|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **WINDA DWI SALISTIANNA**  **G1F024004** | **Class, Objek, dan Method** | **18 September 2024** |

**UNIT 1 KELAS (CLASS)**

**Latihan 1:**  
1.1.     Perbaiki pesan kesalahan Contoh 1!  
1.2.     Analisa ciri-ciri lain Kelas Manusia yang dapat menjadi  
           a. atribut variabel, dan  
           b. perilaku/ behavior!

JAWAB:

Sebuah gambar berisi teks, cuplikan layar, Font, tampilan

Deskripsi dibuat secara otomatis1.1 Masalah utama dari kode Anda adalah ketidaksesuaian nama kelas dan nama constructor. Nama kelas yang dideklarasikan adalah Manusia, tetapi constructor yang digunakan adalah Manusia1. Jadi kode yang benar adalah sebagai berikut :

Gambar 1.1 kode class

Source kode :

**package** Kelas;

**public** **class** Manusia { // deklarasi kelas

// deklarasi atribut Manusia dalam variabel

String nama;

String rambut;

// deklarasi constructor

**public** Manusia(String nama, String rambut) {

System.***out***.println("Nama saya : " + nama +

"\nWarna Rambut : " + rambut);

}

// deklarasi method utama

**public** **static** **void** main(String[] args) {

Manusia satu = **new** Manusia("Winda", "hitam");

}

}

Penjelasan :

Kode di atas mendefinisikan sebuah kelas Manusia dalam paket Kelas. Kelas ini memiliki dua atribut: nama dan rambut, yang keduanya adalah variabel bertipe String. Atribut-atribut ini digunakan untuk menyimpan informasi mengenai nama dan warna rambut seseorang. Constructor Manusia diatur untuk menerima dua parameter bertipe String, yaitu nama dan rambut. Namun, dalam constructor ini, variabel-variabel ini tidak digunakan untuk menginisialisasi atribut kelas; alih-alih, mereka langsung digunakan dalam pernyataan System.out.println untuk mencetak informasi ke konsol. Di dalam metode main, yang merupakan titik awal eksekusi program, sebuah objek Manusia bernama satu dibuat dengan nama "Winda" dan warna rambut "hitam". Ketika objek ini diinstansiasi, constructor Manusia dipanggil dan mencetak informasi tentang nama dan warna rambut objek tersebut. Penting untuk dicatat bahwa dalam implementasi ini, meskipun constructor menerima parameter dan mencetak informasi, atribut kelas nama dan rambut tidak diinisialisasi dengan nilai-nilai ini. Sebuah gambar berisi teks, Font, cuplikan layar, putih

Deskripsi dibuat secara otomatisJadi, meskipun atribut tersedia, nilai-nilai tersebut tidak digunakan di luar constructor.

*Gambar 1.2 Output*

1.2

a). Atribut variabel adalah data atau informasi yang disimpan dalam objek dari sebuah kelas. Jadi atribut variabel dalam gambar 1.1 adalah :

* Nama : menyimpan nama dari objek manusia
* rambut: Menyimpan warna rambut dari objek Manusia

b). perilaku/ behavior merujuk pada cara objek dari sebuah kelas berinteraksi dengan data dan objek lain serta bagaimana mereka melakukan aksi tertentu. **Constructor** Manusia(String nama, String rambut): Constructor ini diresponsibel untuk menginisialisasi atribut nama dan rambut ketika objek Manusia dibuat. Constructor ini juga mencetak informasi tentang nama dan warna rambut objek.

**UNIT 2 OBJEK**

**Latihan 2:**2.1.  Evaluasi penyebab kesalahan dan perbaiki kode tersebut!  
2.2. Apabila Ortu memiliki data variabel umur = 25 dan jenis kelamin = P (untuk Perempuan), rekomendasikan constructor dengan parameter yang baru untuk ditambahkan dalam program!

JAWAB:

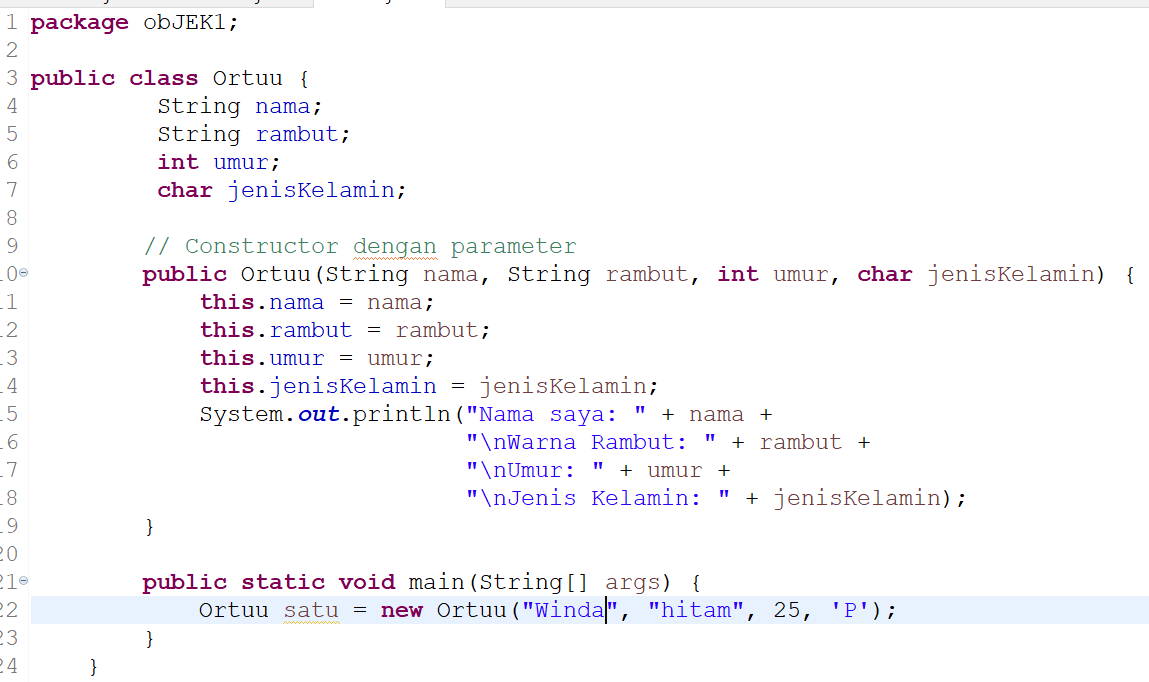
2.1  **Nama Constructor Salah:** Nama constructor seharusnya sama dengan nama kelas. Pada kode Anda, constructor dideklarasikan sebagai ortu, tetapi nama kelas adalah Ortu. Constructor harus menggunakan nama kelas yang tepat, yaitu Ortu.**Parameter Constructor Tidak Dideklarasikan:** Constructor yang Anda buat tidak memiliki parameter, tetapi di dalam metode main, Anda mencoba untuk membuat objek Ortu dengan parameter ("Putri", "hitam"). Anda perlu mendeklarasikan constructor dengan parameter yang sesuai. **Variabel nama dan rambut Tidak Dideklarasikan:** Dalam constructor, Anda mencoba untuk mencetak nama dan rambut, tetapi variabel ini tidak dideklarasikan dalam kelas. Jadi kode yang benar Sebuah gambar berisi teks, cuplikan layar, software, Font

Deskripsi dibuat secara otomatisadalah :

*Gambar 2.1 kode objek*

Penjelasan:

Kelas `Ortu` dalam kode Java yang Anda berikan memiliki atribut `nama` dan `rambut`, serta constructor yang menginisialisasi kedua atribut tersebut dengan parameter yang diberikan. Constructor juga mencetak nilai dari `nama` dan `rambut` ke konsol. Namun, dalam metode `main`, objek `Ortu` diinstansiasi dengan parameter `nama` yang kosong dan `rambut` yang diatur ke "hitam". Ini dapat menimbulkan masalah jika nama kosong dianggap tidak valid dalam konteks aplikasi Anda. Untuk memastikan data yang lebih valid dan mencegah potensi kesalahan, Anda bisa menambahkan validasi pada constructor untuk memeriksa apakah `nama` adalah string kosong atau null.

2.2

*Gambar 2.2 Kode Objek*

Penjelasan :

Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatisSama seperti penjrlasan gambar 2.1 kode objek digambar 2.2 hanya menambahkan variabel umur dan jeniskelamin mengunakan constructor dengan parameter yang baru.

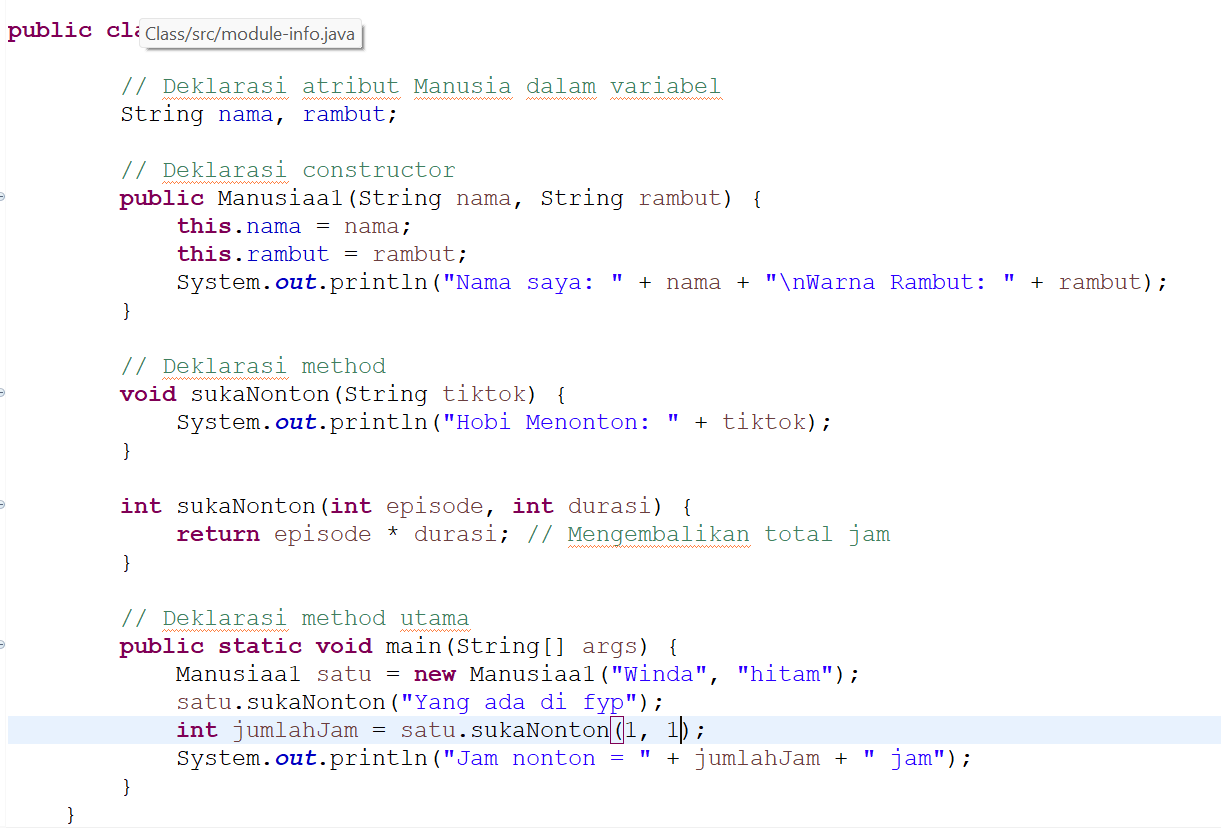
*Gambar 2.3 Output*

**UNIT 3 METHOD**

**Latihan 3:**  
3.1.  Evaluasi penyebab kesalahan dan perbaiki kode tersebut!  
3.2.  Ubahlah method dan constructor Contoh 3 sesuai dengan perilaku/ behavior anda  
3.3.  Berdasarkan Contoh 3 dan Latihan 3.2. simpulkan perbedaan:  
        a) constructor overloading dan overriding  
        b) method overloading, dan method overriding  
        c) method yang mengembalikan nilai dan method tidak mengembalikan nilai

JAWAB:

3.1 Kesalahan dalam kode tersebut adalah: **Nama Kelas**: Kelas dideklarasikan sebagai Manusia, tetapi constructor menggunakan Manusia1.**Deklarasi Method**: Metode sukaNonton tidak dideklarasikan dengan benar. Perlu menggunakan tanda kurung**Parameter dan Tipe Kembali**: Method sukaNonton harus dideklarasikan dengan parameter dan tipe kembalian yang sesuai.

3.2

*Gambar 3.1 kode Method*

Penjelasan :

Sebuah gambar berisi teks, Font, garis, cuplikan layar

Deskripsi dibuat secara otomatis Kode pada gambar 3.1 mendefinisikan kelas Manusiaa1, yang berfungsi untuk menggambarkan objek manusia dengan atribut seperti nama dan warna rambut. Constructor Manusiaa1 menerima dua parameter, yaitu nama dan rambut, dan menyimpannya dalam atribut yang sesuai. Ketika objek dibuat, constructor akan mencetak nama dan warna rambut ke konsol. Kelas ini juga memiliki dua metode bernama sukaNonton. Metode pertama menerima parameter berupa string, yang mencetak hobi menonton pengguna, sedangkan metode kedua menerima dua parameter integer (episode dan durasi) dan mengembalikan total jam menonton berdasarkan perkalian keduanya. Di dalam metode main, objek satu dibuat dari kelas Manusiaa1 dengan nama "Winda" dan warna rambut "hitam". Kemudian, metode sukaNonton dipanggil untuk mencetak hobi menonton Winda. Selanjutnya, metode kedua sukaNonton dipanggil dengan parameter 1 untuk episode dan 1 untuk durasi, yang mengembalikan nilai total jam menonton.

*Gambar 3.2 Output Method*

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

* **Overloading**: Membuat beberapa constructor dengan nama yang sama tetapi dengan parameter yang berbeda.
* **Overriding**: Tidak berlaku untuk constructor, karena constructor tidak dapat diwarisi.

b) **Method Overloading dan Overriding**:

* **Overloading**: Memiliki beberapa metode dengan nama yang sama tetapi parameter yang berbeda dalam kelas yang sama.
* **Overriding**: Metode di subclass yang memiliki nama dan parameter yang sama dengan metode di superclass.

c) **Method yang Mengembalikan Nilai dan Method yang Tidak Mengembalikan Nilai**:

* **Mengembalikan Nilai**: Memiliki tipe kembali (seperti int, String, dll.) dan menggunakan return untuk mengembalikan nilai.
* **Tidak Mengembalikan Nilai**: Tipe kembali adalah void, dan tidak menggunakan return untuk mengembalikan nilai.

**UNIT 4 EXTENDS**

**Latihan 4:**  
4.1. Evaluasi method yang dimiliki class Anak extends Ortu dengan method di class Ortu!  
       Apakah penulisan method ini sudah efisien?  
4.2. Setelah dirunning di JDoodle, catat waktu eksekusinya.   
       Rekomendasikan perbaikan penulisan kode method untuk dapat mengefisienkan waktu eksekusi!

Jawab :

4.1 Kelas Anak memiliki beberapa method yang mengoverride method di kelas Ortu. Metode sukaMenonton dalam kelas Anak ada dua versi: satu dengan parameter int dan String, dan satu lagi yang mengoverride method induk dengan parameter String. Penulisan ini bisa dianggap efisien jika tujuan adalah untuk memberikan perilaku yang berbeda. Namun, mengulang metode yang sama dengan nama yang sama dan parameter yang sama di kedua kelas tidak diperlukan. Jika hanya satu versi yang dibutuhkan, metode yang tidak perlu bisa dihapus untuk mengurangi kompleksitas.

4.2 Source code:

**package** obJEK1;

**public** **class** Ortuu {

// membuat kelas induk

**void** sukaMenonton(String a) {

System.***out***.println("Nonton " + a);

}

**void** sukaMembaca(String a) {

System.***out***.println("Suka Baca " + a);

}

**public** **static** **void** main(String [] args) {

System.***out***.println("Sifat Orang Tua :");

Ortuu objekO = **new** Ortuu(); // memanggil objek induk

objekO.sukaMenonton("Berita");

objekO.sukaMembaca("Koran");

System.***out***.println("\n Sifat Anak :");

Anak objekA = **new** Anak(); //memanggil objek anak

objekA.sukaMenonton(9, "Film Drakor");

objekA.sukaMembaca("Novel Beautiful Eyes");

} }

**class** Anak **extends** Ortuu {

**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 :");

Ortuu objekO = **new** Ortuu(); // memanggil objek induk

objekO.sukaMenonton("Berita"); // memanggil sifat spesifik induk

objekO.sukaMembaca("Koran");

System.***out***.println("\n Sifat Anak :");

Anak objekA = **new** Anak(); //memanggil objek anak

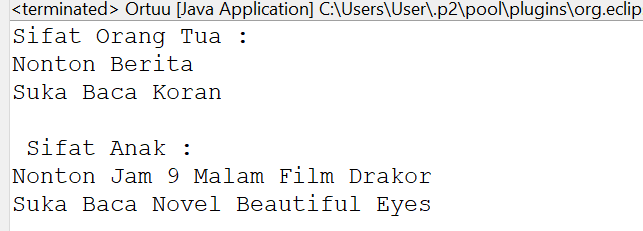
objekA.sukaMenonton(9, "Film Drakor");

objekA.sukaMembaca("Novel Beautiful Eyes");

}

}

Penjelasan :

 Kode di atas mendefinisikan dua kelas, `Ortuu` dan `Anak`, di mana `Anak` merupakan subclass dari `Ortuu`. Kelas `Ortuu` berisi dua metode: `sukaMenonton(String a)` dan `sukaMembaca(String a)`, yang masing-masing mencetak aktivitas menonton dan membaca berdasarkan input yang diberikan. Di dalam metode `main` kelas `Ortuu`, sebuah objek dari kelas ini dibuat dan metode dipanggil untuk menampilkan sifat orang tua. Selanjutnya, objek dari kelas `Anak` dibuat untuk memanggil metode `sukaMenonton` dengan parameter yang berbeda, menunjukkan adanya method overloading. Kelas `Anak` memiliki metode yang sama dengan `Ortuu`, yaitu `sukaMembaca`, tetapi juga memiliki dua versi dari metode `sukaMenonton`: satu dengan parameter `(int a, String b)` untuk mencetak informasi waktu menonton dan satu lagi yang mengoverride metode induk untuk menjaga kesamaan perilaku. Dalam metode `main` kelas `Anak`, objek `Ortuu` dan `Anak` dibuat untuk menampilkan sifat kedua kelas. Kode ini menggambarkan konsep pewarisan dan polymorphism, di mana subclass dapat menyesuaikan atau memperluas perilaku dari superclass. Namun, ada duplikasi dalam penulisan kode, terutama dalam mendefinisikan method yang sama di kedua kelas.

*Gambar 4.1 Output*

**KESIMPULAN**

Dalam rangkaian latihan ini, kita telah membahas berbagai konsep dalam pemrograman berbasis objek, termasuk kelas, objek, metode, dan pewarisan. Kita mengidentifikasi dan memperbaiki kesalahan dalam kode, serta menganalisis atribut dan perilaku yang relevan. Selain itu, kita mengevaluasi efisiensi metode dalam subclass dan superclass, membedakan antara overloading dan overriding, serta mengidentifikasi metode yang mengembalikan nilai versus yang tidak. Dari analisis ini, pentingnya penulisan kode yang bersih dan efisien serta penggunaan prinsip-prinsip pemrograman yang baik menjadi sangat jelas, untuk memastikan bahwa kode tidak hanya berfungsi dengan benar, tetapi juga mudah dipelihara dan diperluas di masa depan.