BANYAK BENTUK (POLYMORPHISM)

Pertemuan ke-10

Sub-CPMK

- *Mahasiswa mampu menyelesaikan masalah dengan menggunakan salah satu pilar OOP yaitu Banyak Bentuk dalam konsep Pemrograman Berorientasi Objek (PBO). (C4, A4).*

Materi

1. Konsep Banyak Bentuk
2. Jenis-Jenis Banyak Bentuk

1. Konsep Banyak Bentuk

- Banyak bentuk dapat dimaknai sebagai modul yang memiliki nama sama, namun memiliki tingkah laku (behaviour) yang berbeda sehingga kode program implementasinya juga berbeda.
- Dalam mengimplementasikan banyak bentuk harus memenuhi:

Lanj…

- Method yang dipanggil harus melalui variabel dari basis kelas (class) atau superkelas (superclass).
- Method yang dipanggil juga harus menjadi method dari basis class.
- Signature method harus sama baik pada superclass maupun subclass.
- Method access attribute pada subclass tidak boleh lebih terbatas dari basis class.

Lanj…

- Banyak bentuk adalah pemakaian method dengan nama yang sama pada kelas-kelas yang berbeda dan memungkinkan method yang tepat dieksekusi berdasarkan konteks yang memangilnya.
- Banyak bentuk memiliki arti "satu nama, banyak bentuk".

Lanj…

- Hal ini disebut juga sebagai overloading yang berarti menggunakan hal yang sama untuk tujuan yang berbeda.
- Dengan menggunakan banyak bentuk kita dapat membuat banyak fungsi dengan nama yang sama tetapi memiliki daftar argumen yang berbeda.

Lanj…

- Banyak bentuk dapat diilustrasikan sebagai berikut: "buah pepaya itu masih **mentah**". "proposal penjualan masih **mentah**, perlu ada kajian ulang". Kata "**mentah**" pada contoh dapat diaplikasikan pada berbagai objek dan dapat diinterpretasikan ke dalam beberapa makna. Dalam mengimplementasikan banyak bentuk dapat berbasis pada **interface** dan berbasis pada **pewarisan**.

1.1 Berbasis Interface

- Untuk mengimplementasikan banyak bentuk dengan interface kita mengimplementasikan interface dengan cara berbeda di beberapa kelas. Selanjutnya aplikasi client bisa menggunakan implementasi baru atau lama. Keuntungan dari banyak bentuk berbasis interface yaitu kita tidak perlu mengkompilasi ulang aplikasi client untuk bisa bekerja dengan implementasi interface yang baru.

1.2 Berbasis Pewarisan

- Sebagian besar bahasa pemrograman beroirentasi objek menyediakan implementasi banyak bentuk melalui pewarisan. Banyak bentuk berbasis pewarisan ini melibatkan pendefinsian method di kelas induk dan meng-**override**-nya dengan implementasi baru di kelas turunan. Dalam implementasi banyak bentuk dengan pewarisan kita dapat menerapkan overloading dan overriding.

1.2 Berbasis Pewarisan (Lanj..)

Buat kelas dengan nama: clsBangun

Kelas: clsBangun

```
1.   package luasbangun;
2.
3.   public class clsBangun
4.   {
5.       public void hitung(int x,int y)
6.       {
7.           System.out.println("Luas bangun: ");
8.       }
9.   }
```

1.2 Berbasis Pewarisan (Lanj..)

Buat kelas dengan nama: clsSegiEmpat

Kelas: clsSegiEmpat

```
1.   package luasbangun;
2.
3.   public class clsSegiEmpat extends clsBangun
4.   {
5.       public void hitung (int x, int y)
6.       {
7.           System.out.println("Segi empat: "+ x*y);
8.       }
9.   }
```

1.2 Berbasis Pewarisan (Lanj..)

Buat kelas dengan nama: clsSegiTiga

Kelas: clsSegiTiga

```
1.   package luasbangun;
2.
3.   public class clsSegiTiga extends clsBangun
4.   {
5.       public void hitung (int x, int y)
6.       {
7.           System.out.println("Segi tiga: "+ 0.5*x*y);
8.       }
9.   }
```

1.2 Berbasis Pewarisan (Lanj..)

Buat program di main program

Main program

```
1.   package luasbangun;
2.   import java.util.Scanner;
3.
4.   public class LuasBangun {
5.
6.       public static void main(String[] args)
7.       {
8.           // TODO code application logic here
9.           Scanner input = new Scanner (System.in);
10.
11.          clsBangun objBangun = new clsBangun();
12.          clsSegiEmpat objSegiEmpat = new clsSegiEmpat();
13.          clsSegiTiga objSegiTiga = new clsSegiTiga();
```

1.2 Berbasis Pewarisan (Lanj..)

```
14.
15.             int m,n;
16.             System.out.printf("Masukkan Nilai X: ");
17.             m = input.nextInt();
18.             System.out.printf("Masukkan Nilai Y: ");
19.             n = input.nextInt();
20.             objBangun.hitung(m, n);
21.
22.             objBangun = objSegiEmpat;
23.             objBangun.hitung(m, n);
24.
25.             objBangun = objSegiTiga;
26.             objBangun.hitung(m, n);
27.         }
28.     }
```

1.2 Berbasis Pewarisan (Lanj..)

Contoh keluaran program seperti berikut.

2. Jenis-Jenis Banyak Bentuk

- Ada dua jenis polimorfisme yaitu:
  - polimorfisme statis dan
  - polimorfisme dinamis.

2.1 Polymorphism Statis

- Untuk melakukan polimorfisme statis dapat dilakukan dengan **overloading fungsi** dan **overloading operator**. Contoh polimorfisme statis adalah method yang menggunakan method final atau method private. Pada saat kompilasi Java mengetahui method mana yang memanggil dengan memeriksa argumennya.

2.1 Polymorphism Statis (Lanj..)

2.1.1 Overloading Fungsi

- Overloading fungsi (method) merupakan kemampuan suatu fungsi melakukan tugas yang berbeda, dengan nama fungsi sama.
- Namun yang membedakan tipe datanya.

2.1 Polymorphism Statis (Lanj..)

2.1.2 Overloading Operator

- Untuk melakukan overloading operator Java tidak mendukung untuk melakukan overloading operator.

2.2 Polymorphism Dinamis

- Polimorfisme dinamis juga dikenal polimorfisme run-time. Dalam hal ini compiler Java tidak tahu method mana yang dipanggil pada waktu kompilasi. Method override adalah contoh polimorfisme run-time. Dalam hal ini method overrid adalah pemanggil melalui variabel referensi kelas super. Jenis-jenis polimorfisme dinamis pada Java.

2.2 Polymorphism Dinamis

2.2.1 Fungsi Virtual

- Yang dimaksud dengan polimorfisme fungsi virtual tidak lain adalah fungsi yang kinerjanya dapat diganti dalam kelas warisan dengan fungsi argumen atau signatur yang sama. Fungsi virtual tidak dapat dinyatakan sebagai private.
- Dalam fungsi ini kita mendapat peringatan (pesan) jika kita tidak menggunakan kata kuci Virtual atau New.

Ringkasan:

- Salah satu dari empat pilar pemrograman berorientasi objek adalah banyak bentuk (*polymorphism*).
- Banyak bentuk dapat dimaknai sebagai modul yang memiliki nama sama, namun memiliki tingkah laku (behaviour) yang berbeda sehingga kode program implementasinya juga berbeda.

Latihan Mandiri

- Buatlah sebuah program berorientasi objek yang mengimplementasikan konsep polymorphism dengan konsep interface.
- Buatlah sebuah program berorientasi objek yang mengimplementasikan konsep polymorphism dengan konsep inheritance.

TERIMA KASIH