

**Template Lembar Kerja Individu dan Kelompok**

<b>Nama &amp; NPM</b>	<b>Topik:</b>	<b>Tanggal:</b>
<b>Delta Setyawan G1F024056</b>	<b>Kelas Java</b>	<b>17 September 2024</b>
<b>[Nomor Soal] Identifikasi Masalah:</b>		
<ol style="list-style-type: none"><li>1) Uraikan permasalahan dan variabel</li><li>2) Rincikan sumber informasi yang relevan (buku / webpage)</li><li>3) Uraikan rancangan solusi yang diusulkan (jika ada).</li><li>4) Analisis susunan solusi, parameter solusi (jika ada).</li></ol>		
<b>[Nomor Soal] Analisis dan Argumentasi</b>		
<ol style="list-style-type: none"><li>1) Uraikan rancangan solusi yang diusulkan.</li><li>2) Analisis solusi, kaitkan dengan permasalahan.</li></ol>		
<b>[Nomor Soal] Penyusunan Algoritma dan Kode Program</b>		
<ol style="list-style-type: none"><li>1) Rancang desain solusi atau algoritma</li><li>2) Tuliskan kode program dan luaran<ol style="list-style-type: none"><li>a) Beri komentar pada kode</li><li>b) Uraikan luaran yang dihasilkan</li><li>c) Screenshot/ Capture potongan kode dan hasil luaran</li></ol></li></ol>		
<b>[Nomor Soal] Kesimpulan</b>		
<ol style="list-style-type: none"><li>1) Analisa<ol style="list-style-type: none"><li>a) Susunlah kesimpulan berdasarkan permasalahan, algoritma, dan kode program!</li><li>b) Apakah dasar alasan pengambilan keputusan Anda untuk kasus ini?</li></ol></li><li>2) Evaluasi<ol style="list-style-type: none"><li>a) Apa konsekuensi dari skenario pemrograman ini?</li><li>b) Evaluasi input, proses, dan luaran yang dihasilkan! (jika ada)</li></ol></li><li>3) Kreasi<ol style="list-style-type: none"><li>a) Apakah ada pengetahuan baru yang dikembangkan dan konsep baru sebagai usulan solusi?</li><li>b) Konstruksikan hubungan antara variabel yang berbeda dengan konsep yang anda ketahui! (jika ada)</li></ol></li></ol>		

### [No. 1] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variabel

```
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!

### [No.1] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara
  - a. kelas Manusia hanya memiliki atribut tanpa perilaku yang jelas. Hal ini berarti objek Manusia yang dibuat hanya bisa menyimpan data (nama dan rambut) tetapi tidak bisa "melakukan" sesuatu yang lebih kompleks, seperti berbicara, berjalan, atau memperkenalkan diri. Kekurangan perilaku dalam objek sering kali mengakibatkan kelas yang kurang efektif dalam menggambarkan tindakan nyata.
  - b. perilaku atau method merepresentasikan aksi atau fungsi yang bisa dilakukan oleh objek dari kelas tersebut. Pada kelas Manusia, kita bisa menambahkan perilaku yang mencerminkan aksi-aksi yang umumnya dilakukan oleh manusia dalam dunia nyata.
- 2) Perbaiki kode program dengan cara:

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

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

    public static void main(String[] args) {
        Manusia aksesclass = new Manusia();
        aksesclass.nama = "Iqbal Ferdinand Putra";
        aksesclass.rambut = "Hitam";
        System.out.println(aksesclass.nama);
        System.out.println(aksesclass.rambut);
    }
}
```

### [No.1 ] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
  - (a) Mulai
  - (b) Cari permasalahan

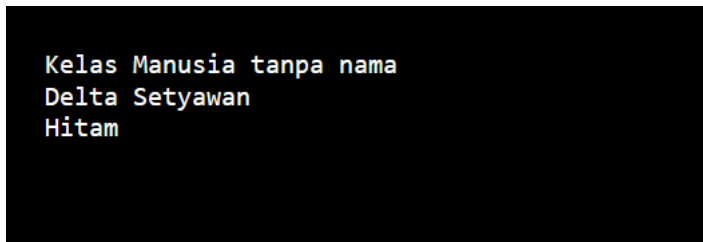
- (c) Selesaikan permasalahan
- (d) Selesai
- 2) Kode program dan luaran

a. Kode program

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

    // deklarasi constructor tanpa parameter
    public Manusia() {
        System.out.println("Kelas Manusia tanpa nama");
    }
    public static void main(String[] args) {
        Manusia aksesclass = new Manusia();
        aksesclass.nama = "Delta Setyawan";
        aksesclass.ambut = "Hitam";
        System.out.println(aksesclass.nama);
        System.out.println(aksesclass.ambut);
    }
}
```

b. Luaran



```
Kelas Manusia tanpa nama
Delta Setyawan
Hitam
```

b. Analisa luaran yang dihasilkan

Luaran sudah sesuai dengan kode program yang disusun.

Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

## [No.1] Kesimpulan

### Analisa

Atribut seperti nama dan rambut memberikan identitas pada objek Manusia. Namun, tanpa perilaku, objek ini tidak bisa berfungsi secara efektif. Dengan menambahkan metode, kelas tersebut menjadi lebih berguna dan siap untuk diterapkan dalam konteks aplikasi yang lebih besar.

Kelas yang baik harus memiliki keseimbangan antara data yang disimpannya (atribut) dan aksi yang bisa dilakukan (metode). Dalam contoh ini, Manusia sudah memiliki atribut yang cukup, dan dengan menambahkan metode, kita membuatnya lebih dinamis.

## [No.2] Identifikasi Masalah:

- 2) Uraikan permasalahan dan variabel

```
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");
    }
}
```

## **Luaran 2:**

Nama saya : Putri  
Warna Rambut : hitam

## **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?

### **[No.2] Analisis dan Argumentasi**

- 3) Saya mengusulkan permasalahan ini dapat diatasi dengan cara
  - a. Mengembangkan constructor di kelas “anak” yang akan mewarisi dari kelas “ortu”. Sehingga, atribut tambahan yang lebih spesifik untuk keturunan dapat dimasukkan ke dalam constructor.
- 4) Alasan solusi ini karena
  - a. Pewarisan (inheritance) pada Object Oriented Programming subclass bisa mengambil atribut dan perilaku dari parent class, serta menambahkan atribut baru yang lebih spesifik untuk menunjukkan karakteristik unik dari subclass.
- 5) Perbaiki kode program dengan cara

```
public class Kelas {  
    //deklarasi constructor  
    public Kelas(String nama, String rambut, String kulit, int umur, int tinggi) {  
        this.nama=nama;  
        this.rambut=rambut;  
        this.kulit=kulit;  
        this.umur=umur;  
        this.tinggi=tinggi;  
        System.out.println(" Nama saya : "+ nama +  
            "\n Warna Rambut : " + rambut "\n Warna kulit : " + kulit "\n Umur : " + umur + "\n  
Tinggi : " + tinggi);  
    }  
    public static void main (String[] args) {  
        Kelas satu = new Kelas("Delta setyawan", "hitam", "Sawo mateng", 18, 170);  
    }  
}
```

### **[No.2] Penyusunan Algoritma dan Kode Program**

- 3) Algoritma
  - (e) Mulai
  - (f) Cari permasalahan
  - (g) Selesaikan permasalahan
  - (h) Selesai
- 4) Kode program dan luaran

#### a. Kode program

```
public class Kelas {
    String nama, rambut, kulit;
    int umur, tinggi;
    //deklarasi constructor
    public Kelas(String nama, String rambut, String kulit, int umur, int tinggi) {
        this.nama=nama;
        this.ambut=rambut;
        this.kulit=kulit;
        this.umur=umur;
        this.tinggi=tinggi;
        System.out.println(" Nama saya : "+ nama + "\n Warna Rambut : " + rambut + "\n Warna kulit : " + kulit + "\n Umur : " +
        umur + "\n Tinggi : " + tinggi);
    }
    public static void main (String[] args) {
        Kelas satu = new Kelas("Delta setyawan", "hitam", "Sawo mateng", 18, 170);
    }
}
```

#### b. Luaran

```
Nama saya : Delta setyawan
Warna Rambut : hitam
Warna kulit : Sawo mateng
Umur : 18
Tinggi : 170
```

##### a) Analisa luaran yang dihasilkan

Luaran sudah sesuai dengan kode program yang disusun.

Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

### [No.2] Kesimpulan

#### Analisa

Pada program itu saya menggunakan bentuk kelas public karena memungkinkan akses dari luar paket sehingga objek dapat dibuat di kelas manapun. Perbaikan program dengan menambahkan atribut warna mata dan tinggi badan dilakukan karena struktur Java mengharuskan setiap objek memiliki detail yang spesifik, dan constructor adalah tempat yang tepat untuk menerima data tersebut saat objek dibuat.

### [No.3] Identifikasi Masalah:

#### 3) Uraikan permasalahan dan variabel

```
public class Manusia {
    //deklarasi atribut Manusia dalam variabel
    String nama, rambut;

    //deklarasi constructor
    public Manusal(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

### [No.3] Analisis dan Argumentasi

- 6) Analisa perbedaan deklarasi constructor, method, dan method utama!
  - a. Constructor adalah method khusus yang digunakan untuk menginisialisasi objek saat dibuat. Constructor memiliki nama yang sama dengan nama kelas dan tidak memiliki tipe kembalian. Pada kode di atas, constructor adalah `Manusia(String nama, String rambut)`. Constructor ini dipanggil saat objek `Manusia` dibuat dengan argumen yang diberikan.
  - b. Method adalah blok kode yang dieksekusi ketika dipanggil, dan biasanya berisi serangkaian instruksi untuk melakukan operasi tertentu. Method `sukaNonton(String film)` pada kode di atas menampilkan hobi menonton seseorang, dan dipanggil setelah objek dibuat. Tidak seperti constructor, method dapat memiliki tipe kembalian.
  - c. Method utama (`main`) adalah method yang menjadi titik masuk program Java dan digunakan untuk menjalankan aplikasi. Tanpa method utama, program tidak akan dapat dijalankan. Pada contoh di atas, method utama adalah `public static void main(String[] args)`, yang digunakan untuk membuat objek `Manusia` dan memanggil method `sukaNonton`.
- 7) Tentukan kapan Anda perlu menggunakan constructor dan method?
  - a. Constructor digunakan saat Anda ingin menginisialisasi objek dengan nilai-nilai awal tertentu. Constructor dipanggil otomatis saat objek dibuat menggunakan `new`.
  - b. Method digunakan untuk menjalankan fungsi-fungsi tertentu setelah objek dibuat. Method bisa digunakan untuk mengubah atau mengambil data, atau untuk melakukan operasi tertentu yang terkait dengan objek tersebut.
- 8) Uraikan perbedaan berikut:
  - a) constructor overloading dan overriding
  - b) method overloading, dan method overriding
  - c) method yang mengembalikan nilai dan method tidak mengembalikan nilai

**a) Constructor Overloading dan Constructor Overriding:**

- Overloading: Proses memiliki lebih dari satu constructor dengan parameter berbeda dalam satu kelas. Ini memberikan fleksibilitas dalam pembuatan objek dengan berbagai parameter.
- Overriding: Tidak ada "constructor overriding" di Java karena constructor tidak diwariskan dari superclass ke subclass.

**b) Method Overloading dan Method Overriding:**

- Overloading: Membuat beberapa method dengan nama yang sama tetapi parameter yang berbeda di dalam kelas yang sama. Ini memungkinkan method untuk menangani berbagai jenis input.
- Overriding: Proses mendefinisikan kembali method dari superclass di subclass dengan tipe kembalian dan parameter yang sama. Method overriding memungkinkan subclass untuk memberikan implementasi spesifik dari method yang diwarisi.

**c) Method yang mengembalikan nilai dan method yang tidak mengembalikan nilai:**

- Method yang mengembalikan nilai adalah method yang memiliki tipe kembalian (misalnya int, String, dll.) dan mengembalikan nilai saat dieksekusi.
- Method yang tidak mengembalikan nilai (seperti void) hanya menjalankan instruksi tertentu dan tidak mengembalikan apapun.

9) Perbaiki kode program dengan cara

```
public class Manusia {  
    //deklarasi atribut Manusia dalam variabel  
    String nama, rambut;  
  
    //deklarasi constructor  
    public Manusia(String nama, String rambut) {  
        System.out.println(" Nama saya : " + nama +  
            "\n Warna Rambut : " + rambut);  
    }  
  
    //deklarasi method  
    void sukaNonton(String film) {  
        System.out.println(" Hobi : " + film);  
    }  
  
    //deklarasi method utama  
    public static void main( String[] args) {  
        Manusia satu = new Manusia("Delta Setyawan", "hitam");  
        satu.sukaNonton("Bermain game");  
    }  
}
```

**[No.3] Penyusunan Algoritma dan Kode Program**

- 5) Algoritma
  - (i) Mulai
  - (j) Cari permasalahan dalam kode program
  - (k) Selesaikan permasalahan dalam kode program
  - (l) Selesai
- 6) Kode program dan luaran

a. Kode program

```
public class Manusia {  
    //deklarasi atribut Manusia dalam variabel  
    String nama, rambut;  
  
    //deklarasi constructor  
    public Manusia(String nama, String rambut) {  
        System.out.println(" Nama saya : " + nama +  
            "\n Warna Rambut : " + rambut);  
    }  
  
    //deklarasi method  
    void sukaNonton(String film) {  
        System.out.println(" Hobi : " + film);  
    }  
  
    //deklarasi method utama  
    public static void main( String[] args) {  
        Manusia satu = new Manusia("Delta Setyawan", "hitam");  
        satu.sukaNonton("Bermain game");  
    }  
}
```

b. Luaran

```
Nama saya : Delta Setyawan  
Warna Rambut : hitam  
Hobi : Bermain game
```

b) Analisa luaran yang dihasilkan

Luaran sudah sesuai dengan kode program yang disusun.

Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

### [No.3] Kesimpulan

#### Analisa

Pada program ini, saya menggunakan kelas public untuk memastikan bahwa kelas Manusia dapat diakses dari mana saja dalam aplikasi Java. Program ini menggunakan constructor untuk menginisialisasi objek dengan nama dan warna rambut, serta method sukaNonton untuk menampilkan hobi menonton seseorang. Kesalahan awal terjadi pada penamaan constructor (Manusia1), yang tidak sesuai dengan nama kelas, dan sudah diperbaiki dengan mengganti nama constructor menjadi Manusia. Algoritma program ini sederhana, dimulai dengan menciptakan objek Manusia dan kemudian memanggil method sukaNonton untuk menampilkan hobi objek tersebut.

### [No.4] Identifikasi Masalah:

4) Uraikan permasalahan dan variabel

```
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**

## 5) **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!

10) Bandingkan method yang dimiliki class Anak extends Ortu dengan method di class Ortu!

- a. Kelas Anak mewarisi method dari kelas Ortu, tetapi method sukaMenonton di-*override* dalam kelas Anak untuk menambah fleksibilitas (dengan overloading) dan perubahan perilaku.
- b. Method sukaMembaca di kelas Anak juga di-*override* untuk mengubah output sesuai dengan karakteristik anak.

11) Perbaiki kode program dengan cara

```
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 One Piece"); //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.sukaBermain("Mobile Legends");
    } }

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);
    }
    void sukaBermain(String a) {
        System.out.println("Suka Bermain " + 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 One Piece"); //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.sukaBermain("Mobile Legends");
}
}
```

#### **[No.4] Penyusunan Algoritma dan Kode Program**

- 7) Algoritma
  - (m) Mulai
  - (n) Cari permasalahan dalam kode program
  - (o) Selesaikan permasalahan dalam kode program
  - (p) Selesai
- 8) Kode program dan luaran
  - a. Kode program

```

1 public class Ortu {           // membuat kelas induk
2     void sukaMenonton(String a) { // method induk spesifik
3         System.out.println("Nonton " + a);
4     }
5     void sukaMembaca(String a) { // method induk umum bisa diubah anak
6         System.out.println("Suka Baca " + a);
7     }
8
9     public static void main(String [] args) {
10        System.out.println("Sifat Orang Tua :");
11        Ortu objekO = new Ortu(); // memanggil objek induk
12        objekO.sukaMenonton("Berita"); // memanggil sifat spesifik induk
13        objekO.sukaMembaca("Koran"); // memanggil method dengan variabel dapat diubah
14
15        System.out.println("\n Sifat Anak :");
16        Anak objekA = new Anak(); //memanggil objek anak
17        objekA.sukaMenonton(9, "Film One Piece"); //memanggil sifat spesifik anak yang diturunkan induk
18        objekA.sukaMembaca("Komik One Piece"); //memanggil method ke induk yang otomatis diturunkan tanpa deklarasi ulang di anak
19        objekA.sukaBermain("Mobile Legends");
20    }
21
22    class Anak extends Ortu {
23        void sukaMenonton(int a, String b) {
24            System.out.println("Nonton Jam " + a + " Malam " + b);
25        }
26        void sukaMenonton(String a) { // method induk spesifik
27            System.out.println("Nonton " + a);
28        }
29        void sukaMembaca(String a) { // method induk umum bisa diubah anak
30            System.out.println("Suka Baca " + a);
31        }
32        void sukaBermain(String a) {
33            System.out.println("Suka Bermain " + a);
34        }
35
36        public static void main(String [] args) {
37            System.out.println("Sifat Orang Tua :");
38            Ortu objekO = new Ortu(); // memanggil objek induk
39            objekO.sukaMenonton("Berita"); // memanggil sifat spesifik induk
40            objekO.sukaMembaca("Koran"); // memanggil method dengan variabel dapat diubah
41
42            System.out.println("\n Sifat Anak :");
43            Anak objekA = new Anak(); //memanggil objek anak
44            objekA.sukaMenonton(9, "Film One Piece"); //memanggil sifat spesifik anak yang diturunkan induk
45            objekA.sukaMembaca("Komik One Piece"); //memanggil method ke induk yang otomatis diturunkan tanpa deklarasi ulang di anak
46            objekA.sukaBermain("Mobile Legends");
47        }
48    }

```

#### b. Luaran

Sifat Orang Tua :

Nonton Berita

Suka Baca Koran

Sifat Anak :

Nonton Jam 9 Malam Film One Piece

Suka Baca Komik One Piece

Suka Bermain Mobile Legends

#### c) Analisa luaran yang dihasilkan

Luaran sudah sesuai dengan kode program yang disusun.

Tipe data yang ditampilkan telah sesuai dengan kebutuhan dan permintaan data.

#### [No.4] Kesimpulan

##### Analisa

Pada program ini, saya menggunakan bentuk kelas public karena memungkinkan kelas tersebut diakses dari luar kelas, yang sesuai dengan konsep aksesibilitas pada OOP. Penggunaan kelas induk dan anak dalam bentuk public juga memfasilitasi pewarisan, sehingga atribut dan method yang ada di kelas induk bisa digunakan secara bebas di kelas anak. Perbaikan program dengan menambahkan method baru di kelas Anak seperti `sukaBermain` dilakukan untuk menunjukkan fleksibilitas dalam pewarisan dan memungkinkan kelas turunan memiliki perilaku atau kemampuan tambahan

### **Refleksi**

Dari soal di atas saya sedikit mengetahui tentang topik Kelas Java yang terdiri 4 kelas yaitu kelas(class), kelas objek, kelas method, dan kelas Extends. Dari ke empat kelas saya kesulitan dalam menjawab di kelas method, awalnya saya kira udh ada luarannya ternyata masih ada error dari kode tersebut, jadi saya harus memperbaiki kode program nya.