

MODUL V

INHERITANCE DAN POLYMORPHISM

A. TUJUAN

- ✓ Memahami konsep dasar pemrograman berorientasi objek.
- ✓ Mampu mengimplementasikan konsep *inheritance* dan *polymorphism* di dalam program.
- ✓ Mampu menyelesaikan kasus-kasus sederhana berbasis konsep *inheritance* dan *polymorphism*.

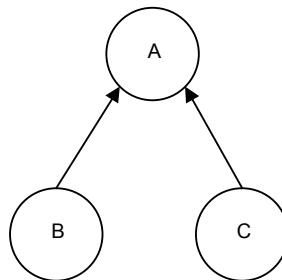
B. PETUNJUK

- Awali setiap aktivitas dengan doa, semoga berkah dan mendapat kemudahan.
- Pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar.
- Kerjakan tugas-tugas praktikum dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas.

C. DASAR TEORI

1. Pewarisan (*Inheritance*)

Istilah *inheritance* (pewarisan) mengacu pada kemampuan dari sebuah kelas untuk mewarisi *state* dan *behavior* kelas lain. Dengan demikian, atribut-atribut dan method-method kelas yang diwarisi (superkelas) secara intrinsik menjadi bagian dari kelas yang mewarisinya (subkelas). Terlepas dari warisan yang telah diperoleh, subkelas dapat menambahkan atau memodifikasi atribut-atribut dan method-method superkelas.

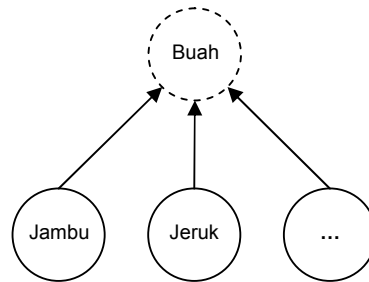


Gambar 1. Hubungan pewarisan

Konsep *inheritance* melahirkan sejumlah pasangan istilah yang menggambarkan hubungan antara dua kelas terkait, seperti superkelas-subkelas, supertipe-subtipe, kelas dasar-kelas turunan, *ancestor-descendant*, *parent-heir*, dan leluhur-turunan.

2. Kelas Abstrak

Kelas abstrak (*abstract class*) adalah kelas yang mengandung konsep abstrak, dan tidak akan pernah bisa diinstansiasi. Kelas abstrak didefinisikan dengan tujuan untuk digunakan dan diperluas oleh kelas lain. Dengan demikian, kelas ini merupakan cikal bakal superkelas.



Gambar 2. Kelas abstrak Buah

3. Interface

Interface merupakan suatu tipe abstrak yang mendefinisikan komunikasi antara dua entitas. Interface merepresentasikan sebuah kontrak, di mana kelas yang mengimplementasikan interface harus menerapkan tiap-tiap aspek interface secara nyata sebagaimana yang telah didefinisikan.

Tujuan utama penggunaan interface adalah memungkinkan kelas-kelas yang mirip untuk memiliki *behavior* standar. Jadi, interface memiliki sedikit kemiripan dengan kelas abstrak, di mana keduanya sama-sama didesain untuk digunakan oleh kelas lain.

Di balik beberapa persamaan antara interface dan kelas abstrak, terdapat perbedaan-perbedaan di antara keduanya. Mengacu pada karakteristik keduanya, setidaknya ada dua perbedaan mendasar yang bisa kita garisbawahi.

- Di dalam kelas abstrak boleh terdapat implementasi nyata dari suatu method. Keadaan ini berbeda sekali dengan interface, di mana semua method harus berupa deklarasi abstrak, dan tidak boleh ada implementasi sama sekali.
- Suatu kelas hanya boleh mewarisi sebuah kelas, tetapi ia dapat mengimplementasikan lebih dari satu interface.

4. Polimorfisme (*Polymorphism*)

Polimorfisme secara harfiah dapat diartikan banyak bentuk. Konsep ini memiliki arti kemampuan untuk mendefinisikan perilaku yang berbeda. Singkatnya, secara teknis, method atau konstruktor dengan nama sama dapat memiliki perilaku berbeda bergantung pada argumen atau tipe objeknya. Jadi, kata kunci untuk merepresentasikan konsep polimorfisme adalah: *satu nama, banyak bentuk*.

Konsep polimorfisme membentuk paradigma pemrograman yang ampuh yang mampu menyederhanakan definisi *client* dan secara dinamis mendukung perubahan keterhubungan antarobjek saat *runtime*.

D. LATIHAN

1. Pewarisan

Dalam hubungan pewarisan, kelas turunan merupakan implementasi nyata dari kelas dasar. Untuk lebih memahami konsep pewarisan, ikuti langkah-langkah berikut:

1. Buat aplikasi Windows (dengan memilih template **Windows Application**)
2. Tambahkan item kelas baru melalui menu **Project > Add Class**. Simpan dengan nama **Person.vb**.
3. Lengkapi body kelas `Person` seperti berikut:

```
Public Class Person

    Private strName As String

    Sub New()
        strName = "Anonymous"
    End Sub

    Public Property Name() As String
        Get
            Return strName
        End Get
        Set(ByVal value As String)
            strName = value
        End Set
    End Property

    Public Sub PrintInfo()
        Console.WriteLine("Method Objek Person di-invoke...")
    End Sub

End Class
```

4. Simpan kelas `Person`.

Untuk mengimplementasikan pewarisan, VB.NET menyediakan keyword `Inherits`.

1. Masih di project yang sama, tambahkan item kelas baru dan simpan dengan nama **Student.vb**.
2. Lengkapi body kelas Student seperti berikut:

```
' Kelas Student mewarisi kelas Person
Public Class Student
    Inherits Person

    Private mNim As Integer

    ' Properti Nim
    Public Property NIM() As Integer
        Get
            Return mNim
        End Get
        Set(ByVal value As Integer)
            mNim = value
        End Set
    End Property
End Class
```

3. Simpan kelas Student.

Langkah selanjutnya adalah menguji fungsionalitas kelas turunan (Student). Misalkan di sini kita menggunakan aplikasi Windows untuk pengujian.

1. Tambahkan sebuah button di form.
2. Berikan event Click, dan lengkapi kode event handler-nya.

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

    ' Menciptakan objek Student
    Dim student As New Student
    ' Menggunakan properti kelas turunan (Student)
    student.NIM = 123
    Console.WriteLine("NIM: " & student.NIM)

    ' Menggunakan properti kelas induk
    student.Name = "Agus"
    Console.WriteLine("Nama: " & student.Name)

    ' Memanggil method kelas induk
    student.PrintInfo()
End Sub
```

3. Jalankan aplikasi dan amati hasilnya.



Perlu diperhatikan, pembuatan kelas juga dapat dilakukan di dalam modul. Dengan kata lain, sebuah template Module bisa mengandung lebih dari satu definisi kelas; cara penggunaan kelas-kelas ini sama seperti definisi reguler—dalam satu file tersendiri.

Overriding Method

Pada hubungan pewarisan, membawa secara intrinsik semua atribut dan method tidak selalu dikehendaki. Adakalanya kita ingin memodifikasi perilaku dari superkelas. Langkah ini kita lakukan dengan cara meng-*override* method superkelas. Subkelas dapat mengesampingkan method yang didefinisikan di superkelas dengan menyediakan implementasi baru.

Di VB.NET, implementasi *overriding* memerlukan tahapan khusus.

1. Agar method `PrintInfo()` di superkelas dapat di-*override*, tambahkan keyword `Overridable`.

```
Public Class Person
    ' Member lainnya tetap

    ' Mengindikasikan bahwa method dapat di-override
    Public Overridable Sub PrintInfo()
        Console.WriteLine("Method Objek Person di-invoke...")
    End Sub
End Class
```

2. Selanjutnya, untuk meng-*override*, gunakan keyword `Overrides`.

```
Public Class Student
    Inherits Person

    ' Member lainnya tetap

    ' Meng-override method superkelas
    Public Overrides Sub PrintInfo()
        Console.WriteLine("Method Objek Student di-invoke...")
    End Sub
End Class
```

3. Untuk mengetahui pengaruh *overriding*, panggil method `PrintInfo()`.

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

    ' Menciptakan objek Student
    Dim student As New Student

    ' Memanggil method yang di-override
    student.PrintInfo()
End Sub
```

Dalam konteks *overriding*, fungsionalitas method superkelas akan ditindas oleh subkelas. Meski demikian, kita masih dapat mengakses method superkelas dengan memanfaatkan keyword `MyBase`—ekuivalen dengan `super` di Java.

```

Public Class Student
    Inherits Person

    ' Member lainnya tetap

    ' Meng-override method superkelas
    Public Overrides Sub PrintInfo()
        ' Memanggil method superkelas
        MyBase.PrintInfo()

        Console.WriteLine("Method Objek Student di-invoke...")
    End Sub
End Class

```



Perhatikan, di beberapa bahasa lain—seperti Java, implementasi overriding tidak memerlukan keyword-keyword khusus—seperti `Overrideable` dan `Overrides`.

2. Kelas Abstrak

Kelas abstrak di VB.NET didefinisikan dengan menggunakan keyword `MustInherit`. Keyword ini sekaligus mengindikasikan bahwa kelas abstrak harus diperluas oleh kelas lainnya.

Sebagai contoh, buat kelas abstrak `Buah` seperti berikut:

1. Tambahkan item kelas baru dan simpan dengan nama **Buah.vb**.

```

' Mendeklarasikan class dengan keyword MustInherit
Public MustInherit Class Buah

    ' Deklarasi member abstrak dengan MustOverride
    Public MustOverride Sub GetFlavor()

    ' Method reguler
    Public Sub GetFamily()
        Console.WriteLine("Keluarga Buah-buahan")
    End Sub
End Class

```

2. Tambahkan item kelas baru dan simpan dengan nama **Jeruk.vb**.

```

Public Class Jeruk
    Inherits Buah

    ' Harus meng-override (mengimplementasikan) GetFlavor()
    Public Overrides Sub GetFlavor()
        Console.WriteLine("Asam")
    End Sub
End Class

```

3. Perhatikan, setiap kelas yang mewarisi kelas abstrak harus mengimplementasikan method-method abstrak di superkelas.

3. Interface

Interface memungkinkan kita untuk mendefinisikan fitur-fitur sebagai kelompok kecil dari member-member yang berhubungan. Di VB.NET, interface didefinisikan dengan menggunakan pernyataan `Interface`.

1. Tambahkan item baru dengan template **Module**.
2. Definisikan dua buah interface, misalnya `IPrintable` dan `IWritable`.

```
Module Module1

    ' Interface IWritable
    Interface IWritable

        Sub Write()

    End Interface

    ' Interface IPrintable
    Interface IPrintable

        Sub Print()

    End Interface

End Module
```

3. Untuk mengimplementasikan interface, kita menggunakan keyword `Implements`.

```
Public Class InterfaceDemo
    Implements IPrintable

    Public Sub Print() Implements IPrintable.Print
        Console.WriteLine("Print...")
    End Sub

End Class
```

4. Sama seperti penggunaan kelas abstrak, method-method di interface harus diimplementasikan oleh kelas-kelas yang menggunakannya.

Berbeda dengan pewarisan, sebuah kelas dapat mengimplementasikan lebih dari satu interface.

```
' Mengimplementasikan lebih dari satu (multiple) interface
Public Class InterfaceDemo
    Implements IPrintable, IWritable

    Public Sub Print() Implements IPrintable.Print
        Console.WriteLine("Print...")
    End Sub

    Public Sub Write() Implements Module1.IWritable.Write
        Console.WriteLine("Writing...")
    End Sub

End Class
```

Latihan Kecil

Apa pendapat Anda mengenai pendefinisian kelas dan interface berikut. Jelaskan!

```
Public Class Something
    Inherits IPrintable

End Class

Interface ISomething
    Inherits IPrintable

End Interface
```

4. Polimorfisme

Pada hakekatnya, polimorfisme dapat diklasifikasikan menjadi dua jenis: *overloading* dan *overriding* (lihat kembali pembahasan di awal).

Overloading Method

Overloading method adalah kemampuan untuk mendefinisikan beberapa method di sebuah kelas dengan nama sama. Ini mengimplikasikan bahwa method yang di-*overload* harus memiliki jumlah atau tipe argumen berbeda. Adapun jika jumlah dan tipe argumennya sama, maka urutannya harus berbeda.

1. Tambahkan item kelas baru dan simpan dengan nama **OverloadDemo.vb**.

```
Public Class OverloadDemo

    ' Overloading method GetTotal

    Public Function GetTotal() As Integer
        Console.WriteLine("Tanpa Parameter di-invoke")
        Return 1
    End Function

    Public Function GetTotal(ByVal a As Integer) As Integer
        Console.WriteLine("Parameter Integer di-invoke")
        Return 1
    End Function

    Public Function GetTotal(ByVal a As Integer, _
        ByVal b As Double) As Integer
        Console.WriteLine(
            "Parameter Integer dan Double di-invoke")
        Return 1
    End Function

    Public Function GetTotal(ByVal b As Double, _
        ByVal a As Integer) As Integer
        Console.WriteLine(
            "Parameter Double dan Integer di-invoke")
        Return 1
    End Function

End Class
```

2. Pemanggilan kode *overloading*—dari sisi kompiler—didasarkan pada argumen yang dikirimkan.

```
Private Sub Button2_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button2.Click

    Dim od As New OverloadDemo

    ' Memanggil method yang di-overload

    od.GetTotal()
    ' Output: Tanpa Parameter di-invoke

    od.GetTotal(1)
    ' Output: Parameter Integer di-invoke

    od.GetTotal(1, 3.5)
    ' Output: Parameter Integer dan Double di-invoke

    od.GetTotal(3.5, 2)
    ' Output: Parameter Double dan Integer di-invoke

End Sub
```

3. Jalankan aplikasi dan amati hasilnya.

Latihan Kecil

Apa yang terjadi jika di dalam kelas `OverloadDemo` ditambahkan method-method berikut. Jelaskan!

```
Public Sub GetTotal()
    Console.WriteLine("GetTotal")
End Sub

Public Function GetTotal() As String
    Return ""
End Function
```

E. TUGAS PRAKTIKUM

1. Buat kelas abstrak `Shape` dengan sebuah method abstrak bernama `GetArea()`. Definisikan juga dua subkelas dari `Shape` dengan nama `Rectangle` dan `Circle`. Gunakan rumus penghitungan luas untuk mengimplementasikan method `GetArea()`.