Nama & NPM	Topik:	Tanggal:
Sindi Putri Utami **G1F024053**	**Kelas Java**	**12 September 2024**

[1] Identifikasi Masalah:

1.1. Analisa ciri-ciri umum Kelas Manusia yang dapat menjadi
a. atribut variabel, dan
b. perilaku/ behavior untuk method!

[1] Analisis dan Argumentasi

Pada soal masih ada pesan kesalahan, saya memperbaiki program supaya menghasilkan output. Saya menambahkan parameter dalam constructor. Disini ada dua parameter, yaitu parameter nama dan rambut. Ini akan mencetak informasi tentang nama dan rambut ketika objek dari kelas Manusia dibuat. Disini saya menambahkan method suka nonton dan method main.

a. Atribut variable

Atribut variabel adalah elemen-elemen yang menyimpan data dalam sebuah kelas. Dalam kelas Manusia, atribut variabel yang dideklarasikan pada program ini adalah:

- String nama: Menyimpan nama dari objek Manusia.
  String adalah tipe data yang digunakan untuk menyimpan urutan karakter. **nama** adalah variabel yang akan menyimpan nama seseorang (disini saya menyimpan “Utami”).

- String rambut: Menyimpan warna rambut dari objek Manusia.

  Sama dengan nama, rambut juga bertipe String. Ini akan menyimpan warna rambut dari objek Manusia (disini saya menyimpan "hitam").

b. perilaku/ behavior untuk method

Method adalah blok kode yang melakukan operasi tertentu dan biasanya terkait dengan objek atau kelas di mana metode tersebut didefinisikan. Method dapat mengakses dan memodifikasi atribut variabel objek dan juga dapat menerima parameter dan mengembalikan nilai.
Dalam kelas Manusia, terdapat satu metode yang didefinisikan yaitu:

- void sukaNonton(String film)
  void menunjukkan bahwa method ini tidak mengembalikan nilai apapun.
  sukaNonton adalah nama method. Nama ini digunakan untuk memanggil method tersebut dari objek Manusia. Metode sukaNonton digunakan untuk menunjukkan aktivitas atau hobi yang disukai oleh objek Manusia. Disini, film adalah jenis aktivitas atau hobi yang dipilih dan dicetak oleh method ini.
  Method ini menerima satu parameter bertipe String yang dinamai film. Parameter ini digunakan untuk menentukan film atau aktivitas yang disukai oleh objek Manusia.

Penjelasan kode:

- Manusia satu = new Manusia("Utami", "hitam");
  Ini membuat objek baru dari kelas Manusia dengan nama satu dan memberikan nilai "Utami" untuk nama dan warna rambut "hitam".
- satu.sukaNonton("Nonton Drakor");
  Ini memanggil metode sukaNonton pada objek satu dan mengirimkan nilai "Nonton Drakor". Metode ini kemudian mencetak "Hobi : Nonton Drakor" ke layar.

[1] Penyusunan Algoritma dan Kode Program

1) Algoritma:

1. Mulai
2. Deklarasi kelas
3. Deklarasi variable
4. Deklarasi Konstruktor
5. Deklarasi method
6. Metode main
7. Selesai

2) Kode program dan luaran

```java
public class Manusia {  // deklarasi kelas
    // deklarasi variabel
    String nama ;
    String rambut ;

     // deklarasi constructor
    public Manusia(String nama, String rambut) {
        System.out.println("nama : " + nama +
        "\nwarna rambut : " + rambut);

    }
    void sukaNonton(String film) {
        System.out.println("Hobi : " + film);
    }
    public static  void main(String []args) {
        Manusia satu= new Manusia ("Utami", "hitam");
        satu.sukaNonton("Nonton Drakor");
    }
}
```

```
java -cp /tmp/YYFuVXYaOU/Manusia
nama : Utami
warna rambut : hitam
Hobi : Nonton Drakor

=== Code Execution Successful ===
```

Luaran sudah sesuai dengan program yang disusun.

[1] Kesimpulan

Program tersebut mengimplementasikan kelas Manusia dengan atribut nama dan rambut, serta method sukaNonton. Metode sukaNonton mencetak hobi atau aktivitas yang disukai. Program membuat objek Manusia, menetapkan nilai nama dan warna rambut, lalu memanggil method sukaNonton untuk menampilkan informasi hobi ke layar.

[2] 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?

[2]Analisis dan Argumentasi

2.1. Pada program saya menambahkan biodata saya seperti, nama, jenis kelamin, warna rambut, warna kulit, tinggi badan, berat badan, dan hobi.
Saya menambahkan nama, jenis kelamin, warna rambut, warna kulit, tinggi badan, dan berat badan di dalam variabel constructor. Saya memasukkan hobi ke dalam method bernama **sukaNonton** yang memiliki parameter String film.

2.2. Jika kelas Ortu akan memiliki keturunan, ini adalah analisis mengenai sifat (atribut), constructor, dan perilaku positif (behavior) yang bisa diturunkan:

1. Sifat(atribut)
   Atribut yang bisa diturunkan dari kelas Ortu antara lain:

- Nama: Menggambarkan identitas individu.
  Misalnya, sinta
- Jenis Kelamin**:** Menentukan gender (laki-laki atau perempuan).
  Misalnya, jenis kelamin perempuan
- Warna Rambut**:** Mewakili karakteristik fisik yang bisa diwariskan.

Misalnya, warna rambut hitam

- Warna Kulit**:** Sifat fisik yang mungkin diturunkan dari orang tua.
  Misalnya, warna kulit cokelat
- Tinggi Badan**:** Karakteristik fisik yang dapat diwariskan.
  Misalnya, tinggi 150
- Berat Badan**:** Faktor yang dapat dipengaruhi oleh genetik dan gaya hidup.
  Misalnya, berat badan 30

2. Constructor

- Constructor Parameter: Keturunan dapat memiliki constructor yang menerima parameter untuk inisialisasi atribut yang diwarisi, seperti nama, jenisKelamin, rambut, dan lain-lain.
- Constructor Default: Keturunan juga bisa memiliki constructor tanpa parameter yang menginisialisasi atribut dengan nilai default.

3. perilaku positif (behavior)
   Perilaku positif yang bisa diwariskan atau ditambahkan pada subclass:
   Hobi: Misalnya, nonton anime
   .

[2] Penyusunan Algoritma dan Kode Program

1) Algoritma
   1. Mulai
   2. Deklarasi Kelas Ortu
   3. Deklarasi Method sukaNonton
   4. Deklarasi Method main
   5. Selesai

2) Kode program luaran

```java
public class Ortu {
    // Deklarasi constructor ortu
    public Ortu(String nama, String jenisKelamin, String rambut,
        String kulit, int tinggiBadan, int beratBadan)  {
   // nama dan rambut adalah variabel constructor
    System.out.println(" Nama           : " + nama + "\n Jenis
        Kelamin : " + jenisKelamin + "\n Warna Rambut  : " + rambut
        + "\n Warna Kulit   : " + kulit + "\n Tinggi Badan  : " +
        tinggiBadan + "\n Berat Badan   : " + beratBadan);
    }
   // Method sukaNonton
    void sukaNonton(String film) {
        System.out.println(" Hobi           : " + film);
    }
    // Method main
    public static void main (String[] args) {
        Ortu satu = new Ortu("Sindi", "Perempuan", "Hitam", "coklat"
            , 163, 43 );
        satu.sukaNonton("Nonton Drakor");

    }
}
```

```
java -cp /tmp/OkC8kEWyU7/Ortu
 Nama           : Sindi
 Jenis Kelamin : Perempuan
 Warna Rambut  : Hitam
 Warna Kulit   : coklat
 Tinggi Badan  : 163
 Berat Badan   : 43
 Hobi           : Nonton Drakor

=== Code Execution Successful ===
```

Luaran sudah sesuai dengan program yang disusun.

[2] Kesimpulan

Kelas Ortu digunakan untuk menyimpan dan menampilkan informasi tentang orang tua, seperti nama, jenis kelamin, warna rambut, warna kulit, tinggi badan, dan berat badan. Kelas ini memiliki constructor yang menerima parameter untuk menginisialisasi atribut dan mencetak informasi saat objek dibuat. Selain itu, terdapat method sukaNonton yang mencetak hobi menonton film berdasarkan parameter yang diberikan. Dalam method main, objek Ortu dibuat dan method sukaNonton dipanggil untuk menunjukkan hobi menonton.

Program ini mendemonstrasikan cara kerja constructor dan method dalam Java untuk membuat dan mengelola objek dengan cara yang terstruktur.

[3] 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

[3] Analisis dan Argumentasi

3.1. perbedaan deklarasi constructor, method, dan method utama

1. Deklarasi Konstruktor:
   - Constructor adalah metode khusus yang dipanggil secara otomatis ketika objek dari kelas dibuat. Tujuan utama constructor adalah untuk memberikan nilai awal pada atribut objek..
   - Nama constructor harus sama dengan nama kelas. Di sini, **Manusia1** adalah nama constructor, dan harus sama dengan nama kelas (namanya salah karena disini nama constructor adalah Manusia1 sedangkan nama kelas adalah Manusia, seharusnya constructor harus bernama Manusia).
   - Constructor tidak memiliki tipe pengembalian. Constructor secara otomatis mengembalikan objek dari kelas yang bersangkutan.

2. Deklarasi Method
   - Method adalah blok kode yang melakukan suatu tugas atau operasi tertentu. Method dipanggil untuk menjalankan fungsinya, yang bisa melibatkan pengolahan data atau memberikan hasil tertentu. Method bisa memiliki tipe pengembalian yang menunjukkan jenis data yang dikembalikan atau void jika tidak mengembalikan nilai.
   - Nama method bisa berbeda dari nama kelas dan harus mengikuti aturan penamaan method (biasanya camelCase), Ini berarti nama method dimulai dengan huruf kecil dan setiap kata berikutnya dimulai dengan huruf kapital tanpa spasi.
   - Method bisa memiliki tipe pengembalian (misalnya int, String, double) atau tidak mengembalikan nilai (void).
   - Disini method ini berfungsi untuk mencetak hobi menonton yang diinginkan. Method ini dapat dipanggil setelah objek dibuat untuk menjalankan aksi tertentu (mencetak hobi).

3. Deklarasi method utama
   - Method utama atau buasa disebut method main adalah metode khusus di Java yang memberi tahu di mana harus mulai menjalankan program. Ketika menjalankan program Java, komputer akan mencari method main untuk memulai.
   - Nama method ini adalah main, yang merupakan cara umum dalam Java untuk method yang memulai program.
   - Method main selalu memiliki tipe pengembalian void dan menerima parameter berupa array String[].
   - Di sini, objek Manusia diciptakan dengan memanggil constructor, dan kemudian method sukaNonton dipanggil untuk mencetak hobi menonton.

3.2. Kapan perlu menggunakan constructor dan method

- Constructor
  Constructor digunakan ketika ingin menetapkan nilai awal untuk atribut objek ketika objek baru dibuat. Constructor sering digunakan untuk mengatur kondisi awal objek, sehingga objek mulai dalam keadaan yang diinginkan.

Disini digunakan untuk menetapkan nilai awal untuk atribut objek ketika objek Manusia dibuat (misalnya, nama dan rambut).

- Method

  Method digunakan untuk melakukan operasi atau tindakan pada objek, seperti memproses data, menjalankan logika bisnis, atau mengubah status objek. Method bisa dipanggil kapan saja setelah objek dibuat, sesuai kebutuhan.

  Disini method sukaNonton digunakan untuk mencetak informasi tambahan tentang hobi menonton.

3.3. Uraikan perbedaan constructor overloading dan overriding, method overloading, dan method overriding, method yang mengembalikan nilai dan method tidak mengembalikan nilai

a) constructor overloading dan overriding

- constructor overloading

  Constructor overloading adalah kemampuan untuk memiliki lebih dari satu constructor dalam satu kelas, dengan nama yang sama tetapi parameter yang berbeda (jumlah atau tipe parameter).

  Di codingan ini, tidak ada contoh constructor overloading karena hanya ada satu constructor. Jika menambahkan lebih banyak constructor dengan parameter yang berbeda, itu akan menjadi contoh overloading.

- Constructor overriding

  Constructor overriding tidak ada dalam Java. Constructor tidak dapat dioverride karena tidak ada pewarisan untuk constructor seperti halnya method.

  Constructor Overriding Tidak berlaku dalam codingan ini, karena constructor tidak dapat dioverride seperti method.

b) method overloading, dan method overriding

- Method Overloading

  Terjadi ketika dua atau lebih method dalam satu kelas memiliki nama yang sama tetapi dengan parameter yang berbeda (tipe, jumlah, atau urutan).

  Di codingan ini, Jika kita menambahkan method sukaNonton dengan parameter berbeda (misalnya, satu dengan parameter String dan satu dengan parameter int), itu akan menjadi contoh method overloading.

  Method overriding terjadi ketika kelas turunan menyediakan implementasi khusus untuk method yang sudah didefinisikan dikelas induk. Method yang diganti harus memiliki nama, tipe pengembalian, dan parameter yang sama dengan method di kelas induk.

- Method overriding

  Method overriding memungkinkan untuk memodifikasi cara kerja method dari kelas induk agar sesuai dengan kebutuhan kelas yang lebih spesifik. Ini membuat kode kita lebih fleksibel dan mudah diatur, sehingga bisa digunakan untuk berbagai objek dengan cara yang konsisten.

  Jika kelas Manusia memiliki turunan (subclass), dan turunan tersebut mendefinisikan method sukaNonton dengan cara yang berbeda, itu akan menjadi contoh method overriding.

c) method yang mengembalikan nilai dan method tidak mengembalikan nilai

- Method yang Mengembalikan Nilai

  Method ini mengembalikan sebuah nilai dari tipe data tertentu setelah menyelesaikan eksekusi. Method ini harus memiliki tipe pengembalian dan menggunakan return untuk mengembalikan nilai.

Jika mendefinisikan method yang mengembalikan String (misalnya, String getNama() { return nama; }), itu adalah method yang mengembalikan nilai.

- Method Tidak Mengembalikan Nilai
  Method ini memiliki tipe pengembalian void, yang berarti method ini tidak mengembalikan nilai. Method sukaNonton dalam codingan ini adalah contoh method yang tidak mengembalikan nilai, karena dideklarasikan dengan void.

[3] Penyusunan Algoritma dan Kode Program

1) Algoritma
   1.Mulai
   2.Deklarasi kelas
   3.Deklarasi atribut
   4. Deklarasi Constructor
   5. Deklarasi Method
   6. Deklarasi Method Utama (main)
   7.selesai

2) Kode program dan luaran

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

    //deklarasi constructor
    public Manusia(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");
    }
}
```

```
java -cp /tmp/qPe9IK6Oq3/Manusia
 Nama saya : Putri
 Warna Rambut : hitam
 Hobi Menonton : Drakor

=== Code Execution Successful ===
```

Luaran sudah sesuai dengan program yang disusun.

[3] Kesimpulan

Kode Java mendefinisikan kelas Manusia dengan atribut nama dan rambut. Constructor Manusia (bukan Manusia1) digunakan untuk menginisialisasi atribut dan mencetak informasi nama serta warna rambut saat objek dibuat. Method sukaNonton mencetak hobi menonton film yang diberikan sebagai parameter. Dalam method main, objek Manusia dibuat dengan nama "Putri" dan rambut "hitam", dan method sukaNonton dipanggil untuk menampilkan hobi menonton "Drakor". Kode yang benar menghasilkan output:

Nama saya : Putri
Warna Rambut : hitam Hobi
Menonton : Drakor

[4] 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!

[4] Analisis dan Argumentasi

4.1. Perbandingan method kelas anak dan kelas ortu

1. Kelas anak

- Method sukaMenonton(int a, String b)
  Method ini merupakan overload dari method sukaMenonton di kelas Ini mencetak aktivitas menonton dengan dua parameter (jam dan nama film).
  Contoh: objekA.sukaMenonton(9, "Film Drakor");
- Method sukaMenonton(String a)
  Method ini mewarisi method dari kelas Ortu, tetapi dapat diubah (overriding) untuk memberikan implementasi yang sama.
- Method sukaMembaca(String a)
- Method ini juga mewarisi method dari kelas Ortu dan dapat diubah. Implementasinya dapat berbeda jika ada perubahan di dalamnya.
  Contoh: objekA.sukaMembaca("Komik One Piece");

2. Kelas ortu
- Method suka menonton(Sting a)
  Method ini mencetak aktivitas menonton dengan parameter string.
  Contoh: objekO.sukaMenonton("Berita");
- Method suka membaca(String a)
  Method ini mencetak aktivitas membaca dengan parameter string.
  Contoh: objekO.sukaMembaca("Koran");

4.2. Dalam kelas anak saya menambahkan method sukaBermain(String permanan).
Method ini mencetak aktivitas bermain dengan parameter string.
Contoh: objekA.sukaBermain ("Sepak Bola");

[4] Penyusunan Algoritma dan Kode Program

1) Algoritma
1. Mulai
2. Definisikan Kelas Ortu
3. Deklarasikan Method main dalam Kelas Ortu
4. Definisikan Kelas Anak yang Mengextends Ortu
5. Selesai

2) Kode program dan luaran

```
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
        objekA.sukaBermain("Sepak Bola");
    }    }

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);
    }

    void sukaBermain(String permainan) {
        System.out.println("Suka Bermain " + permainan);
    }

public static void main(String [] args) {
        System.out.println("Sifat Orang Tua :");
```

```
java -cp /tmp/OeTpnbS7SI/Ortu
Sifat Orang Tua :
Nonton Berita
Suka Baca Koran

 Sifat Anak :
Nonton Jam 9 Malam Film Drakor
Suka Baca Komik One Piece
Suka Bermain Sepak Bola

=== Code Execution Successful ===
```

```
39      Ortu objekO = new Ortu();      // memanggil objek induk
40      objekO.sukaMenonton("Berita");         // memanggil sifat
            spesifik induk
41      objekO.sukaMembaca("Koran");      // memanggil method dengan
            variabel dapat diubah
42
43      System.out.println("\n Sifat Anak :");
44      Anak objekA = new Anak();     //memanggil objek anak
45      objekA.sukaMenonton(9, "Film Drakor");         //memanggil sifat
            spesifik anak yang diturunkan induk
46      objekA.sukaMembaca("Komik One Piece"); //memanggil method ke
            induk yang otomatis diturunkan tanpa deklarasi ulang di anak
47      objekA.sukaBermain("Sepak Bola");
48      }
49  }
```

Luaran sudah sesuai dengan program yang disusun.

[4] Kesimpulan

Program ini menunjukkan konsep pewarisan dalam pemrograman berorientasi objek dengan kelas ortu sebagai induk dan kelas anak sebagai turunan. Kelas ortu memiliki method untuk mengekspresikan aktivitas, sedangkan kelas Anak mewarisi method tersebut dan menambahkan method baru, serta melakukan overriding. Program ini menggambarkan bagaimana objek Anak dapat memanggil method dari kelas induk dan mengubah perilaku tertentu. Secara keseluruhan, kode ini mengilustrasikan bagaimana pewarisan memungkinkan pengembangan fungsionalitas tanpa perlu menduplikasi kode.

Refleksi

Minggu ini, saya belajar tentang pemrograman berorientasi objek (OOP) dalam Java, fokus pada kelas, objek, dan pewarisan. Tantangan utama adalah memahami penggunaan yang tepat antara constructor dan method, serta membedakan antara method overloading dan overriding.