Nama & NPM	Topik:	Tanggal:
Michelia Erza Annadhira G1F024035	Kelas (Class)	14 September 2024

[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, dan
b. perilaku/ behavior untuk method!

- Ciri-ciri umum yang dapat menjadi atribut variabel
  Dalam konteks kelas manusia, ciri-ciri yang melekat pada setiap objek dari kelas tersebut, antara lain :
  - nama (String), nama seseorang merupakan identitas unik yang menjadi ciri khas setiap manusia
  - rambut (String), rambut seseorang merupakan karakteristik fisik yang membedakan satu orang dengan yang lain
  - umur (int), menyimpan usia manusia
  - jenisKelamin (String), menyimpan informasi jenis kelamin apakah dia laki-laki atau perempuan
  - beratBadan (float), menyimpan berat badan dalam kilogram
  - tinggiBadan (float), menyimpan tinggi badan dalam meter
  - warnaWarna (String), warna mata seseorang juga merupakan salah satu ciri fisik yang membedakan satu manusia dari yang lain

- Ciri-ciri umum yang dapat menjadi perilaku/behavior untuk method
  perilaku atau method dalam kelas manusia merupakan tindakan atau fungsi yang dapat dilakukan oleh objek tersebut, antara lain :
  - Berjalan(), menampilkan bahwa manusia sedang berjalan
  - Berbicara(), menampilkan bahwa manusia sedang berbicara
  - Tidur(), menampilkan bahwa manusia sedang tidur
  - Membaca(), menampilkan bahwa manusia sedang membaca
  - Makan(), menampilkan bahwa manusia sedang makan

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

Video penjelasan pembelajaran dapat diakses pada **Chanel Youtube Rumah Ilmu Raflesia**
https://www.youtube.com/channel/UC8B9rghd3dBiS6OKonLMyIw

Video Materi 1 tentang Kelas, Objek, Method – https://www.youtube.com/watch?v=60IdOc8m8Es

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

[2] Analisis dan Argumentasi

1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara mengembangkan kelas Manusia dengan atribut variabel dan perilaku/behavior untuk method.
2) Alasan solusi ini karena kelas Manusia saat ini hanya memiliki atribut variabel 'nama' dan 'rambut', serta konstruktor tanpa parameter yang tidak berguna secara praktis.
3) Perbaikan kode program dengan cara mengembangkan kelas Manusia dengan atribut variabel yang lebih lengkap dan method-method yang relevan.

[3] Penyusunan Algoritma dan Kode Program

- Algoritma
  a) Mulai
  b) Deklarasi kelas Manusia dengan dua variabel (nama, rambut)
  c) Deklarasi constructor kelas Manusia dengan parameter nama dan rambut
  d) Deklarasi method sukaNonton dengan parameter film
  e) Pada main method :
     - Buat objek baryu sky dari kelas Manusia dengan nama "Jina" dan warna rambut "Merah"
     - Panggil method sukaNonton pada objek sky dengan parameter "Marvel"
  f) Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan
  g) Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan
  h) Selesai

```
1  public class Manusia {  // deklarasi kelas
2      // deklarasi variabel
3      String nama;
4      String rambut;
5
6      // deklarasi constructor tanpa parameter
7      public Manusia(String nama, String rambut) {
8          System.out.println("Nama : " + nama +
9          "\n Warna rambut : " + rambut);
10     }
11
12     void sukaNonton(String film) {
13         System.out.println("Hobi menonton : " + film);
14     }
15
16     public static void main(String[] args) {
17         Manusia sky = new Manusia ("Jina", "Merah");
18         sky.sukaNonton("Marvel");
19     }
20 }
```

Luaran (Output) :

Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi

[4] Kesimpulan

- Dari permasalaha di atas dapat saya simpulkan bahwa permasalahan diselesikan dengan mendesain kelas Manusia yang memiliki atribut dan perilaku. Algoritma dan kode program

Java mencakup constructor dengan dan tanpa parameter, serta metode yang mencerminkan perilaku manusia seperti berjalan(), minum(), dan tidur (). Kode itu berhasil merepresentasikan objek manusia dengan fleksibilitas dan menghasilkan output yang sesuai. ➢ Pengambilan keputusan ini didasarkan pada prinsip-prinsip OOP (Object-Oriented-Programming), fleksibilitas dalam pembuatan objek, dan kemudahan untuk mengelola atribut dan perilaku objek melalui method yang terstruktur.

Nama & NPM	Topik:	Tanggal:
Michelia Erza Annadhira G1F024035	Objek	14 September 2024

[1] Identifikasi Masalah:

1) Uraikan permasalahan dan variabel

Salin dan tempel kode program berikut ke Eclipse atau JDoodle.

```
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?

- Susun kembali kode di contoh 2 dengan menambahkan data ciri-ciri Anda di dalam variabel constructor :
  - Saya menambah umur, warna kulit, dan tinggi badan di dalam variabel constructor :

```
public Ortu(String nama, String rambut, int umur, String
warnaKulit, int tinggiBadan)  {
 //nama, rambut, umur, waraKulit, dan tinggiBadan adalah
variabel constructor
   System.out.println(" Nama saya : "+ nama +
         "\n Warna Rambut : " + rambut +
         "\n Umur : " + umur +
         "\n Warna kulit : " + warnaKulit +
         "\n Tinggi Badan : " + tinggiBadan + “cm” +
         "\n Warna mata : " + warnaMata);
```

  - Dan mengisi public static bedasarkan data yang diminta :

```
public static void main (String[] args) {
   Ortu satu = new Ortu("Miche", "hitam", 18, “sawo matang”,
148, “coklat”);
```

- Jika Anda memiliki keturunan, atribut dan perilaku yang mungkin akan diturunakan, antara lain :
  - Atribut (Sifat yang bias diwariskan)
    - Warna rambut, jika warna rambut saya hitam, adam kemungkinan anak saya juga memiliki rambut hitam
    - Tinggi badan, genetic mempengaruhi tinggi badan, sehinga bias jadi anak saya mewarisi tinggi badan saya
    - Warna kulit, juga merupakan atribut yang dapat diwariskan dari orang tua ke anak

- Perilaku (Behavior)
  - Kejujuran, jika saya mengajarkan pentingnya kejujuran,perilaku ini dapat diwariskan sebagai nilai moral yang kuat
  - Ketegasan, jika saya cenderung tegas dalam bertindak, anak mungkin akan meniru cara saya membuat keputusan dengan tegas
  - Bekerja keras, sesuatu yang bias diwariskan melalui moted dari kelas induk ke kelas anak

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

Video penjelasan pembelajaran dapat diakses pada **Chanel Youtube Rumah Ilmu Raflesia**
https://www.youtube.com/channel/UC8B9rghd3dBiS6OKonLMyIw

Video Materi 1 tentang Kelas, Objek, Method – https://www.youtube.com/watch?v=60IdOc8m8Es

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

[2] Analisis dan Argumentasi

a) Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambahkan atribut dan perilaku lain yang lebih spesifik di dalam kelas “Ortu”.
b) Alasan solusi ini karena hal ini memungkinkan saya untuk memberikan informasi yang lebih lengkap tentang ciri-ciri dan perilaku yang dapat diwariskan oleh keterunan.
c) Perbaikan kode program dengan cara menambahkan beberapa atribut dan metode yang relevan dengan ciri-ciri dan perilaku.

[3] Penyusunan Algoritma dan Kode Program

❖ Algoritma
  a) Mulai
  b) Deklarasi atribut kelas Ortu (nama, rambut, umur, warnaKulit, tinggiBadan, warnaMata)
  c) Deklarasi constructor kelas Ortu dengan parameter :
     - Nama, rambut, umur, warnakulit, tinggiBadan, dan warnaMata
  d) Deklarasi method kerjaKeras dengan parameter b (String) dan print kalimat :
     - “Dia adalah orang yang selalu ” diikuti dengan nilai dari b
  e) Deklarasi method sukaMembaca dengan parameter c (String) dan print kalimat :
     - “Suka membaca ” diikuti dengan nilai dari c
  f) method main :
     - Buat objek Ortu bernama sky lalu isi dengan parameter sesuai dengan data diri
     - Setelah objek dibuat print informasi objek
  g) Panggil method kerjaKeras pada objek sky dengan parameter “Bekerja keras”
  h) Panggil method sukaMembaca pada objek sky dengan parameter “Novel”
  i) Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan
  j) Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan
  k) Selesai

```java
public class Ortu {  // deklarasi kelas
    // deklarasi variabel
    String nama;
    String rambut;
    int umur;
    String warnaKulit;
    int tinggiBadan;
    String warnaMata;

    // deklarasi constructor tanpa parameter
    public Ortu(String nama, String rambut, int umur, String warnaKulit, int tinggiBadan, String warnaMata) {
        System.out.println("Nama : " + nama +
        "\n Warna rambut : " + rambut +
        "\n Umur : " + umur +
        "\n Warna kulit : " + warnaKulit +
        "\n Tinggi badan : " + tinggiBadan + " cm" +
        "\n Warna mata : " + warnaMata);
    }

    void kerjaKeras(String b) {
        System.out.println("Dia adalah orang yang selalu " + b);
    }

    void sukaMembaca(String c) {
        System.out.println("Suka membaca " + c);
    }

    public static void main(String[] args) {
        Ortu sky = new Ortu ("Miche", "Hitam", 18, "sawo matang", 148, "coklat");
        sky.kerjaKeras("Bekerja keras");
        sky.sukaMembaca("Novel");
    }
}
```

Luaran (Output)

Output Generated Files

```
Nama : Miche
 Warna rambut : Hitam
 Umur : 18
 Warna kulit : sawo matang
 Tinggi badan : 148 cm
 Warna mata : coklat
Dia adalah orang yang selalu Bekerja keras
Suka membaca Novel
```

Compiled and executed in 1.216 sec(s)

Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi

[4] Kesimpulan

- Pada program ini saya menggunakan constructor karena constructor memungkinakan inisialisasi atribut objek saat objek dibuat, memberikan fleksibilitas unutk memasukkan nilai-nilai spesifik seperti nama, warna rambut, umur, warna kulit, tinggi badan, dan lainya.
- Perbaikkan program dengan menambahkan metode untuk mencerminkan perilaku seperti bersantai() karena struktur Java mengharuskan perilaku objek didefinisikan melalui metode, yang memungkinkan objek tidak hanya memiliki data, tetapi juga fungsi untuk berinteraksi atau bertindak sesuai dengan atribut yang dimiliki.

Nama & NPM	Topik:	Tanggal:
Michelia Erza Annadhira G1F024035	**Method**	**14 September 2024**

[1] Identifikasi Masalah:

A. Uraikan permasalahan dan variabel

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");
    }
}
```

Luaran 3:

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

Latihan 3:

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

- Perbedaan deklarasi constructor, method, dan method utama :
  - Constructor, digunakan untuk menginisialisasi objek ketika objek tersebut dibuat. Pada kode program di atas, constructor `Manusia1` digunakan untuk menetapkan nilai awal atribut `nama` dan `rambut` pada objek `Manusia`.
    Pada kode program ini, ada kesalahan pada nama constructor, nama constructor haruslah sama dengan nama kelas, sehingga seharusnya `Manusia` dan bukan `Manusia1`.
  - Method, blok kode yang dapat dipanggil untuk melakukan tugas tertentu. Method memiliki nama, parameter, dan tipe pengembalian.
    Method `sukaNonton` pada kode program digunakan untuk mencetak hobi menonton seseorang berdasarkan input parameter `film`.
  - Method Utama (main method), titik masuk dari program Java. Di sini `Manusia` dibuat dan method `sukaNonton` dipanggil.
    Method `main` memiliki tanda tangan khusus `public static void main(String[] args)`. Ini adalah method statis yang dijalankan pertama kali

saat program dimulai.

- Kapan kita perlu menggunakan constructor dan method :
  - Constructor, digunakan ketika kita ingin menginisialisasi nilai-nilai dari suatu objek pada saat objek tersebut dibuat. Constructor secara otomatis dipanggil ketika keyword `new` digunakan untuk membuat objek.
  - Method, digunakan ketika kita ingin melakukan suatu operasi atau aksi berulang kali pada objek. Method dapat dipanggil kapan saja setelah objek dibuat, dan bias dipanggil lebih dari sekali.
- Uraikan perbedaan berikut :
  a) Constructor overloading dan overriding
     - Constructor Overloading, terjadi ketika ada beberapa constructor dalam satu kelas yang memiliiki nama sama tetapi parameter yang berbeda. Contohnya, `Manusia(String nama)` dan `Manusia(String nama, String rambut`.
     - Constructor Overriding, tidak bias di-override karena constructor hanya berlaku untuk kelas yang mendeklarasikan dan tidak diwariskan ke subclass.

  b) Method overloading dan method overriding
     - Method Overloading, terjadi ketika ada beberapa method dalam satu kelas yang memiliki nama sama tetapi berbeda jumlah atau tipe parameter. Contohnya, `sukaNonton(String film)` dan `sukaNonton(String film, int jam)`.
     - Method Overriding, teradi ketika subclass mendefinisikan kembali method yan sudah ada di superclass dengan signature yang sama. Contohnya, jika kelas `anak` mewarisis dari `Manusia` dan menimpa method `sukaNonton` dengan implementasi yang berbeda.

  c) Method yang menggembalikan nilai dan method tidak mengembalikan nilai
     - Mengembalikan nilai, method ini mengembalikan hasil nilai berupa suatu tipe data tertentu. Missal, method denan return tipe `int` akan mengembalikan bilangan bulat. Contohnya, `int hitungUmur()` akan mengembalikan umur sebagai bilangan bulat.
     - Tidak mengembalikan nilai (void method), method ini tidak menggunakan nilai hanya menjalankan logika tertentu. Contohnya, `void sukaNonton(String film)` hanya mencetak sesuatu ke layar tanpa mengembalikan nilai.

B. Rincikan sumber informasi yang relevan (buku / webpage)

Video penjelasan pembelajaran dapat diakses pada **Chanel Youtube Rumah Ilmu Raflesia**
https://www.youtube.com/channel/UC8B9rghd3dBiS6OKonLMyIw

Video Materi 1 tentang Kelas, Objek, Method – https://www.youtube.com/watch?v=60IdOc8m8Es

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

[2] Analisis dan Argumentasi

I. Saya mengusulkan permasalahan ini dapat diatasi dengan cara mengganti nama constructor dari `Manusia1` menjadi `Manusia`, agar sesuai dengan nama kelas.

II. Alasan solusi ini karena dalam Java, nama constructor harus sama persis denan nama kelas. Jika nama constructor tidak sama, maka program tidak akan dapat di komplikasi dengan benar.

III. Perbaikan kode program dengan cara mengubah baris contructor dari :
`public Manusia1(String nama, String rambut) {`

```
   System.out.println(" Nama saya : "+ nama +
   "\n Warna Rambut : " + rambut);
```

menjadi :

```
public Manusia(String nama, String rambut) {
   System.out.println(" Nama saya : "+ nama +
   "\n Warna Rambut : " + rambut);
```

[3] Penyusunan Algoritma dan Kode Program

- Algoritma
  a. Mulai
  b. Deklarasi kelas `Manusia` dengan dua atribut `nama` dan `rambut`.
  c. Buat constructor dalam kelas `Manusia` yang menerima dua parameter (nama dan rambut)
  d. Deklarasikan method `sukaNonton(String film)` untuk mencetak hobi menonton dari objek `Manusia`.
  e. Dalam method `main`, buat objek baru dari kelas `Manusia` dengan parameter `"putri"` dan `"hitam"`.
  f. Panggil method `sukaNonton` pada objek yang telah dibuat dengan argumen `"Drakor"`.
  g. Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan
  h. Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan
  i. Selesai

```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");
    }
}
```

Luaran (Output) :

Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi

[4] Kesimpulan

- Pada program ini saya menggunakan constructor karena constructor digunakan untuk menginisialisasi objek dengan nilai atribut yang diberikan saat objek dibuat. Constructor juga memastikan bahwa setiap objek memiliki nilai awal yang ditetapkan secara langsung ketika

diciptakan.

- Perbaikkan program dengan menambahkan penamaan constructor yang benar yaitu menggantinya menjadi nama yang sama dengan nama kelas karena struktur java mengharuskan constructor memiliki nama yang sama persis dengan kelasnya. Jika tidak sama, program tidak akan dapat berjalan dengan baik yang menimbulkan eror.

Nama & NPM	Topik:	Tanggal:
Michelia Erza Annadhira G1F024035	**Extends**	**14 September 2024**

[1] Identifikasi Masalah:

I. Uraikan permasalahan dan variabel

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
    }
}
```

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

- Perbandingan method yang dimiliki `class Anak extends Ortu` dengan method di `class Ortu`:
  - Di class `Anak`, method `sukaMenonton(int a, String b)` adalah overloading dari method `sukaMenonton(String a)` di class `Ortu`. Itu berarti method ini menerima parameter tambahan untuk menambah fungsi baru.
  - Method `sukaMenonton(String a)` di class `Anak` melakukan override method dari class `Ortu`. Menggantikan perilaku method yang sama di class `Ortu` untuk mencetak output yang berbeda.
  - Method `sukaMenonton(String a)` juga di-override di class `Anak` untuk mencetak output yang berbeda dari method di class `Ortu`, meskipun tetap menggunakan parameter yang sama
- Ubahlah dengan menambahkan objek anak dengan method yang berbeda :
  Di sini saya menambahkan method baru `sukaOlahraga(String a)` di class `Anak`. Method ini mencetak aktivitas olahraga yang disukai oleh objek anak. Method ini tidak ada di class `Ortu`, sehingga hanya tersedia di class `Anak`.
  Pada `main()` method, objek `objek` dari class `Anak` memanggil method baru `sukaOlahraga("Bola Basket")`.

II. Rincikan sumber informasi yang relevan (buku / webpage)

Video penjelasan pembelajaran dapat diakses pada **Chanel Youtube Rumah Ilmu Raflesia**
https://www.youtube.com/channel/UC8B9rghd3dBiS6OKonLMyIw

Video Materi 1 tentang Kelas, Objek, Method – https://www.youtube.com/watch?v=60IdOc8m8Es

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

[2] Analisis dan Argumentasi

a. Saya mengusulkan permasalahan ini dapat diatasi dengan cara menambahkan method baru pada class `Anak` yang tidak ada di class `Ortu`, seperti method `sukaOlahraga`.
b. Alasan solusi ini karena class `Anak` dapat memiliki behavior atau kemampuan baru selain yang diwarisi dari class `Ortu`. Dengan menambahkan method yang unik di class `Anak`, kita dapat memodifikasi dan memperluas fungsional dari class `Ortu` tanpa harus mengubah kode induknya.
c. Perbaikan kode program dengan cara menambahkan method `sukaolahraga(Sring a)` pada class `Anak`, kemudian memanggil method tersebut melalui objek `Anak` di method `main()` class `Anak`. Hal ini memungkinkan anak untuk memiliki aktivitas tambahan yang tidak dimiliki oleh induknya.

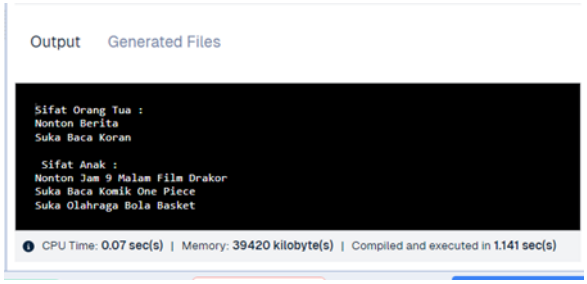
[3] Penyusunan Algoritma dan Kode Program

- Algoritma

a. Mulai
b. Buat class `Ortu` :
   - Tambahkan method `sukaMenonton` yang menerima satu parameter bertipe String
   - Tambahkan method `sukaMembaca` yang menerima satu parameter bertipe String
c. Buat class `Anak` yang meng-extend class `Ortu` :
   - Override method `sukaMenonton(String a)` dengan logika baru
   - Tambahkan method baru `sukaMenonton(int a, String b)` yang menerima dua parameter (integer dan String)
   - Override method `sukaMembaca(String a)` dari class `Ortu` dengan logika baru
   - Tambahkan method baru `sukaOlahraga(String a)` yang menerima satu parameter bertipe String
d. Di dalam class `Anak`, buat method `main()` :
   - Print output dari class `Ortu`.
   - Buat objek dari class `Ortu`.
   - Panggil method `sukaMenonton("Berita")` pada objek `Ortu`.
   - Panggil juga method `sukaMembaca("Koran")` pada objek `Ortu`.
   - Print output 'sifat anak'
   - Buat objek dari class `Anak`.
   - Panggil method `sukaMenonton(9, "Film Drakor")`, `sukaMembaca("Komik One Piece")`, dan `sukaOlahraga("Bola Basket")` pada objek `Anak`.
e. Jalankan kode untuk memeriksa apakah program bias dijalankan dengan baik dan tidak ada kesalahan
f. Simpan kode program yang telah diperbaiki dan catat perubahan yang telah dilakukan
g. Selesai

```
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.sukaOlahraga("Bola Basket");             //method baru di class 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);
    }
    void sukaOlahraga(String a) {
        System.out.println("Suka Olahraga " + a);   //method baru di class Anak
    }

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.sukaOlahraga("Bola Basket");             //memanggil method baru di class Anak
    }
}
```

Luaran (Output) :



Kode program sudah disusun dengan struktur yang benar. Hal ini dapat dilihat dari luaran (output) yang sesuai dengan kode program dan tidak terjadi eror lagi

[4] Kesimpulan

- Pada program ini saya menggunakan inharintance (pewarisan) karena class `Anak` perlu

mewarisi method dari class `Ortu` tanpa perlu mendeklarasikan ulang method yang sama. Dengan inheritance (pewarisan), kita bisa memanfaatkan kembali kode dari class induk, sekaligus menambahkan perilaku di class anak sesuai kebutuhan.

- Perbaikkan program dengan menambahkan method baru `sukaOlahraga(String a)` di class `Anak` karena struktur java mengharuskan setiap class anak yang ingin memiliki fitur baru (behavior atau method baru) harus mendefinisikan method tersebut di dalam class anak secraa eksplisit. Method baru memberikan tambahan kemampuan pada class Anak, tanpa memengaruhi class induk secara langsung

Refleksi

(Tuliskan singkat tentang pengalaman belajar, pemaknaan pengetahuan yang baru, tantangan yang dihadapi pada minggu tersebut. Ringkasan singkat dari semua soal, bukan per soal)

Selama minggu ini, saya telah belajar tentang konsep penting dalam pemprograman, khususnya incharitance, method overriding, dan method overloading dalam Java. Pengalaman ini memberikan pemahaman lebih mendalam tentang bagaiman pewarisan mempermudah pengunaan kembali kode dan bagaimana method overriding memungkinkan kita untuk mengubah perilaku method yang diwarisi tanpa mengubah kode induknya.

Saya juga memahami bahwa Java memungkinkan kita menambahkan fungsional baru di class anak tanpa harus memodifikasi class induk, yang mendukung prinsip modularitas dan pemeliharaan kode yang lebih mudah.

Tantangan yang saya hadapi adalam agak sulit memahami kapan harus menggunakan overriding disbanding overloading serta bagaimana memastikan struktur program tetap rapid an tidak berlebihan.