

Nama & NPM	Topik:	Tanggal:
Zahra Sari Fhadilah G1F024025	Kelas (Class)	12 September 2024

[1] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variable

Contoh 1:

```
public class Manusia { // deklarasi kelas
    // deklarasi variabel
    String nama;
    String rambut;

    // deklarasi constructor tanpa parameter
    public Manusia() {
        System.out.println("Kelas Manusia tanpa nama");
    }
}
```

Latihan 1:

- 1.1. Analisa ciri-ciri umum Kelas Manusia yang dapat menjadi
 - a. atribut variabel, dan
 - b. perilaku/ behavior untuk method!

- 2) Rincikan sumber informasi yang relevan (buku / webpage)

Video Materi 1 tentang Kelas, Objek, Method –
<https://www.youtube.com/watch?v=60IdOc8m8Es>

[1.] Analisis dan Argumentasi

- 1) Analisis

- a. Atribut variable

Atribut variable merupakan karakteristik yang dimiliki suatu objek, atribut ini mendefinisikan keadaan atau informasi yang berkaitan dengan objek tersebut. Pada soal terdapat atribut variable berupa nama dan rambut.

- Nama : Atribut ini menyimpan nama orang. Tipe data yang digunakan adalah String.
- Rambut : Atribut ini menyimpan informasi tentang warna rambut. Tipe data yang digunakan juga String

- b. Perilaku/ behavior untuk method

Perilaku atau behavior merujuk pada tindakan atau fungsi yang dapat dilakukan oleh objek. Hal ini diimplementasikan dalam bentuk method dalam kelas, metode ini memungkinkan objek untuk melakukan operasi tertentu dan berinteraksi dengan atribut atau objek lainnya.

- Makan : Metode yang menggambarkan Tindakan makan, bisa menerima parameter seperti jenis makanan.

2) Argumentasi

Untuk mengetahui atribut variable dan perilaku/behavior untuk method kita harus memahami kode program dengan baik. Karena kode program ini akan lebih mudah di Analisa dan dipahami Ketika kita paham dan mengerti apa yang harus dilakukan.

[1) Kode Program

1) Tuliskan kode program dan luaran

a) Beri komentar pada kode

```
1 public class Orang { // deklarasi kelas
2     // deklarasi variabel
3     String nama , rambut;
4
5     // deklarasi constructor
6     public Orang(String nama , String rambut) {
7         this.nama = nama;
8         this.rambut = rambut;
9         System.out.println("Nama : " + nama + "\n Warna Rambut : " + rambut);
10    }
11
12    // Metod untuk makan
13    public void makan(String makanan) {
14        System.out.println(nama + " sedang makan " + makanan + ".");
15    }
16
17    // Deklarasi method utama
18    public static void main(String[] args) {
19        Orang orang1 = new Orang("Zahra Sari Fhadilah" , "hitam");
20        orang1.makan("bakso");
21    }
22 }
```

Kode program menunjukkan luaran yang sudah sesuai dengan yang diinginkan, disana saya menambahkan perilaku/ behavior untuk method

b) luaran yang dihasilkan

Output Generated Files

```
Nama : Zahra Sari Fhadilah
Warna Rambut : hitam
Zahra Sari Fhadilah sedang makan bakso.
```

i CPU Time: 0.07 sec(s) | Memory: 39124 kilobyte(s) | Compiled and executed in 1.062 sec(s)

Luaran menunjukkan perilaku/ behavior dari objek yaitu sedang makan bakso

[1.] Kesimpulan

- 1) Pada struktur kode program saya menggunakan atribut variable berupa nama dan rambut dengan menggunakan tipe data String. Saya juga menambahkan perilaku/ behavior untuk method yang dilakukan oleh objek, dimana objek itu bernama Zahra Sari Fhadilah dan memiliki rambut warna hitam sebagai karakteristiknya, dan sedang makan bakso sebagai perilaku/ behavior untuk methodnya.

Hal ini dapat dilihat dari luaran(outputnya) :

Nama : Zahra Sari Fhadilah

Warna Rambut : hitam

Zahra Sari Fhadilah sedang makan bakso.

Nama & NPM	Topik:	Tanggal:
Zahra Sari Fhadilah G1F024025	Objek	12 September 2024
[2.] Identifikasi Masalah:		
<p>1) Uraikan permasalahan dan variable</p> <p>Contoh 2: Salin dan tempel kode program berikut ke Eclipse atau JDoodle.</p> <pre> public class Ortu { //deklarasi constructor public Ortu(String nama, String rambut) { //nama dan rambut adalah variabel constructor System.out.println(" Nama saya : "+ nama + "\n Warna Rambut : " + rambut); } public static void main (String[] args) { Ortu satu = new Ortu("Putri", "hitam"); } } </pre> <p>Luaran 2: Nama saya : Putri Warna Rambut : hitam</p> <p>Latihan 2: 2.1. Susun kembali kode di contoh 2 dengan menambahkan data ciri-ciri Anda di dalam variabel constructor! 2.2. Apabila nanti Anda akan memiliki keturunan, analisa sifat (atribut), constructor, dan perilaku positif (behavior) apa yang akan diturunkan?</p> <p>2) Rincikan sumber informasi yang relevan (buku / webpage) <u>Video Materi 1 tentang Kelas, Objek, Method –</u> https://www.youtube.com/watch?v=60IdOc8m8Es</p>		
[2.] Analisis dan Argumentasi		
<p>1) Analisis</p> <p>Apabila nanti Anda akan memiliki keturunan, analisa sifat (atribut), constructor, dan perilaku positif (behavior) apa yang akan diturunkan?</p> <p>Pewarisan sifat atau elemen yang diturunkan memungkinkan satu kelas untuk mewarisi atribut dan perilaku dari kelas lain. Berdasarkan analisis terhadap kode program yang saya buat, beberapa hal yang dapat diturunkan kepada keturunan adalah sebagai berikut:</p> <ul style="list-style-type: none"> • Sifat Salah satu sifat (atribut) yang bisa diwarisi oleh anak dari kelas induk adalah warna rambut, misalnya, anak dapat mewarisi rambut berwarna hitam dari orang tua. • Constructor Constructor akan digunakan Ketika ingin membuat objek dari kelas anak untuk menginisialisasi atribut yang akan diwariskan. • Perilaku positif (behavior) Hal ini merujuk pada perilaku positif yang dimiliki orang tua (kelas induk) akan menurun ke perilaku anak-anaknya (kelas anak). Misalnya sikap empati, 		

jika orang tua memiliki sifat ini maka anaknya cenderung akan meniru sikap empati yang dimiliki oleh orang tuanya.

[2.] Kode Program

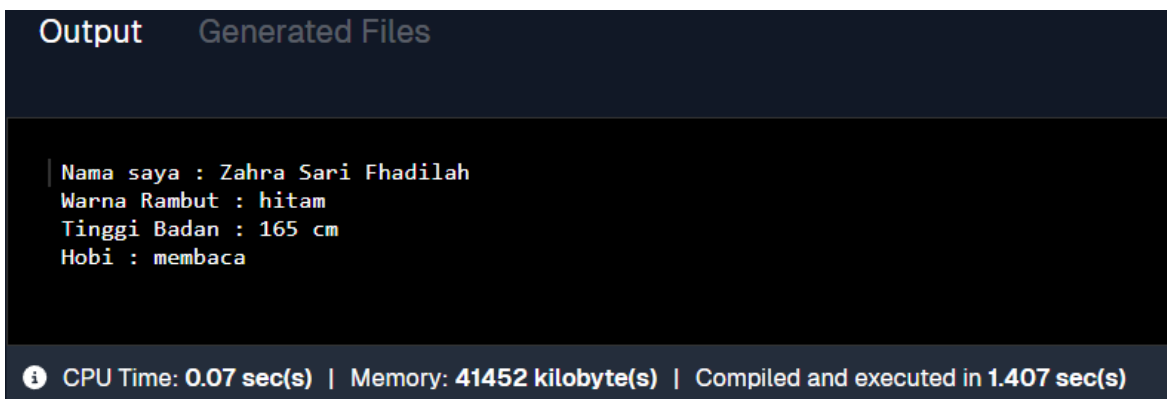
1) Tuliskan kode program dan luaran

a) Beri komentar pada kode

```
1 public class Watashi {
2     // Deklarasi constructor dengan atribut tambahan
3     public Watashi(String nama, String rambut, int tinggi, String hobi) {
4         // Menampilkan informasi yang dimasukkan
5         System.out.println(" Nama saya : " + nama +
6                             "\n Warna Rambut : " + rambut +
7                             "\n Tinggi Badan : " + tinggi + " cm" +
8                             "\n Hobi : " + hobi);
9     }
10
11     public static void main(String[] args) {
12         // Membuat objek dengan atribut tambahan
13         Watashi satu = new Watashi("Zahra Sari Fhadilah", "hitam", 165, "membaca");
14     }
15 }
```

Pada kode program saya menambahkan beberapa atribut di dalam variabel constructor yaitu tinggi dan hobi. Dengan int sebagai tipe data dari tinggi dan String sebagai tipe data dari hobi.

b) Luaran yang dihasilkan



```
Output    Generated Files

Nama saya : Zahra Sari Fhadilah
Warna Rambut : hitam
Tinggi Badan : 165 cm
Hobi : membaca

CPU Time: 0.07 sec(s) | Memory: 41452 kilobyte(s) | Compiled and executed in 1.407 sec(s)
```

Luaran menunjukkan ciri ciri yang saya miliki, luaran sudah sesuai dengan kode program yang ada.

[2.] Kesimpulan

- 1) Pada kode program kita bisa mengetahui bahwa dalam pemrograman kita berorientasi objek dapat dipakai untuk mempresentasikan pewarisan sifat dan perilaku seperti kehidupan nyata.
- 2) Hal ini memberikan fleksibilitas dalam pemrograman dengan memungkinkan penggunaan Kembali kode, sekaligus memberi ruang bagi kelas anak untuk menambahkan atau memodifikasi sifat dan perilaku sesuai kebutuhan.

Nama & NPM	Topik:	Tanggal:
Zahra Sari Fhadilah G1F024025	Method	12 September 2024

[3.] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variabel

Contoh 3: Salin dan tempel kode program berikut ke Eclipse atau JDoodle.

```
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(String film) {
        System.out.println(" Hobi Menonton : " + film);
    }

    //deklarasi method utama
    public static void main( String[] args) {
        Manusia satu = new Manusia("Putri", "hitam");
        satu.sukaNonton("Drakor");
    }
}
```

Luaran 3:

```
Nama saya : Putri
Warna Rambut : hitam
Hobi Menonton : Drakor
```

Latihan 3:

- 3.1. Analisa perbedaan deklarasi constructor, method, dan method utama!
- 3.2. Tentukan kapan Anda perlu menggunakan constructor dan method?
- 3.3. Uraikan perbedaan berikut:
 - a) constructor overloading dan overriding
 - b) method overloading, dan method overriding
 - c) method yang mengembalikan nilai dan method tidak mengembalikan nilai

- 2) Rincikan sumber informasi yang relevan (buku / webpage)

Video Materi 1 tentang Kelas, Objek, Method –
<https://www.youtube.com/watch?v=60IdOc8m8Es>

[3.] Analisis dan Argumentasi

1) Analisis

a) Analisa perbedaan deklarasi constructor, method, dan method utama

- **Constructor**
Merupakan method khusus yang digunakan untuk menginisialisasi objek Ketika dibuat. Dalam contoh kode program, constructor dinyatakan sesuai dengan nama kelas, misal nama kelasnya adalah Manusia maka constructor juga memiliki nama Manusia. Constructor dipanggil secara otomatis saat objek baru dibuat.
- **Method**
Merupakan blok kode yang bisa dieksekusi oleh objek untuk melakukan tindakan tertentu, method memiliki nama yang bisa bebas dipilih dan bisa dipanggil kapanpun oleh objek.
- **Method utama**
Merupakan titik awal eksekusi dalam program di Java, method utama merupakan method yang akan dijalankan pertama kali Ketika program dijalankan.

b) Kapan constructor dan method perlu digunakan

- **Constructor**
Digunakan untuk menginisialisasi objek saat pertama kali dibuat, jika suatu objek memerlukan nilai awal untuk atribut-atributnya maka constructor diperlukan.
- **Method**
Digunakan Ketika ingin objek melakukan suatu Tindakan setelah objek dibuat, method bisa dipanggil kapan saja sesuai kebutuhan.

c) Uraikan perbedaan berikut

- **Constructor overloading dan overriding**
Constructor overloading terjadi ketika sebuah kelas memiliki beberapa constructor dengan parameter yang berbeda, memungkinkan objek dibuat dengan cara yang berbeda. Namun, constructor tidak bisa di-*override* karena tidak diwariskan oleh subclass, sehingga konsep overriding tidak berlaku untuk constructor.
- **Method overloading dan overriding**
Method overloading adalah saat kita membuat beberapa method dengan nama yang sama tetapi berbeda parameter dalam satu kelas. Ini memungkinkan pemanggilan method dengan berbagai input. Di sisi lain, method overriding adalah ketika sebuah subclass menyediakan implementasi yang berbeda untuk method yang sudah ada di superclass, menggunakan nama dan parameter yang sama, namun dengan perilaku yang berbeda.
- **Method yang mengembalikan nilai dan method yang tidak mengembalikan nilai**
Method yang mengembalikan nilai memberikan output berupa tipe data tertentu setelah eksekusi dan menggunakan return untuk mengirimkan nilai tersebut. Sementara itu, method yang tidak mengembalikan nilai menggunakan tipe void dan hanya melakukan aksi tertentu tanpa memberikan nilai balik.

[3.] Kode Program

1) Tuliskan kode program dan luaran

a) Kode

```
1 public class Manusia {  
2     //deklarasi atribut Manusia dalam variabel  
3     String nama, rambut;  
4  
5     //deklarasi constructor  
6     public Manusia(String nama, String rambut) {  
7         System.out.println(" Nama saya : " + nama +  
8         "\n Warna Rambut : " + rambut);  
9     }  
10  
11     //deklarasi method  
12     void sukaNonton(String film) {  
13         System.out.println(" Hobi Menonton : " + film);  
14     }  
15  
16     //deklarasi method utama  
17     public static void main( String[] args) {  
18         Manusia satu = new Manusia("Putri", "hitam");  
19         satu.sukaNonton("Drakor");  
20     }  
21 }
```

b) Luaran yang dihasilkan

Output Generated Files

```
Nama saya : Putri  
Warna Rambut : hitam  
Hobi Menonton : Drakor
```

i CPU Time: 0.05 sec(s) | Memory: 39012 kilobyte(s) | Compiled and executed in 1.244 sec(s)

Kode program yang dijalankan sudah sesuai dengan hasil luaran atau output yang ditampilkan oleh sistem

[3.] Kesimpulan

- 1) Dari sini, saya memahami bahwa ada perbedaan mendasar antara constructor dan method. Constructor secara otomatis dijalankan saat objek dibuat dan berfungsi untuk menginisialisasi nilai awal objek, sementara method dapat dipanggil kapan saja untuk menjalankan tindakan tertentu pada objek.
- 2) Saya juga menyadari bahwa overloading memungkinkan baik constructor maupun method memiliki versi berbeda berdasarkan parameter yang diterima, sedangkan overriding hanya berlaku untuk method dan memungkinkan subclass mengganti implementasi dari superclass. Selain itu, method dapat dikelompokkan menjadi yang mengembalikan nilai dan yang hanya menjalankan aksi tanpa mengembalikan hasil.

Nama & NPM	Topik:	Tanggal:
Zahra Sari Fhadilah G1F024025	Extends	13 September 2024

[4.] Identifikasi Masalah:

1) Uraikan permasalahan dan variabel

Contoh 4: Salin dan tempel kode program berikut ke JDoodle. Kemudian catat waktu eksekusinya.

```
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
}
}
```

Luaran 4:

Sifat Orang Tua :

Nonton Berita

Suka Baca Koran

Sifat Anak :

Nonton Jam 9 Malam Film Drakor

Suka Baca Komik One Piece

Latihan 4:

4.1. Bandingkan method yang dimiliki `class Anak extends Ortu` dengan method di `class Ortu` !

4.2. Ubahlah Contoh 4 dengan menambahkan objek anak dengan method yang berbeda!

2) Rincikan sumber informasi yang relevan (buku / webpage)

Video Materi 2 tentang extends – <https://www.youtube.com/watch?v=6qULMlcv-eg>

[4.] Analisis dan Argumentasi

- 1) solusi yang diusulkan.
Dengan cara menambahkan method baru pada class Anak yang berbeda dari method di class Ortu. Pada kode program saya modifikasi contoh dengan menambahkan method sukaOlahraga(String olahraga) pada class Anak.

[4.] Penyusunan Algoritma dan Kode Program

1.) Algoritma

- Mulai
- Buat Kelas Ortu:
 - Definisikan method sukaMenonton(String a) untuk mencetak "Nonton" + a.
 - Definisikan method sukaMembaca(String a) untuk mencetak "Suka Baca" + a.
- Di method main Kelas Ortu:
 - Cetak "Sifat Orang Tua".
 - Buat objek Ortu.
 - Panggil method sukaMenonton("Berita").
 - Panggil method sukaMembaca("Koran").
- Buat Kelas Anak (extends Ortu):
 - Override method sukaMenonton(int a, String b) untuk mencetak "Nonton Jam" + a + " Malam" + b.
 - Override method sukaMenonton(String a) untuk mencetak "Nonton" + a.
 - Override method sukaMembaca(String a) untuk mencetak "Suka Baca" + a.
 - Tambahkan method sukaOlahraga(String olahraga) untuk mencetak "Suka berolahraga" + olahraga.
- Di method main Kelas Anak:
 - Cetak "Sifat Anak".
 - Buat objek Anak.
 - Panggil method sukaMenonton(9, "Film Drakor").
 - Panggil method sukaMembaca("Komik One Piece").
 - Panggil method sukaOlahraga("Badminton").
- Selesai

2) Tuliskan kode program dan luaran

a) Beri komentar pada kode

```
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 objek0 = new Ortu(); // memanggil objek induk
    objek0.sukaMenonton("Berita"); // memanggil sifat spesifik induk
    objek0.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
    objekA.sukaOlahraga("Badminton"); //memanggil method baru
}

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);
    }
    //method baru
    void sukaOlahraga(String olahraga) {
        System.out.println("Suka berolahraga " + olahraga);
    }
}

public static void main(String [] args) {
    System.out.println("Sifat Orang Tua :");
    Ortu objek0 = new Ortu(); // memanggil objek induk
    objek0.sukaMenonton("Berita"); // memanggil sifat spesifik induk
    objek0.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
    objekA.sukaOlahraga("Badminton"); //memanggil method baru
}
```

b) Luaran yang dihasilkan

```
Output    Generated Files

Sifat Orang Tua :
Nonton Berita
Suka Baca Koran

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

CPU Time: 0.07 sec(s) | Memory: 39088 kilobyte(s) | Compiled and executed in 1.473 sec(s)
```

Luaran yang dihasilkan telah bertambah dari contoh yang dilampirkan karna saya menambah objek anak dengan method yang berbeda.

[4.] Kesimpulan

- 1) Dari kode program ini, saya mengetahui bahwa kita dapat menambahkan method baru pada class turunan (Anak) yang berbeda dari method di class induk (Ortu) melalui proses overriding dan penambahan method khusus. Saya menambahkan method `sukaOlahraga(String olahraga)` pada class Anak, sehingga class Anak dapat memiliki

fungsiionalitas tambahan yang tidak dimiliki oleh class Ortu, namun tetap dapat mewarisi dan mengubah perilaku method yang sama dari class induk

Refleksi:

Dari tugas yang telah saya kerjakan, saya mendapatkan banyak pengetahuan baru, terutama dalam memahami konsep kelas, objek, method, dan pewarisan dalam pemrograman berorientasi objek. Saya belajar bagaimana membuat atribut dan perilaku yang dapat diwariskan dari kelas induk ke kelas turunan, serta bagaimana kita dapat menambahkan fungsiionalitas baru melalui method overriding dan overloading. Selain itu, tugas ini juga membantu saya memahami penggunaan constructor dan method yang lebih mendalam, serta bagaimana konsep-konsep ini diterapkan dalam berbagai skenario pemrograman.