Nama & NPM	Topik:	Tanggal:
Farrel Alvaro Alinskie .M G1F024024	**Unit 1 kelas (class)**	**1 oktober 2024**

[No. 1] Identifikasi Masalah:

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

   //deklarasi constructor
   public Manusia1 (String nama) {
         System.out.println(" Nama saya : "+ nama +
         "\n Warna Rambut : " + rambut);
   }

   //deklarasi method utama
   public static void main( String[] args) {
         Manusia1 satu = new Manusia1("Putri", "hitam");
   }
 }
```

Luaran 1:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
   The constructor Manusia1(String, String) is undefined
   at Manusia1.main(Manusia1.java:13)
```

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!

[No.1] Jawaban

1.1 program yang telah di perbaiki

```java
public class Manusia {

    String nama, rambut;


    public Manusia(String nama, String rambut) {
        this.nama = nama;
        this.rambut = rambut;
        System.out.println("Nama saya : " + this.nama +
                           "\nWarna Rambut : " + this.rambut);
    }

    public static void main(String[] args) {
        Manusia satu = new Manusia("Putri", "hitam");
    }
}

```

output:

```
Nama saya : Putri
Warna Rambut : hitam

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

1.2 **a. Atribut Variabel**

Atribut-atribut yang dapat ditambahkan dalam kelas **Manusia** bisa mencakup berbagai informasi tentang manusia. Berdasarkan konteks, berikut adalah contoh atribut yang dapat digunakan:

- String nama: Nama manusia.
- String rambut: Warna rambut manusia.
- int umur: Umur manusia.
- String jenisKelamin: Jenis kelamin manusia (laki-laki/perempuan).
- String alamat: Alamat tempat tinggal manusia.

b. Perilaku/Behavior

Perilaku atau method yang dapat dimiliki oleh kelas **Manusia** bisa mencakup tindakan atau sifat manusia tersebut. Berikut adalah beberapa contoh perilaku:

- void berkenalan(): Metode yang mencetak pesan perkenalan.

- void tidur(): Metode yang menunjukkan bahwa manusia tersebut tidur.
- void berbicara(String pesan): Metode yang menunjukkan manusia dapat berbicara dengan menyampaikan pesan.
- void beraktivitas(): Metode yang menggambarkan kegiatan yang dilakukan oleh manusia.

Nama & NPM	Topik:	Tanggal:
Farrel Alvaro Alinskie .M G1F024024	**Unit 2 objek**	**1 oktober 2024**

[No. 2] Identifikasi Masalah:
Objek adalah bentuk keturunan dari kelas, sehingga otomatis memiliki atribut dan method dari kelas.

Deklarasi objek: NamaKelas NamaObjek = new NamaKelas(parameter);

Contoh 2: Salin dan tempel kode program berikut ke Eclipse atau JDoodle.

```
public class Ortu {
   //deklarasi constructor  (variabel constructor)
   public ortu   {
         //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:
```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
   The constructor Ortu(String, String) is undefined

   at Ortu.main(Ortu.java:9)
```

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

[No.2]jawaban

2.1Penyebab Kesalahan: Kesalahan terjadi karena konstruktor Ortu tidak didefinisikan dengan benar. Dalam kode, tidak ada konstruktor yang menerima parameter String untuk nama dan rambut. Selain itu, ada kesalahan dalam penulisan nama konstruktor yang seharusnya memiliki huruf besar untuk nama kelas (Ortu), bukan huruf kecil (ortu).

Perbaikan Kode: Berikut adalah perbaikan yang perlu dilakukan:

- Definisikan konstruktor Ortu dengan parameter untuk nama dan rambut.
- Atur nilai variabel nama dan rambut di dalam konstruktor.

2.2

```java
public class Ortu {

    String nama, rambut;
    int umur;
    char jenisKelamin;

    public Ortu(String nama, String rambut, int umur, char
        jenisKelamin) {
        this.nama = nama;
        this.rambut = rambut;
        this.umur = umur;
        this.jenisKelamin = jenisKelamin;
        System.out.println("Nama saya : " + this.nama +
                           "\nWarna Rambut : " + this.rambut +
                           "\nUmur : " + this.umur +
                           "\nJenis Kelamin : " + this.jenisKelamin);

    }

    public static void main(String[] args) {
        Ortu satu = new Ortu("Putri", "hitam", 25, 'P');
    }
}
```

Dengan perbaikan dan penambahan di atas, kelas **Ortu** sekarang memiliki konstruktor yang benar dan dapat menerima atribut tambahan umur dan jenisKelamin. Kode ini sekarang siap untuk dijalankan tanpa kesalahan kompilasi, dan akan mencetak semua informasi yang diberikan saat objek Ortu dibuat.

Nama & NPM	Topik:	Tanggal:
Farrel Alvaro Alinskie .M G1F024024	**Unit 3 method**	**1 oktober 2024**

[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  {
      System.out.println(" Hobi Menonton : " + film);
   }

   int sukaNonton {
      episode*durasi;
   }

   //deklarasi method utama
   public static void main( String[] args) {
         Manusia satu = new Manusia("Putri", "hitam");
         satu.sukaNonton("Drakor");
         int jumlahJam = satu.sukaNonton(2, 2);
         System.out.println("Jam nonton = " +jumlahJam + " jam");
   }
}
```

Luaran 3:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
   The method sukaNonton(String) is undefined for the type Manusia1
   The method sukaNonton(int, int) is undefined for the type Manusia1
   at Manusia1.main(Manusia1.java:23)
```

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

[No.3]jawaban

3.1 Penyebab Kesalahan:

1. Nama Kelas dan Konstruktor yang Tidak Cocok: Anda mendeklarasikan konstruktor public Manusia1, tetapi nama kelasnya adalah Manusia. Harusnya nama konstruktor mengikuti nama kelas.
2. Metode sukaNonton Tidak Dideklarasikan dengan Benar: Anda tidak mendefinisikan metode dengan benar. Metode harus memiliki tanda kurung () setelah nama metode, bahkan jika tidak ada parameter.
3. Metode sukaNonton Kedua Tidak Didefinisikan dengan Tipe Kembali yang Benar: Jika ingin mengembalikan nilai, harus ada pernyataan return.
4. Variabel film, episode, dan durasi Tidak Dideklarasikan: Variabel ini digunakan tetapi tidak didefinisikan sebelumnya.

Program yang benar

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

   // Deklarasi konstruktor
   public Manusia(String nama, String rambut) {
      this.nama = nama; // Menetapkan nilai nama
      this.rambut = rambut; // Menetapkan nilai rambut
      System.out.println("Nama saya : " + nama +
                  "\nWarna Rambut : " + rambut);
   }

   // Metode untuk menampilkan hobi menonton film
   void sukaNonton(String film) {
      System.out.println("Hobi Menonton: " + film);
   }

   // Metode untuk menghitung total jam menonton
   int sukaNonton(int episode, int durasi) {
      return episode * durasi; // Mengembalikan total jam
   }

   // Deklarasi metode utama
   public static void main(String[] args) {
      Manusia satu = new Manusia("Putri", "hitam");
      satu.sukaNonton("Drakor");
      int jumlahJam = satu.sukaNonton(2, 2);
      System.out.println("Jam nonton = " + jumlahJam + " jam");
   }
}
```

3.2
```
public class Manusia {
   // Deklarasi atribut Manusia
   String nama, rambut;
   int umur; // Menambahkan atribut umur

   // Deklarasi konstruktor
   public Manusia(String nama, String rambut, int umur) {
      this.nama = nama;
      this.rambut = rambut;
      this.umur = umur;
      System.out.println("Nama saya : " + nama +
                  "\nWarna Rambut : " + rambut +
                  "\nUmur : " + umur + " tahun");
   }

   // Metode untuk menampilkan hobi
   void sukaNonton(String film) {
      System.out.println("Hobi Menonton: " + film);
   }

   // Metode untuk menghitung total jam menonton
   int sukaNonton(int episode, int durasi) {
      return episode * durasi; // Mengembalikan total jam
   }

   public static void main(String[] args) {
      Manusia satu = new Manusia("Putri", "hitam", 25);
      satu.sukaNonton("Drakor");
      int jumlahJam = satu.sukaNonton(2, 2);
      System.out.println("Jam nonton = " + jumlahJam + " jam");
   }
}
```

3.3

a) Constructor Overloading dan Overriding

- **Constructor Overloading**: Ini terjadi ketika ada beberapa konstruktor dengan nama yang sama tetapi dengan parameter yang berbeda dalam satu kelas. Contoh: memiliki dua konstruktor Manusia dengan parameter yang berbeda (satu dengan dua parameter, satu dengan tiga parameter).
- **Constructor Overriding**: Ini tidak terjadi dalam Java, karena konstruktor tidak dapat di-override seperti metode. Namun, kita dapat menggunakan pewarisan untuk membuat konstruktor di kelas turunan yang berbeda dari kelas induk.

b) Method Overloading dan Method Overriding

- **Method Overloading**: Ini terjadi ketika ada beberapa metode dalam satu kelas dengan nama yang sama tetapi dengan parameter yang berbeda. Misalnya, dua metode sukaNonton dengan parameter berbeda (satu menerima String, yang lain menerima dua int).
- **Method Overriding**: Ini terjadi ketika metode di kelas turunan memiliki implementasi yang berbeda dari metode di kelas induk. Contoh, jika ada metode sukaNonton di kelas induk

Manusia dan kita mendefinisikan kembali metode tersebut di kelas turunan dengan implementasi yang berbeda.

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

- **Method yang Mengembalikan Nilai**: Metode ini menggunakan tipe pengembalian dan memiliki pernyataan return. Contohnya adalah int sukaNonton(int episode, int durasi), yang mengembalikan total jam menonton.
- **Method yang Tidak Mengembalikan Nilai**: Metode ini dideklarasikan dengan tipe void dan tidak memiliki pernyataan return. Contohnya adalah void sukaNonton(String film), yang hanya mencetak hobi menonton tetapi tidak mengembalikan nilai.

Nama & NPM	Topik:	Tanggal:
Farrel Alvaro Alinskie .M G1F024024	**Unit 4 extend**	**1 oktober 2024**

[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
} }
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. 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!

[No.4]jawaban
4.1**Evaluasi Metode:**

1. **Overloading dan Overriding**: Kelas Anak memiliki dua metode sukaMenonton, satu yang menerima parameter int dan String (overloading), dan satu lagi yang menimpa metode dari kelas induk Ortu yang juga menerima String (overriding). Ini adalah praktik yang baik dalam OOP, tetapi penggunaan nama yang sama untuk dua metode yang berbeda bisa membingungkan jika tidak didokumentasikan dengan baik.
2. **Pengulangan Kode**: Metode sukaMembaca di kelas Anak menimpa metode yang sama dari kelas Ortu. Meskipun ini sah, tidak ada alasan untuk mendeklarasikannya ulang jika tidak ada perubahan dalam implementasinya. Ini menciptakan pengulangan kode yang tidak perlu.
3. **Efisiensi Penulisan Kode**: Kode dalam metode sukaMenonton(String a) di kelas Anak sama dengan yang ada di kelas Ortu, yang menunjukkan bahwa penulisan ini bisa lebih efisien dengan tidak mengulangi metode tersebut di kelas anak.

Kesimpulan: Penulisan metode sudah fungsional, tetapi tidak sepenuhnya efisien karena ada pengulangan yang tidak perlu. Menggunakan inheritance dan overriding dengan bijak dapat mengurangi jumlah kode yang ditulis tanpa mengorbankan fungsionalitas.

4.2Untuk meningkatkan efisiensi penulisan kode, berikut adalah beberapa rekomendasi:

1. **Menghapus Pengulangan Metode**: Jika implementasi metode di kelas Anak sama dengan di kelas Ortu, cukup gunakan metode dari kelas induk tanpa menimpa atau mendeklarasikannya kembali.
2. **Menggunakan Method Overloading dengan Bijak**: Jika Anda perlu mempertahankan dua versi dari metode sukaMenonton, pastikan penamaan dan parameter jelas sehingga tidak membingungkan.

Kode yang Diperbaiki:

```
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);
   }
}

class Anak extends Ortu {
   // Method overloading
   void sukaMenonton(int a, String b) {
      System.out.println("Nonton Jam " + a + " Malam " + b);
   }

   // Tidak perlu menimpa jika tidak ada perubahan implementasi
   // void sukaMenonton(String a) { // Ini bisa dihapus jika sama dengan induk

   // }

   // Jika tidak ada perubahan implementasi, hapus
   // void sukaMembaca(String a) {
   //    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("\nSifat 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
    }
}
```