



UNIVERSITAS IPWIJA

Pertemuan Ke 10

Pengenalan OOP

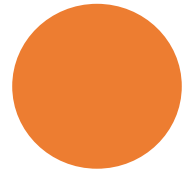
Muhamad Maulana Rachman , S.Kom., M.Kom





Materi yang disampaikan :

- Pengenalan OOP
- Pewarisan / Inheritance
- Polymorphism





Pengenalan OOP

• Pengantar OOP

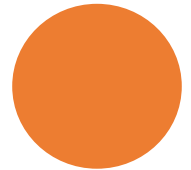
Secara umum, sebuah program komputer terdiri atas kode dan data yang dapat berupa variable maupun konstanta. Kode dan data tersebut kemudian diatur sehingga dapat bekerja sama untuk menghasilkan program keluaran yang akan digunakan untuk menyelesaikan sebuah permasalahan.

Pada model pemrograman prosedural alur kerja program berorientasi pada process (Process-Oriented)

Dalam pemrograman berorientasi objek, pendekatan yang dilakukan adalah dengan memodelkan sistem menjadi objek-objek. Objek dapat didefinisikan sebagai suatu entitas yang memiliki data dan method.

Object Oriented Programming (OOP) atau yang lebih dikenal dengan pemrograman berorientasi objek adalah suatu Teknik atau pendekatan baru dalam dunia pemrograman.

Selanjutnya ketika OOP ternyata lebih mampu menyelesaikan masalah daripada teknik prosedural, sebagian besar programmer mulai menggunakan teknik OOP.





Pengenalan OOP

• Pengantar OOP

Dalam prosedural programming kebanyakan fungsi dalam sebuah program ditulis dalam beberapa modul atau dapat lebih dari satu modul tergantung dari jenis aplikasi yang dibuat.

Dengan teknik OOP, Kalian tidak perlu mengubah keseluruhan program yang bermasalah tersebut. Anda cukup mengubah coding program yang bermasalah saja.

Beberapa hal yang merupakan keuntungan dari konsep pemrograman berbasis object (PBO/ OOP) adalah:

- Objek-objek yang dibuat bersifat reusable, sehingga dapat digunakan untuk program-program lain.
- Struktur program lebih jelas, trackable (kesalahan mudah dilacak), dan mudah untuk dikembangkan



Pengenalan OOP

- **Pengertian Class**

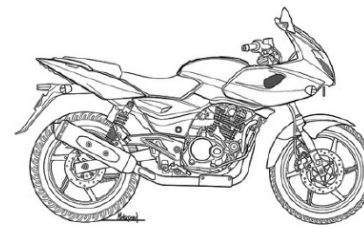
Class adalah rancangan/ sketsa/ blueprint dari sebuah objek. Sebelum kita dapat membuat sebuah objek maka kita harus membuat rancangannya terlebih dahulu.

Secara umum class memiliki dua macam anggota yaitu field dan method.

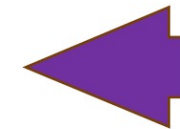
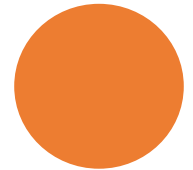
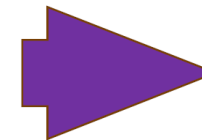
Field dapat diartikan sebagai atribut dari object, sedangkan method dapat diartikan sebagai aksi/ tindakan yang dapat dilakukan oleh sebuah object

Contohnya:

“jika sebuah perusahaan ingin membuat motor keluaran terbaru, maka sebelumnya perusahaan tersebut harus membuat rancangannya terlebih dahulu, rancangan tersebut bisa berupa gambar/ sketsa.”



OBJEK



CLASS

Pengenalan OOP

- **Main Class (Kelas Utama)**

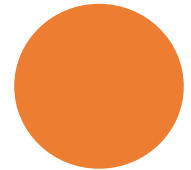
Main class adalah class yang didalamnya terdapat main method atau program utama, yang mana pada saat pertamakali program di compile dan dijlkn, mak compiler kan menerjemahkan coding yang ada pada main class. Bentuk main class pada microsoft visual studio 2022 adalah sebagai berikut.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Latihan
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Main class dalam C# selalu bernama Class Program

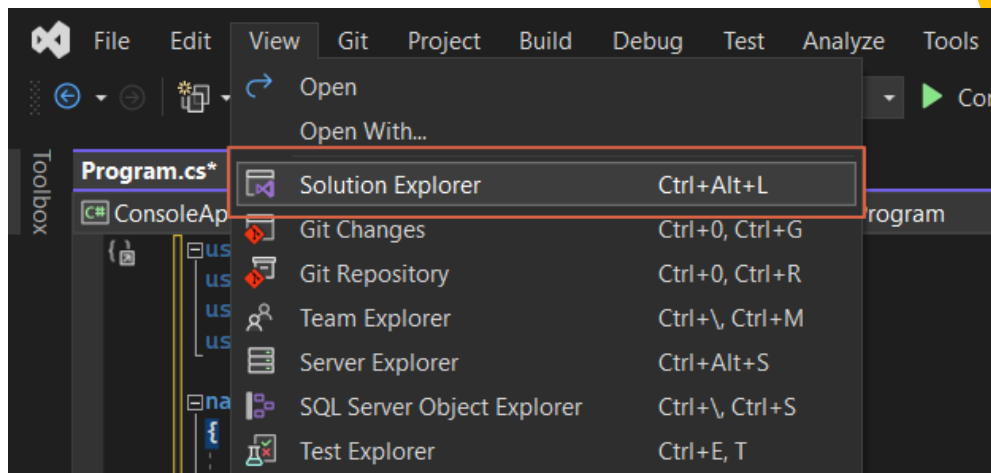
Main Method



Pengenalan OOP

- **Cara Membuat Kelas Baru**

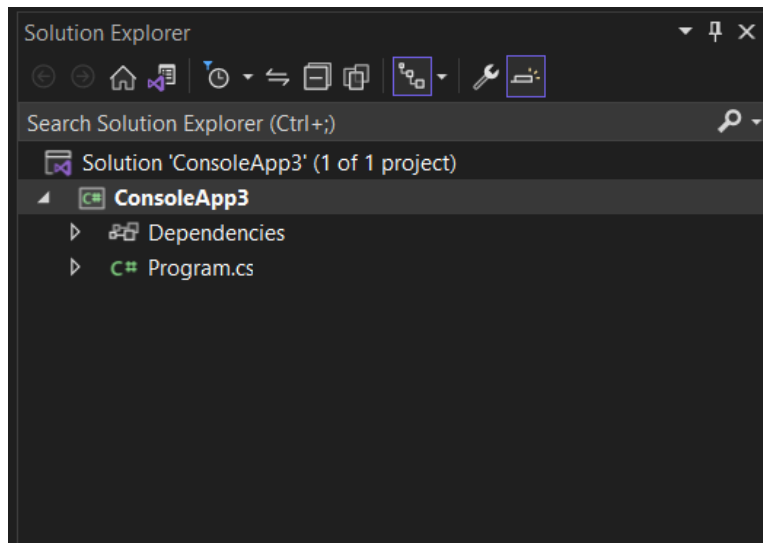
1. Klik Menu View > Pilih Solution Explorer
Note : Jika Solution Explorer tidak muncul



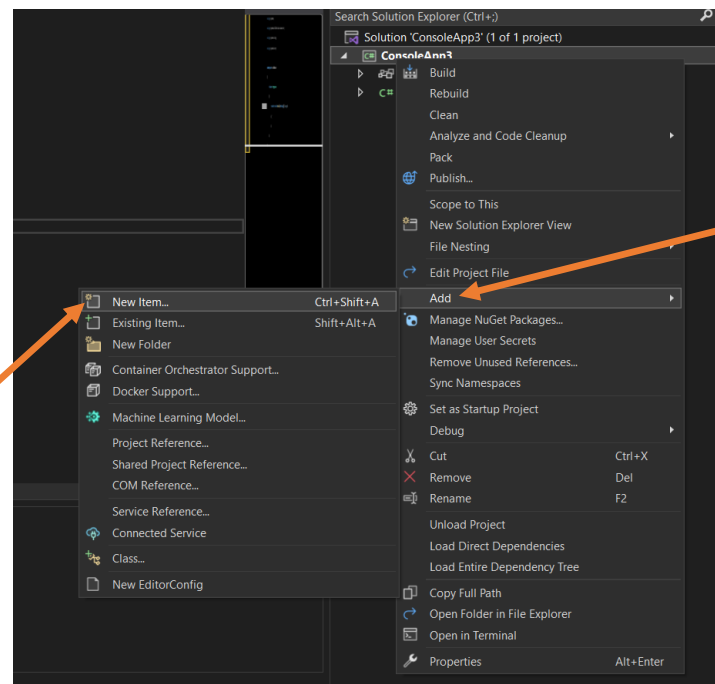
Pengenalan OOP

- **Cara Membuat Kelas Baru**

2. Jendela Solution Explorer akan muncul disebelah kanan



3. Klik Kanan pada nama project kalian lalu pilih add, pilih new item

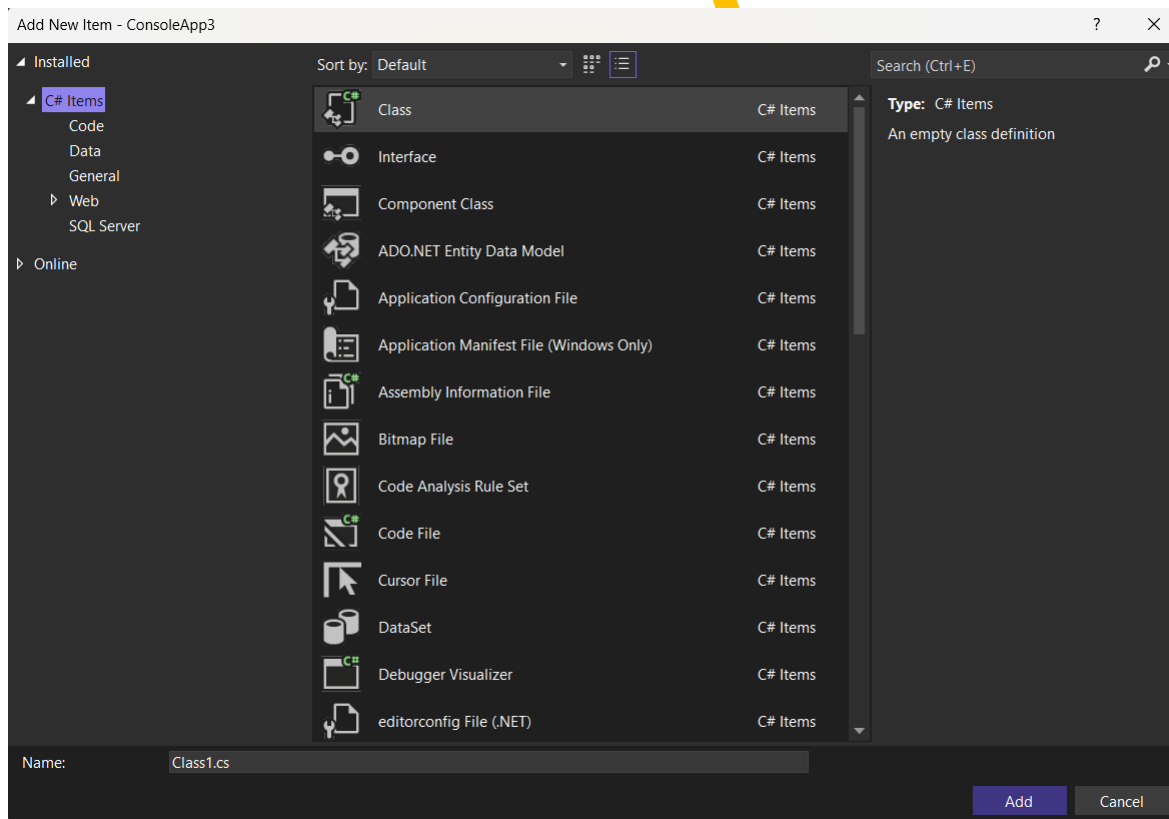


Pengenalan OOP



- **Cara Membuat Kelas Baru**

4. Muncul jendela Add New Item, Pilih Class, beri nama class pada textbox name, lalu klik add

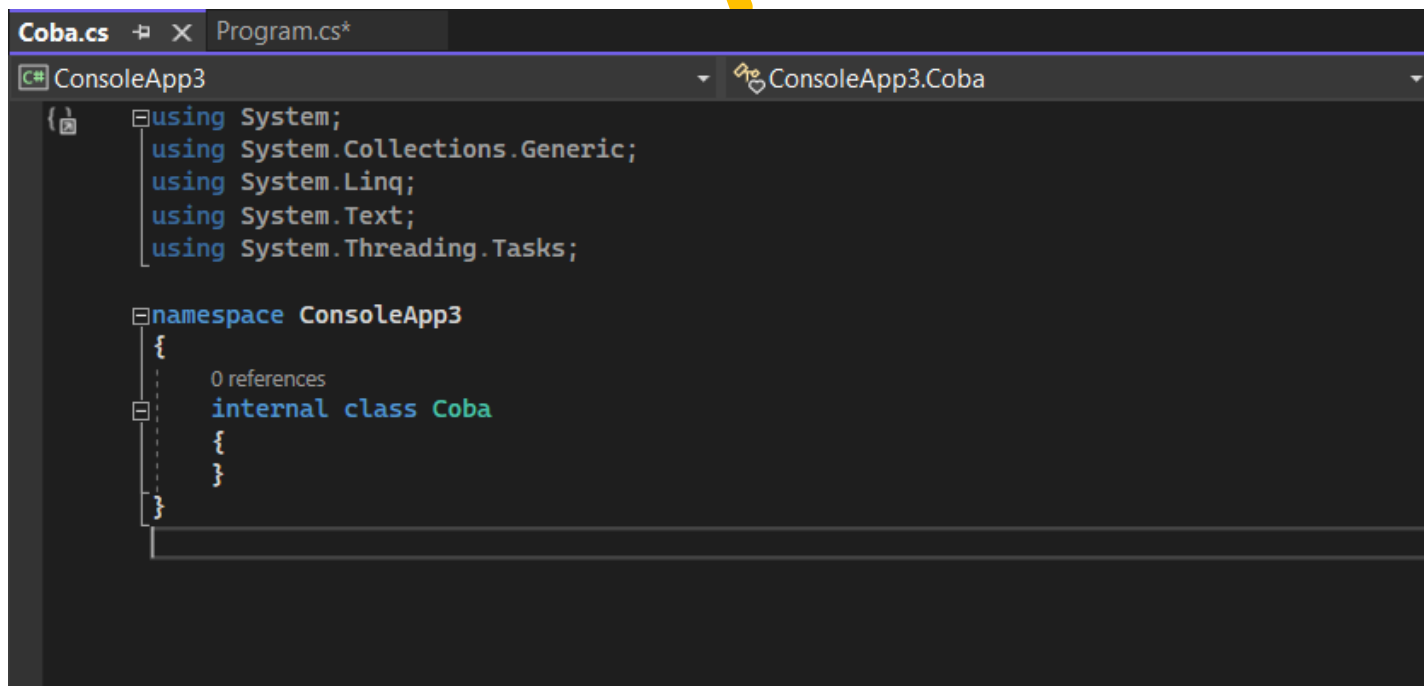


Pengenalan OOP



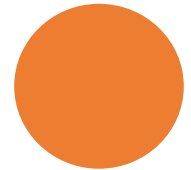
- **Cara Membuat Kelas Baru**

5. Kalian akan melihat terdapat satu kelas baru dengan nama class yang telah kalian buat



```
Coba.cs Program.cs*
ConsoleApp3
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp3
{
    0 references
    internal class Coba
    {
    }
}
```



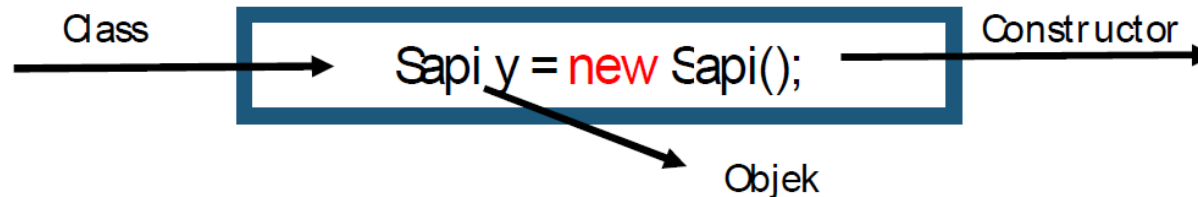
Pengenalan OOP

• Object

Seperti yang telah dibahas pada sub bab class, class merupakan sketsa/ blue print dari sebuah objek, dengan begitu object adalah realisasi dari sebuah object. Dalam kehidupan nyata object menurut Kamus Meriam Webster adalah suatu material yang dapat dirasakan oleh panca indera. Sedangkan objek didalam software Buat (software object) adalah konsep software yang dibundel bersama-sama. Terdiri dari data dan fungsi.

Instansiasi Objek

Instansiasi objek adalah proses pembuatan objek software didalam pemrograman berorientasi objek. Berikut ini adalah cara menginstantiasi objek



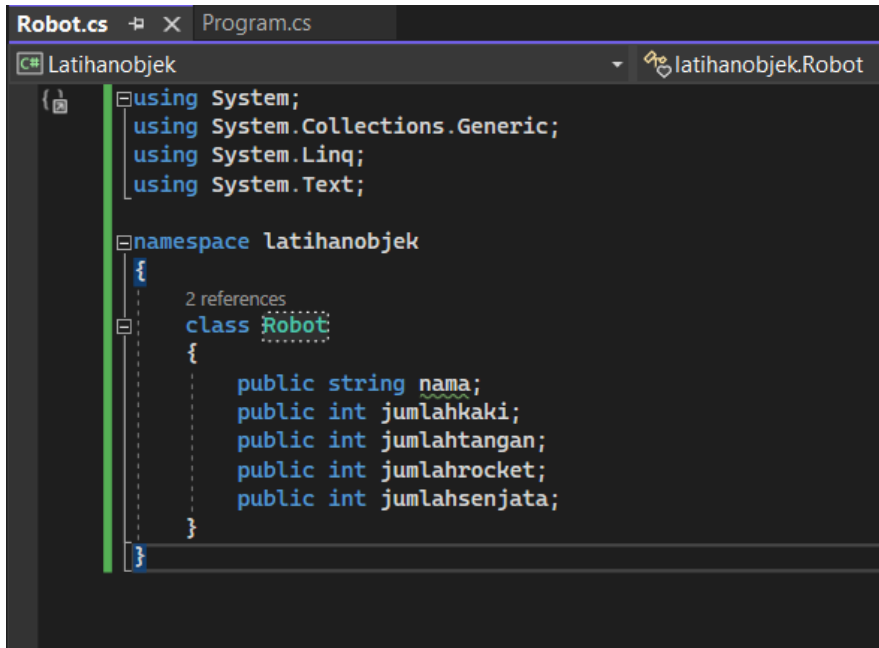
Constructor adalah method khusus yang didefinisikan didalam class dan akan dipanggil secara otomatis tiap kali terjadi instantiasi object.



Pengenalan OOP

• Contoh penggunaan object

1. Buatlah sebuah project baru
2. Beri nama project Latihanobjek
3. Beri nam class dengan nama "Robot.cs" lalu berikan coding berikut



```
Robot.cs Program.cs
C# Latihanobjek latihanobjek.Robot
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

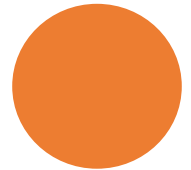
namespace latihanobjek
{
    2 references
    class Robot
    {
        public string nama;
        public int jumlahkaki;
        public int jumlahtangan;
        public int jumlahrocket;
        public int jumlahsenjata;
    }
}
```

Instantiasi object pada program.cs

```
namespace latihanobjek
{
    class program
    {
        static void Main(string[] args)
        {
            Robot robot1 = new Robot();

            robot1.nama = "Astroboy";
            robot1.jumlahkaki = 2;
            robot1.jumlahtangan = 2;
            robot1.jumlahrocket = 8;
            robot1.jumlahsenjata = 2;

            Console.WriteLine("\n Nama Robot adalah {0}", robot1.nama);
            Console.WriteLine(" Jumlah kakinya ada {0}", robot1.jumlahkaki);
            Console.WriteLine(" Jumlah tangannya ada {0}", robot1.jumlahtangan);
            Console.WriteLine(" Jumlah rocket ada {0}", robot1.jumlahrocket);
            Console.WriteLine(" jumlah senjata ada {0}", robot1.jumlahsenjata);
        }
    }
}
```





Pengenalan OOP

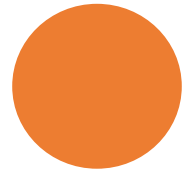
- **Method**

Pada pemrograman berorientasi object pun terdapat fungsi/ function yang lebih sering disebut dengan method. Pada pertemuan sebelumnya kita telah banyak menggunakan method seperti `static void main(String[] args)` juga merupakan method yang disebut sebagai main method.

Main method adalah sebuah method yang akan dijalankan paling pertama pada saat program dijalankan.

Jika kita membuat suatu aplikasi dengan Console Application, secara default akan terdapat method Static Void Main untuk menjalankan aplikasi.

Method bertipe void berarti bahwa method tersebut tidak dapat mengembalikan nilai apapun pada method tersebut. Tetapi jika pada suatu method tidak terdapat void maka method tersebut harus mengembalikan sebuah nilai.



Pengenalan OOP

- **Tambahkan kode program berikut ini pada class Robot.cs**

```
namespace latihanobjek
{
    2 references
    class Robot
    {
        //Attribut / variable
        public string nama;
        public int jumlahkaki;
        public int jumlahtangan;
        public int jumlahrocket;
        public int jumlahsenjata;

        //Method
        1 reference
        public void Jalan()
        {
            Console.WriteLine("\n {0} sedang jalan", this.nama);
        }

        1 reference
        public void MengeluarkanSenjata()
        {
            Console.WriteLine(" {0} Mengeluarkan senjata ", this.nama);
        }

        1 reference
        public void Terbang()
        {
            Console.WriteLine(" {0} Mengeluarkan rocket ", this.nama);
        }
    }
}
```



Pengenalan OOP

- **Mengakses method pada Program.cs**

```
namespace latihanobjek
{
    0 references
    class program
    {
        0 references
        static void Main(string[] args)
        {
            //Instansi objek
            Robot robot1 = new Robot();

            //Mengakses Attribut
            robot1.nama = "Astroboy";
            robot1.jumlahkaki = 2;
            robot1.jumlahtangan = 2;
            robot1.jumlahrocket = 8;
            robot1.jumlahsenjata = 2;

            //Display
            Console.WriteLine("\n Nama Robot adalah {0}", robot1.nama);
            Console.WriteLine(" Jumlah kakinya ada {0}", robot1.jumlahkaki);
            Console.WriteLine(" Jumlah tangannya ada {0}", robot1.jumlahtangan);
            Console.WriteLine(" Jumlah rocket ada {0}", robot1.jumlahrocket);
            Console.WriteLine(" jumlah senjata ada {0}", robot1.jumlahsenjata);

            //Mengakses Method
            robot1.Jalan();
            robot1.MengelurakanSenjara();
            robot1.Terbang();

            Console.Read();
        }
    }
}
```





Pengenalan OOP

- **Encapsulation**

Enkapsulasi adalah sebuah metode untuk menyembunyikan elemen tertentu dari sebuah class. Fitur ini penting karena atribut yang ada didalam class merupakan informasi penting yang terkadang harus disembunyikan agar tidak dapat diakses secara langsung oleh user.

Sebagai contoh misalnya :

Programmer membuat sebuah class Bernama "Sapi" didalam class sapi terdapat atribut "public int umur" ini artinya atribut umur dapat diakses secara public dengan artian dapat diakses oleh siapa saja dengan syarat data yang dimasukkan harus berupa integer(bilangan bulat).

Bagaimana jika seorang user memasukkan umur sapi tersebut dengan bilangan negatif (-1),.



Pengenalan OOP

- **Access Identifier**

Access Identifier berfungsi untuk menentukan siapa saja yang dapat mengakses (membaca/ mengubah) data-data (attribut & method) didalam sebuah class. Access Identifier yang sering digunakan dalam pemrograman bahasa C# ada 3 yaitu:

- a. Public**


Menyatakan bahwa anggota class tersebut (attribut/ method/ property) boleh diakses oleh siapa saja (class yang lain).

- b. Private**

Menyatakan bahwa anggota class tersebut (attribut/ method/ property) hanya boleh diakses oleh dirinya sendiri (class itu sendiri).

- c. Protected**

Menyatakan bahwa anggota class tersebut (attribut/ method/ property) hanya boleh diakses oleh dirinya sendiri dan turunan-turunan classnya (class yang menurunkan sifat-sifat dari class tersebut).



```
using System.Text;

namespace latihanobjek
{
    2 references
    class Robot
    {
        //Attribut / variable
        public string nama;
        private int jumlahkaki;
        private int jumlahtangan;
        private int jumlahrocket;
        private int jumlahsenjata;

        //Method
        1 reference
        public void Jalan()
        {
            Console.WriteLine("\n {0} sedang jalan", this.nama);
        }

        1 reference
        public void MengeluarkanSenjata()
        {
            Console.WriteLine(" {0} Mengeluarkan senjata ", this.nama);
        }

        1 reference
        public void Terbang()
        {
            Console.WriteLine(" {0} Mengeluarkan rocket ", this.nama);
        }
    }
}
```

Pengenalan OOP

- **Sekarang tambahkan kode program yang ada pada Robot.cs**

```
class Robot
{
    //Attribute / variable
    public string nama;
    private int jumlahkaki=2;
    private int jumahtangan=2;
    private int jumlahrocket=2;
    private int jumlahsenjata=8;

    //Method untuk membuat attribut bertipe private terdisplay
    public int lihatkaki()
    {
        return this.jumlahkaki;
    }
    public int lihatsenjata()
    {
        return this.jumlahsenjata;
    }
    public int lihatrocket()
    {
        return this.jumlahrocket;
    }
    public int lihattangan()
    {
        return this.jumahtangan;
    }
}
```

- **Ubah Juga kode program yang ada pada "Program.cs"**

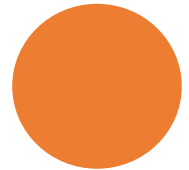
```
class Program
{
    static void Main(string[] args)
    {
        //Instantiasi objek
        Robot robot1 = new Robot();

        //Mengakses attribut
        robot1.nama = "Astroboy";

        //Display
        Console.WriteLine("\n Nama Robot adalah {0}", robot1.nama);
        Console.WriteLine(" Jumlah kaki {0} ada {1}", robot1.nama, robot1.lihatkaki());
        Console.WriteLine(" Jumlah tangan {0} ada {1}", robot1.nama, robot1.lihattangan());
        Console.WriteLine(" Jumlah Senjata {0} ada {1}", robot1.nama, robot1.lihatsenjata());
        Console.WriteLine(" Jumlah Rocket {0} ada {1}", robot1.nama, robot1.lihatrocket());

        //Mengakses method;
        robot1.Jalan();
        robot1.mengeluarkanSenjata();
        robot1.Terbang();

        Console.Read();
    }
}
```



Pengenalan OOP

- **Property**

Dalam bahasa pemrograman C#, terdapat cara lain untuk membungkus field/ atribut, yaitu dengan membuat property. Property memiliki fungsi yang sama dengan method getter/ setter. Dalam property method getter/ setter disatukan dalam sebuah property. Dengan demikian, kita tidak perlu lagi membuat 2 buah method dengan nama yang berbeda. Agar lebih jelas, mari langsung saja kita praktekan.

- **Buatlah sebuah project baru dengan nama "latihanproperty"**
- **Lalu tambahkan kelas baru dengan nama "Malware.cs"**
- **Berikan kode program berikut ini pada "Malware.cs"**

```
namespace Latihanproperty
{
    2 references
    class Malware
    {
        public string nama;
        private int size;
        private String jenis_malware;
        private String kemampuan;
    }
}
```

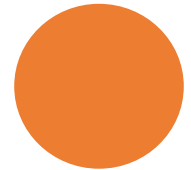
Lakukan Hal yang sama pada attribut-attribut yang lainnya

- **Membuat property**
Blok attribute lalu ctrl+R+E
Lalu apply

```
namespace Latihanproperty
{
    2 references
    class Malware
    {
        public string nama;
        private int size;
        private String jenis_malware;
        private String kemampuan;

        2 references
        public int Size { get => size; set => size = value; }
        2 references
        public string Jenis_malware { get => jenis_malware; set => jenis_malware = value; }
        2 references
        public string Kemampuan { get => kemampuan; set => kemampuan = value; }
    }
}
```

Property



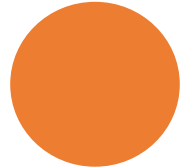
Pengenalan OOP

- **Buatlah sebuah objek pada "Program.cs" kemudian ketikkan program dibawah ini**

```
{
    class Program
    {
        static void Main(string[] args)
        {
            Malware Malware1 = new Malware();

            Malware1.nama = "BackdoorWin32.Exe";
            Malware1.Size = 14;
            Malware1.Kemampuan = "Shut Down setiap 30 menit";
            Malware1.Jenis_malware = "Trojan";

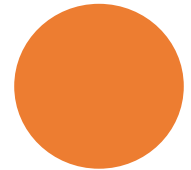
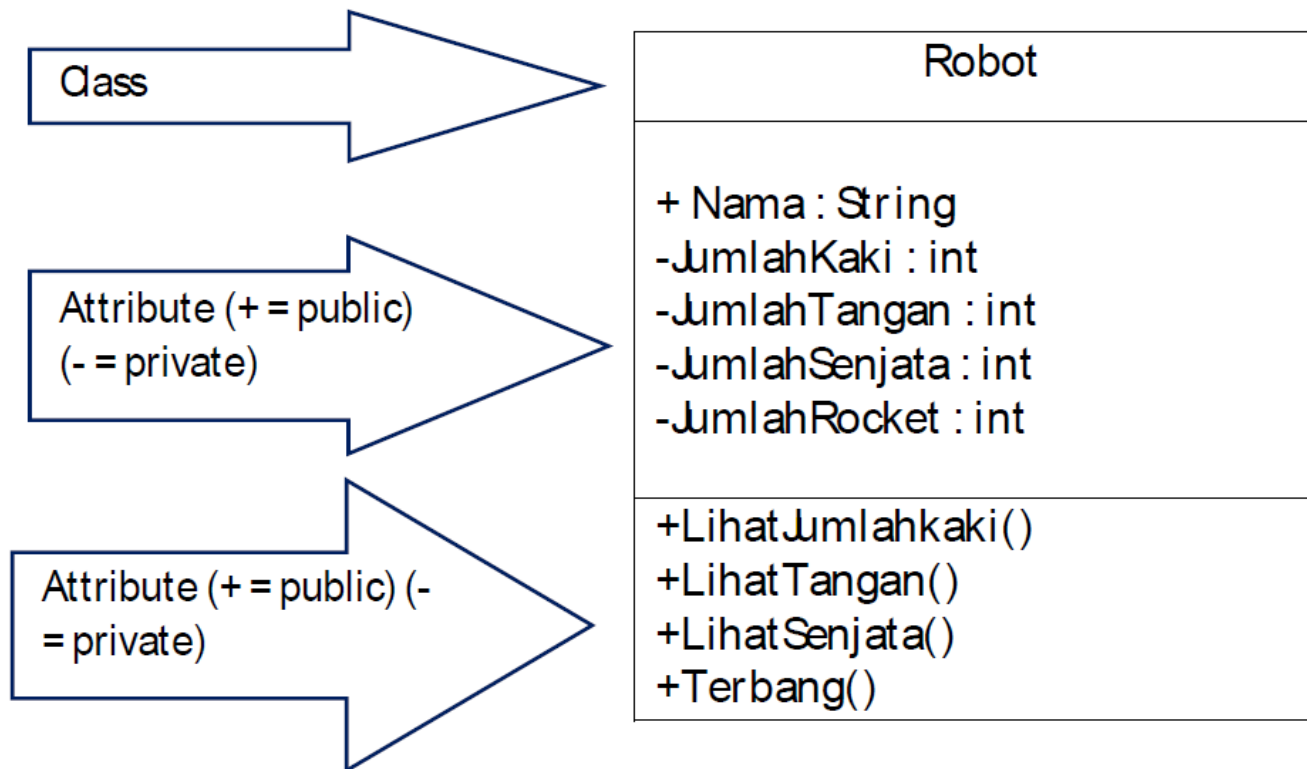
            Console.WriteLine("\n Nama malware berikut ini adalah {0}", Malware1.nama);
            Console.WriteLine(" Malware {0} termasuk jenis {1}", Malware1.nama, Malware1.Jenis_malware);
            Console.WriteLine(" Malware {0} berkapasitas {1}Mb ", Malware1.nama, Malware1.Size);
            Console.WriteLine(" Kemampuan {0} adalah {1} ", Malware1.nama, Malware1.Kemampuan);
            Console.Read();
        }
    }
}
```



Pengenalan OOP

- **Hubungan antara Class, Attribute, Method**

Hubungan antara Class, Attribute , dan method biasa digambarkan dalam bentuk Class Diagram seperti contoh dibawah ini:



Pewarisan/ Inheritance

Inheritance/ Pewarisan adalah suatu cara pembuatan class baru dengan menggunakan kembali class yang sudah didefinisikan sebelumnya dengan menambahkan attribute dan method baru. Sehingga demikian class baru yang dibuat tetap memiliki attribute dan method yang dimiliki oleh class induknya. Pada konsep pewarisan/ inheritance terdapat beberapa istilah yang perlu diketahui yaitu:

- Sub Class, digunakan untuk menunjukkan class anak atau turunan secara hirarkis dari super class
- Super Class, digunakan untuk menunjukkan class induk secara hirarkis dari sub class (class anak)

Studi Kasus

Buatlah sebuah project yang berisi tiga class berikut ini:

Virus	Worm	Trojan
+ nama: String + size : int + kemampuan : String + banyak : int	+ nama : String + size : int + kemampuan : String	+ nama : String + size : int + kemampuan : String
+ Menyerang() : void + MemperbanyakDiri() : void	+ Menyerang() : void + MenginfeksiRegistry() : void + Menghapus NTLDR() : void	+ Menyerang() : void + MenyembunyikanFile() : void + MemblokirCMD() : void



Pewarisan/ Inheritance

Berikut ini adalah kode program dari bagan diatas:

Virus.cs

```
class Virus
{
    public String nama;
    public int size;
    public String kemampuan;
    public int banyak;

    public void menyerang()
    {
        Console.WriteLine(" {0} menyerang dengan {1} ", this.nama, this.kemampuan);
    }

    public void MemperbanyakDiri()
    {
        Console.WriteLine(" {0} memperbanyak file hingga {1} kali", this.nama, this.banyak*5);
    }
}
```





Pewarisan/ Inheritance

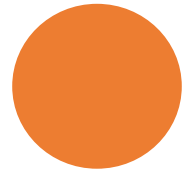
Berikut ini adalah kode program dari bagan diatas:

Trojan.cs

```
class Trojan
{
    public String nama;
    public int size;
    public String kemampuan;

    public void Menyerang()
    {
        Console.WriteLine(" {0} menyerang dengan {1} ", this.nama, this.kemampuan);
    }

    public void MenyembunyikanFile()
    {
        Console.WriteLine(" {0} Sembunyikan File yang ada di C:/Program File/ Microsoft Office", this.nama);
    }
    public void MemblokirCMD()
    {
        Console.WriteLine(" {0} Blokir setiap kegiatan melalui CommandPrompt", this.nama);
    }
}
```





Pewarisan/ Inheritance

Berikut ini adalah kode program dari bagan diatas:

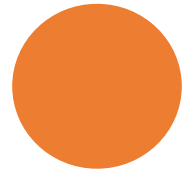
Worm.cs

```
class Worm
{
    public String nama;
    public int size;
    public String kemampuan;

    public void menyerang()
    {
        Console.WriteLine(" {0} menyerang dengan {1} ", this.nama, this.kemampuan);
    }

    public void MenginfeksiRegistry()
    {
        Console.WriteLine(" {0} Menginfeksi Registry ", this.nama);
    }

    public void MenghapusNTLDR()
    {
        Console.WriteLine(" {0} Menghapus NTLDR ", this.nama);
    }
}
```



Pewarisan/ Inheritance

Berikut ini adalah kode program dari bagan diatas:

Program.cs

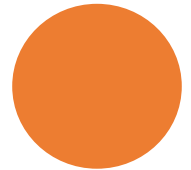
```
static void Main(string[] args)
{
    Virus virus1 = new Virus();
    Trojan Trojan1 = new Trojan();
    Worm Worm1 = new Worm();

    //pengaksesan attribut
    virus1.nama = "Sality32.exe";
    virus1.size = 32;
    virus1.kemampuan = "Menghapus File Penting";
    virus1.banyak = 5;
    Trojan1.nama = "BackdoorWin32.exe";
    Trojan1.size = 14;
    Trojan1.kemampuan = "Shutdown setiap 30 menit";
    Worm1.nama = "Brontox.exe";
    Worm1.size = 23;
    Worm1.kemampuan = "Sleep Setiap 15 menit";

    //display
    Console.WriteLine("\n Nama Virus: {0} kapasitas {1} Mb", virus1.nama, virus1.size);
    Console.WriteLine(" Nama Trojan: {0} kapasitas {1} Mb", Trojan1.nama, Trojan1.size);

    //pengaksesan method
    virus1.menyerang();
    virus1.MemperbanyakDiri();
    Trojan1.Menyerang();
    Trojan1.MenyembunyikanFile();
    Trojan1.MemblokirCMD();
    Worm1.menyerang();
    Worm1.MenginfeksiRegistry();
    Worm1.MenghapusNTLDR();

    Console.Read();
}
```



Pewarisan/ Inheritance



CLASS DIAGRAM WITH INHERITANCE

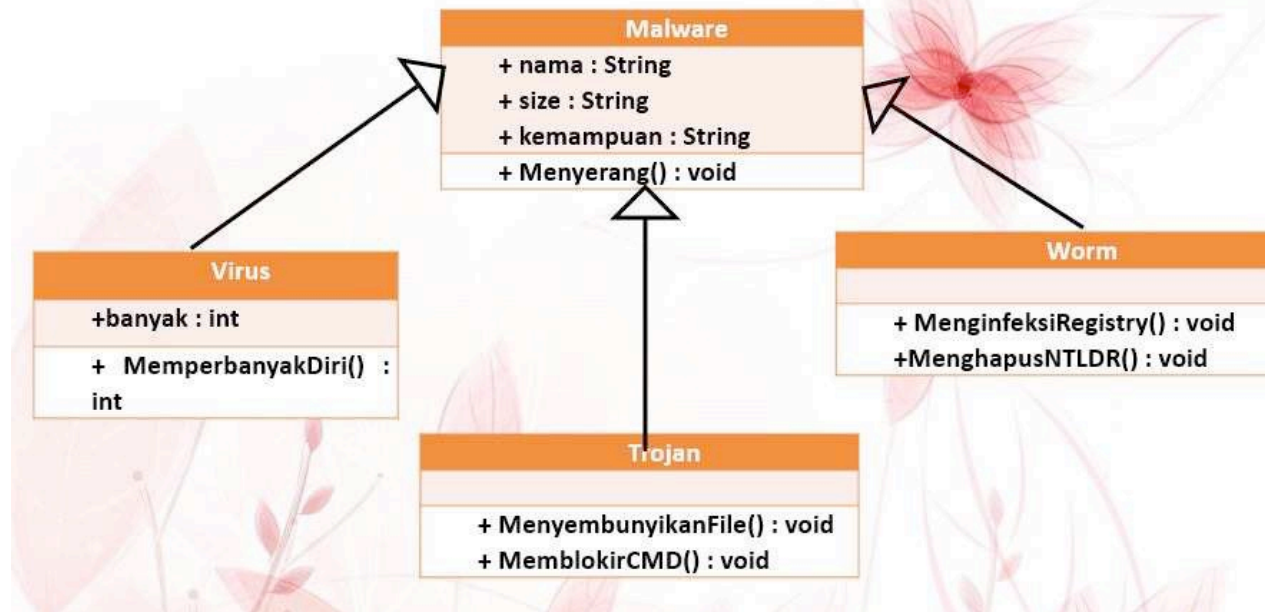


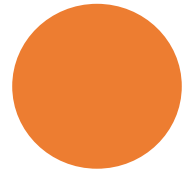
diagram diatas programmer perlu membuat empat kelas baru yaitu kelas malware, virus, trojan, worm. Mengapa lebih mudah menggunakan inheritance? Karena setiap malware (virus/ trojan/ worm) sama sama memiliki attribut nama, size, dan kemampuan, juga memiliki method Menyerang().

Pewarisan/ Inheritance

Malware.cs

```
class Malware
{
    public string nama;
    public int size;
    public string kemampuan;

    public void Menyerang()
    {
        Console.WriteLine("\n {0} menyerang dengan cara {1}", this.nama, this.kemampuan);
    }
}
```



Virus.cs

```
class Virus: Malware
{
    public int banyak;

    public void MemperbanyakDiri()
    {
        Console.WriteLine(" {0} memperbanyak diri sebanyak {1} kali", this.nama, this.banyak*5);
    }
}
```

Menandakan class virus turunan dari class malware

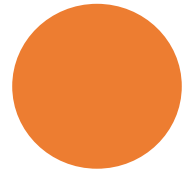


Pewarisan/ Inheritance

Torjan.cs

```
class Trojan : Malware
{
    public void menyembunyikanFile()
    {
        Console.WriteLine(" Sembunyikan file yang ada di C:/Program File/Microsoft Office");
    }
    public void memblokirCMD()
    {
        Console.WriteLine(" Blokir semua aktifitas yang menggunakan Command Prompt");
    }
}
```

Menandakan class torjan turunan dari class malware



Worm.cs

```
class Worm : Malware
{
    public void menginfeksiRegistry()
    {
        Console.WriteLine(" {0} menginfeksi registry", this.nama);
    }
    public void menghapusNTLDR()
    {
        Console.WriteLine(" {0} menghapus NT Loader", this.nama);
    }
}
```

Menandakan class worm turunan dari class malware



Pewarisan/ Inheritance

Program.cs

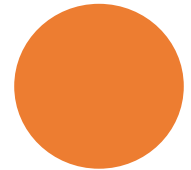
```
class Program
{
    static void Main(string[] args)
    {
        Virus virus1 = new Virus();
        Trojan trojan1 = new Trojan();
        Worm worm1 = new Worm();

        virus1.nama = "Sality32.exe";
        virus1.size = 32;
        virus1.banyak = 10;
        virus1.kemampuan = "Menghapus File Penting";
        virus1.Menyerang();
        virus1.MemperbanyakDiri();

        trojan1.nama = "BackdoorWin32.exe";
        trojan1.kemampuan = " Shutdown setiap 30 menit";
        trojan1.Menyerang();
        trojan1.menyembunyikanFile();

        worm1.nama = "Brontox.exe";
        worm1.size = 14;
        worm1.kemampuan = "Sleep Setiap 15 menit";
        worm1.Menyerang();
        worm1.menginfeksiRegistry();
        worm1.menghapusNTLDR();

        Console.Read();
    }
}
```





Virtual dan Override Methode

Dengan menggunakan konsep inheritance programmer dapat mempermudah pengerjaan program menjadi lebih simple dan efektif dengan dapat menggunakan method yang sama dengan yang ada pada parent class (super class), akan tetapi ada kalanya method yang diwarisi oleh super class pada sub class tidak sesuai dengan tempatnya.

Misalkan

saja programmer membuat class dengan nama "Hewan.cs" sebagai parent class, kemudian terdapat sub class dengan nama "Buaya.cs", dan "Banteng.cs", lalu dalam class "Hewan.cs" terdapat method menyerang(), maka secara otomatis buaya dan banteng memiliki method menyerang, akan tetapi pada realnya (pada kehidupan nyata) buaya dan banteng memiliki cara menyerang yang berbeda, lalu bagaimana cara inheritance dapat menurunkan method untuk sesuatu yang lebih spesifik lagi?

"Yaitu dengan cara membuat override method, override method yaitu menempa method yang ada pada parent class untuk diterapkan pada sub class dengan cara member awalan keyword "Override" pada nama method yang sama yang ada pada parent/ super class."



Virtual dan Override Methode

Sedangkan virtual adalah keyword yang digunakan pada method yang ada pada parent class yang nantinya akan di override melalui method yang ada di sub class. Jadi jika pada class turunan/ sub class ditambahkan keyword override, maka pada super class method yang nantinya akan di override/ ditimpa menggunakan keyword "Virtual". Berikut adalah contoh penerapan virtual dan override method.

Malware.cs

```
class Malware
{
    public string nama;
    public int size;
    public string kemampuan;

    public virtual void Menyerang()
    {
        Console.WriteLine("\n {0} menyerang dengan cara {1}", this.nama, this.kemampuan);
    }
}
```

Menandakan menyerang dapat di overried

Virus.cs

```
class Virus: Malware
{
    public int banyak;

    public void MemperbanyakDiri()
    {
        Console.WriteLine(" {0} memperbanyak diri sebanyak {1} kali", this.nama, this.banyak*5);
    }

    public override void Menyerang()
    {
        Console.WriteLine(" Virus {0} melakukan serangan ", this.nama);
    }
}
```

Menimpa void menyerang() yg ada pada parent



Polymorphism

Polimorfisme digunakan untuk menyatakan suatu nama yang merujuk pada beberapa fungsi (Sinaga, 2004). Pada polimorfisme, rujukan dapat dilakukan pada beberapa tipe objek. Hal ini dilakukan karena setiap objek dimungkinkan memiliki instruksi yang berbeda. Dalam mengimplementasikan polimorfisme, perlu diperhatikan hal-hal sebagai berikut (Rickyanto, 2005):

1. Method yang dipanggil harus melalui method superclass
2. Method yang dipanggil juga harus merupakan method yang ada pada superclass
3. Signature method harus sama baik yang ada pada superclass ataupun yang ada ada subclass
4. Method access attribute pada subclass tidak boleh lebih terbatas dari pada yang ada di super class



Overload Method

Method overloading adalah membuat dua atau lebih method yang bernama sama, namun dengan jumlah/ jenis parameter yang berbeda. Hal ini dilakukan untuk mempermudah pemakaian method yang telah dibuat. Ciri khas dari method overloading adalah berada dalam class yang sama. Sebagai contoh mari simak kode program berikut ini.

- Buatlah sebuah project baru dengan nama latihan overload
- Buatlah sebuah class baru dengan nama siswa
- Ikuti kode program berikut ini

Siswa.cs

```
class Siswa
{
    public string nim;
    public string nama;
    public int n_UTS;
    public double IPK;

    public void cetak(String NamaSiswa)
    {
        this.nama = NamaSiswa;
        Console.WriteLine("\n Namanya adalah {0} ", this.nama);
    }

    public void cetak(int nilai_uts)
    {
        this.n_UTS = nilai_uts;
        Console.WriteLine(" Nilai UTS {0} adalah {1} ", this.nama, this.n_UTS);
    }

    public void cetak(Double ipk)
    {
        this.IPK = ipk;
        Console.WriteLine(" IPK {0} adalah {1} ", this.nama, this.IPK);
    }
}
```



Overload Method

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace latihanoverloadlagi
{
    class Program
    {
        static void Main(string[] args)
        {
            Siswa Siswa1 = new Siswa();

            Siswa1.cetak("Hani Harafani");
            Siswa1.cetak(70);
            Siswa1.cetak(3.33);
            Console.Read();
        }
    }
}
```

