|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **Kelas Java** | **09 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

Analisa ciri-ciri umum Kelas Manusia yang dapat menjadi

a. atribut variabel, dan

b. perilaku/ behavior untuk method!

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. Atribut variabel dalam kelas Manusia merepresentasikan ciri-ciri atau sifat yang dimiliki oleh setiap objek (instansi) dari kelas tersebut. Dalam kode yang diberikan, terdapat dua atribut:

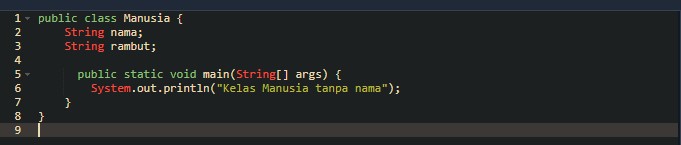
* nama: Merupakan atribut bertipe String yang digunakan untuk menyimpan nama seorang manusia. Ini adalah ciri khas yang membedakan setiap individu manusia.
* rambut: Juga bertipe String, atribut ini digunakan untuk menggambarkan warna atau jenis rambut seseorang. Ini adalah salah satu ciri fisik yang dapat bervariasi antar individu.

1. Method dalam kelas Manusia merepresentasikan tindakan atau perilaku yang dapat dilakukan oleh objek dari kelas tersebut. Dalam kode yang diberikan, hanya terdapat satu constructor tanpa parameter.

* Constructor adalah method khusus yang digunakan untuk menginisialisasi objek ketika dibuat. Constructor dalam contoh ini hanya mencetak pesan ke konsol.

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

1. Algoritma
2. Mulai: Menandai awal dari algoritma.
3. Deklarasi Kelas Manusia: Membuat sebuah kelas baru.
4. Deklarasi Atribut: Menambahkan atribut nama dan rambut ke kelas.
5. Deklarasi Konstruktor: Menambahkan konstruktor tanpa parameter ke kelas.
6. Cetak Pesan: Mencetak pesan "Kelas Manusia tanpa nama".
7. Selesai: Menandai akhir dari algoritma.
8. Kode program dan luaran



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Kode di atas hanya menampilkan pesan "Kelas Manusia tanpa nama" karena belum ada objek dari kelas Manusia yang dibuat dan dimanfaatkan.

**[No.1] Kesimpulan**

1. **Evaluasi**

Kode di atas merupakan contoh dasar dari sebuah kelas dalam Java. Meskipun fungsional, kode ini masih memiliki potensi untuk dikembangkan menjadi lebih kompleks dan bermanfaat. Dengan menambahkan objek, menggunakan atribut, dan menambahkan metode, kelas Manusia dapat menjadi representasi yang lebih baik dari konsep manusia dalam program Java.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **Kelas Java** | **09 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

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?

Contoh 2:

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.1] Analisis dan Argumentasi**

1. Atribut (Sifat)

Umur: Menambah atribut untuk menyimpan informasi tentang umur.

Jenis Kelamin: Menambahkan atribut untuk menyimpan jenis kelamin (misalnya, laki-laki atau perempuan).

Hobi: Menambahkan atribut untuk menyimpan daftar hobi atau minat.

Berat Badan: Menambah atribut untuk berat badan.

1. Constructor

dapat mengembangkan constructor di subclass untuk menerima atribut baru sambil memanggil constructor superclass untuk memastikan semua informasi yang relevan ditetapkan.

1. Perilaku Positif (Behavior)

Berbicara: Metode yang memungkinkan objek untuk mengeluarkan pernyataan atau berbicara.

Berolahraga: Metode untuk mengupdate tinggi badan atau berat badan berdasarkan aktivitas fisik.

Belajar: Metode untuk meningkatkan pengetahuan atau keterampilan, bisa menjadi representasi dari proses belajar.

Menampilkan Informasi: Metode untuk menampilkan semua informasi tentang objek secara ringkas.

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

Algoritma

1. Mulai Program
2. Deklarasi Kelas

* Nama kelas: Saya

1. Deklarasi Constructor

* Constructor Saya menerima tiga parameter:

String nama

String rambut

int tinggibadan

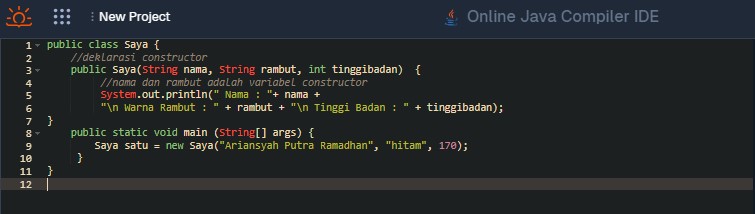
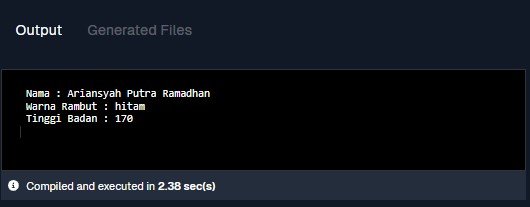
1. Dalam Constructor

* Tampilkan informasi menggunakan System.out.println:
* Tampilkan Nama
* Tampilkan Warna Rambut
* Tampilkan Tinggi Badan

1. Metode Main

* Buat objek satu dari kelas Saya dengan parameter:
* nama: "Ariansyah Putra Ramadhan"
* rambut: "hitam"
* tinggibadan: 170

1. Akhiri Program
2. Kode program dan luaran



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Secara keseluruhan, program ini berfungsi dengan baik dan menghasilkan output yang jelas menampilkan informasi tentang objek yang diciptakan.

**[No.1] Kesimpulan**

1. **Evaluasi**

Program mendemonstrasikan penggunaan konstruktor dengan baik, mengizinkan pengisian nilai untuk atribut objek saat dibuat. Output yang dihasilkan informatif dan mudah dibaca, menjelaskan setiap atribut dengan jelas.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **Kelas Java** | **09 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

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

Contoh 3: Salin dan tempel kode program berikut ke Eclipse atau JDoodle.

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.1] Analisis dan Argumentasi**

3.1 Analisis Perbedaan Deklarasi

Constructor:

Memiliki nama yang sama dengan nama kelas, Tidak memiliki tipe kembalian (void), Digunakan untuk menginisialisasi objek saat objek dibuat.

Method:

Memiliki nama yang berbeda dari nama kelas, Bisa memiliki tipe kembalian atau void, Digunakan untuk melakukan tindakan atau operasi tertentu pada objek.

Method Utama (main):

Titik masuk eksekusi program, Bertipe static, sehingga bisa dipanggil tanpa membuat objek dari kelas tersebut, Digunakan untuk menjalankan program secara keseluruhan.

3.2 Kapan Menggunakan Constructor dan Method

Constructor:

Digunakan saat membuat objek baru untuk memberikan nilai awal pada atribut-atribut objek tersebut. Memastikan objek dalam keadaan valid saat dibuat.

Method:

Digunakan untuk:

Melakukan tindakan atau operasi pada objek (misalnya, mengubah nilai atribut, melakukan perhitungan), Membungkus sekumpulan logika yang sering digunakan, Memisahkan tanggung jawab dalam suatu kelas.

3.3 Perbedaan

a) Constructor Overloading dan Overriding

Constructor Overloading:

- Membuat beberapa constructor dengan parameter yang berbeda dalam satu kelas.

- Digunakan untuk memberikan fleksibilitas dalam membuat objek dengan inisialisasi yang berbeda.

- Tidak ada constructor overriding karena constructor tidak bisa di-override.

b) Method Overloading dan Overriding

Method Overloading:

- Membuat beberapa method dengan nama yang sama tetapi parameter yang berbeda dalam satu kelas atau kelas turunan.

- Digunakan untuk melakukan tindakan yang sama tetapi dengan cara yang berbeda berdasarkan parameter yang diberikan.

- Method Overriding:

- Mendefinisikan ulang method yang sudah ada di kelas induk dalam kelas turunan.

- Digunakan untuk memberikan implementasi yang berbeda pada method yang sama di kelas turunan.

c) Method yang Mengembalikan Nilai dan Method Tidak Mengembalikan Nilai

Method yang Mengembalikan Nilai:

- Memiliki tipe kembalian yang spesifik (misalnya, int, String, boolean).

- Hasil pemrosesan method akan dikembalikan ke pemanggil method.

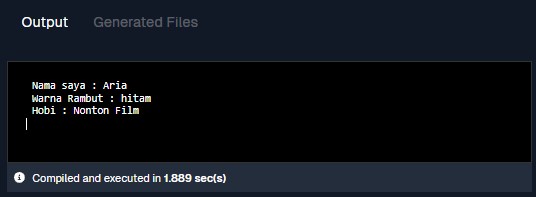
- Method Tidak Mengembalikan Nilai (void):

- Tidak memiliki tipe kembalian.

- Biasanya digunakan untuk melakukan tindakan sampingan seperti mencetak output ke layar, mengubah nilai variabel global, atau memanggil method lain

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

1. Algoritma
2. Mulai
3. Buat kelas Manusia
4. Buat objek Aria
5. Cetak info Aria
6. Selesai
7. Kode program dan luaran



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Program mencetak nama dan warna rambut dari objek satu yang telah dibuat. Ini menunjukkan bahwa konstruktor berfungsi dengan baik dalam menginisialisasi objek. Program juga mencetak hobi menonton dari objek satu. Ini menunjukkan bahwa metode sukaNonton berhasil dipanggil dan mencetak pesan yang sesuai dengan argumen yang diberikan.

**[No.1] Kesimpulan**

1. **Evaluasi**

Kode program di atas merupakan titik awal yang baik untuk belajar pemrograman berorientasi objek. Namun, masih banyak potensi untuk dikembangkan menjadi program yang lebih kompleks dan bermanfaat.

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariansyah Putra Ramadhan**  **G1F024080** | **Kelas Java** | **09 Oktober 2024** |

**[No. 1] Identifikasi Masalah:**

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!

Contoh 4: Salin dan tempel kode program berikut ke JDoodle. Kemudian catat waktu eksekusinya.

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  
    }  
}

Compiled and executed in 1.955 sec(s)

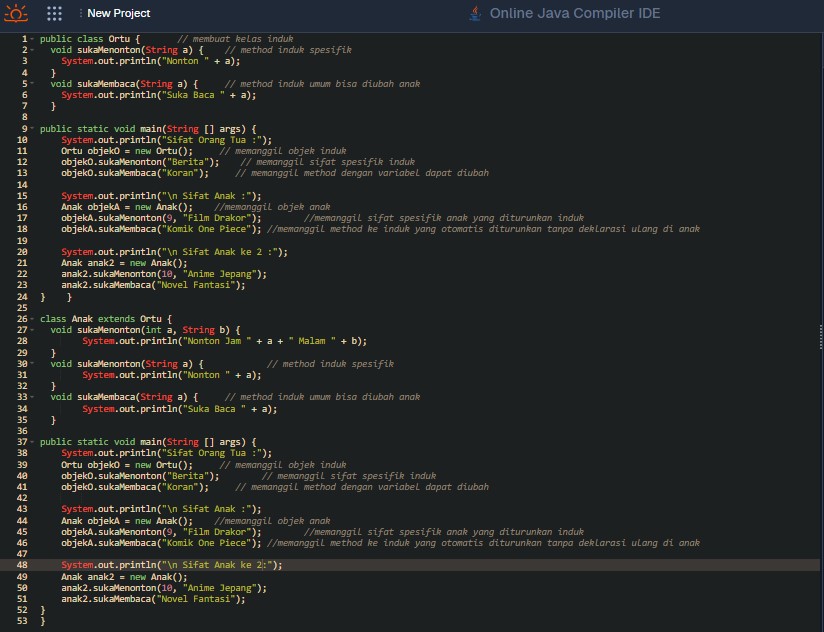
**[No.1] Analisis dan Argumentasi**

4.1. Bandingkan method yang dimiliki class Anak extends Ortu dengan method di class Ortu!

Metode sukaMenonton(String a): Metode ini di-override di kelas Anak. Ada dua versi: satu dengan parameter String dan satu lagi dengan parameter int dan String.

Metode sukaMembaca(String a): Metode ini di-override di kelas Anak, namun dengan implementasi yang sama.

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

1. Algoritma
2. Buat model "Orang Tua" dan "Anak".
3. Beri kemampuan pada "Orang Tua" untuk menonton dan membaca.
4. Buat "Anak" mewarisi kemampuan dari "Orang Tua".
5. Tambahkan kemampuan khusus pada "Anak" untuk menyebutkan waktu menonton.
6. Buat contoh "Orang Tua" dan "Anak".
7. Tampilkan apa yang dilakukan oleh "Orang Tua" dan "Anak" (misalnya, menonton apa, membaca apa, dan jam berapa menonton).
8. Kode program dan luaran 



1. Screenshot/ Capture potongan kode dan hasil luaran

Beri komentar pada kode yang di Screenshot

1. Analisa luaran yang dihasilkan   
   Program ini akan menghasilkan output ke konsol, menampilkan serangkaian kalimat yang menggambarkan aktivitas yang dilakukan oleh objek Ortu (orang tua) dan objek Anak (anak).

**[No.1] Kesimpulan**

1. **Evaluasi**

Program terbagi menjadi kelas-kelas yang jelas (Ortu dan Anak), meningkatkan keterbacaan dan kemudahan perawatan kode. Output program juga mudah dipahami dan memberikan gambaran yang jelas tentang perilaku objek-objek yang dibuat. Program ini hanya mensimulasikan aktivitas menonton dan membaca. Untuk menjadi lebih kompleks, bisa ditambahkan aktivitas lainnya seperti makan, tidur, bekerja, dan sebagainya.