

| | | |
|---------------------|--------|-------------------|
| Nama & NPM | Topik: | Tanggal: |
| Bagas Satrio Winata | | 17 September 2024 |

[No.1] Identifikasi Masalah:

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

[No.1] Analisis dan Argumentasi

- 1) Permasalahan pada kode program dapat di selesaikan dengan cara menambahkan atribut didalam public manusia dan menambahkan method untuk perilaku
- 2) Alasan saya mengusulkan Solusi tersebut karena di dalam public manusia harus di deklarasikan atribut yang ingin di gunakan supaya atribut bisa dibaca oleh program dan juga menambahkan method void sukaGame untuk menampilkan perilaku
- 3) Perbaikan kode program dengan cara mengisi public manusia dengan atribut string nama dan rambut dan menambahkan metode void sukaGame serta menambahkan public static void main(String []args) untuk menampilkan hasil dari metode

[No.1] Algoritma dan Kode Program

1. Algoritma
 - Mulai program
 - Mendeklarasikan public class manusia
 - Mendeklarasikan variabel String nama dan rambut didalam public class manusia
 - Mendeklarasikan public manusia dan void sukaGame didalam public class manusia
 - Mendeklarasikan System.out.println() didalam public class manusia
 - Mendeklarasikan public static void main() didalam public class manusia
 - Program selesai
2. Kode program

The screenshot shows an IDE with a Java file named 'Manusia.java'. The code defines a public class 'Manusia' with two instance variables, 'nama' and 'rambut', and a constructor that prints the class name. It also includes a 'sukaGame' method that prints a message with a film name, and a 'main' method that creates a 'Manusia' object and calls 'sukaGame'.

```
1 public class Manusia { // deklarasi kelas
2     // deklarasi variabel
3     String nama ;
4     String rambut ;
5
6     // deklarasi constructor tanpa parameter
7     public Manusia(String nama, String rambut) {
8         System.out.println("nama : " + nama +
9             "\nwarna rambut : " + rambut);
10    }
11
12    void sukaGame(String film) {
13        System.out.println("Hobi Main Game : " + film);
14    }
15    public static void main(String []args) {
16        Manusia baru = new Manusia ("Bagas", "hitam");
17        baru.sukaGame("FF");
18    }
19 }
```

The right-hand side of the IDE shows the 'Input/Output' panel. It indicates the language version is 'JDK 21.0.0' and that the program is running interactively. The output shows the results of the program execution:

```
nama : Bagas
warna rambut : hitam
Hobi Main Game : FF
```

At the bottom of the output panel, it states 'Compiled and executed in 1.242 sec(s)'.

[No.1] Kesimpulan

Pada program itu saya menggunakan bentuk kelas public karena kelas ini perlu diakses oleh kelas lain atau dari luar file tempat kelas tersebut berada. Perbaikan program dengan menambahkan atribut String nama dan String rambut serta method void sukaGame karena struktur Java mengharuskan kelas berisi atribut dan method untuk mendefinisikan data dan perilaku dari objek yang dibuat.

[No.2] Identifikasi Masalah:

```
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");  
    }  
}
```

[No.2] Analisis dan Argumentasi

- 1) Masalah pada program dapat diatasi dengan cara menambahkan ciri-ciri saya dan menambahkan perilaku
- 2) Alasan saya mengusulkan Solusi ini karena apabila ingin menganalisa keturunan, analisa sifat (atribut), constructor, dan perilaku positif (behavior) apa yang akan diturunkan kita harus menambahkan ciri-ciri dan perilaku kita terlebih dahulu
- 3) Perbaikan kode program dengan cara menambahkan variabel String jenisKelamin, String kulit, int tinggiBadan, double beratBadan
- 4) Analisa apabila

Atribut dan perilaku positif yang bisa diturunkan kepada keturunan dari kelas Ortu dapat mencakup beberapa hal:

1. Atribut (Sifat) yang Diturunkan:

- Warna rambut hitam bisa menjadi sifat fisik yang diturunkan .
- Warna Kulit sawo matang yang bisa diturunkan sebagai sifat fisik.
- Jenis Kelamin Ini akan tetap tergantung pada keturunan.
- Tinggi Badan dan Berat Badan Faktor-faktor ini dapat mencerminkan genetik atau pola hidup dan bisa jadi diturunkan.

2. Constructor yang Diturunkan:

Constructor dapat digunakan untuk menginisialisasi sifat-sifat keturunan, seperti nama, warna rambut, jenis kelamin, warna kulit, tinggi badan, dan berat badan. Jadi, pada saat objek anak dibuat, constructor ini akan menetapkan sifat awal mereka berdasarkan apa yang diturunkan dari ortu.

3. Perilaku/Behavior yang Diturunkan:

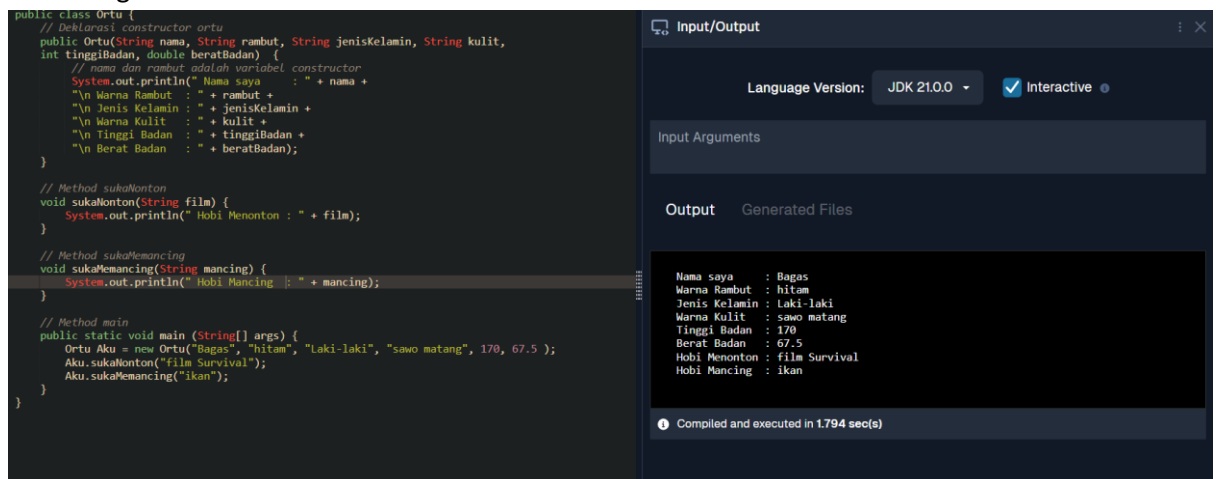
- Hobi Menonton Kebiasaan atau minat seperti suka menonton film survival bisa menjadi perilaku yang diturunkan atau dipengaruhi oleh orang tua.
- Hobi Memancing Kebiasaan suka memancing juga bisa diturunkan sebagai minat atau aktivitas keluarga jadi, bisa jadi diturunkan ke anak .

[No.2] Algoritma dan Kode Program

1) Algoritma

- Mulai program
- Mendeklarasikan public class ortu
- Mendeklarasikan public ortu didalam public class ortu
- Mendeklarasikan void sukaNonton dan sukaMemancing didalam public ortu
- Mendeklarasikan public static void main(String[] args) didalam public ortu
- Program selesai

2) Kode Program



```
public class Ortu {  
    // Deklarasi constructor ortu  
    public Ortu(String nama, String rambut, String jenisKelamin, String kulit,  
        int tinggiBadan, double beratBadan) {  
        // nama dan rambut adalah variabel constructor  
        System.out.println(" Nama saya      : " + nama +  
            "\n Warna Rambut   : " + rambut +  
            "\n Jenis Kelamin  : " + jenisKelamin +  
            "\n Warna Kulit   : " + kulit +  
            "\n Tinggi Badan  : " + tinggiBadan +  
            "\n Berat Badan   : " + beratBadan);  
    }  
  
    // Method sukaNonton  
    void sukaNonton(String film) {  
        System.out.println(" Hobi Menonton : " + film);  
    }  
  
    // Method sukaMemancing  
    void sukaMemancing(String mancing) {  
        System.out.println(" Hobi Mancing  : " + mancing);  
    }  
  
    // Method main  
    public static void main (String[] args) {  
        Ortu Aku = new Ortu("Bagas", "hitam", "Laki-laki", "sawo matang", 170, 67.5 );  
        Aku.sukaNonton("film Survival");  
        Aku.sukaMemancing("ikan");  
    }  
}
```

Input/Output

Language Version: JDK 21.0.0 ☒ Interactive

Input Arguments

Output Generated Files

```
Nama saya      : Bagas  
Warna Rambut   : hitam  
Jenis Kelamin  : Laki-laki  
Warna Kulit   : sawo matang  
Tinggi Badan  : 170  
Berat Badan   : 67.5  
Hobi Menonton : film Survival  
Hobi Mancing  : ikan
```

Compiled and executed in 1.794 sec(s)

[No.2] Kesimpulan

Kesimpulan:

1. Kode program telah diperbaiki dengan menambahkan atribut baru seperti jenis kelamin, kulit, tinggi badan, dan berat badan, yang menggambarkan ciri-ciri fisik tambahan dari objek yang dibuat. Hal ini memungkinkan untuk mencerminkan lebih banyak karakteristik dari orang tua yang bisa diturunkan ke keturunan.
2. Penambahan Perilaku metode suka nonton dan suka memancing ditambahkan untuk menunjukkan perilaku atau hobi yang dapat diturunkan dari orang tua kepada anak. Ini memperlihatkan bahwa selain atribut fisik, perilaku atau kebiasaan juga bisa menjadi bagian dari sifat yang diturunkan.
3. Constructor yang ditambahkan mencakup lebih banyak atribut, sehingga saat objek Ortu dibuat, sifat-sifat seperti nama, warna rambut, jenis kelamin, tinggi badan, dan berat badan langsung diinisialisasi. Ini membuat proses pembuatan objek menjadi lebih lengkap dan realistis dalam menggambarkan sifat-sifat yang diwarisi.
4. Program ini menunjukkan bagaimana karakteristik fisik dan perilaku bisa diwariskan melalui kelas induk Ortu. Contoh seperti hobi menonton film survival dan memancing diimplementasikan untuk menunjukkan bahwa minat juga bisa diturunkan atau dipengaruhi oleh orang tua.

[No.3] Identifikasi Masalah:

```
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");  
    }  
}
```

[No.3] Analisis dan Argumentasi

- 1) Permasalahan ini dapat diatas dengan cara menghapus angka 1 yang ada di public manusia
- 2) Perbedaan deklarasi constructor, method, dan method utama:
 - Constructor adalah method khusus yang dipanggil secara otomatis saat objek dari kelas dibuat. Constructor digunakan untuk menginisialisasi atribut objek. Dalam kode, terdapat constructor Manusia1(String nama, String rambut), yang menginisialisasi atribut nama dan rambut saat objek Manusia dibuat. Namun, terdapat kesalahan dalam kode karena nama constructor harus sama dengan nama kelas (Manusia), bukan Manusia1
 - Method adalah fungsi yang dapat dipanggil setelah objek dibuat untuk melakukan operasi tertentu. Contoh dalam kode ini adalah method sukaNonton(String film), yang menampilkan hobi menonton dari objek. Method ini bisa dipanggil kapan saja setelah objek terbentuk.
 - Method Utama Main Method public static void main(String[] args) adalah titik masuk utama eksekusi program Java. Di dalam method ini, objek Manusia dibuat, dan constructor serta method dipanggil. Semua eksekusi program dimulai dari method ini.
- 3) Kapan perlu menggunakan constructor dan method?
 - Constructor Digunakan ketika Anda ingin menginisialisasi nilai-nilai atribut objek segera setelah objek tersebut dibuat. Jika Anda perlu memastikan bahwa objek sudah memiliki data yang diperlukan saat dibuat, constructor adalah tempat yang tepat untuk melakukannya.

- Method Digunakan untuk menjalankan logika atau operasi setelah objek terbentuk. Method bisa digunakan berulang kali untuk menjalankan operasi tertentu, misalnya, method `sukaNonton` digunakan untuk menampilkan hobi dari objek yang dibuat.
- 4) Uraikan perbedaan berikut:
- a) Constructor Overloading vs Constructor Overriding:
 - Overloading Terjadi ketika ada beberapa constructor dalam satu kelas dengan jumlah atau tipe parameter yang berbeda. Setiap constructor melakukan tugas yang sama namun dengan cara yang sedikit berbeda tergantung pada parameter yang diterima.
 - Overriding Tidak berlaku untuk constructor. Constructor tidak bisa di-*override* karena constructor tidak diwariskan ke subclass, melainkan dibuat ulang pada kelas turunan.
 - b) Method Overloading vs Method Overriding:
 - Overloading Terjadi ketika beberapa method memiliki nama yang sama dalam satu kelas tetapi dengan parameter yang berbeda (jumlah atau tipe data). Ini memungkinkan method yang sama untuk menangani berbagai jenis input.
 - Overriding Terjadi ketika subclass mendefinisikan ulang method yang sudah ada di superclass. Method ini harus memiliki nama, tipe kembalian, dan parameter yang sama dengan method di superclass, tetapi memberikan implementasi yang berbeda.
 - c) Method yang Mengembalikan Nilai vs Method yang Tidak Mengembalikan Nilai:
 - Method yang Mengembalikan Nilai Method ini mengembalikan nilai setelah menyelesaikan tugasnya. Nilai ini dapat berupa tipe data apa pun seperti `int`, `String`, atau objek, dan dikembalikan dengan kata kunci.
 - Method yang Tidak Mengembalikan Nilai Method ini tidak mengembalikan nilai apa pun dan dideklarasikan dengan kata kunci `void`. Method hanya menjalankan tugasnya dan selesai tanpa ada nilai yang dikembalikan.

[No.3] Algoritma dan Kode Program

- 1) Algoritma
 - Mulai program
 - Mendeklarasikan `public class manusia`
 - Mendeklarasikan variabel `String nama` dan `rambut` didalam `public class manusia`
 - Mendeklarasikan `public manusia` dan `void sukaGame` didalam `public class manusia`
 - Mendeklarasikan `System.out.println()` didalam `public class manusia`
 - Mendeklarasikan `public static void main()` didalam `public class manusia`
 - Program selesai
- 2) Kode program

```

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("Bagas", "hitam");
19         satu.sukaNonton("horor");
20     }
21 }

```

Input/Output

Language Version: JDK 21.0.0 ☒ Interactive

Input Arguments

Output Generated Files

```

Nama saya : Bagas
Warna Rambut : hitam
Hobi Menonton : horor

```

Compiled and executed in 1.877 sec(s)

[No.3] Kesimpulan

Pada program itu saya menggunakan bentuk kelas public karena kelas Manusia harus dapat diakses dari luar packagenya, terutama dari method main yang menjalankan program utama. Perbaiki program dengan menambahkan nama constructor yang sesuai dengan nama kelas karena struktur Java mengharuskan constructor memiliki nama yang sama dengan kelasnya untuk menginisialisasi objek dengan benar.

[No.4] Identifikasi Masalah:

```

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
    }
}

```

[No.4] Analisis dan Argumentasi

- 1) Permasalahan pada kode program dapat diatasi dengan cara menambahkan perilaku pada anak
- 2) Alasan saya mengusulkan Solusi ini agar bisa melakukan anak extend ortu

[No.4] Algoritma dan Kode Program

- 1) Algoritma
 - Mulai Program
 - Mendeklarasikan public class ortu
 - Mendeklarasikan void suka nonton dan suka membaca didalam public class ortu
 - Mendeklarasikan public static void main(String []args) didalam public class ortu
 - Mendeklarasikan class anak extends ortu didalam public class ortu
 - Mendeklarasikan void suka menonton, suka membaca dan suka Game didalam class anak extends ortu
 - Mendeklarasikan public static void main() didalam class anak extends ortu
 - Mendeklarasikan System.out.println() di setiap metode yang di gunakan
 - Program selesai
- 2) Kode Program

```
1: public class Ortu { // membuat kelas induk
2:     void sakaMenonton(String a) { // method induk spesifik
3:         System.out.println("Nonton " + a);
4:     }
5:     void sakaMembaca(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 objek1 = new Ortu(); // memanggil objek induk
12:     objek1.sakaMenonton("Herita"); // memanggil sifat spesifik induk
13:     objek1.sakaMembaca("Koran"); // memanggil method dengan variabel dapat diubah
14: }
15: System.out.println("\n Sifat Anak :");
16: Anak objek2 = new Anak(); // memanggil objek anak
17: objek2.sakaMenonton("Film Drakor"); // memanggil sifat spesifik anak yang diturunkan induk
18: objek2.sakaMembaca("Anak One Piece"); // memanggil method ke induk yang otomatis diturunkan tanpa deklarasi ulang di anak
19: objek2.sakaGame("Mobile Legend");
20: }
21: }
22: class Anak extends Ortu {
23:     void sakaMenonton(int a, String b) {
24:         System.out.println("Nonton Jam " + a + " Malam " + b);
25:     }
26:     void sakaMenonton(String a) { // method induk spesifik
27:         System.out.println("Nonton " + a);
28:     }
29:     void sakaMembaca(String a) { // method induk umum bisa diubah anak
30:         System.out.println("Suka Baca " + a);
31:     }
32: }
33: // metode baru
34: void sakaGame(String a) { // method induk umum bisa diubah anak
35:     System.out.println("Suka Bermain Game " + a);
36: }
37: }
38: }
39: public static void main(String [] args) {
40:     System.out.println("Sifat Orang tua :");
41:     Ortu objek1 = new Ortu(); // memanggil objek induk
42:     objek1.sakaMenonton("Herita"); // memanggil sifat spesifik induk
43:     objek1.sakaMembaca("Koran"); // memanggil method dengan variabel dapat diubah
44: }
45: System.out.println("\n Sifat Anak :");
46: Anak objek2 = new Anak(); // memanggil objek anak
47: objek2.sakaMenonton("Film Drakor"); // memanggil sifat spesifik anak yang diturunkan induk
48: objek2.sakaMembaca("Anak One Piece"); // memanggil method ke induk yang otomatis diturunkan tanpa deklarasi ulang di anak
49: objek2.sakaGame("Mobile Legend");
50: }
51: }
```

Input/Output

Language Version: JDK 21.0.0 Interactive

Input Arguments

Output Generated Files

Sifat Orang tua :
Nonton Herita
Suka Baca Koran

Sifat Anak :
Nonton Jam 9 Malam Film Drakor
Suka Baca Komik One Piece
Suka Bermain Game Mobile Legend

Compiled and executed in 1.28 sec(s)

[No.4] Kesimpulan

Pada program itu, saya menggunakan bentuk kelas publik karena kelas ini harus dapat diakses oleh kelas atau objek lain di luar paketnya, sehingga memungkinkan program untuk berjalan dengan baik ketika kita ingin memanggil objek dari kelas lain (kelas Anak mewarisi kelas Ortu).