Academic Year 2021/2022

Objek dan Kelas

Pemrograman Berorientasi Objek

Rofilde Hasudungan
Lecturer
Department of Informatic

Universitas Muhammadiyah Kalimantan Timur

November 15, 2021

Outline

Kelas dan Objek

1. Pada pembahasan sebelumnya kita telah menggunakan variabel untuk menyimpan suatu nilai dengan tipe data seperti int, short, float, double, dan String. Variabel-variabel tersebut hanya dapat menampung satu buah nilai/data.
2. Untuk menampung data yang banyak/lebih dari satu, maka kita dapat menggunakan array.
3. Namun, array hanya dapat menampung data yang memiliki tipe data yang sama,
4. padahal informasi dari suatu data kadang bervariasi. Contohnya mahasiswa memiliki nim (String), umur (int), nama (String) dan lain sebagainya. Contoh lainnya, misalkan pembalap MotoGP memiliki nama, nomor, dan tim. Jika kalian bermain game, suatu karakter memiliki data atau kadang disebut atribut seperti kekuatan serangan, pertahanan, reward dan lain sebagainya.

```
1
2  public class InfoPembalap {
3    public static void main(String[] args) {
4      String nama1 = "Valentino Rossi";
5      String alias1 = "Rossi";
6      short nomor1 = 46;
7      float tinggi1= 1.76f;
8      String tim1  = "Yamaha Satelite";
9
10     String nama2 = "Francesco Bagnia";
11     String alias2 = "Pecco";
12     short nomor2 = 38;
13     float tinggi2= 1.86f;
14     String tim2  = "Ducati";
15
16     System.out.println("Nama : " + nama1);
17     System.out.println("Alias : " + alias1);
18     System.out.println("Nomor : " + nomor1);
19     System.out.println("Tinggi : " + tinggi1);
20     System.out.println("Tim : " + tim1);
```

```
21      System.out.println("------------------");
22      System.out.println("Nama : " + nama2);
23      System.out.println("Alias : " + alias2);
24      System.out.println("Nomor : " + nomor2);
25      System.out.println("Tinggi : " + tinggi2);
26      System.out.println("Tim : " + tim2);
27    }
28
29 }
```

Kelas dan Objek

1. Pada contoh sebelumnya, terdapat data pembalap yang memiliki nama, alias, nomor, tinggi, dan tim, dimana tipe data dari tiap atribut pembalap atau variabel tersebut berbeda.
2. Membuat variable-variabel yang terpisah untuk tiap-tiap data seperti contoh di atas tidaklah efisien, terutama seperti contoh di atas terdapat banyak pembalap.
3. Kita perlu sejenis struktur data untuk mendeskripsikan pembalap tersebut, sehingga data-data tersebut dapat dijadikan satu unit yang dapat diakses dan dimanajemen dengan mudah.
4. Untuk mendeskripsikan suatu data yang kompleks seperti pembalap, mahasiswa dan lain sebagainya dapat menggunakan kelas tersendiri. Berikut ialah contoh kelas untuk mendeskripsikan data (objek) pembalap.

Mendeskripsikan Objek (Struktur data)

Untuk membuat data (objek) seperti pembalap, mahasiswa, etc. menjadi satu unit, kita dapat membuat sebuah kelas untuk mendeskripsikan data tersebut. Kelas yang dibuat sama dengan kelas yang telah dibuat sebelumnya. Namun, format umumnya ialah sebagai berikut:

```
1 MODIFIER class NamaKelas {
2   // atribut
3   // method
4 }
```

Contoh Kelas untuk Objek (data) Pembalap

Sebagai contoh berikut ialah kelas untuk objek pembalap.

```
1 public class Pembalap {
2   String nama;
3   String alias;
4   short nomor;
5   float tinggi;
6   String tim;
7 }
```

Perhatikan didalam kelas Pembalap, terdapat beberapa variabel yang digunakan untuk menampung data pembalap. Variabel-variabel tersebut digunakan untuk menampung data(nilai) yang terkait dengan pembalap (objek).
Selanjutnya kita dapat menggunakan kelas tersebut untuk membuat objek seperti contoh berikut:

```
1
2  public class InfoPembalap {
3    public static void main(String[] args) {
4      Pembalap rossi = new Pembalap();
5      rossi.nama = "Valentino Rossi";
6      rossi.alias = "Rossi";
7      rossi.nomor = 46;
8      rossi.tinggi = 1.76F;
9      rossi.tim = "Yamaha Satelite";
10
11     Pembalap pecco = new Pembalap();
12     pecco.nama = "Francesco Bagnia";
13     pecco.alias = "Pecco";
14     pecco.nomor = 38;
15     pecco.tinggi = 1.86f;
16     pecco.tim = "Ducati";
17
18     Pembalap[] pembalap = new Pembalap[];
19     pembalap[0]  =rossi;
20     pembalat[1] = pecco;
```

```
21      for(Pembalap p : pembalap) {
22        System.out.println("Nama : " + p.nama);
23        System.out.println("Alias : " + p.alias);
24        System.out.println("Nomor : " + p.nomor);
25        System.out.println("Tinggi : " + p.tinggi);
26        System.out.println("Tim : " + p.tim);
27        System.out.println("-------------------");
28      }
29    }
30  }
```

Format Deklarasi Variabel Objek

Dari contoh di atas, penggunaan kelas Pembalap sebagai tipe data sama dengan penggunaan tipe data yang lain seperti int, double, String, etc. Sehingga hal ini seperti kita memiliki atau membuat tipe data sendiri. Apa yang telah kita buat sebelumnya (kelas Pembalap) disebut dengan tipe data objek. Seperti yang ditunjukan pada contoh kode InfoPembalap, untuk menggunakan tipe data Pembalap ditunjukan pada baris ke 4 dan 11. Format umumnya ialah berikut:

```
1 NamaKelas var = new NamaKelas();
```

Array dari Objek

Samahalnya dengan variabel dengan tipe data lain seperti int, double, long, sort, float, String, etc. Kita juga dapat membuat array dari Tipe Data Objek yang kita buat dengan format yang sama. Perhatikan format berikut:

```
1 NamaKelas[] var = new NamaKelas[PANJANG_ARRAY];
```

Dengan menggunakan format di atas, kita dapat merubah kelas InfoPembalap sehingga seperti berikut:

```
1  public class InfoPembalap {
2    public static void main(String[] args) {
3      Pembalap rossi = new Pembalap();
4      rossi.nama = "Valentino Rossi";
5      rossi.alias = "Rossi";
6      rossi.nomor = 46;
7      rossi.tinggi = 1.76F;
8      rossi.tim = "Yamaha Satelite";
9
10     Pembalap pecco = new Pembalap();
11     pecco.nama = "Francesco Bagnia";
12     pecco.alias = "Pecco";
13     pecco.nomor = 38;
14     pecco.tinggi = 1.86f;
15     pecco.tim = "Ducati";
16
17     Pembalap[] pembalap = new Pembalap[2];
18
19     for(Pembalap p : pembalap) {
20       System.out.println("Nama : " + p.nama);
```

```
21          System.out.println("Alias : " + p.alias);
22          System.out.println("Nomor : " + p.nomor);
23          System.out.println("Tinggi : " + p.tinggi);
24          System.out.println("Tim : " + p.tim);
25          System.out.println("-------------------");
26      }
27    }
28  }
```

Dengan menggunakan array, kita dapat mengakses objek (data) pembalap dengan perulangan, misalkan seperti pada code di atas digunakan untuk menampilkan datanya (19-26). Tentunya kode di atas lebih efisien dari kode sebelumnya (tanpa array) dan terasa lebih efisien lagi ketika kita bekerja dengan jumlah data (objek) dan variabel yang banyak.

Objek sebagai Parameter dan Nilai Balik

Sama seperti variabel dengan tipe data lain seperti int, double, dan String, Objek yang kita buat dapat digunakan sebagai parameter ataupun nilai balik dari suatu method. Perhatikan contoh berikut:

```
1  public class InfoPembalap {
2    public static void main(String[] args) {
3      Pembalap[] daftarPembalap = dataPembalap();
4      tampilkanPembalap(daftarPembalap);
5    }
6
7    public static Pembalap[] dataPembalap() {
8      Pembalap rossi = new Pembalap("Valentino Rossi", "Rossi", 46, 1.76
       F, "Yamaha Satelite");
9      Pembalap pecco = new Pembalap("Francesco Bagnia", "Pecco", 38,
       1.86F, "Ducati");
10     Pembalap mir = new Pembalap("J. Mir", "Mir", 77, 1.86F, "Suzuki");
11
12     Pembalap[] pembalap = new Pembalap()[3];
13     pembalap[0] = rossi;
14     pembalap[1] = pecco;
15     pembalap[2] = mir;
16
17     return pembalap;
18   }
```

```
19
20    public static void tampilkanPembalap(Pembalap[] pembalap) {
21      for(Pembalap p : pembalap) {
22        System.out.println("Nama : " + p.nama);
23        System.out.println("Alias : " + p.alias);
24        System.out.println("Nomor : " + p.nomor);
25        System.out.println("Tinggi : " + p.tinggi);
26        System.out.println("Tim : " + p.tim);
27        System.out.println("-------------------");
28      }
29    }
30  }
```

Atribut

1. Perhatikan kelas Pembalap. Struktur kelas yang dibuat sedikit berbeda dari yang telah dibuat pada pembahasan-pembahasan sebelumnya.
2. Pada kelas ini (eg. Pembalap), "variabel" yang kita buat diletakan **langsung** didalam kelas, bukan didalam method main seperti pada contoh sebelum-sebelumnya.
3. "Variabel-variabel" tersebut didalam paradigma pemrograman berorientasi objek sering disebut dengan atribut, karena ia mendeskripsikan data atau atribut dari objek (pembalap, mahasiswa, dll).
4. Sehingga, untuk mengakses (menggunakan) atribut (variabel) tersebut HARUS melalui variabel objek.
5. Istilah lain dari atribut yang sering digunakan ialah properti, dan field, namun pada pembahasan ini kita akan lebih sering menggunakan istilah atribut.

Mendefinisikan Atribut

1. Layaknya variabel, atribut didefinisikan (dibuat) dengan menyertakan tipe datanya.
2. Atribut sangat mirip sekali variabel, namun atribut dibuat didalam kelas dan diluar method.
3. Kemudian, atribut seperti method memiliki modifier.

Format Atribut

```
class NamaKelas {
  MODIFIER TIPEDATA namaAtribut1;
  MODIFIER TIPEDATA namaAtribut2;
  MODIFIER TIPEDATA namaAtribut;
}
```

1. MODIFIER, Dapat berupa public, private, protected atau tidak ada sama sekali (default). Detail kegunaan MODIFIER akan dibahas kemudian.
2. TIPEDATA, dapat berupa tipe data yang telah ada seperti Numerik (Primitif), atau Objek seperti String, Scanner atau Objek yang lainnya.
3. namaAtribut ialah atribut, dimana penamaannya seperti penamaan variabel.

Method dari Objek

Suatu objek selain memiliki atribut juga memiliki method. Method pada objek didefinisikan pada kelas dengan cara yang sama seperti yang telah dibahas pada materi sebelumnya. Berikut ialah format dalam mendeklarasikan method:

```
1 Modifier ReturnType namaMethod(param1, param2,.., parameters) {
2   // instruksi-instruksi
3 }
```

Ubahlah kode Pembalap.java dan InfoPembalap.java seperti pada slide berikut

```
1  public class Pembalap {
2    String nama;
3    String alias;
4    short nomor;
5    float tinggi;
6    String tim;
7
8    public void info() {
9      System.out.println("Nama : " + this.nama);
10     System.out.println("Alias : " + this.alias);
11     System.out.println("Nomor : " + this.nomor);
12     System.out.println("Tinggi : " + this.tinggi);
13     System.out.println("Tim : " + this.tim);
14   }
15 }
```

```
1  public class InfoPembalap {
2    public static void main(String[] args) {
3      Pembalap rossi = new Pembalap();
4      rossi.nama = "Valentino Rossi";
5      rossi.alias = "Rossi";
6      rossi.nomor = 46;
7      rossi.tinggi = 1.76F;
8      rossi.tim = "Yamaha Satelite";
9
10     Pembalap pecco = new Pembalap();
11     rossi.nama = "Francesco Bagnia";
12     rossi.alias = "Pecco";
13     rossi.nomor = 38;
14     rossi.tinggi = 1.86f;
15     rossi.tim = "Ducati";
16
17     Pembalap[] pembalap = new Pembalap[2];
18     pembalap[0] = rossi;
19     pembalap[1] = pecco;
20     for(Pembalap p : pembalap) {
```

```
21          p.info();
22          System.out.println("-------------------");
23      }
24    }
25  }
26
27
```

Method Constructor

Pada objek, terdapat suatu method khusus, dimana method ini dijalankan saat pertama kali suatu method dibuat (*initiated*). Method ini disebut dengan method constructor. Beberapa keunikan dari method ini ialah

1. Dijalankan saat pertama kali objek dibuat,
2. Tidak memiliki return value,
3. Nama method identik dengan nama kelas.

```
1  public class Pembalap {
2    String nama;
3    String alias;
4    short nomor;
5    float tinggi;
6    String tim;
7    String gender;
8
9    public Pembalap() {
10     this.gender = "Pria";
11   }
12
13   public Pembalap(String nama, String alias, short nomor, float tinggi
       , String tim) {
14     this.nama = nama;
15     this.alias = alias;
16     this.nomor = nomor;
17     this.tinggi = tinggi;
18     this.tim = tim;
19   }
```

```
20
21    public void info() {
22      System.out.println("Nama : " + this.nama);
23      System.out.println("Alias : " + this.alias);
24      System.out.println("Nomor : " + this.nomor);
25      System.out.println("Tinggi : " + this.tinggi);
26      System.out.println("Tim : " + this.tim);
27    }
28  }
```

Method Constructor untuk Inisialisasi Nilai Attribut

Salah satu kegunaan dari method constructor ialah banyak digunakan untuk inisialisasi nilai atribut objek, sehingga nilai objek dapat disi dengan efisien.

```
1  public class Pembalap {
2    String nama;
3    String alias;
4    short nomor;
5    float tinggi;
6    String tim;
7    String gender;
8
9    public Pembalap() {
10     this.gender = "Pria";
11   }
12
13   public Pembalap(String nama, String alias, short nomor, float tinggi
       , String tim) {
14     this.nama = nama;
15     this.alias = alias;
16     this.nomor = nomor;
17     this.tinggi = tinggi;
18     this.tim = tim;
19   }
```

```
20
21    public void info() {
22      System.out.println("Nama : " + this.nama);
23      System.out.println("Alias : " + this.alias);
24      System.out.println("Gender : " + this.gender);
25      System.out.println("Nomor : " + this.nomor);
26      System.out.println("Tinggi : " + this.tinggi);
27      System.out.println("Tim : " + this.tim);
28    }
29  }
```

```
1  public class InfoPembalap {
2    public static void main(String[] args) {
3      Pembalap rossi = new Pembalap("Valentino Rossi", "Rossi", 46, 1.76
       F, "Yamaha Satelite");
4      Pembalap pecco = new Pembalap("Francesco Bagnia", "Pecco", 38,
       1.86F, "Ducati");
5      Pembalap mir = new Pembalap("J. Mir", "Mir", 77, 1.86F, "Suzuki");
6
7      Pembalap[] pembalap = new Pembalap()[3];
8      pembalap[0] = rossi;
9      pembalap[1] = pecco;
10     pembalap[2] = mir;
11
12     for(Pembalap p : pembalap) {
13       p.info();
14       System.out.println("--------------------");
15     }
16   }
17 }
```

Method Kelas Vs Method Objek

1. Pada materi sebelumnya telah dibahas bahwa kita dapat membuat method pada kelas dan method tersebut **dimiliki** oleh kelas.
2. Kemudian, pada materi ini, suatu objek juga dapat memiliki method (method tersebut **dimiliki** oleh objek).
3. Meskipun, method harus dibuat didalam sebuah kelas, namun kepemilikannya berbeda.
4. Sehingga, kita dapat katakan bahwa kita dapat men-setting kepemilikan dari method, yakni dimiliki oleh kelas atau dimiliki oleh objek.
5. Method yang dimiliki oleh kelas, dapat digunakan (akses) secara langsung dengan menggunakan format:

```
1 NamaKelas.namaMethod(...,parameters);
```

6. dari kode contoh yang telah diberikan, bisakah kalian tunjukan yang mana method kelas dan yang mana method objek?

Method Kelas Vs Method Objek

1. Sedangkan method yang dimiliki oleh objek, digunakan dengan terlebih dahulu membuat objek dari kelas seperti format berikut

```
1 NamaKelas variabelObjek = new NamaKelas(..., parameters);
2 variabelObjek.namaMethod(...,parameters);
```

2. Untuk men-setting suatu method dimiliki oleh Kelas, maka kita harus menambahkan keyword **static** pada method tersebut. Sedangkan method yang dimiliki oleh objek tidak menggunakan keyword ini.

Keyword this, Scope Variabel, Ownership dan Method

Perhatikan atribut pada kelas Pembalap dan parameter pada method Pembalap sebagai berikut:

```
1  public class Pembalap {
2    String nama;
3    String alias;
4    short nomor;
5    float tinggi;
6    String tim;
7    String gender;
8    ...
9    public Pembalap(String nama, String alias, short nomor, float tinggi
       , String tim) {
10     this.nama = nama;
11     this.alias = alias;
12     this.nomor = nomor;
13     this.tinggi = tinggi;
14     this.tim = tim;
```

Lingkup Variabel dan Ownership

```
1 public Pembalap(String nama, String alias, short nomor, float tinggi,
      String tim) {
2   this.nama = nama;
3   this.alias = alias;
4   this.nomor = nomor;
5   this.tinggi = tinggi;
6   this.tim = tim;
7 }
```

Pada **method Pembalap** terdapat 5 parameter. Dari sisi kepemilikan, parameter tersebut ialah variabel yang dimiliki oleh method Pembalap, sehingga tidak dapat diakses oleh method yang lain, atau dikatakan hanya diakses pada lingkup lokal (local scope), sehingga ia dapat dikatakan variabel lokal (local variable). Sedangkan, variabel (atribut) yang dimiliki oleh objek (didalam kelas), dapat diakses oleh semua method yang terdapat pada objek yang sama.

Keyword this, Scope Variabel dan Ownership Atribut

```
1 public Pembalap(String nama, String alias, short nomor, float tinggi,
      String tim) {
2   this.nama = nama;
3   this.alias = alias;
4   this.nomor = nomor;
5   this.tinggi = tinggi;
6   this.tim = tim;
7 }
```

Pada **method Pembalap** terdapat 5 parameter (variabel). Nama-nama parameter pada method tersebut mungkin ada yang sama dengan nama atribut dari objek. Hal ini dibolehkan, karena skop dan kepemilikannya berbeda. Namun, didalam penulisan atribut (variabel objek) menggunakan keyword **this**. Keyword this ialah merujuk pada objek itu sendiri, sehingga jika terdapat variabel objek (atribut) nama, dan parameter/variabel method nama, maka jika ditulis `this.nama` artinya mengakses atribut (variabel objek) dari objek.

Keyword this

```
1 public void info() {
2   System.out.println("Nama : " + nama);
3   System.out.println("Alias : " + alias);
4   System.out.println("Gender : " + gender);
5   System.out.println("Nomor : " + nomor);
6   System.out.println("Tinggi : " + tinggi);
7   System.out.println("Tim : " + tim);
8 }
```

1. Penggunaan keyword `this` digunakan untuk merujuk objek.
2. Ia digunakan untuk mengakses atribut atau method didalam method yang dimiliki oleh objek.
3. Keyword `this` dapat **tidak dituliskan** untuk mengakses atribut (variabel objek), jika pada method tersebut tidak ada variabel lokal dengan nama yang sama dengan variabel objek (atribut). .