

Template Lembar Kerja Individu dan Kelompok

Nama & NPM	Topik:	Tanggal:
Karina Hadiyah Ramadona G1F024040	Kelas (Class)	18 September 2024
[No.1] Identifikasi Masalah:		
<p>1) Uraikan permasalahan dan variable</p> <p>Contoh 1:</p> <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>Luaran 1:</p> <p>Exception in thread "main" java.lang.Error: Unresolved compilation problem: The constructor Manusia1(String, String) is undefined at Manusia1.main(Manusia1.java:13)</p> <p>Latihan 1:</p> <p>1.1. Perbaiki pesan kesalahan Contoh 1!</p> <p>1.2. Cermati contoh 1. susun kode menggunakan constructor dengan parameter data pribadi anda!</p> <p>2) Rincikan sumber informasi yang relevan (buku / webpage)</p> <p>https://www.youtube.com/watch?v=60ldOc8m8Es</p>		
[No.1] Analisis dan Argumentasi		
<p>1) Uraikan rancangan solusi yang diusulkan.</p> <p>Saya mengusulkan untuk mengubah kelas yang awalnya manusia1 menjadi Manusia. Kemudian saya juga mengusulkan untuk menambahkan String untuk mengakses atribut.</p> <p>2) Analisis solusi, kaitkan dengan permasalahan.</p> <p>Alasan saya mengusulkan solusi ini adalah karena hal ini menyebabkan konstruktor tidak dapat dipanggil dengan nama yang berbeda dari kelasnya.</p>		
[No.1] Penyusunan Algoritma dan Kode Program		
<p>1) Rancang desain solusi atau algoritma</p> <ol style="list-style-type: none"> Buat variable Rancang Solusi dari permasalahan Analisis Jika sudah yakin benar, maka coba untuk di run Selesai. <p>2) Tuliskan kode program dan luaran</p>		

```

: New Project Online Java Compiler ID

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

    //deklarasi constructor
    public Manusia (String nama, String rambut, String umur, String tinggibadan) {
        System.out.println(" Nama saya : "+ nama +
            "\n Warna Rambut : " + rambut+
            "\n Umur : " + umur+
            "\n Tinggi Badan : " +tinggibadan);
    }

    //deklarasi method utama
    public static void main( String[] args) {
        Manusia satu = new Manusia("Karina", "hitam", "17", "156");
    } }

```

Luaran

Output Generated Files

```

Nama saya : Karina
Warna Rambut : hitam
Umur : 17
Tinggi Badan : 156
|

```

Compiled and executed in 1.283 sec(s)

Source code:

```

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

    //deklarasi constructor
    public Manusia (String nama, String rambut, String umur, String tinggibadan) {
        System.out.println(" Nama saya : "+ nama +
            "\n Warna Rambut : " + rambut+
            "\n Umur : " + umur+
            "\n Tinggi Badan : " +tinggibadan);
    }

    //deklarasi method utama
    public static void main( String[] args) {
        Manusia satu = new Manusia("Karina", "hitam", "17", "156");
    } }

```

[No.1] Kesimpulan

- 1) Evaluasi
 - a) Apa konsekuensi dari skenario pemrograman ini?

Skenario pemrograman ini menunjukkan penerapan prinsip pemrograman berorientasi objek yang baik. Meskipun ada tantangan, seperti kompleksitas dan kebutuhan dokumentasi, manfaatnya jauh lebih besar. Keterbacaan kode yang meningkat dan fleksibilitas dalam penambahan fitur membuat pengembangan lebih efisien. Kunci keberhasilan terletak pada manajemen yang baik dan dokumentasi, sehingga kode dapat dipahami dan dirawat dengan mudah oleh siapa pun di masa depan.

Nama & NPM	Topik:	Tanggal:
Karina Hadiyah Ramadona G1F024040	Objek	18 September 2024
[No.2] Identifikasi Masalah:		
1) Uraikan permasalahan dan variable		

```

public class Ortu {
    //deklarasi constructor (variabel constructor)
    public ortu {
        //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:

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

The constructor Ortu(String, String) is undefined

at Ortu.main(Ortu.java:9)

Latihan 2:

2.1. Evaluasi penyebab kesalahan dan perbaiki kode tersebut!

2.2. Apabila nanti Anda akan memiliki keturunan, analisa sifat (atribut) dan constructor sebagai Ortu apa yang akan diturunkan (gunakan data karakter pribadi anda) ?

2.3. Rancanglah kode program untuk sifat (atribut) dan constructor overloaded dari Latihan 2.2!

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


<https://www.youtube.com/watch?v=60ldOc8m8Es>

[No.2] Analisis dan Argumentasi

- 1) Uraikan rancangan solusi yang diusulkan.
Mengubah public ortu menjadi public Ortu agar sesuai dengan nama kelasnya.
- 2) Analisis solusi, kaitkan dengan permasalahan.
Alasan saya mengusulkan solusi ini adalah karena hal ini menyebabkan konstruktor tidak dapat dipanggil dengan nama yang berbeda dari kelasnya.

[No.2] Penyusunan Algoritma dan Kode Program

- 1) Rancang desain solusi atau algoritma
 - a) Buat variable
 - b) Rancang Solusi dari permasalahan
 - c) Analisis
 - d) Jika sudah yakin benar, maka coba untuk di run
 - e) Selesai.
- 2) Tuliskan kode program dan luaran

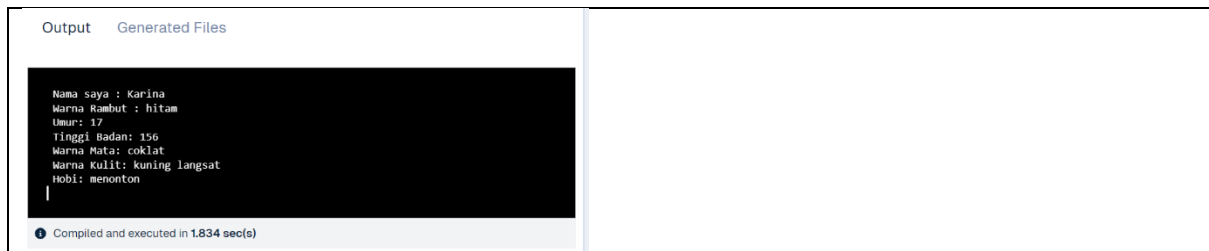


```

1 public class Ortu {
2     //deklarasi constructor (variabel constructor)
3     public Ortu (String Nama, String rambut, String umur, String tinggibadan, String warnakulit) {
4         //nama dan rambut adalah variabel constructor
5         System.out.println(" Nama saya : "+ Nama +
6             "\n Warna Rambut : " + rambut +
7             "\n Umur: " + umur +
8             "\n Tinggi Badan: " + tinggibadan +
9             "\n Warna Mata: " + warnamata +
10            "\n Warna Kulit: " + warnakulit +
11            "\n Hobi: " + hobi);
12    }
13    public static void main (String[] args) {
14        Ortu satu = new Ortu("Karina", "hitam", "17", "156", "coklat", "kuning lang");
15    } }

```

Luaran



Source code:

```
public class Ortu {
    //deklarasi constructor (variabel constructor)
    public Ortu (String Nama, String rambut, String umur, String tinggibadan, String warnamata,
String warnakulit, String hobi) {
        //nama dan rambut adalah variabel constructor
        System.out.println(" Nama saya : "+ Nama +
"\n Warna Rambut : " + rambut +
"\n Umur: " + umur +
"\n Tinggi Badan: " + tinggibadan +
"\n Warna Mata: " + warnamata +
"\n Warna Kulit: " + warnakulit +
"\n Hobi: " + hobi);
    }
    public static void main (String[] args) {
        Ortu satu = new Ortu("Karina", "hitam", "17", "156", "coklat", "kuning langsung", "menonton");
    } }
```

[No.2] Kesimpulan

- 1) Evaluasi
 - a) Apa konsekuensi dari skenario pemrograman ini?

Skenario pemrograman ini menunjukkan penerapan prinsip pemrograman berorientasi objek yang baik. Meskipun ada tantangan, seperti kompleksitas dan kebutuhan dokumentasi, manfaatnya jauh lebih besar. Keterbacaan kode yang meningkat dan fleksibilitas dalam penambahan fitur membuat pengembangan lebih efisien. Kunci keberhasilan terletak pada manajemen yang baik dan dokumentasi, sehingga kode dapat dipahami dan dirawat dengan mudah oleh siapa pun di masa depan.

Nama & NPM	Topik:	Tanggal:
Karina Hadiyah Ramadona G1F024040	Method	18 September 2024

[No.3] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variable

```
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");
} }
```

Luaran 3:

Exception in thread "main" java.lang.Error: Unresolved compilation problems:

The method sukaNonton(String) is undefined for the type Manusia1

The method sukaNonton(int, int) is undefined for the type Manusia1

at Manusia1.main(Manusia1.java:23)

Latihan 3:

3.1. Evaluasi penyebab kesalahan dan perbaiki kode tersebut!

3.2. Berdasarkan Latihan 2.2. Anda sudah punya kode program untuk atribut dan constructor sebagai Ortu.

Kembangkanlah kode program untuk method dari Ortu dengan data perilaku pribadi

Anda yang menggunakan:

- a) method overloading,
- b) method dengan return value
- c) method tanpa return value

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

<https://www.youtube.com/watch?v=6qULMlcv-eg>

[No.3] Analisis dan Argumentasi

- 1) Uraikan rancangan solusi yang diusulkan.
Saya mengusulkan untuk menambahkan String pada void suka nonton untuk mengakses atribut. Kemudian sya juga menambahkan return.
- 2) Analisis solusi, kaitkan dengan permasalahan.
Alasan saya mengusulkan solusi ini adalah karena hal Ini menyebabkan konstruktor tidak dapat dipanggil dengan nama yang berbeda dari kelasnya.

[No.3] Penyusunan Algoritma dan Kode Program

- 1) Rancang desain solusi atau algoritma
 - a) Buat variable
 - b) Rancang Solusi dari permasalahan
 - c) Analisis
 - d) Jika sudah yakin benar, maka coba untuk di run
 - e) Selesai.
- 2) Tuliskan kode program dan luaran

```

1 public class Ortu {
2     // Deklarasi constructor
3     public Ortu(String Nama, String rambut, String umur, String tinggibadan, String warnamata, String warnakulit, String hobi) {
4         System.out.println("Nama saya: " + Nama +
5             "\nWarna Rambut: " + rambut +
6             "\nUmur: " + umur +
7             "\nTinggi Badan: " + tinggibadan +
8             "\nWarna Mata: " + warnamata +
9             "\nWarna Kulit: " + warnakulit +
10            "\nHobi: " + hobi);
11     }
12
13     // Method overloading
14     public void tampilkanHobi() {
15         System.out.println("Hobi saya: menonton");
16     }
17
18     public void tampilkanHobi(String tambahan) {
19         System.out.println("Hobi saya: menonton dan " + tambahan);
20     }
21
22     // Method dengan return value
23     public int hitungTahunDariUmur() {
24         return 2023 - 17; // Misalkan umur yang dihardcode
25     }
26
27     // Method tanpa return value
28     public void sapa() {
29         System.out.println("Halo, saya Karina. Selamat datang!");
30     }
31
32     public static void main(String[] args) {
33         Ortu satu = new Ortu("Karina", "hitam", "17", "156", "coklat", "kuning la

```

Luaran

Output Generated Files

```

Nama saya: Karina
Warna Rambut: hitam
Umur: 17
Tinggi Badan: 156
Warna Mata: coklat
Warna Kulit: kuning langsung
Hobi: menonton
Hobi saya: menonton
Hobi saya: menonton dan berolahraga
Tahun lahir saya: 2006
Halo, saya Karina. Selamat datang!
|

```

Compiled and executed in 1.393 sec(s)

Source code:

```

public class Ortu {
    // Deklarasi constructor
    public Ortu(String Nama, String rambut, String umur, String tinggibadan, String warnamata,
String warnakulit, String hobi) {
        System.out.println("Nama saya: " + Nama +
            "\nWarna Rambut: " + rambut +
            "\nUmur: " + umur +
            "\nTinggi Badan: " + tinggibadan +
            "\nWarna Mata: " + warnamata +
            "\nWarna Kulit: " + warnakulit +
            "\nHobi: " + hobi);
    }

    // Method overloading
    public void tampilkanHobi() {
        System.out.println("Hobi saya: menonton");
    }

    public void tampilkanHobi(String tambahan) {
        System.out.println("Hobi saya: menonton dan " + tambahan);
    }

    // Method dengan return value
    public int hitungTahunDariUmur() {
        return 2023 - 17; // Misalkan umur yang dihardcode
    }
}

```

```
// Method tanpa return value
public void sapa() {
    System.out.println("Halo, saya Karina. Selamat datang!");
}

public static void main(String[] args) {
    Ortu satu = new Ortu("Karina", "hitam", "17", "156", "coklat", "kuning langsung", "menonton");

    // Menggunakan metode yang ditambahkan
    satu.tampilkanHobi();
    satu.tampilkanHobi("berolahraga");
    int tahunLahir = satu.hitungTahunDariUmur();
    System.out.println("Tahun lahir saya: " + tahunLahir);
    satu.sapa();
}
}
```

[No.3] Kesimpulan

- 1) Evaluasi
 - a) Apa konsekuensi dari skenario pemrograman ini?

Secara keseluruhan, skenario ini memberikan gambaran yang baik tentang bagaimana memanfaatkan fitur pemrograman berorientasi objek, tetapi juga menunjukkan tantangan yang bisa muncul seiring dengan kompleksitas yang meningkat.

Nama & NPM	Topik:	Tanggal:
Karina Hadiyah Ramadona G1F024040	Extends	18 September 2024

[No.4] Identifikasi Masalah:

- 1) Uraikan permasalahan dan variable


```
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. Evaluasi method yang dimiliki Contoh 4 pada class Anak extends Ortu dengan method di class Ortu.

Simpulkan hasil evaluasi Anda agar method ini menjadi efisien!

4.2. Setelah dirunning di JDoodle, catat waktu eksekusinya.

Susun kembali kode program yang dapat mengefisienkan waktu eksekusi!

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

<https://www.youtube.com/watch?v=60IdOc8m8Es>

[No.4] Analisis dan Argumentasi

1) Uraikan rancangan solusi yang diusulkan.

a) Duplikasi Kode: Kode di kelas Anak mengulang metode main dari kelas Ortu.

Sebaiknya, hanya satu metode main yang digunakan untuk meningkatkan efisiensi.

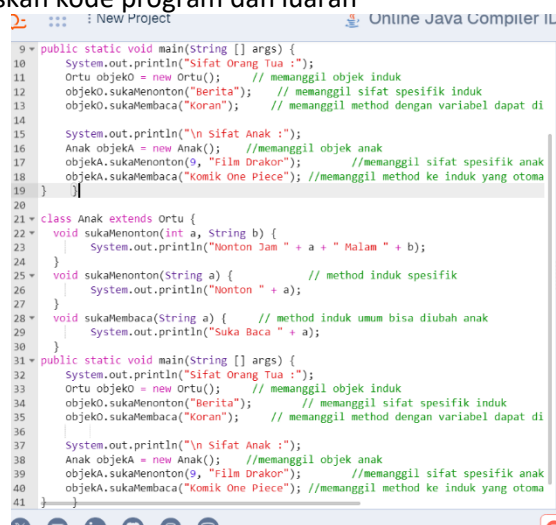
- b) Konsistensi Metode: Metode `sukaMenonton` di kelas `Anak` memiliki dua versi, yang bisa membingungkan. Sederhanakan dengan mempertahankan satu metode yang lebih jelas.
- c) Anotasi `@Override`: Gunakan anotasi `@Override` saat mengoverride metode untuk meningkatkan keterbacaan dan menghindari kesalahan.
- d) Pemisahan Tanggung Jawab: Jika metode berbeda secara signifikan antara induk dan anak, pertimbangkan untuk menghilangkan metode di kelas induk yang tidak diperlukan.
- e) Keterbacaan: Tambahkan komentar yang jelas untuk setiap metode agar lebih mudah dipahami.

2) Analisis solusi, kaitkan dengan permasalahan.

Alasan saya mengusulkan solusi ini adalah karena hal ini menyebabkan konstruktor tidak dapat dipanggil dengan nama yang berbeda dari kelasnya.

[No.4] Penyusunan Algoritma dan Kode Program

- 1) Rancang desain solusi atau algoritma
 - a) Buat variable
 - b) Rancang Solusi dari permasalahan
 - c) Analisis
 - d) Jika sudah yakin benar, maka coba untuk di run
 - e) Selesai.
- 2) Tuliskan kode program dan luaran



```

9 public static void main(String [] args) {
10     System.out.println("Sifat Orang Tua :");
11     Ortu objek0 = new Ortu(); // memanggil objek induk
12     objek0.sukaMenonton("Berita"); // memanggil sifat spesifik induk
13     objek0.sukaMembaca("Koran"); // memanggil method dengan variabel dapat di
14
15     System.out.println("\n Sifat Anak :");
16     Anak objekA = new Anak(); //memanggil objek anak
17     objekA.sukaMenonton(9, "Film Drakor"); //memanggil sifat spesifik anak
18     objekA.sukaMembaca("Komik One Piece"); //memanggil method ke induk yang otoma
19 }
20
21 class Anak extends Ortu {
22     void sukaMenonton(int a, String b) {
23         System.out.println("Nonton Jam " + a + " Malam " + b);
24     }
25     void sukaMenonton(String a) { // method induk spesifik
26         System.out.println("Nonton " + a);
27     }
28     void sukaMembaca(String a) { // method induk umum bisa diubah anak
29         System.out.println("Suka Baca " + a);
30     }
31 }
32 public static void main(String [] args) {
33     System.out.println("Sifat Orang Tua :");
34     Ortu objek0 = new Ortu(); // memanggil objek induk
35     objek0.sukaMenonton("Berita"); // memanggil sifat spesifik induk
36     objek0.sukaMembaca("Koran"); // memanggil method dengan variabel dapat di
37
38     System.out.println("\n Sifat Anak :");
39     Anak objekA = new Anak(); //memanggil objek anak
40     objekA.sukaMenonton(9, "Film Drakor"); //memanggil sifat spesifik anak
41     objekA.sukaMembaca("Komik One Piece"); //memanggil method ke induk yang otoma
42 }

```

Luaran

Output Generated Files

```

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 1.951 sec(s)

Source code:

```

public class Ortu { // Kelas induk
    void sukaMenonton(String a) { // Method induk spesifik
        System.out.println("Nonton " + a);
    }
}

```

```

void sukaMembaca(String a) { // Method induk umum yang bisa diubah anak
    System.out.println("Suka Baca " + a);
}

public void tampilkanSifat() {
    System.out.println("Sifat Orang Tua :");
    sukaMenonton("Berita"); // Memanggil sifat spesifik induk
    sukaMembaca("Koran"); // Memanggil method yang dapat diubah
}
}

class Anak extends Ortu {
    void sukaMenonton(int a, String b) { // Method spesifik anak
        System.out.println("Nonton Jam " + a + " Malam " + b);
    }

    // Menggunakan method dari induk
    @Override
    void sukaMembaca(String a) { // Override method induk
        System.out.println("Suka Baca " + a);
    }

    public void tampilkanSifat() {
        System.out.println("\nSifat Anak :");
        sukaMenonton(9, "Film Drakor"); // Memanggil sifat spesifik anak
        sukaMembaca("Komik One Piece"); // Memanggil method dari induk
    }
}

public class Main {
    public static void main(String[] args) {
        Ortu objekO = new Ortu(); // Memanggil objek induk
        objekO.tampilkanSifat(); // Menampilkan sifat orang tua

        Anak objekA = new Anak(); // Memanggil objek anak
        objekA.tampilkanSifat(); // Menampilkan sifat anak
    }
}

```

[No.4] Kesimpulan

- 1) Evaluasi
 - a) Apa konsekuensi dari skenario pemrograman ini?

Skenario pemrograman ini menunjukkan konsep pewarisan dan polimorfisme dengan baik, memungkinkan kelas `Anak` untuk mewarisi dan mengubah metode dari kelas `Ortu`. Ini meningkatkan keterbacaan dan reuse kode, tetapi bisa menyebabkan kebingungan jika tidak dikelola dengan baik. Penggunaan anotasi `@Override` disarankan untuk meningkatkan kejelasan. Secara keseluruhan, program ini efektif dalam menerapkan prinsip pemrograman berorientasi objek, dengan perhatian pada manajemen kompleksitas.