PAIK6401 / AIK21341
Pemrograman Berorientasi Objek

10 Generik

Informatika FSM Universitas Diponegoro

Capaian Pembelajaran

CPL03

Mampu menerapkan ilmu dan teknologi informasi dalam proses penyelesaian permasalahan, yang meliputi analisis permasalahan kompleks, pemodelan, pendefinisian kebutuhan, perancangan, implementasi dan evaluasi terhadap sistem, proses, komponen, dan program.

CPL05

Mampu menghasilkan rancangan, mengimplementasikan, dan mengevaluasi solusi berbasis algoritma dengan mempertimbangkan aspek kompleksitas

CPL07

Mampu memilih, mengadaptasi atau membuat, kemudian menerapkan teknik, sumber daya, kakas komputasi moderen dengan tepat pada aktivitas komputasi kompleks, serta memahami keterbatasannya

Capaian Pembelajaran

CPMK05-2:

Mampu menerapkan konsep teoretis bidang pengetahuan dan keterampilan Ilmu Komputer dalam menyelesaikan permasalahan kompleks dengan pemikiran komputasional untuk pengambilan keputusan.

CPMK10-2:

Mampu menghasilkan rancangan dan Mengimplementasi solusi berbasis algoritma untuk permasalahan kompleks.

Capaian Pembelajaran

Sub CPMK05-2 dan Sub CPMK10-2:

1.Mampu menerapkan (C3) konsep enkapsulasi, kelas, dan algoritma siklus hidup objek dengan mendemonstrasikan (P3) dalam bahasa pemrograman tertentu.

2.Mampu menerapkan (C3) konsep dan konsekuensi pewarisan dengan mengkonstruksi (P4) kelas dalam bahasa pemrograman tertentu.

3.Mampu menganalisis (C4) polimorfisme dan generik dengan mengembangkan (P4) kasus dalam bahasa pemrograman tertentu.

4.Mampu mendesain (C6) koleksi objek persisten dengan mendemonstrasikan (P3) penyelesaian permasalahan kompleks.

5.Mampu memadukan (C6) prinsip rancangan berorientasi objek dengan paradigma lain yang relevan.

Bahan Kajian

1. Objek, Kelas, Enkapsulasi, dan Information Hiding
2. Inheritance, Overloading, Overriding, Kelas Abstrak dan Interface, Eksepsi dan Asersi
3. Polimorfisme dan Generik
4. Koleksi Objek Persisten
5. Desain Berorientasi Objek, Multiparadigma

Generik

- Spesifik = khusus, jenis/tipe sudah ditentukan.
- Generik = umum, jenis/tipe belum ditentukan saat pendefinisian suatu kelas. Jenis/tipe ditentukan saat kelas generic tersebut digunakan dalam kamus.
- Generik bisa berlaku pada kelas atau pada metode.
  Class Kelas<Generik>
  method Fun<Generik>() -> Generik

1. Generik pada Kelas

- Definisi:
  Class Datum <Generik> has …
- Kamus:
  A : Datum <Spesifik>
- Algoritma:
  A <- new Datum <>()

Kelas Datum yang isinya Generik

```
class Datum<Generik> has
    isi : Generik
    function getIsi() -> Generik
        -> this.isi
    procedure setIsi( input x : Generik )
        this.isi <- x
```

Contoh Suatu Keluarga Kelas

```
class Kupu has
  procedure gerak()
class Ulat inherits Kupu has
  procedure gerak()
    output "ulat merayap"
class Kepompong inherits Kupu has
  procedure gerak()
    output "kepompong diam"
class KupuDewasa inherits Kupu has
  procedure gerak()
    output "kupu terbang"
```

Aplikasi Kelas Generik

```
class Main
 constructor Main()
  kamus
   K : Ulat
   anu : Datum < Kupu >
  algoritma
   K <- new Ulat
   anu <- new Datum<>()
   anu.setIsi(K)
   anu.getIsi().gerak() {ulat merayap}
   anu.setIsi( new Kepompong )
   anu.getIsi().gerak() {kepompong diam}
   anu.setIsi( new KupuDewasa )
   anu.getIsi().gerak() {kupu terbang}
```

Pikirkan!
1. Ada berapa variabel?
2. Ada berapa objek?

2. Generik pada Metode

- Definisi:
  class MetodeGenerik has
    function Fun <Generik>(parameters) → Generik
    procedure Pro <Generik>(parameters)
- Kamus:
  A : Datum <Spesifik>
  variabel : Datum <Spesifik>
  M : MetodeGenerik
- Algoritma:
  variabel <- M.Fun(A)
  M.Pro(A)

Contoh Kelas Bermetode Generik

```
class ContohMetodeGenerik has
  function Fungsi <Generik> (D:Datum<Generik>) -> Generik
    -> D.getIsi()
  procedure Prosedur <Generik> (D:Datum<Generik>)
    D.getIsi().method() {tergantung metode yang diketahui}
```

Aplikasi Kelas Bermetode Generik

```
class Main has
 constructor Main()
  kamus
   C : ContohMetodeGenerik
   Anu : Datum<Kupu>
  algoritma
   C <- new ContohMetodeGenerik()
   Anu <- new Datum<Ulat>()
   C.Prosedur(Anu)          {ulat merayap}
   C.Fungsi(Anu).gerak()    {ulat merayap}
```

Contoh Lain Kelas Bermetode Generik

```
class Picker has {metode generik}
    function peek<T>(a: array of T)-> T
        -> a[1]
class Main has
 constructor Main()
  kamus
   b : array [1...10] of string
  algoritma
   b[1] <- "Semarang"
   b[2] <- "Jogja"
   b[3] <- "Solo"
   output ( new Picker).peek(b) {Semarang}
```

Membatasi Generik

- Upper Bounded Generic
  - Membatasi masukan kelas hanya anak dari suatu kelas generic pembatas, alamiah
  - Kata kunci **inherits**
- Lower Bounded Generic
  - Membatasi masukan kelas hanya induk dari suatu kelas generic pembatas
  - Kata kunci **super**

Upper Bounded Generic

- Kelas Pembatas sebagai hierarki kelas tertinggi, Generik hanya berlaku pada dirinya sendiri dan keturunannya

```
class Datum<Generik inherits Pembatas > has
    isi : Generik
    function getIsi() -> Generik
        -> this.isi
    procedure setIsi( input x : Generik )
        this.isi <- x
```

Lower Bounded Generic

- Kelas Generik sebagai hierarki kelas terendah, Generik hanya berlaku pada dirinya sendiri dan leluhurnya

```
class Datum< ? super Generik > has
    isi : Generik
    function getIsi() -> Generik
        -> this.isi
    procedure setIsi( input x : Generik )
        this.isi <- x
```

Referensi

1.Panji Wisnu Wirawan, Indra Waspada, Satriyo Adhy. 2018. Buku Ajar Pemrograman Berorientasi Objek.
2.Herbert Schildt. 2019. Java The Complete Reference, 11th edition.
3.Vaskaran Sarcar. 2020. Interactive Object-Oriented Programming in Java.
4.Robert C. Martin. 2013. Agile Software Development, Principles, Patterns, and Practices, 1st edition.
5.Peter Sestoft. 2017. Programming Language Concepts, 2nd edition.