

Nama & NPM	Topik:	Tanggal:
Aditya Bagas Setiawan(G1F024051)	Class/Kelas	16/09/2024
[Nomor 1] Identifikasi Masalah:		
1.1. Perbaiki pesan kesalahan Contoh 1! 1.2. Analisa ciri-ciri lain Kelas Manusia yang dapat menjadi a. atribut variabel, dan b. perilaku/ behavior!		
[Nomor 1] Analisis dan Argumentasi		
<div>1.1</div> <div>Contoh 1:</div> <pre> public class Manusia { // deklarasi kelas //deklarasi atribut Manusia dalam variabel String nama, rambut; //deklarasi constructor public Manusia1 (String nama) { System.out.println(" Nama saya : "+ nama + "\n Warna Rambut : " + rambut); } //deklarasi method utama public static void main(String[] args) { Manusia1 satu = new Manusia1("Putri", "hitam"); } } </pre> <p>(Pada contoh yang diberikan terdapat beberapa kesalah,yang pertama pada nama kelas tidak dan constructur berbeda karena hal itu kode program menjadi eror dan tidak mengeluarkan luaran yang semestinya. Yang kedua,pada deklarasi constructor hanya terdapat String nama tetapi kode program tersebut ingin menginialisasi dua atribut(nama dan rambut. Yang ketiga,dalam constructor kita perlu menyimpan nilai menggunakan <i>this.nama</i> dan <i>this.rambut</i> untuk membedakan antara parameter dan atribut.)</p> <p>Setelah saya melakukan perbaikan pada contoh 1,berikut ini adalah kode program dan luaran yang dapat berjalan dan tidak mengalami error</p>		

```

1 public class Manusia { // deklarasi kelas
2     //deklarasi atribut Manusia dalam variabel
3     String nama, rambut;
4
5     //deklarasi constructor
6     public Manusia (String nama,String rambut) {
7         this.nama = nama;
8         this.ambut = rambut;
9         System.out.println(" Nama saya : "+ nama +
10             "\n Warna Rambut : " + rambut);
11     }
12
13     //deklarasi method utama
14     public static void main( String[] args) {
15         Manusia satu = new Manusia("Putri", "hitam");
16     }
17 }

```

```

Nama saya : Putri
Warna Rambut : hitam

```

Compiled and executed in 1.835 sec(s)

1.2

Ciri ciri kelas lain pada kelas manusia:

1.Atribut variabel

- a.umur : (int umur)
- b.tinggi badan : (double tinggi)
- c.alamat : (String alamat)

2.Behavior/perilaku

- a.Berbicara : metode yang bisa menampilkan perilaku berbicara dan menyampaikan pesan dari perilaku tersebut.

```

1 public void bicara(String pesan) {
2     System.out.println(nama + " berkata: " + pesan);
3 }

```

- b.Tidur : metode yang bisa menampilkan perilaku tidur dan waktu tidur

```

1 public void tidur(int jam) {
2     System.out.println(nama + " tidur selama " + jam + " jam.");
3 }

```

[Nomor 1] Kesimpulan

Kode program dalam kelas Manusia bertujuan untuk menampilkan nama dan jenis rambut,walaupun masih banyak kesalahan pada kode program tersebut.

Setelah perbaikan, kelas manusia memiliki dua atribut utama yaitu *nama* dan *rambut* yang sebelumnya hanya satu atribut yaitu *nama*, sehingga mampu mencetak informasi melalui constructor.program ini menunjukkan bagaimana kelas Manusia dapat dibuat dan diinisialisasi, serta bagaimana dapat ditampilkan.

Jika kita mengembangkan kode tersebut lebih lanjut,kelas ini dapat ditambahkan atribut dan metode lain yang menunjukkan karakteristik dan perilaku manusia.

Nama & NPM	Topik:	Tanggal:
Aditya Bagas Setiwan(G1F024051)	Objek	(16/09/2024)

[Nomor 2] Identifikasi Masalah:

- 2.1. Evaluasi penyebab kesalahan dan perbaiki kode tersebut!
- 2.2. Apabila Ortu memiliki data variabel umur = 25 dan jenis kelamin = P (untuk Perempuan), rekomendasikan constructor dengan parameter yang baru untuk ditambahkan dalam program!

[Nomor 2] Analisis dan Argumentasi

2.1

Pada contoh kode program masih ditemukan masalah sebagai berikut:

1. Nama Constructor Salah: Nama constructor seharusnya sama dengan nama kelas. Dalam kode, nama constructor adalah ortu, yang tidak sesuai dengan nama kelas Ortu. Harusnya nama constructor adalah Ortu.
2. Parameter Constructor Tidak Didefinisikan: Constructor harus didefinisikan dengan parameter yang sesuai dan menyimpan nilai ke dalam variabel instance.
3. Variabel Instance Tidak Dideklarasikan: Sehingga perlu mendeklarasikan variabel instance seperti nama dan rambut di dalam kelas.

Setelah saya melakukan perbaikan kode program akan terlihat seperti:

```

1 public class Ortu {
2     // Deklarasi variabel
3     private String nama;
4     private String rambut;
5
6     // Constructor dengan parameter
7     public Ortu(String nama, String rambut) {
8         this.nama = nama;
9         this.ambut = rambut;
10        System.out.println("Nama saya: " + nama + "\nWarna Rambut: " + rambut);
11    }
12
13    public static void main(String[] args) {
14        Ortu satu = new Ortu("Putri", "hitam");
15    }
16 }
17

```

```

Nama saya: Putri
Warna Rambut: hitam

```

Setelah dilakukan perbaikan kode tersebut dapat berjalan dengan baik dan sesuai yang saya harapkan.

2.2

ika kita ingin menambahkan variabel umur dan jenisKelamin, kita perlu menambahkan variabel tersebut dan memperbarui constructor serta method untuk mencetak informasi.

Perubahan yang saya lakukan:

- Tambahkan variabel umur dan jenisKelamin.
- Tambahkan constructor baru dengan parameter yang sesuai.
- Perbarui constructor untuk menerima parameter baru dan menyimpannya ke dalam variabel instance.

```
1 public class Ortu {
2     // Deklarasi variabel
3     private String nama;
4     private String rambut;
5     private int umur;
6     private char jenisKelamin;
7
8     // Constructor dengan parameter
9     public Ortu(String nama, String rambut, int umur, char jenisKelamin) {
10        this.nama = nama;
11        this.rambut = rambut;
12        this.umur = umur;
13        this.jenisKelamin = jenisKelamin;
14        System.out.println("Nama saya: " + nama +
15                           "\nWarna Rambut: " + rambut +
16                           "\nUmur: " + umur +
17                           "\nJenis Kelamin: " + (jenisKelamin == 'P' ? "Perempuan" : "Laki-laki"));
18    }
19
20    public static void main(String[] args) {
21        Ortu satu = new Ortu("Putri", "hitam", 25, 'P');
22    }
23 }
```

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

[Nomor 2] Kesimpulan

Pada soal 1.1 masalah yang saya temukan adalah Nama constructor tidak sesuai dengan nama kelas (ortu harus Ortu), tidak ada parameter dalam constructor, dan variabel instance (nama dan rambut) tidak dideklarasikan. Dan perbaikan yang saya lakukan untuk mengatasi masalah tersebut dengan cara,nama constructor diubah menjadi Ortu, variabel instance nama dan rambut dideklarasikan, dan constructor diperbarui untuk menerima dan menyimpan parameter nama dan rambut.

Pada soal 1.2 saya melakukan penambahan variabel umur dan jenisKelamin pada kelas.dan juga melakukan perubahan dengan membuat constructor baru dengan parameter tambahan (umur dan jenisKelamin), dan perbarui output untuk mencetak informasi lengkap termasuk umur dan jenis kelamin.

Nama & NPM	Topik:	Tanggal:
Aditya Bagas Setiawan(G1F024051)	Method	16/09/2024

[Nomor 3] Identifikasi Masalah:

- 3.1. Evaluasi penyebab kesalahan dan perbaiki kode tersebut!
- 3.2. Ubahlah method dan constructor Contoh 3 sesuai dengan perilaku/ behavior anda
- 3.3. Berdasarkan Contoh 3 dan Latihan 3.2. simpulkan perbedaan:
 - a) constructor overloading dan overriding
 - b) method overloading, dan method overriding
 - c) method yang mengembalikan nilai dan method tidak mengembalikan nilai

[Nomor 3] Analisis dan Argumentasi

3.1

```
public class Manusia {  
    //deklarasi atribut Manusia dalam variabel  
    String nama, rambut;  
  
    //deklarasi constructor  
    public Manusia1(String nama, String rambut) {  
        System.out.println(" Nama saya : "+ nama +  
            "\n Warna Rambut : " + rambut);  
    }  
  
    //deklarasi method  
    void sukaNonton {  
        System.out.println(" Hobi Menonton : " + film);  
    }  
  
    int sukaNonton {  
        episode*durasi;  
    }  
  
    //deklarasi method utama  
    public static void main( String[] args) {  
        Manusia satu = new Manusia("Putri", "hitam");  
        satu.sukaNonton("Drakor");  
        int jumlahJam = satu.sukaNonton(2, 2);  
        System.out.println("Jam nonton = " +jumlahJam + " jam");  
    }  
}
```

Terdapat beberapa kesalahan pada kode program yang ada di contoh. Yang pertama konstruktor harus memiliki nama yang sama dengan nama kelas, contoh yang diberikan memiliki perbedaan pada konstruktor (publicmanusia1) dan nama kelasnya (Manusia). Yang kedua pada deklarasi void *suka Nonton*, tidak ada tanda kurung pada nama metode ini. Yang ketiga metode *sukanonton()* mencoba menggunakan variable *film*, namun variabel ini tidak didefinisikan. Yang keempat metode *sukaNonton* dideklarasikan dengan nama yang sama tetapi memiliki tipe data int dan tidak ada parameter yang sesuai. Untuk overloading metode, kita harus memberikan parameter yang berbeda. Yang kelima pada bagian *satu.sukanonton("Drakor")*, metode *sukaNonton* harus memiliki parameter yang sesuai.

Setelah melakukan perbaikan, berikut kode program yang saya perbaiki:

```

1 public class Manusia {
2     // Deklarasi atribut Manusia
3     String nama, rambut;
4
5     // Deklarasi constructor
6     public Manusia(String nama, String rambut) {
7         this.nama = nama;
8         this.ambut = rambut;
9         System.out.println("Nama saya : " + nama +
10             "\nWarna Rambut : " + rambut);
11     }
12
13     // Deklarasi method pertama
14     void sukaNonton(String film) {
15         System.out.println("Hobi Menonton: " + film);
16     }
17
18     // Deklarasi method kedua
19     int sukaNonton(int episode, int durasi) {
20         return episode * durasi;
21     }
22
23     // Deklarasi method utama
24     public static void main(String[] args) {
25         Manusia satu = new Manusia("Putri", "hitam");
26         satu.sukaNonton("Drakor");
27         int jumlahJam = satu.sukaNonton(2, 2);
28         System.out.println("Jam nonton = " + jumlahJam + " jam");
29     }
30 }
31

```

```

Nama saya : Putri
Warna Rambut : hitam
Hobi Menonton: Drakor
Jam nonton = 4 jam

```

3.2

```

1 public class Manusia {
2     // Deklarasi atribut Manusia
3     String nama, umur, hobi;
4
5     // Deklarasi konstruktor
6     public Manusia(String nama, String umur, String hobi) {
7         this.nama = nama;
8         this.umur = umur;
9         this.hobi = hobi;
10        System.out.println("Nama saya : " + nama +
11            "\nUmur : " + umur +
12            "\nHobi : " + hobi);
13    }
14
15    // Deklarasi method untuk olahraga
16    void sukaOlahraga(String olahraga) {
17        System.out.println("Saya suka olahraga " + olahraga);
18    }
19
20    // Deklarasi method untuk menghitung total waktu olahraga
21    int sukaOlahraga(int perhari, int lamaOlahraga) {
22        return perhari * lamaOlahraga;
23    }
24
25    // Deklarasi method utama
26    public static void main(String[] args) {
27        Manusia satu = new Manusia("Aditya", "18 Tahun", "bermain game");
28
29        // Memanggil method sukaOlahraga
30        satu.sukaOlahraga("Bersepeda");
31        int totalOlahraga = satu.sukaOlahraga(1, 2);
32        System.out.println("Total waktu olahraga = " + totalOlahraga + " jam/hari");
33    }
34 }
35

```

```

Nama saya : Aditya
Umur : 18 Tahun
Hobi : bermain game
Saya suka olahraga Bersepeda
Total waktu olahraga = 2 jam/hari
|

```

3.3

a) Constructor Overloading dan Constructor Overriding

-Constructor Overloading: Terjadi ketika ada lebih dari satu konstruktor dalam satu kelas dengan parameter yang berbeda-beda. Misalnya, bisa ada konstruktor tanpa parameter dan konstruktor dengan beberapa parameter.

-Constructor Overriding: Tidak ada konsep overriding pada konstruktor karena konstruktor tidak diwariskan seperti method biasa.

b) Method Overloading dan Method Overriding

-Method Overloading: Terjadi ketika ada beberapa metode dengan nama yang sama tetapi parameter berbeda (jumlah atau tipe). Tujuannya adalah memungkinkan metode yang sama dipanggil dengan variasi argumen yang berbeda.

-Method Overriding: Terjadi ketika subclass menyediakan implementasi spesifik untuk metode yang sudah ada di superclass. Tujuannya untuk mengubah perilaku metode yang diwariskan dari kelas induk.

c) Method yang Mengembalikan Nilai dan Method Tanpa Mengembalikan Nilai

-Method yang Mengembalikan Nilai: Metode ini mengembalikan sebuah nilai kepada pemanggilnya dan biasanya memiliki tipe pengembalian (misalnya, int, String, double).

-Method Tanpa Mengembalikan Nilai: Metode ini tidak mengembalikan nilai apa pun dan menggunakan kata kunci void.

[Nomor 3] Kesimpulan

Saya mempelajari cara memperbaiki kesalahan kode Java seperti penamaan konstruktor, penggunaan metode yang tepat, dan variabel yang sesuai. Selain itu, saya memodifikasi kode untuk menambahkan fitur baru (seperti hobi olahraga) dan memahami perbedaan antara *constructor overloading*, *method overloading*, *method overriding*, serta metode yang mengembalikan nilai dan yang tidak. Soal-soal ini menguatkan konsep dasar tentang sintaks, fleksibilitas kode, serta perbedaan penggunaan dan pengelolaan metode dalam pemrograman Java.

Nama & NPM	Topik:	Tanggal:
Aditya Bagas Setiawan(G1F024051)	Extends	16/09/2024

[Nomor 4] Identifikasi Masalah:

- 4.1. Evaluasi method yang dimiliki class Anak extends Ortu dengan method di class Ortu!
Apakah penulisan method ini sudah efisien?
- 4.2. Setelah dirunning di JDoodle, catat waktu eksekusinya.
Rekomendasikan perbaikan penulisan kode method untuk dapat mengefisienkan waktu eksekusi!

[Nomor 4] Analisis dan Argumentasi

4.1

Kelas Ortu:

Metode `sukaMenonton(String a)` dan `sukaMembaca(String a)` cukup sederhana dan jelas. Tidak ada masalah dengan efisiensi karena implementasinya langsung sesuai dengan fungsinya.

Kelas Anak:

Metode Overloading:

- `sukaMenonton(int a, String b)` dan `sukaMenonton(String a)`: Memiliki dua versi dari metode `sukaMenonton` di kelas Anak. Ini adalah contoh overloading metode, yang biasanya baik untuk memberikan berbagai cara untuk melakukan aksi yang sama dengan parameter berbeda.
- Namun, metode `sukaMenonton(String a)` di kelas Anak memiliki implementasi yang sama dengan kelas Ortu. Jika metode ini tidak mengalami perubahan, maka bisa dihindari pengulangannya di kelas Anak dan cukup menggunakan yang ada di kelas Ortu.

Metode Override:

- `sukaMembaca(String a)` di kelas Anak: Ini meng-override metode yang sama dari kelas Ortu. Jika implementasinya identik, tidak ada keuntungan dari overriding kecuali ada rencana untuk mengubah implementasinya di masa depan.

Untuk meningkatkan efisiensi, berikut adalah langkah-langkah yang bisa diambil:

- Hapus Metode yang Redundan: Jika `sukaMenonton(String a)` dan `sukaMembaca(String a)` di kelas Anak tidak memiliki perubahan, hapus metode ini dari kelas Anak dan gunakan implementasi dari kelas Ortu.
- Gunakan Polimorfisme Secara Bijaksana: Gunakan overriding hanya jika perlu menyesuaikan perilaku di subclass.

Dengan menghapus metode yang tidak diperlukan atau redundan, kode menjadi lebih bersih dan lebih mudah untuk dirawat.

4.2

```
public class Ortu { // membuat kelas induk
    void sukaMenonton(String a) { // method induk spesifik
        System.out.println("Nonton " + a);
    }
    void sukaMembaca(String a) { // method induk umum bisa diubah anak
        System.out.println("Suka Baca " + a);
    }
}

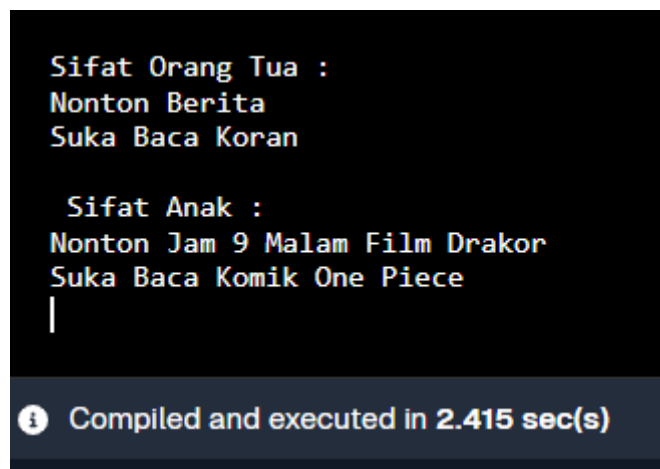
public static void main(String [] args) {
    System.out.println("Sifat Orang Tua :");
    Ortu objekO = new Ortu(); // memanggil objek induk
    objekO.sukaMenonton("Berita"); // memanggil sifat spesifik induk
    objekO.sukaMembaca("Koran"); // memanggil method dengan variabel dapat diubah

    System.out.println("\n Sifat Anak :");
    Anak objekA = new Anak(); //memanggil objek anak
    objekA.sukaMenonton(9, "Film Drakor"); //memanggil sifat spesifik anak yang diturunkan induk
    objekA.sukaMembaca("Komik One Piece"); //memanggil method ke induk yang otomatis diturunkan tanpa deklarasi ulang di anak
}

class Anak extends Ortu {
    void sukaMenonton(int a, String b) {
        System.out.println("Nonton Jam " + a + " Malam " + b);
    }
    void sukaMenonton(String a) { // method induk spesifik
        System.out.println("Nonton " + a);
    }
    void sukaMembaca(String a) { // method induk umum bisa diubah anak
        System.out.println("Suka Baca " + a);
    }
}

public static void main(String [] args) {
    System.out.println("Sifat Orang Tua :");
    Ortu objekO = new Ortu(); // memanggil objek induk
    objekO.sukaMenonton("Berita"); // memanggil sifat spesifik induk
    objekO.sukaMembaca("Koran"); // memanggil method dengan variabel dapat diubah

    System.out.println("\n Sifat Anak :");
    Anak objekA = new Anak(); //memanggil objek anak
    objekA.sukaMenonton(9, "Film Drakor"); //memanggil sifat spesifik anak yang diturunkan induk
    objekA.sukaMembaca("Komik One Piece"); //memanggil method ke induk yang otomatis diturunkan tanpa deklarasi ulang di anak
} }
```



```
Sifat Orang Tua :
Nonton Berita
Suka Baca Koran

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

Compiled and executed in 2.415 sec(s)
```

Menurut saya dengan waktu 2.4 detik kode program tersebut masih bisa di sederhanakan supaya dapat menghemat waktu eksekusi.

Berikut ini adalah contoh kode program yang saya buat untuk mengefesiensi waktu eksekusi:

```

1 public class Ortu {
2     void sukaMenonton(String a) {
3         System.out.println("Nonton " + a);
4     }
5
6     void sukaMembaca(String a) {
7         System.out.println("Suka Baca " + a);
8     }
9
10    public static void main(String[] args) {
11        System.out.println("Sifat Orang Tua :");
12        Ortu objekO = new Ortu();
13        objekO.sukaMenonton("Berita");
14        objekO.sukaMembaca("Koran");
15
16        System.out.println("\nSifat Anak :");
17        Anak objekA = new Anak();
18        objekA.sukaMenonton(9, "Film Drakor");
19        objekA.sukaMembaca("Komik One Piece");
20    }
21 }
22
23 class Anak extends Ortu {
24     void sukaMenonton(int a, String b) {
25         System.out.println("Nonton Jam " + a + " Malam " + b);
26     }
27 }
28

```

```

Sifat Orang Tua :
Nonton Berita
Suka Baca Koran

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

```

Compiled and executed in 1.346 sec(s)

Dengan menyederhanakan kode tersebut dapat memotong waktu sekitar 1.2 detik ,sekitar 50% waktu yang dibutuhkan untuk menjalankan program pada contoh.

[Nomor 4] Kesimpulan

Dalam evaluasi penulisan metode di kelas Ortu dan Anak, ditemukan bahwa meskipun implementasi metode sudah sesuai fungsi dasar, terdapat potensi perbaikan untuk meningkatkan efisiensi:

1. Efisiensi Metode:

- Overloading: Metode sukaMenonton(int a, String b) di kelas Anak efektif untuk parameter berbeda, namun duplikasi sukaMenonton(String a) di kelas Anak yang sama dengan di kelas Ortu tidak diperlukan jika implementasinya identik.
- Override: Metode sukaMembaca(String a) di kelas Anak yang meng-override kelas Ortu dengan implementasi yang sama tidak efisien tanpa perubahan.

2. Rekomendasi Perbaikan:

- Hapus Duplikasi Kode: Singkirkan metode di subclass yang identik dengan superclass untuk menghindari duplikasi.
- Optimalisasi Kode: Mengurangi duplikasi dan memastikan konsistensi kode dapat meningkatkan keterbacaan dan mempermudah pemeliharaan.
- Pengujian: Pastikan untuk menguji aplikasi setelah melakukan perubahan untuk memastikan fungsionalitas tetap sesuai harapan.

Dengan mengimplementasikan perbaikan ini, efisiensi kode dapat ditingkatkan dan kode akan lebih bersih serta lebih mudah dikelola.

REFLEKSI

Dengan mengerjakan soal *kelas*, *objek*, *method* dan *extends* saya mempelajari hal baru yang sebelumnya tidak saya ketahui sedikitpun dan juga menurut saya pembelajaran minggu ini tergolong seru karena dapat memecahkan masalah dengan mengerjakan soal langsung.