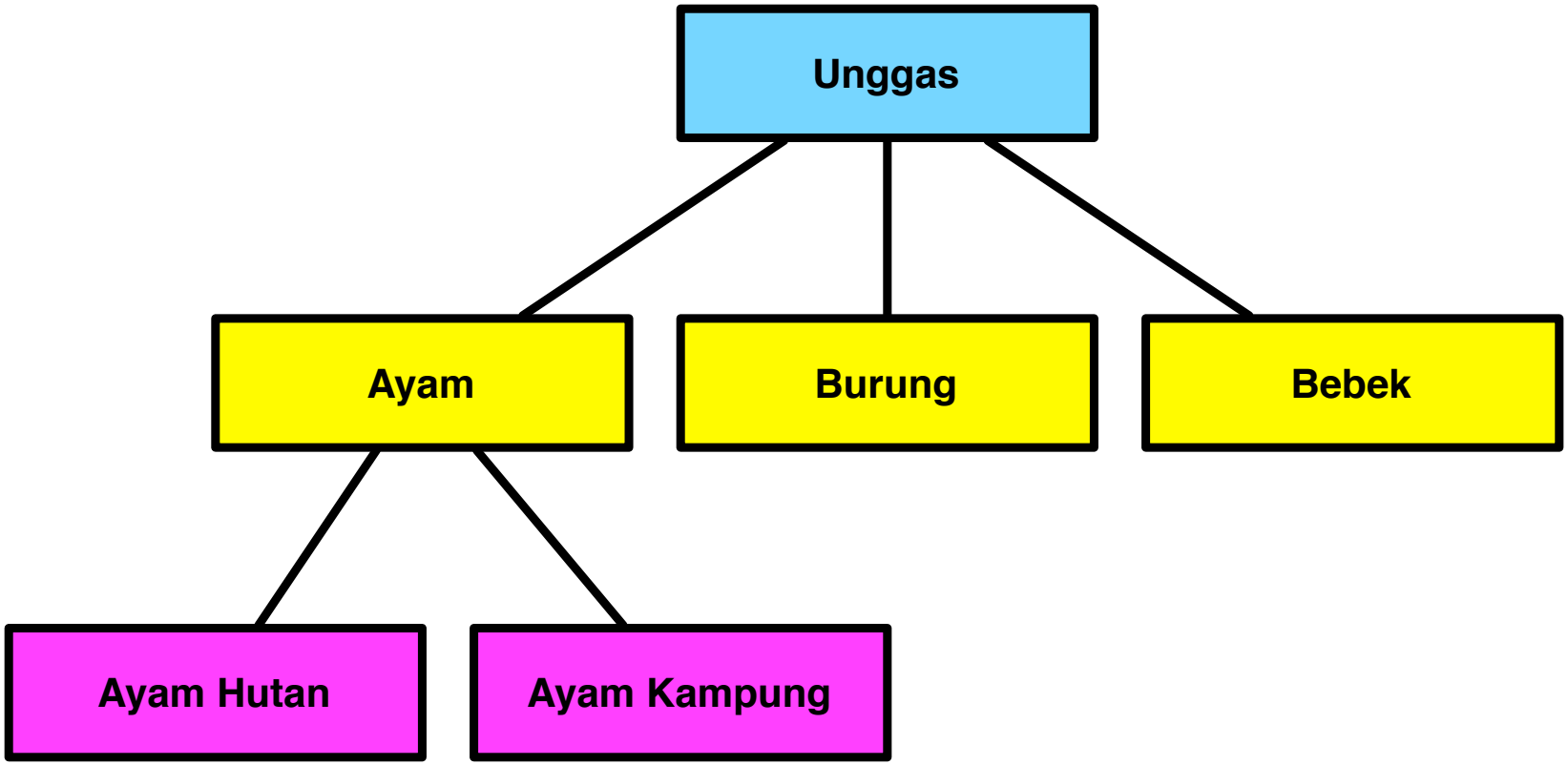
# Pengertian Pewarisan (Inheritance)

- **Pewarisan** atau ***inheritance*** pada pemograman berorientasi objek merupakan suatu hubungan dua buah kelas atau lebih dimana ada kelas yang memiliki atribut dan metode yang sama dengan kelas lainnya berserta atribut dan metode tambahan yang merupakan sifat kelas khusus kelas yang menjadi turunannya
- Kelas yang merupakan kelas turunan biasa disebut kelas anak (*subclass*) dan kelas yang menjadi dasar penurunan adalah kelas orang tua (*superclass*)
- Pewarisan (inheritance) adalah kemampuan class untuk menurunkan sifat-sifat ke class turunannya.
- Dalam inheritance terjadi pewarisan properti (ciri) dan perilaku (behaviour) dari kelas induk ke kelas anak. Misalnya kelas turunan dapat menggunakan properti dan method yang ada pada class induknya.
- Semua properti yang dimiliki oleh kelas induk akan dimiliki oleh kelas anak dengan sendirinya sehingga terjadi “Code Sharing” antara class induk dan class anak.
- Kelas yang dijadikan kelas induk merupakan kelas yang general sedangkan kelas anak merupakan kelas induk yang memiliki sesuatu yang spesifik.

# Inheritance



# Contoh Pewarisan (1)



# Contoh Pewarisan (2)

- Sebagai kelas dasar adalah Unggas
  - Salah satu sifat Unggas adalah bertelur dan bersayap
- Kelas turunan pertama adalah Ayam, Burung dan Bebek
  - Tiga kelas turunan ini mewarisi sifat kelas dasar Unggas yaitu bertelur dan bersayap
  - Selain mewarisi sifat kelas dasar, masing-masing kelas turunan mempunyai sifat khusus, Ayam berkokok, Burung terbang dan bebek berenang.
- Kelas Ayam mempunyai kelas turunan yaitu Ayam Kampung dan Aram Hutan
- Dua kelas ini mewarisi sifat kelas Ayam yang berkokok. Tetapi dua kelas ini juga punya sifat yang berbeda, yaitu:
  - Ayam kampung berkokok panjang halus
  - Ayam Hutan berkokok pendek dan kasar

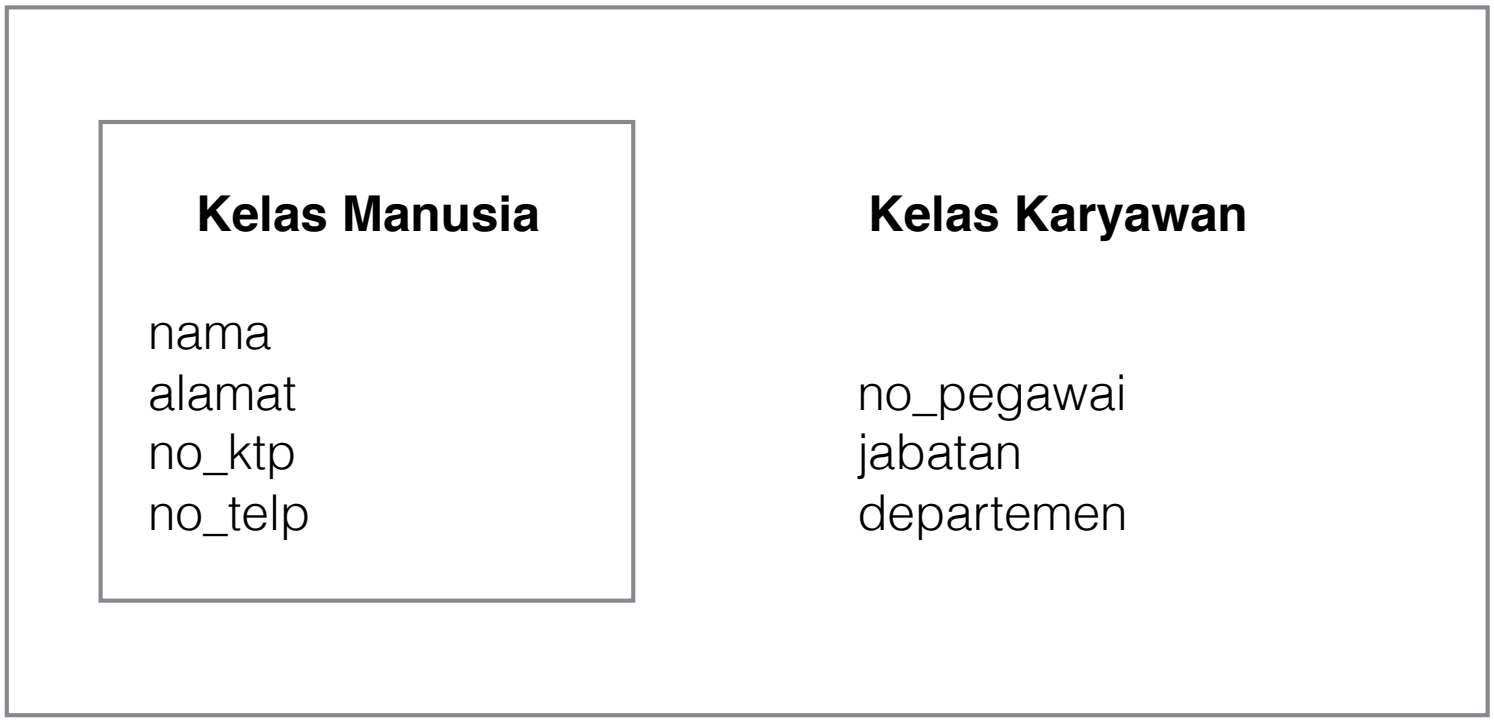
- Contoh kelas Manusia dan kelas Karyawan

**Kelas Manusia**

nama
alamat
no_ktp
no_telp

**Kelas Karyawan**

nama
alamat
no_ktp
no_telp
no_pegawai
jabatan
departemen


Kelas Manusia
nama
alamat
no_ktp
no_telp
Kelas Karyawan
no_pegawai
jabatan
departemen

# Access Identifier

- Access identifier berfungsi untuk menentukan siapa saja yang dapat mengakses (baik membaca atau mengubah) anggota dari sebuah class.
- Ada beberapa macam access identifier:
  - public (+)
  - private (-)
  - protected (#)

- **Public** : menyatakan bahwa anggota class tersebut boleh diakses oleh siapa saja (method/property dari class lain)
- **Private** : menyatakan bahwa anggota class tersebut hanya dapat diakses oleh dirinya sendiri (method/property dari class yang sama)
- **Protected** : menyatakan sesuatu yang sama dapat memiliki berbagai bentuk dan perilaku yang berbeda

# Pembatasan Inheritance

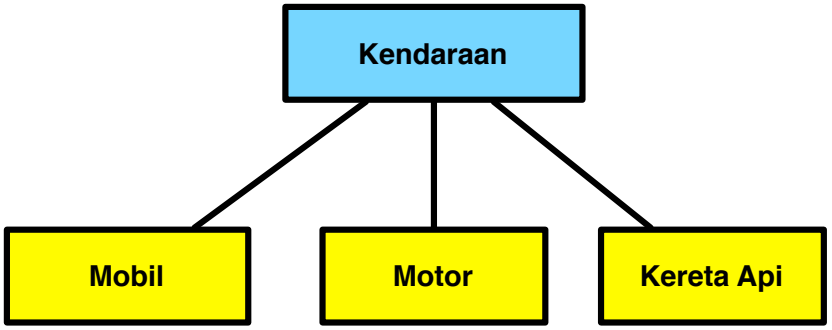
- Data dan fungsi yang dapat diwariskan hanya yang bersifat **public** dan **protected**
- Untuk data dan fungsi private tetap tidak dapat diwariskan. Hal ini disebabkan sifat **private** yang hanya dapat diakses dari dalam kelas saja.
- Sifat pewarisan ini menyebabkan kelas-kelas dalam pemograman berorientasi objek membentu hirarki kelas mulai dari kelas dasar, kelas turunan pertama, kelas turunan kedua, dan seterusnya.

- Struktur pewarisan:

```
<access_modifier> class <nama_class_turunan> : <nama_class_induk>
```

- Berdasarkan struktur pewarisan diatas, pembuatan class melalui inheritance pada C# menggunakan operator "**:**".

# Contoh




```
class kendaraan
{
  …
}
   class mobil:kendaraan
   {
     …
   }
   class motor:kendaraan
   {
     …
   }
   class kereta_api:kendaraan
   {
     …
   }
```

# Contoh Program Inheritance

```
class Mamalia
{
  public int umur;
  public String jenisMakanan;
  public void menyusui()
  {
    Console.WriteLine("Menyusui anaknya”);
  }
  public void melahirkan()
  {
    Console.WriteLine("Melahirkan anak”);
  }
}
  class Paus : Mamalia
  {
    public void berenang()
    {
      Console.WriteLine(“Berenang...");
    }
  }
```

# Interface

- Interface merupakan bentuk kelas khusus dimana kelas tersebut tidak memiliki kode program maupun data. Dalam interface hanya terdapat deklarasi method tapi tidak ada implementasi kodenya. Oleh karena tidak memiliki data dan implementasi kode, objek tidak bisa diciptakan dari interface.
- Sebuah interface dibuat dengan menggunakan keyword interface. Beberapa hal yang perlu diperhatikan dalam menulis interface adalah sebagai berikut:
    1. Interface tidak boleh diturunkan dari kelas atau struct lain.
    2. Interface hanya mendeklarasikan method. Properti tidak boleh didelarasikan dalam interface. Begitu juga dengan tipe data seperti enumerations, kelas lain, atau struct.
    3. Method yang dideklarasikan dalam interface tidak boleh memiliki akses level. Hal ini dikarenakan semua method yang dideklarasikan dalam interface memiliki akses level public.
    4. Interface tidak boleh mendeklarasikan constructor dan destructor.

- Contoh deklarasi interface:

```
interface IBidangDatar
{
   String GetID();
   double HitungLuas();
}
```

Pada contoh diatas dideklarasikan sebuah interface bernama IbidangDatar. Dalam interface tersebut memiliki 2 buah method yaitu GetID dan HitungLuas. Perhatikan deklarasi method tersebut, dalam deklarasi method tersebut tidak dideklarasikan akses level method. Hal ini bukan berarti hak akses method tersebut mengikuti akses level default yaitu private akan tetap akses level kedua method tersebut sudah ditetapkan public. Dalam interface diatas, method yang dideklarasikan tidak memiliki implementasi. Hanya deklarasi tipe data, nama method dan daftar argumen jika ada.

- ***Saran untuk pendefinisian interface dimulai dengan "I". Method pada interface tidak perlu menggunakan acces modifier karena semua method akan bersifat public.***

```
interface iweapon
{
  string shoot();
  string reload();
}
  class shotgun : iweapon
  {
    string iweapon.shoot()
    {
    }
  ....
  string iweapon.reload()
    {
    }
      ...
  }
```

- Objek tidak bisa diciptakan dari sebuah interface. Kegunaan interface memang bukan sebagai “cetakan” untuk membuat objek akan tetapi lebih berfungsi sebagai “kontrak”. Interface harus diimplementasikan oleh kelas lain agar dapat berguna.
- Kelas lain yang mengimplementasikan sebuah interface harus mengimplementasikan seluruh method yang dideklarasikan dalam interface tersebut. Error akan muncul jika ada sebuah method yang dideklarasikan dalam interface tidak didefinisikan pada kelas yang mengimplementasikan interface tersebut.
- Sebuah kelas bisa mengimplementasikan lebih dari satu interface. Jika sebuah kelas mengimplementasikan lebih dari satu interface, maka setiap method yang ada dalam interfaceyang diimplementasikan harus didefinisikan dalam kelas yang mengimplementasikan interface tersebut. Hal ini berbeda dengan inheritance dimana pada inheritance sebuah kelas hanya boleh diturunkan dari 1 kelas lain, tidak boleh lebih

- Contoh penggunaan interface dapat dilihat pada contoh berikut ini.

```
class PersegiPanjang : IBidangDatar
{
    public String ID;
    public double panjang;
    public double lebar;
    public String GetID()
    {
        return ID;
    }
        public double HitungLuas(){
        double luas = panjang * lebar;
        return luas;
    }
}
```

# Abstract Class

- Kelas abstrak merupakan kelas yang didefinisikan tidak lengkap. Sama halnya dengan interface, kelas abstrak tidak bisa digunakan untuk membuat objek. Untuk membuat objek, kelas abstrak harus diturunkan terlebih dahulu. Namun berbeda dengan interface, pada kelas abstrak bisa terdapat deklarasi properti dan method. Secara umum, kelas abstrak sama seperti kelas biasa, namun ada bagian-bagian tertentu yang dibiarkan kosong untuk dikustomisasi oleh kelas turunannya.
- Kelas abstrak dibuat dengan menggunakan kata kunci abstract. Sebuah kelas abstrak bisa dibuat dari sebuah kelas yang diturunkan dari kelas lainnya.

- Berikut ini adalah contoh kelas abstrak.

```
abstract class BidangDatar
{
   public String ID;
   public double panjang;
   public double lebar;

   public String GetID()
   {
   return ID;
   }
   abstract public double HitungLuas( );
}
```

Pada contoh diatas terdapat kelas abstrak BidangDatar. Dalam kelas abstrak tersebut terdapat sebuah method abstrak bernama HitungLuas(). Method abstrak ini merupakan method yang harus didefinisikan dalam kelas turunan BidangDatar tersebut.

- Contoh implementasi kelas abstrak dapat dilihat pada kode dibawah ini.

```
class Persegi : BidangDatar
{
   public override double HitungLuas()
    {
     return panjang * lebar;
    }
}
```

- Method HitungLuas() yang mendefinisikan abstract method HitungLuas() pada kelas abstrak BidangDatar harus menggunakan keyword override.

# Overriding

- Pada pertemuan sebelumnya telah dibahas tentang polymorphisme. Salah satu bentuk polymorphisme adalah overriding. Overriding berarti mendefinisikan ulang method yang terdapat dalam kelas induk. Method yang bisa dioverriding hanyalah method yang memiliki keyword virtual di kelas induknya dan method yang meng-overriding harus menggunakan keyword override
- Overriding ini sering terjadi saat implementasi method pada kelas induk tidak sesuai dengan perilaku kelas anaknya. Misalnya saja kelas Mamalia. Asumsikan saja kelas Mamalia memiliki method pindah. Pada umumnya setiap Mamalia pindah tempat dengan cara berjalan atau berlari. Namun Paus memiliki cara yang berbeda dan tidak memungkinkan untuk berjalan atau berlari. Kelas Paus tidak mungkin menggunakan method pindah yang diwariskan kelas induknya kepadanya karena cara berpindah kelas Paus berbeda dengan kelas Mamalia pada umumnya. Oleh karena itu method pindah yang diwariskan dari induknya perlu didefinisikan ulang agar sesuai dengan kondisi kelas Paus.

- Contoh implementasi dari kasus diatas adalah sebagai berikut

```
Class Mamalia
{
  public virtual void pindah()
  {
    Console.WriteLine("Lari...");
  }
}
class Kucing : Mamalia
  {
  }
class Paus : Mamalia
  {
    public override void pindah(){
    Console.WriteLine(“Berenang…”);
  }
}
```

Pada contoh diatas dapat dilihat ada 3 kelas yaitu Mamalia, Kucing dan Paus. Kelas Kucing dan Paus merupakan turunan dari kelas Mamalia. Oleh karena method pindahnya kelas Paus memiliki metode yang berbeda dengan dengan metode pindah pada kelas induknya, maka pada kelas Paus perlu didefinisikan kembali metode pindah yang dimilikinya.

# Perbedaan Interface dan Abstract Class

- Cara penggunaan Abstract Class adalah diturunkan, sedangkan cara penggunaan Interface adalah diimplementasikan.
- Pada C# lambang untuk menurunkan class dan menggunakan interfaces sama-sama menggunakan tanda ":" (titik dua). Tetapi untuk membedakan biasanya nama interface diawali dengan huruf I di depan misal: IEnumerable, IDisposable, dll.
- Sebuah class hanya dapat diturunkan dari satu abstract class tapi dapat menggunakan lebih dari satu interfaces.
- Method dan member variable pada abstract class boleh sudah ada isinya, sedangkan pada interfaces semua belum ada implementasinya.
- Pada Abstract Class semua method / member variable yang abstract harus diimplementasikan di class turunannya.
- Pada Interface semua member variable dan method harus diimplementasikan di class yang menggunakan interface tersebut.
- Access Modifier pada method dan member variable di Interface secara implisit adalah public.

# Sealed Class dan Sealed Method

- Sealed class merupakan class yang tidak dapat diturunkan/diwariskan. Pada sealed class terdapat sealed method yang juga tidak dapat diturunkan. Penguncian kelas dilakukan dengan menggunakan keyword sealed seperti contoh dibawah ini.

```
sealed class armor
{
   sealed string buy();
}
```

- Penggunaan keyword sealed tersebut mengunci kelas yang bersangkutan untuk tidak bisa mewariskan properti dan method-nya ke kelas lain. Selain berlaku terhadap kelas, keyword sealed juga berlaku untuk method. Method yang menggunakan kata kunci sealed tidak akan diwariskan ke kelas turunannya.