

Template Lembar Kerja Individu

Nama & NPM	Topik:	Tanggal:
Anggun Syavira Trinanda	Kelas java	15 september 2024

[N0.1] 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

Variabel Kelas Manusia terdiri dari dua atribut: nama dan rambut. Atribut nama menyimpan informasi dasar tentang seseorang, sedangkan atribut rambut memberikan detail tambahan tentang karakteristik fisik seseorang. Kedua atribut ini merupakan komponen penting dari identitas objek Manusia.

b. perilaku/ behavior untuk method!

Kelas ini memiliki constructor yang dapat menginisialisasi objek Manusia dengan parameter nama dan rambut. Saat objek dibuat, constructor ini secara langsung mencetak informasi yang relevan ke konsol. Metode tambahan seperti `displayInfo()`—yang menampilkan informasi lengkap tentang objek—dan getter dan setter—yang memungkinkan akses dan perubahan atribut nama dan rambut—dapat ditambahkan untuk meningkatkan fungsionalitas kelas. Oleh karena itu, kelas manusia tidak hanya menyimpan data, tetapi juga memiliki perilaku yang membantu mereka berinteraksi dengan data.

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

[Video Materi 1 tentang Kelas, Objek, Method – https://www.youtube.com/watch?v=60ldOc8m8Es](https://www.youtube.com/watch?v=60ldOc8m8Es)

[Video Materi 2 tentang – https://www.youtube.com/watch?v=6qULMlcv-eg](https://www.youtube.com/watch?v=6qULMlcv-eg)

[No.1] Analisis dan Argumentasi

1) Uraikan rancangan solusi yang diusulkan.

Solusi untuk kelas manusia bertujuan untuk menampilkan informasi dasar tentang seseorang, seperti nama dan jenis rambut. Solusi ini membuat kelas manusia memiliki dua atribut, yaitu nama dan rambut, untuk menyimpan data identitas objek. Selain itu, kelas ini juga memiliki konstruktor yang menerima dua parameter, yaitu nama dan rambut, yang digunakan untuk menginisialisasi atribut dan mencetak informasi tersebut ke konsol saat objek diproses. Metode `main` berfungsi sebagai titik masuk program, di mana objek manusia dibuat dengan memberikan nilai untuk nama dan rambut, menciptakan interaksi yang mudah dan langsung.

2) Analisis solusi, kaitkan dengan permasalahan.

analisis solusi, kelas manusia menangani masalah representasi identitas individu dengan mudah dan jelas. Kelas ini memudahkan pengelolaan data dengan memanfaatkan konstruktor untuk menerima dan menampilkan data. Namun, solusi ini masih memiliki beberapa keterbatasan. Saat ini, kelas hanya membahas dua sifat, membuatnya kurang komprehensif untuk menggambarkan sifat manusia secara keseluruhan.

Fleksibilitas dan fungsionalitas adalah masalah yang mungkin muncul. Misalnya, kelas harus dimodifikasi lebih lanjut jika ingin menambahkan atribut seperti umur, alamat, atau jenis kelamin. Selain itu, penggunaan kelas ini menjadi terbatas karena tidak ada cara untuk mengakses atau mengubah data atribut setelah objek dibuat.

[No.1] Penyusunan Algoritma dan Kode Program

- 1) Rancang desain solusi atau algoritma
 - Definisi Kelas = Buat kelas dengan nama Manusia.
 - Deklarasikan dua atribut: = String nama, rambut
 - Buat konstruktor yang menerima dua parameter:
 - Parameter String nama untuk inisialisasi atribut nama.
 - Parameter String rambut untuk inisialisasi atribut rambut.
 - Dalam konstruktor Cetak informasi nama dan jenis rambut ke konsol.
 - Metode Main
 - Buat metode main sebagai titik masuk program.
 - Dalam metode main, lakukan langkah berikut:
 - Buat objek dari kelas Manusia dengan memberikan nilai untuk nama dan rambut (misalnya, "anggun syavira" dan "pendek").
 - Konstruktor secara otomatis akan dipanggil dan mencetak informasi ke konsol.
- 2) Tuliskan kode program dan luaran
 - a) Beri komentar pada kode

Kode di atas mendefinisikan kelas Manusia, yang menampilkan informasi dasar tentang orang. Kelas ini menyimpan data identitas individu dengan dua atribut, nama dan rambut. Dua parameter ini diberikan kepada constructor kelas untuk menginisialisasi atribut tersebut. Konstruktor akan mencetak nama dan jenis rambut ke konsol saat objek dibuat untuk memberikan umpan balik langsung kepada pengguna. Program dimulai dengan metode main, di mana objek manusia diberi nilai tertentu, seperti "anggun syavira" untuk nama dan "pendek" untuk jenis rambut. Dengan cara ini, saat objek dihidupkan, konstruktor secara otomatis dipanggil dan data yang relevan ditampilkan. Struktur kode ini memungkinkan representasi data individu yang mudah, tetapi dapat diperluas untuk

- b) Uraikan luaran yang dihasilkan
 - Nama saya : anggun
 - Warna Rambut : hitam
- c) Screenshot/ Capture potongan kode dan hasil luaran



```
1 public class Manusia {
2     // deklarasi kelas
3     // deklarasi variabel
4     String nama;
5     String rambut;
6
7     // deklarasi constructor tanpa parameter
8     public Manusia(String nama, String rambut) {
9         System.out.println("nama saya : " + nama);
10        System.out.println("rambut saya : " + rambut);
11    }
12
13    public static void main(String[] args) {
14        Manusia satu = new Manusia("anggun syavira", "pendek");
15    }
16 }
17
18
19
```

Input/Output

Language Version: JDK 21.0.0 ☒ Interactive

Input Arguments

Output Generated Files

```
nama saya : anggun syavira
rambut saya : pendek
```

[N0.1] Kesimpulan

1) Evaluasi

a) Apa konsekuensi dari skenario pemrograman ini?

Skenario pemrograman ini memiliki beberapa konsekuensi. Pertama, kelas `Manusia` dengan konstruktor yang mencetak informasi langsung ke konsol menciptakan interaksi yang sederhana, tetapi kurang fleksibel. Jika ingin menyimpan informasi atau mengakses atribut di kemudian hari, saat ini tidak ada metode untuk melakukannya. Selain itu, desain ini mengandalkan output ke konsol, yang mungkin tidak cocok untuk aplikasi yang lebih besar, di mana data perlu diproses lebih lanjut atau disimpan dalam format lain. Jika atribut atau perilaku baru ingin ditambahkan di masa mendatang, modifikasi pada kelas akan diperlukan, yang bisa menyebabkan kompleksitas tambahan jika tidak direncanakan dengan baik.

Template Lembar Kerja Individu

Nama & NPM	Topik:	Tanggal:
Anggun Syavira Trinanda	Kelas java	17 september 2024

[N0.2] Identifikasi Masalah:

1) 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?

-Sifat (Atribut) Kelas Ortu terdiri dari beberapa sifat yang menunjukkan sifat fisik dan identitas seseorang. Atribut-atribut ini termasuk: Nama dan warna rambut menyimpan nama dan warna rambut. Jenis kelamin dan warna kulit menyimpan warna kulit dan tinggi badan menyimpan tinggi badan dalam cm dan berat badan menyimpan berat badan dalam kg. Untuk menggambarkan sifat orang tua yang dapat diturunkan kepada anak, sifat-sifat ini dapat diwariskan oleh kelas anak (turunan). untuk menggambarkan karakteristik orang tua yang dapat diturunkan kepada anak.

-Dalam kelas Ortu, konstruktor konstruktor dirancang untuk menginisialisasi objek dengan atribut yang telah disebutkan sebelumnya. Dalam kasus ini, konstruktor ini menerima sejumlah parameter untuk mengisi nilai atribut. Kelebihan: Constructor memungkinkan kita memastikan bahwa setiap objek Ortu yang dibuat memiliki semua informasi yang diperlukan. Ini membuatnya lebih jelas dan meminimalkan kesalahan saat menginisialisasi objek.

-Perilaku Positif (behavior) Perilaku atau perilaku positif yang dapat berasal dari kelas Ortu termasuk: Pencetakan Informasi: Metode pembuatan yang mencetak informasi ciri-ciri individu saat objek dibuat menunjukkan perilaku informatif. Ini dapat membantu dalam memberikan umpan balik kepada pengguna dan debugging. Pewarisan: Jika kelas Anak diturunkan dari kelas Ortu, anak tersebut akan mewarisi semua constructor dan atribut. Ini berarti bahwa objek Anak akan memiliki semua karakteristik orang tua, yang memungkinkan kita untuk membuat hierarki objek yang mencerminkan hubungan keluarga. Kemampuan Overriding: Guru anak dapat mengubah perilaku yang ada dengan menambahkan teknik baru untuk menunjukkan perilaku unik anak atau mengubah cara informasi dipresentasikan.

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

[Video Materi 1 tentang Kelas, Objek, Method – https://www.youtube.com/watch?v=60ldOc8m8Es](https://www.youtube.com/watch?v=60ldOc8m8Es)

[Video Materi 2 tentang – https://www.youtube.com/watch?v=6qULMlcv-eg](https://www.youtube.com/watch?v=6qULMlcv-eg)

[No.2] Analisis dan Argumentasi

1) Uraikan rancangan solusi yang diusulkan.

Solusi yang disarankan berkonsentrasi pada pembuatan kelas "Ortu" yang dapat secara rinci menunjukkan karakteristik individu dalam pemrograman berorientasi objek. Dengan menggunakan atribut seperti "nama", "rambut", "jenis kelamin", "warna kulit", "tinggi badan", dan "berat badan", kelas ini memberikan gambaran lengkap tentang orang tersebut. Konstruktor kelas "Ortu" dirancang untuk menerima semua atribut sebagai parameter, memastikan bahwa setiap objek yang dibuat memiliki data yang lengkap dan diinisialisasi dengan benar, dan mencetak informasi tersebut secara langsung untuk umpan balik pengguna. Metode "main" adalah titik awal eksekusi, di mana objek "Ortu" dibuat dengan semua atribut yang dibutuhkan. Pengembangan tambahan, seperti pewarisan ke kelas lain, penambahan teknik tambahan, dan validasi data input, dapat dilakukan dengan desain ini. Akibatnya, desain ini membuat kode lebih mudah dibaca dan lebih terorganisir.

2) Analisis solusi, kaitkan dengan permasalahan.

Analisis solusi yang ditawarkan oleh kelas Ortu menunjukkan metode yang efektif untuk menyelesaikan masalah yang sering terjadi dalam pengelolaan data individu dalam pemrograman. Kelas ini mendefinisikan atribut seperti nama, rambut, jenis kelamin, warna kulit, tinggi badan, dan berat badan, yang memungkinkan representasi data secara sistematis dan terorganisir, yang membuatnya lebih mudah dipahami dan dikelola. Konstruktor memastikan bahwa setiap objek yang dibuat memiliki semua data yang diperlukan untuk menghindari kesalahan data yang tidak lengkap dan memberikan umpan balik langsung kepada pengguna. Selain itu, pengembangan lebih lanjut, seperti pembuatan kelas turunan yang dapat mewarisi atribut dari kelas Ortu, memungkinkan penambahan fitur baru menjadi lebih fleksibel, didukung oleh rancangan ini. Pendekatan berorientasi objek yang digunakan meningkatkan keterbacaan dan kemudahan pemeliharaan karena memastikan bahwa setiap bagian kode memiliki tanggung jawab yang jelas. Secara umum, kelas Ortu memberikan dasar yang kuat untuk pengembangan aplikasi lebih lanjut, sambil memastikan bahwa kode tetap mudah dipahami dan dikelola, yang sangat penting dalam konteks pengembangan perangkat lunak yang membutuhkan keandalan dan kemudahan perawatan.

[No.2] Penyusunan Algoritma dan Kode Program

1) Rancang desain solusi atau algoritma

- Mulai
- Definisikan kelas Ortu
- Deklarasi Constructor:
- Dalam Constructor
- Method main
- akhiri

2) Tuliskan kode program dan luaran

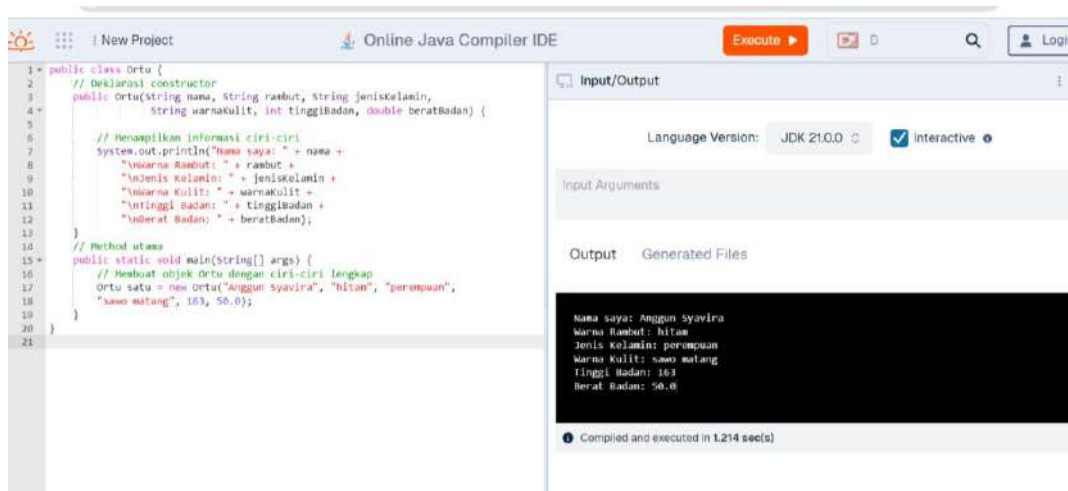
a) Beri komentar pada kode

Kode di atas mendefinisikan kelas `Ortu`, yang berfungsi untuk merepresentasikan karakteristik individu, seperti nama, warna rambut, jenis kelamin, warna kulit, tinggi badan, dan berat badan. Kelas ini memiliki konstruktor yang menerima berbagai parameter untuk menginisialisasi objek dan langsung mencetak informasi ciri-ciri individu ke konsol saat objek dibuat. Ini memungkinkan pengguna untuk dengan mudah melihat data yang diinput tanpa memerlukan method tambahan untuk menampilkannya. Di dalam method `main`, sebuah objek `Ortu` dibuat dengan parameter yang spesifik, sehingga saat program dijalankan, informasi lengkap

tentang individu tersebut akan ditampilkan secara otomatis. Pendekatan ini menunjukkan penerapan prinsip pemrograman berorientasi objek, seperti enkapsulasi dan penggunaan konstruktor, sehingga memudahkan pengelolaan dan representasi data dalam aplikasi.

- b) Uraikan luaran yang dihasilkan
- Nama saya: Anggun Syavira
Warna Rambut: hitam
Jenis Kelamin: perempuan
Warna Kulit: sawo matang
Tinggi Badan: 163
Berat Badan: 50.0

- c) Screenshot/ Capture potongan kode dan hasil luaran



The screenshot shows an online Java IDE with the following code in the editor:

```
1 public class Ortu {
2     // Deklarasi konstruktor
3     public Ortu(String nama, String rambut, String jenisKelamin,
4         String warnaKulit, int tinggiBadan, double beratBadan) {
5
6         // Menampilkan informasi ciri-ciri
7         System.out.println("Nama saya: " + nama +
8             "\nWarna Rambut: " + rambut +
9             "\nJenis kelamin: " + jenisKelamin +
10            "\nWarna kulit: " + warnaKulit +
11            "\nTinggi badan: " + tinggiBadan +
12            "\nBerat Badan: " + beratBadan);
13     }
14
15     // Method utama
16     public static void main(String[] args) {
17         // Membuat objek Ortu dengan ciri-ciri lengkap
18         Ortu satu = new Ortu("Anggun Syavira", "hitam", "perempuan",
19             "sawo matang", 163, 50.0);
20     }
21 }
```

The right-hand pane shows the 'Input/Output' section with the following output:

```
Nama saya: Anggun Syavira
Warna Rambut: hitam
Jenis kelamin: perempuan
Warna Kulit: sawo matang
Tinggi Badan: 163
Berat Badan: 50.0
```

Below the output, it states: 'Compiled and executed in 1.214 sec(s)'.

[NO.2] Kesimpulan

- 1) Evaluasi

- a) Apa konsekuensi dari skenario pemrograman ini?

Dalam skenario pemrograman ini, kelas Ortu memberikan struktur yang jelas untuk menyimpan dan menampilkan data pribadi, yang membuat pengelolaan data lebih mudah. Dengan menggunakan konstruktor, setiap objek Ortu dihidupkan dengan semua informasi ciri-cirinya, sehingga tidak ada data yang terlewat. Namun, metode ini memiliki beberapa kekurangan. Misalnya, informasi ditampilkan langsung setiap kali objek baru dibuat, yang dapat menyebabkan pengulangan data di konsol jika banyak objek diperlukan. Selain itu, desain ini tidak mendukung extensibility, yang berarti pengembang harus memodifikasi kelas secara langsung jika ingin menambahkan atribut baru atau memperluas fungsionalitas. Jika mereka melakukannya, ini akan mengakibatkan kesalahan. Data yang ditampilkan tidak disimpan dalam atribut internal yang dapat diakses atau diubah dalam konteks lain, sehingga fokus pada output dapat menghambat pengembangan lebih lanjut. Oleh karena itu, meskipun skenario ini efektif untuk demonstrasi dasar pemrograman berorientasi objek, terdapat beberapa aspek yang perlu diperhatikan untuk pengembangan dan pengelolaan data yang lebih efisien.

Template Lembar Kerja Individu

Nama & NPM	Topik:	Tanggal:
Anggun Syavira Trinanda	Kelas java	17 september 2024

[N0.3] Identifikasi Masalah:

3) Uraikan permasalahan dan variabel

```
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!

1. Deklarasi Konstruktor: Nama konstruktor dalam kelas Manusia seharusnya sama dengan nama kelas, tetapi pada kode yang diberikan, nama konstruktor ditulis sebagai Manusia1, yang salah; nama kelas seharusnya ditulis sebagai Manusia. Konstruktor menginisialisasi objek kelas dan menerima parameter atribut kelas. Dalam kasus ini, konstruktor menerima dua parameter—nama dan rambut—dan mencetak informasi ini ke konsol saat objek dibuat. Konstruktor memiliki nama kelas yang sama dan tidak memiliki tipe kembalian.
2. Deklarasi Metode : Metode SukaNonton Fungsi kelas ini dapat dipanggil setelah objek dibuat. Metode ini menerima satu syarat, yaitu film, dan mencetak data tentang minat menonton. Metode ini berbeda dari konstruktor karena memiliki tipe kembalian yang dapat menjadi void, yang berarti tidak mengembalikan nilai. Metode ini tidak terbatas pada proses pembuatan objek, dan dapat digunakan berkali-kali pada objek yang sama.
3. Metode utama, main, adalah titik masuk program Java dan merupakan metode statis. Metode ini dideklarasikan dengan kata kunci public static, yang memungkinkan JVM untuk menjalankan program tanpa perlu membuat objek dari kelas tersebut. Di dalam metode ini, objek dari kelas Manusia dibuat, dan metode sukaNonton dipanggil untuk mencetak informasi tentang hobi menonton. Metode utama juga tidak memiliki parameter selain argumen String[] args, yang digunakan untuk menerima input dari baris perintah.

3.2. Tentukan kapan Anda perlu menggunakan constructor dan method?

- Konstruktor: Digunakan saat perlu menginisialisasi atribut objek saat objek dibuat. Cocok untuk memberikan nilai awal pada atribut dan menempatkan logika khusus yang diperlukan saat objek diciptakan.
- Metode: Digunakan untuk mendefinisikan perilaku objek, seperti tindakan yang dapat dilakukan (misalnya, hobi). Metode juga digunakan untuk mengakses atau mengubah atribut setelah objek diciptakan dan dapat dipanggil berulang kali untuk interaksi dinamis dengan objek.

3.3. Uraikan perbedaan berikut:

a) constructor overloading dan overriding

-constructor overloading dan overriding konstruktor overloading: Ketika sebuah kelas memiliki lebih dari satu konstruktor dengan nama yang sama tetapi dengan parameter yang berbeda, seperti jumlah, tipe, atau urutan, ini disebut sebagai konsep.

memungkinkan pembuatan objek yang dapat disesuaikan. Untuk ilustrasi, Anda dapat memiliki satu konstruktor untuk kelas Manusia yang menerima dua parameter, seperti nama dan rambut, dan yang lain untuk kelas Manusia yang menerima hanya satu parameter, seperti nama saja.

-konstruktor Overriding: Konstruktor tidak dapat dioverride, jadi istilah "overriding" tidak berlaku untuknya. Namun, kata kunci "super" dapat digunakan untuk menghubungkan subclass ke konstruktor superclass. Karena setiap kelas memiliki konstruktor unik, ide overriding tidak diterapkan di sini.

b) method overloading, dan method overriding

Ketika sebuah kelas memiliki sejumlah metode dengan nama yang sama tetapi dengan parameter yang berbeda, metode overloading terjadi. Ini memungkinkan metode untuk beroperasi dengan cara yang berbeda tergantung pada jenis dan jumlah argumen yang diberikan. Contohnya, kita dapat memiliki metode tampil dengan satu parameter untuk menampilkan nama, dan metode tampil lain dengan dua parameter untuk menampilkan nama dan umur. Namun, metode overriding terjadi ketika subclass memiliki metode dengan nama dan parameter yang sama dengan metode di superclass. Hal ini memungkinkan subclass untuk melakukan implementasi yang berbeda dari metode tersebut, menggantikan perilaku default yang ditentukan di superclass.

c) method yang mengembalikan nilai dan method tidak mengembalikan nilai

Metode pengembalian nilai menggunakan tipe pengembalian seperti int, String, atau tipe data lainnya, dan kemudian mengembalikan nilai kepada pemanggil dengan kata kunci return. Sebagai contoh, metode int tambah(int a, int b) mengembalikan hasil penjumlahan. Di sisi lain, metode yang tidak mengembalikan nilai didefinisikan dengan tipe pengembalian void, yang berarti metode tersebut hanya melakukan aksi tertentu tanpa mengembalikan nilai. Metode void tampil() hanya mencetak informasi tetapi tidak memberikan nilai kembali kepada pemanggil.

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

[Video Materi 1 tentang Kelas, Objek, Method – https://www.youtube.com/watch?v=60ldOc8m8Es](https://www.youtube.com/watch?v=60ldOc8m8Es)

[Video Materi 2 tentang – https://www.youtube.com/watch?v=6qULMlcv-eg](https://www.youtube.com/watch?v=6qULMlcv-eg)

[No.3] Analisis dan Argumentasi

1) Uraikan rancangan solusi yang diusulkan.

Tujuan dari rancangan solusi kode ini adalah untuk mendefinisikan kelas Manusia yang memiliki atribut nama dan warna rambut. Kelas ini juga memiliki konstruktor yang seharusnya

menginisialisasi atribut-atribut tersebut ketika objek dibuat. Namun, ada kesalahan dalam kode yang menyebabkan penggunaan nama konstruktor Manusia (1), yang seharusnya merupakan nama manusia. Untuk menunjukkan hobi menonton seseorang, metode sukaNonton ditambahkan. Dalam metode main, kita membuat objek dari kelas Manusia dan memanggil metode sukaNonton untuk menunjukkan hobi menonton.

2) Analisis solusi, kaitkan dengan permasalahan.

Ada kesalahan penamaan konstruktor dalam kode ini. Agar konstruktor dapat diinisialisasi dengan benar saat membuat objek, konstruktor harus memiliki nama yang sama dengan kelas, yaitu Manusia. Kesalahan kompilasi akan terjadi karena ketidakcocokan ini.

Selain itu, solusi yang ditawarkan umumnya bertujuan untuk mendefinisikan perilaku dan sifat manusia, tetapi perlu diperbaiki agar berfungsi. Setelah perbaikan, program dapat digunakan untuk membuat objek manusia, seperti menampilkan nama dan warna rambut, dan menampilkan hobi menonton, yang menciptakan interaksi mudah dan fungsionalitas yang diinginkan.

[No.3] Penyusunan Algoritma dan Kode Program

1) Rancang desain solusi atau algoritma

- Deklarasi Kelas: Buat kelas dengan nama Manusia.
- Deklarasi Atribut: string nama, rambut
- Deklarasi Constructor:

Buat constructor dengan dua parameter (string nama, string rambut):

Saat constructor dipanggil, tampilkan informasi nama dan warna rambut dengan format: "Nama saya: [nama]", "Warna Rambut: [rambut]"

- Deklarasi Method: Buat method sukaNonton(String film):

Method ini mencetak hobi menonton film dengan format: "Hobi Menonton: [film]"

- Deklarasi Method main:

Buat method main sebagai titik awal program:

Buat objek satu dari kelas Manusia, dan inisialisasi dengan nama "Putri" dan rambut "hitam".

Panggil method sukaNonton pada objek satu dengan parameter "Drakor"

2) Tuliskan kode program dan luaran

a) Beri komentar pada kode

Kode sebelumnya mendefinisikan kelas manusia dengan dua ciri: nama dan rambut.

Saat objek baru dibuat, Constructor Manusia digunakan untuk menerima dua parameter dan mencetak informasi tentang nama dan warna rambut. Ini menunjukkan bahwa informasi ditampilkan langsung saat objek dibuat. Menerima nama film dan mencetak kesenangan menonton yang dimiliki oleh film juga merupakan bagian dari metode sukanonton. Pada metode main, objek Manusia dengan nama "anggun" dan rambut berwarna "hitam" dibuat. Kemudian, metode sukaNonton digunakan untuk menunjukkan bahwa hobi menonton objek ini adalah "Drakor". Kode ini menggambarkan konsep dasar pemrograman berorientasi objek Java, seperti penggunaan constructor, atribut, dan metode. Kode berhasil mencetak data yang diharapkan.

b) Uraikan luaran yang dihasilkan

Nama saya : anggun

Warna Rambut : hitam

Hobi Menonton : Drakor

c) Screenshot/ Capture potongan kode dan hasil luaran

```
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("anggun", "hitam");
19        satu.sukaNonton("Drakor");
20    }
21 }
```

Input/Output

Language Version: JDK 21.0.0 ☐ Interactive

Input Arguments

Stdin Inputs

Output Generated Files

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

[N0.3] Kesimpulan

1) Evaluasi

a) Apa konsekuensi dari skenario pemrograman ini?

Beberapa hal penting terjadi sebagai akibat dari skenario pemrograman ini. Pertama, meskipun konstruktor mencetak nama dan warna rambut pada saat pembuatan objek Manusia, informasi ini tidak disimpan dalam atribut kelas, sehingga tidak dapat diakses setelah objek dibuat. Kedua, konstruktor yang hanya mencetak informasi dapat membingungkan karena konstruktor biasanya digunakan untuk menginisialisasi atribut. Karena metode ini tidak menghasilkan nilai yang dapat diuji, pengujian unit menjadi lebih sulit. Karena keterbatasan ini, tidak ada cara untuk mengikuti beberapa hobi atau fitur lain dari objek manusia; oleh karena itu, menambah fitur atau teknik baru di masa depan mungkin tidak efektif. Terakhir, informasi tentang nama dan warna rambut dicetak saat objek dibuat, tidak dapat diubah. Ini dapat mengganggu pengalaman pengguna. Untuk meningkatkan kode, gunakan konstruktor untuk

Template Lembar Kerja Individu

Nama & NPM	Topik:	Tanggal:
Anggun Syavira Trinanda	Kelas java	18 september 2024

[N0.4] 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 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
}

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 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
}
}
```

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`!

Kelas Anak, yang berasal dari kelas Ortu, memiliki beberapa metode yang berbeda dari kelas induknya. Pertama, kelas Anak menambahkan metode baru `sukaMenonton(int a, String b)` sebagai contoh overloading. Metode ini menerima dua parameter, jam tayang dan judul tayangan, yang memberikan fleksibilitas tambahan untuk menampilkan tayangan. Selain itu, metode `sukaMenonton(String a)` dan `sukaMembaca(String a)` dari kelas Ortu dioverride oleh kelas Anak. Akibatnya, jika metode ini dipanggil pada objek Anak, versi yang sudah ada di kelas Anak akan dieksekusi. Meskipun perilaku metode yang dioverride tidak berbeda secara fungsional dari perilaku kelas induk, ini memberi kelas Anak kesempatan untuk memperluas atau mengubah perilaku mereka jika diperlukan. Oleh karena itu, kelas Anak tidak hanya mewarisi metode kelas Ortu, tetapi juga meningkatkan fungsionalitas dan fleksibilitasnya melalui penambahan method baru dan pengubahan method yang ada, mencerminkan prinsip pewarisan dan polimorfisme dalam pemrograman berorientasi objek.!

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

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

[Video Materi 1 tentang Kelas, Objek, Method – https://www.youtube.com/watch?v=60ldOc8m8Es](https://www.youtube.com/watch?v=60ldOc8m8Es)

[Video Materi 2 tentang – https://www.youtube.com/watch?v=6qULMlcv-eg](https://www.youtube.com/watch?v=6qULMlcv-eg)

[No.4] Analisis dan Argumentasi

- 1) Uraikan rancangan solusi yang diusulkan.
Solusi ini menggunakan pewarisan dalam pemrograman berorientasi objek untuk mensimulasikan perilaku orang tua dan anak. Dalam rancangan ini, kelas Ortu berfungsi sebagai kelas induk dengan dua method: `sukaMenonton(String a)` dan `sukaMembaca(String a)`, yang menunjukkan aktivitas umum yang dapat dilakukan oleh orang tua. Kelas Anak, yang merupakan turunan dari kelas Ortu, memperluas fungsionalitasnya dengan menambahkan method baru `sukaMenonton(int a, String b)` untuk menunjukkan bahwa anak-anak dapat mengkomunikasikan kegiatan menonton dengan cara yang lebih khusus, seperti dengan menggunakan `m`. Selain itu, kelas Anak mengabaikan metode kelas induk, sehingga objek dari kelas ini dapat berperilaku berbeda saat menggunakan metode yang sama. Perilaku yang telah didefinisikan ditunjukkan dan diuji dengan metode `main` di kedua kelas, dengan mencetak aktivitas masing-masing.
- 2) Analisis solusi, kaitkan dengan permasalahan.
Metode ini bagus untuk menjelaskan hubungan antara orang tua dan anak melalui pewarisan. Dengan menggunakan kelas induk Ortu, kita dapat mendefinisikan sifat dan perilaku umum yang dapat diwariskan kepada kelas Anak. Ini menunjukkan bahwa anak-anak mungkin memiliki sifat dan perilaku yang sama dengan orang tua mereka, tetapi mereka juga dapat memiliki sifat dan perilaku yang berbeda. Metode overloading dan overriding memberikan fleksibilitas yang diperlukan untuk menciptakan perilaku khusus yang lebih sesuai dengan kebutuhan masing-masing kelas.
Solusi ini menangani masalah mengorganisasikan dan mengorganisasikan sifat dan tindakan yang berbeda di antara dua entitas (orang tua dan anak). Metode ini memanfaatkan prinsip pemrograman berorientasi objek dasar seperti pewarisan dan polimorfisme untuk membuat kode yang lebih modular, mudah dibaca, dan mudah digunakan. Hal ini memungkinkan pengembangan lebih lanjut di masa depan, misalnya dengan menambahkan kelas lain yang mewakili anggota keluarga yang berbeda, tanpa perlu mengubah banyak kode yang sudah ada.

[No.4] Penyusunan Algoritma dan Kode Program

- 1) Rancang desain solusi atau algoritma
 - **Mulai Program**
 - **definisikan Kelas Ortu**
Deklarasikan metode sukaMenonton(String a)
Deklarasikan metode sukaMembaca(String a)
Deklarasikan metode sukaOlahraga(String a)
Deklarasikan metode main(String[] args):
 - **Definisikan Kelas Anak (yang mewarisi dari Ortu)**
Deklarasikan metode sukaMenonton(int a, String b)
Deklarasikan metode sukaMenonton(String a)
Deklarasikan metode sukaMembaca(String a)
 - **Deklarasikan metode main(String[] args) dalam kelas Anak**
 - **Akhiri program**

- 2) Tuliskan kode program dan luaran

- a) Beri komentar pada kode

Kode sebelumnya mendefinisikan dua kelas, "Ortu" dan "Anak", berdasarkan gagasan pewarisan dalam pemrograman berorientasi objek. Kelas "Ortu" berfungsi sebagai kelas induk, dan memiliki tiga metode: "sukaMenonton(String a)", "sukaMembaca(String a)", dan "sukaOlahraga(String a)". Dalam metode main, objek "Ortu" dibuat, dan metode-metode tersebut disebut untuk menunjukkan sifat-sifat orang tua. Kelas "Anak", yang merupakan turunan dari "Ortu", mengoverride Metode "main" kelas Anak membuat objek "Anak". Metode kedua kelas dipanggil, termasuk metode "sukaolahraga", yang hanya ada di kelas "Ortu". Hal ini menunjukkan bagaimana anak mewarisi sifat dari orang tua dan juga memiliki perilaku uniknya sendiri. Namun, terdapat redundansi dalam kode karena method `main` di kelas `Anak` terduplikasi dengan `main` di kelas `Ortu`, yang seharusnya cukup didefinisikan satu kali saja untuk menguji semua fungsionalitas.

- b) Uraikan luaran yang dihasilkan

Sifat Orang Tua :

Nonton Berita


Suka Baca Koran

Sifat Anak :

Nonton Jam 9 Malam Film Drakor

Suka Baca Komik One Piece

- c) Suka Olahraga basketScreenshot/ Capture potongan kode dan hasil luaran



The screenshot shows an online Java compiler interface. On the left, the code editor contains the following Java code:

```
1 public class Ortu {
2     void sukaMenonton(String a) {
3         System.out.println("Nonton " + a);
4     }
5     void sukaMembaca(String a) {
6         System.out.println("Suka Baca " + a);
7     }
8     void sukaOlahraga(String a) {
9         System.out.println("Suka Olahraga " + a);
10    }
11 }
12
13 public static void main(String[] args) {
14     System.out.println("Sifat Orang Tua :");
15     Ortu objOrtu = new Ortu();
16     objOrtu.sukaMenonton("Berita");
17     objOrtu.sukaMembaca("Koran");
18 }
19
20 class Anak extends Ortu {
21     void sukaMenonton(int a, String b) {
22         System.out.println("Nonton Jam " + a + " Malam " + b);
23     }
24     void sukaMembaca(String a) {
25         System.out.println("Nonton " + a);
26     }
27     void sukaOlahraga(String a) {
28         System.out.println("Suka Olahraga " + a);
29     }
30 }
```

On the right, the 'Input/Output' panel shows the output of the program:

```
Language Version: JDK 21.0.0
Interactive

Input Arguments:

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 Olahraga Basket

Completed and executed in 1.410 seconds
```

[N0.4] Kesimpulan

1) Evaluasi

a) Apa konsekuensi dari skenario pemrograman ini?

Skenario pemrograman ini menyebabkan redundansi dan ketidakefisienan dalam kode. Pertama, adanya dua pendekatan main yang berbeda di kelas Ortu dan Anak menyebabkan duplikasi logika karena kedua pendekatan ini melakukan hal yang sama, mencetak sifat dari orang tua dan anak. Karena perubahan yang diperlukan dilakukan di dua lokasi berbeda, ukuran kode meningkat. Hal ini juga membuatnya lebih sulit untuk dikelola. Selain itu, tidak ada cara untuk memaksimalkan perilaku pewarisan, seperti memanggil metode induk dari objek anak secara langsung, yang dapat meningkatkan kebersihan dan kejelasan kode, meskipun kelas Anak mewarisi metode dari kelas Ortu. Dengan mengubah struktur ini, programmer dapat membuat desain yang lebih modular dan efektif, yang tidak hanya menyederhanakan pemeliharaan kode tetapi juga meningkatkan efisiensi. meningkatkan pembacaan dan penggunaan kembali kode di masa depan.