

Nama & NPM	Topik:	Tanggal:
Hamzah Rizqullah Rahmad	Kelas (Class),Objek,Method, Extends	18 September 2024

[No. 1] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variabel
Permasalahan yang terjadi pada **Latihan 1.1** adalah terkait dengan penggunaan konstruktor yang salah. Pada kode, konstruktor Manusia1 hanya menerima satu parameter (String nama), namun objek dibuat menggunakan dua parameter, yaitu nama dan rambut. Ini mengakibatkan kesalahan saat program dijalankan.

Variabel yang terkait:

- nama: Menyimpan nama manusia (String).
- rambut: Menyimpan warna rambut manusia (String).

[No.1] Analisis dan Argumentasi

- 1) permasalahan ini dapat diatasi dengan cara
Mengubah deklarasi konstruktor untuk menerima dua parameter, yaitu nama dan rambut, agar sesuai dengan argumen yang diberikan saat pembuatan objek.

[No.1] Penyusunan Algoritma dan Kode Program

- 1) berikan Algoritma
 - Deklarasikan kelas Manusia dengan atribut nama dan rambut.
 - Buat konstruktor yang menerima dua parameter (nama, rambut).
 - Cetak informasi nama dan warna rambut saat objek dibuat.
 - Buat method main untuk mendeklarasikan objek dengan dua parameter.
- 2) Berikan Kode program dan luaran

```

1 public class Manusia {
2     // Deklarasi atribut
3     String nama, rambut;
4
5     // Deklarasi constructor dengan dua parameter
6     public Manusia(String nama, String rambut) {
7         System.out.println("Nama saya : " + nama + "\nWarna Rambut : " + rambut);
8     }
9
10    // Method utama
11    public static void main(String[] args) {
12        Manusia satu = new Manusia("Putri", "hitam");
13    }
14 }
15

```

Gambar 1 (input)

```

Nama saya : Putri
Warna Rambut : hitam

```

Gambar 1 (output)

- a) Analisa luaran yang dihasilkan
Luaran yang dihasilkan sudah sesuai dengan ekspektasi. Kode menampilkan nama dan warna rambut sesuai dengan argumen yang diberikan pada konstruktor. Tipe data dan hasil yang ditampilkan sudah sesuai dengan kebutuhan program.

[No.1] Kesimpulan

1. Analisa

- a) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program!
b) Apakah dasar alasan pengambilan keputusan Anda untuk kasus ini?
- a) Kesalahan pada konstruktor berhasil diatasi dengan memperbaiki penulisan konstruktor agar menerima dua parameter (nama dan rambut). Algoritma dan kode yang diperbaiki sesuai dengan permasalahan yang ada.
- b) Dasar pengambilan keputusan untuk memperbaiki konstruktor adalah karena pesan kesalahan jelas menunjukkan bahwa jumlah parameter yang dimasukkan tidak sesuai dengan yang dideklarasikan. Dengan memperbaiki jumlah parameter pada konstruktor, program dapat berjalan dengan baik.

[No. 2] Identifikasi Masalah:

Uraikan permasalahan dan variabel:

Pada **Latihan 2.1**, masalah terjadi karena nama konstruktor pada kelas **Ortu** tidak sesuai dengan nama kelasnya. Konstruktor dinamai **ortu**, sedangkan nama kelasnya adalah **Ortu**, yang menyebabkan kompiler tidak mengenali konstruktor tersebut. Selain itu, konstruktor tidak memiliki parameter meskipun seharusnya menerima dua parameter, yaitu **nama** dan **rambut**.

Variabel yang terkait:

- nama: Menyimpan nama (String).
- rambut: Menyimpan warna rambut (String).

Pada **Latihan 2.2**, diminta untuk menambahkan dua atribut baru pada kelas **Ortu**, yaitu **umur** dan **jenisKelamin**. Untuk menangani hal ini, konstruktor harus dimodifikasi agar dapat menerima parameter tambahan.

Variabel yang terkait:

- nama: Menyimpan nama (String).
- rambut: Menyimpan warna rambut (String).
- umur: Menyimpan umur (int).
- jenisKelamin: Menyimpan jenis kelamin (String).

[No.2] Analisis dan Argumentasi

- 1) permasalahan ini dapat diatasi dengan cara
Mengubah konstruktor untuk menerima empat parameter: nama, rambut, umur, dan jenisKelamin. Inisialisasi atribut umur dan jenisKelamin di dalam konstruktor, dan tampilkan nilai-nilainya.

[No. 2] Penyusunan Algoritma dan Kode Program


1. Algoritma:

- 1) Deklarasikan kelas Ortu dengan atribut nama, rambut, umur, dan jenisKelamin.
- 2) Buat konstruktor yang menerima empat parameter.
- 3) Inisialisasi keempat atribut tersebut di dalam konstruktor.
- 4) Cetak nilai semua atribut menggunakan konstruktor.

Kode program

```
public class Ortu {  
    // Deklarasi atribut  
    String nama, rambut, jenisKelamin;  
    int umur;  
  
    // Constructor baru dengan empat parameter  
    public Ortu(String nama, String rambut, int umur, String jenisKelamin) {  
        System.out.println("Nama saya : " + nama +  
            "\nWarna Rambut : " + rambut +  
            "\nUmur saya : " + umur +  
            "\nJenis Kelamin : " + jenisKelamin);  
    }  
  
    // Metode utama  
    public static void main(String[] args) {  
        Ortu satu = new Ortu("Putri", "hitam", 25, "P");  
    }  
}
```

Gambar 2 (input)



```
Nama saya : Putri  
Warna Rambut : hitam  
Umur saya : 25  
Jenis Kelamin : P  
|
```

Gambar 2 (output)

a) Analisa Luaran yang Dihasilkan:

- Luaran sesuai dengan yang diharapkan. Kode program berhasil menampilkan semua atribut, termasuk nama, warna rambut, umur, dan jenis kelamin. Perubahan yang dilakukan menghasilkan output yang lengkap dan sesuai dengan tambahan parameter pada konstruktor.

[No. 2] Kesimpulan

Analisa: a) Penambahan atribut **umur** dan **jenisKelamin** pada kelas **Ortu** sudah berjalan dengan baik. Konstruksi kode untuk konstruktor baru memastikan semua informasi dapat diinput dan ditampilkan secara lengkap.

b) Dasar pengambilan keputusan adalah untuk memberikan fleksibilitas lebih pada objek **Ortu**, sehingga lebih banyak informasi yang bisa dimasukkan dan ditampilkan. Dengan memperluas parameter konstruktor, informasi tambahan tentang **umur** dan **jenisKelamin** bisa dimasukkan tanpa masalah.

[No. 3] Identifikasi Masalah

1. Uraikan permasalahan dan variabel:

Pada Latihan 3.1, masalah terjadi karena konstruktor pada kelas **Manusia** salah diberi nama. Nama konstruktor seharusnya sama dengan nama kelas (**Manusia**), namun dinamai **Manusia1**. Selain itu, metode **sukaNonton** tidak dideklarasikan dengan benar, baik dari segi parameter maupun tipe pengembaliannya.

Pada Latihan 3.2, program diminta untuk diubah sesuai dengan preferensi perilaku, di mana hobi yang sebelumnya menonton film diubah menjadi membaca buku. Selain itu, program harus menampilkan jumlah halaman yang dibaca.

Variabel yang terkait:

- nama: Menyimpan nama (String).
- rambut: Menyimpan warna rambut (String).
- jumlahBuku: Menyimpan jumlah buku yang dibaca (int).
- halamanPerBuku: Menyimpan jumlah halaman per buku (int).

Pada Latihan 3.3, program mendemonstrasikan konsep method overloading dan method overriding, serta perbedaan antara metode yang mengembalikan nilai dan metode yang tidak mengembalikan nilai.

[No. 3] Analisis dan Argumentasi

Permasalahan ini dapat diatasi dengan cara:

Mengubah metode `sukaNonton` menjadi `sukaBaca`, yang akan menampilkan hobi membaca buku. Menambahkan metode kedua yang menghitung total halaman yang dibaca berdasarkan jumlah buku dan halaman per buku.

[No. 3] Penyusunan Algoritma dan Kode Program

Algoritma:

1. Deklarasikan kelas **Manusia** dengan atribut **nama** dan **rambut**.
2. Buat konstruktor untuk menginisialisasi atribut.
3. Buat dua metode **sukaBaca**: satu untuk menampilkan jenis buku yang dibaca dan satu lagi untuk menghitung total halaman yang dibaca.
4. Cetak hasil dari kedua metode tersebut.

Kode Program:

```
1 public class Manusia {
2     // Deklarasi atribut
3     String nama, rambut;
4
5     // Deklarasi constructor dengan parameter nama dan warna rambut
6     public Manusia(String nama, String rambut) {
7         System.out.println("Nama saya: " + nama +
8             "\nWarna Rambut: " + rambut);
9     }
10
11     // Method pertama: Hobi membaca buku
12     void sukaBaca(String buku) {
13         System.out.println("Hobi Membaca: " + buku);
14     }
15
16     // Method kedua: Menghitung total halaman yang dibaca
17     int sukaBaca(int jumlahBuku, int halamanPerBuku) {
18         return jumlahBuku * halamanPerBuku;
19     }
20
21     // Method utama
22     public static void main(String[] args) {
23         Manusia satu = new Manusia("Putri", "hitam");
24         satu.sukaBaca("Novel");
25         int totalHalaman = satu.sukaBaca(3, 250);
26         System.out.println("Total halaman dibaca = " + totalHalaman + " halaman");
27     }
28 }
29
```

Gambar 3 (input)

```
Nama saya: Putri
Warna Rambut: hitam
Hobi Membaca: Novel
Total halaman dibaca = 750 halaman
|
```

Gambar 3 (output)

[No. 3] Kesimpulan

Analisa: a) Konsep **method overloading** dan **method overriding** telah berhasil diimplementasikan dengan baik. Algoritma yang digunakan dapat menunjukkan perbedaan antara metode yang memiliki parameter berbeda dan metode yang diubah di kelas turunan.

b) Dasar pengambilan keputusan adalah untuk mendemonstrasikan dua konsep penting dalam pemrograman berorientasi objek, yaitu **overloading** dan **overriding**, untuk memahami fleksibilitas dan kekuatan penggunaan metode dengan parameter dan perilaku yang berbeda.

[No. 4] Identifikasi Masalah

Uraikan permasalahan dan variabel:

Pada **Latihan 4.1**, terdapat dua kelas: **Ortu** dan **Anak** yang meng-extend kelas **Ortu**. Masalah yang diidentifikasi adalah:

- **Method Overriding** yang tidak perlu: Metode **sukaMenonton(String a)** dan **sukaMembaca(String a)** di kelas **Anak** sama persis dengan metode di kelas **Ortu**, sehingga mendefinisikannya ulang di kelas **Anak** tidak memberikan manfaat.
- **Method Overloading** yang relevan: Metode **sukaMenonton(int a, String b)** di kelas **Anak** adalah contoh dari method overloading yang memperluas perilaku dari metode yang ada di kelas **Ortu**.

[No. 4] Analisis dan Argumentasi

1. **Permasalahan ini dapat diatasi dengan cara:**
 - Menghapus metode yang di-override di kelas **Anak** jika metode tersebut tidak mengalami perubahan logika dibandingkan dengan metode di kelas **Ortu**.
 - Mempertahankan metode **sukaMenonton(int a, String b)** di kelas **Anak** untuk menampilkan waktu dan jenis tontonan, yang merupakan contoh dari method overloading yang valid.

[No. 4] Penyusunan Algoritma dan Kode Program

1. **Algoritma:**
 - Deklarasikan kelas **Ortu** dengan metode **sukaMenonton(String a)** dan **sukaMembaca(String a)**.
 - Deklarasikan kelas **Anak** yang meng-extend **Ortu** dan tambahkan metode **sukaMenonton(int a, String b)**.
 - Buat objek dari kedua kelas untuk memanggil metode yang sesuai.
 - Tampilkan hasil dari pemanggilan metode untuk memverifikasi perbaikan.
2. **Kode Program:**

```

1 public class Ortu {
2     // Method di kelas induk
3     void sukaMenonton(String a) {    // method induk spesifik
4         System.out.println("Nonton " + a);
5     }
6
7     void sukaMembaca(String a) {    // method induk yang dapat diubah anak
8         System.out.println("Suka Baca " + a);
9     }
10
11     public static void main(String[] args) {
12         System.out.println("Sifat Orang Tua :");
13         Ortu objek0 = new Ortu();    // memanggil objek induk
14         objek0.sukaMenonton("Berita");    // memanggil sifat spesifik induk
15         objek0.sukaMembaca("Koran");    // memanggil method induk
16
17         System.out.println("\nSifat Anak :");
18         Anak objekA = new Anak();    // memanggil objek anak
19         objekA.sukaMenonton(9, "Film Drakor");    // memanggil method yang overload
20         objekA.sukaMembaca("Komik One Piece");    // memanggil method induk yang diwariskan
21     }
22 }
23
24 class Anak extends Ortu {
25     // Method overloading dengan parameter tambahan
26     void sukaMenonton(int a, String b) {
27         System.out.println("Nonton Jam " + a + " Malam " + b);
28     }
29 }
30

```

Gambar 4 (input)

```

Sifat Orang Tua :
Nonton Berita
Suka Baca Koran

Sifat Anak :
Nonton Jam 9 Malam Film Drakor
Suka Baca Komik One Piece
|

```

Gambar 4 (output)

a) Analisa Luaran yang Dihasilkan:

Luaran sudah sesuai dengan yang diharapkan. Program berhasil menampilkan informasi dari kelas **Ortu** dan **Anak** sesuai dengan metode yang diimplementasikan. Hasil yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data, serta tidak ada redundansi dalam mendefinisikan metode yang sama di kelas turunan.

[No. 4] Kesimpulan

Analisa: a) Program sudah diperbaiki dengan menghapus metode yang tidak perlu di-override di kelas **Anak** dan memanfaatkan method overloading dengan menambahkan parameter pada metode **sukaMenonton**. Ini menghindari redundansi dan menjaga efisiensi kode.

b) Dasar pengambilan keputusan adalah untuk menghindari pendefinisian ulang metode yang tidak mengalami perubahan logika di kelas turunan, serta memanfaatkan method overloading untuk memperluas perilaku metode yang ada dengan parameter tambahan yang relevan. Hal ini mengoptimalkan waktu eksekusi dan meminimalkan redundansi kode.

Refleksi

Minggu ini, saya memperoleh wawasan mendalam tentang konsep dasar dalam pemrograman berorientasi objek, khususnya mengenai konstruktor, metode, dan konsep polimorfisme seperti method overloading dan method overriding. Saya menyadari pentingnya kesesuaian nama konstruktor dengan nama kelas untuk memastikan bahwa objek dapat diinisialisasi dengan benar; kesalahan dalam penamaan konstruktor, seperti yang terjadi pada latihan awal, dapat menyebabkan kegagalan saat pembuatan objek. Selain itu, saya belajar bahwa method overloading memberikan fleksibilitas dalam pemanggilan metode dengan nama yang sama namun parameter yang berbeda, seperti yang terlihat dalam perubahan metode `sukaBaca` dari `sukaNonton` untuk menyesuaikan dengan perilaku baru. Method overriding, di sisi lain, memungkinkan subclass untuk memberikan implementasi khusus dari metode superclass, namun jika tidak ada perubahan logika yang berarti, hal ini bisa menyebabkan redundansi kode yang tidak perlu, seperti yang terjadi pada kelas `Ortu` dan `Anak`.

Latihan tentang penambahan atribut baru, seperti umur dan jenisKelamin, pada kelas memberikan pemahaman tentang cara memodifikasi konstruktor untuk memenuhi kebutuhan data yang lebih kompleks, menunjukkan pentingnya perancangan konstruktor yang fleksibel. Salah satu tantangan utama yang dihadapi adalah mengidentifikasi dan memperbaiki kesalahan dalam deklarasi metode dan konstruktor, termasuk kesalahan penamaan dan metode yang tidak perlu di-override, yang memerlukan perhatian detail untuk memastikan fungsi kode yang tepat. Saya juga harus mempertimbangkan bagaimana mengoptimalkan kode dengan menghindari redundansi dan memanfaatkan fitur OOP secara efektif.

Secara keseluruhan, minggu ini memperdalam pemahaman saya tentang desain dan implementasi kelas dalam pemrograman berorientasi objek. Pengalaman ini membantu saya dalam memperbaiki kesalahan dan meningkatkan keterampilan dalam menulis kode yang lebih efektif dan efisien, serta menekankan pentingnya pengelolaan metode dan efisiensi kode dalam pengembangan perangkat lunak.