Tugas 1

Nama & NPM	Topik:	Tanggal:
M. Jenyfer Aprilino G1F024057	Kelas Java	18/09/2024

[1,1,1,2] 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)
```

1.1 Perbaiki pesan kesalahan Contoh 1!

Jawab: Pada soal masih ada pesan kesalahan:

- **Nama Kelas dan Konstruktor: Kelas dideklarasikan sebagai Manusia, jadi konstruktor juga harus bernama Manusia, bukan Manusia1.**
- **Parameter Konstruktor: Konstruktor yang sebelumnya hanya memiliki satu parameter (String nama) harus memiliki dua parameter (String nama, String rambut) agar sesuai dengan instansiasi objek di main.**
- **Menginisialisasi Atribut: Anda perlu menginisialisasi this.nama dan this.rambut di dalam konstruktor untuk menyimpan nilai yang diterima.**

1.2 Cermati contoh 1. susun kode menggunakan constructor dengan parameter data pribadi anda!

Jawab: Diketahui dari soal:

- **Kelas Person: Kelas ini memiliki tiga atribut: nama, umur, dan alamat.**
- **Konstruktor: Konstruktor menerima tiga parameter untuk menginisialisasi atribut kelas.**
- **Method tampilkanData: Method ini digunakan untuk menampilkan data pribadi yang telah disimpan.**
- **Method main: Di sinilah objek Person dibuat dengan data contoh. Anda dapat mengganti "John Doe", 25, dan "Jalan Contoh No. 123" dengan data pribadi Anda.**

[1.1, 1.2] Penyusunan Algoritma dan Kode Program

1.1
```
public class Manusia {  // deklarasi kelas
   // deklarasi atribut Manusia dalam variabel
   String nama, rambut;

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

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

Luaran:

```
java -cp /tmp/SS1u1CNoKN/Manusia
Nama saya: Putri
Warna Rambut: hitam
=== Code Execution Successful ===
```

1.2
```
public class Person {
   // Deklarasi atribut
   String nama;
   int umur;
   String alamat;

   // Deklarasi konstruktor dengan parameter
   public Person(String nama, int umur, String alamat) {
      this.nama = nama; // Inisialisasi atribut nama
      this.umur = umur; // Inisialisasi atribut umur
      this.alamat = alamat; // Inisialisasi atribut alamat
   }

   // Method untuk menampilkan data pribadi
   public void tampilkanData() {
      System.out.println("Nama: " + this.nama);
      System.out.println("Umur: " + this.umur);
      System.out.println("Alamat: " + this.alamat);
   }

   // Deklarasi method utama
   public static void main(String[] args) {
      // Membuat objek Person dengan data pribadi
      Person saya = new Person("John Doe", 25, "Jalan Contoh No. 123");
      saya.tampilkanData(); // Menampilkan data pribadi
```

```
    }
}
```

Luaran:

```
 java -cp /tmp/DwRU5CitMZ/Person
Nama: John Doe
Umur: 25
Alamat: Jalan Contoh No. 123

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

[1.1, 1.2] Kesimpulan

1. **Konstruktor**: Konstruktor adalah metode khusus dalam sebuah kelas yang digunakan untuk menginisialisasi objek. Dengan menggunakan parameter
2. **Encapsulation**: Penggunaan atribut dan metode dalam kelas membantu dalam menerapkan prinsip encapsulation, yang menjaga data pribadi dan menyediakan cara untuk mengakses dan memanipulasinya.
3. **Praktik Pemrograman Baik**: Menggunakan konstruktor dengan parameter membuat kode lebih fleksibel dan mudah digunakan,.
4. **Contoh Implementasi**: Dalam contoh yang diberikan, kelas Person memungkinkan kita untuk menyimpan dan menampilkan informasi pribadi dengan jelas,

Refleksi

Mendapatkan pemahaman yang baru

Tugas 2

Nama & Npm	**Topik:**	**Tanggal:**
M. Jenyfer Aprilino G1F024057	**Kelas Java**	**18/09/2024**

[2.1, 2.2, 2.3] Identifikasi Masalah:

```
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)
```

2.1 Evaluasi penyebab kesalahan dan perbaiki kode tersebut!

Jawab:

- **Menambahkan variabel nama dan rambut dengan tipe data String.**
- **Mengubah nama constructor menjadi Ortu, sama dengan nama kelas.**
- **Menambahkan parameter pada constructor dan menginisialisasi variabel instance dengan nilai parameter tersebut.**

2.2 Apabila nanti Anda akan memiliki keturunan, analisa sifat (atribut) dan constructor sebagai Ortu apa yang akan diturunkan (gunakan data karakter pribadi anda) ?
Jawab:

- **nama: Nama keturunan.**
- **rambut: Warna rambut keturunan.**
- **sifat: Kepribadian, seperti ramah, ceria, atau pemikir kritis.**
- **hobi: Kegiatan yang disukai, misalnya membaca, bermain musik, atau olahraga.**
- **kecerdasan: Kemampuan belajar, seperti minat dalam matematika atau bahasa.**

2.3 Rancanglah kode program untuk sifat (atribut) dan constructor overloaded dari Latihan 2.2!
Jawab:

- **Kelas Ortu:**
- **Memiliki dua constructor: satu tanpa parameter (default) dan satu dengan parameter untuk inisialisasi nama dan rambut.**
- **Metode displayInfo() untuk menampilkan informasi.**
- **Kelas Keturunan:**
- **Mengextends kelas Ortu dan juga memiliki dua constructor: satu tanpa parameter dan satu dengan parameter yang memanggil constructor dari kelas Ortu.**
- **Atribut tambahan untuk sifat, hobi, dan kecerdasan.**
- **Metode displayKeturunanInfo() untuk menampilkan semua informasi.**
- **Kelas Main:**
- **Di sini, kita membuat objek dari Keturunan menggunakan kedua constructor untuk mendemonstrasikan overloaded constructor.**

[2.1, 2.2, 2.3] Penyusunan Algoritma dan Kode Program

```
2.1 public class Ortu {
    // Deklarasi variabel
    private String nama;
    private String rambut;

    // Constructor dengan parameter
    public Ortu(String nama, String rambut) {
        this.nama = nama; // Menggunakan 'this' untuk membedakan variabel instance
dan parameter
        this.rambut = rambut;
        System.out.println("Nama saya: " + this.nama +
                "\nWarna Rambut: " + this.rambut);
    }

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

```
Luaran: java -cp /tmp/Mp1G9KmzjN/Ortu
        Nama saya: Putri
        Warna Rambut: hitam

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

2.2
```
public class Keturunan extends Ortu {
  private String sifat;
  private String hobi;
  private int kecerdasan; // Misalnya skala 1-10

  // Constructor untuk Keturunan
  public Keturunan(String nama, String rambut, String sifat, String hobi, int
kecerdasan) {
    super(nama, rambut); // Memanggil constructor dari kelas Ortu
    this.sifat = sifat;
    this.hobi = hobi;
    this.kecerdasan = kecerdasan;

    System.out.println("Sifat saya: " + this.sifat +
            "\nHobi saya: " + this.hobi +
            "\nTingkat kecerdasan: " + this.kecerdasan);
  }
}
```

```
Luaran: ERROR!
        /tmp/BWrI9WTR5p/Keturunan.java:1: error: cannot find symbol
        public class Keturunan extends Ortu {
                                       ^
          symbol: class Ortu
        1 error

        === Code Exited With Errors ===
```

2.3
```
public class Ortu {
  private String nama;
  private String rambut;

  // Constructor tanpa parameter
  public Ortu() {
    this.nama = "Tidak Diketahui";
    this.rambut = "Tidak Diketahui";
  }

  // Constructor dengan parameter
  public Ortu(String nama, String rambut) {
    this.nama = nama;
    this.rambut = rambut;
  }

  // Metode untuk menampilkan informasi
  public void displayInfo() {
    System.out.println("Nama saya: " + this.nama +
            "\nWarna Rambut: " + this.rambut);
```

```
    }
}

class Keturunan extends Ortu {
    private String sifat;
    private String hobi;
    private int kecerdasan; // Skala 1-10

    // Constructor tanpa parameter
    public Keturunan() {
        super(); // Memanggil constructor Ortu tanpa parameter
        this.sifat = "Tidak Diketahui";
        this.hobi = "Tidak Diketahui";
        this.kecerdasan = 0;
    }

    // Constructor dengan parameter untuk Keturunan
    public Keturunan(String nama, String rambut, String sifat, String hobi, int
     kecerdasan) {
        super(nama, rambut); // Memanggil constructor Ortu dengan parameter
        this.sifat = sifat;
        this.hobi = hobi;
        this.kecerdasan = kecerdasan;
    }

    // Metode untuk menampilkan informasi keturunan
    public void displayKeturunanInfo() {
        displayInfo(); // Menampilkan informasi dari kelas Ortu
        System.out.println("Sifat saya: " + this.sifat +
                           "\nHobi saya: " + this.hobi +
                           "\nTingkat kecerdasan: " + this.kecerdasan);
    }
}

public class Main {
    public static void main(String[] args) {
        // Menggunakan constructor tanpa parameter
        Keturunan keturunan1 = new Keturunan();
        keturunan1.displayKeturunanInfo();

        System.out.println();

        // Menggunakan constructor dengan parameter
        Keturunan keturunan2 = new Keturunan("Putri", "Hitam", "Ramah", "Membaca",
     8);
        keturunan2.displayKeturunanInfo();
    }
}

Luaran: ERROR!
        /tmp/819EoKQRv0/Ortu.java:54: error: class Main is public, should be
        declared in a file named Main.java
        public class Main {
               ^
```

1 error

=== Code Exited With Errors ===

[2.1, 2.2, 2.3] Kesimpulan

- **Penggunaan Constructor Overload**: Constructor overload memungkinkan kita untuk membuat beberapa versi dari constructor dalam sebuah kelas.
- **Pewarisan (Inheritance)**: Dalam contoh, kelas Keturunan mewarisi sifat-sifat dari kelas Ortu. Ini menunjukkan bagaimana objek dapat memiliki atribut dan metode dari kelas induk, serta menambah atau mengubah atribut sesuai kebutuhan.
- **Pentingnya Metode**: Metode displayInfo() dan displayKeturunanInfo() digunakan untuk menampilkan informasi dari objek. Ini menunjukkan prinsip pemrograman yang baik, yaitu memisahkan logika penyimpanan data dan presentasi data.
- **Penggunaan Encapsulation**: Dengan mendeklarasikan atribut sebagai private, kita memastikan bahwa data tidak dapat diakses langsung dari luar kelas, melainkan melalui metode yang telah ditentukan, menjaga integritas data.

Refleksi

Mendapatkan pemahaman yang baru

Tugas 3

Nama & NPM	**Topik:**	**Tanggal:**
M. Jenyfer Aprilino G1F024057	**Kelas Java**	**18/09/2024**

[Nomor Soal] 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)
```

3.1 Evaluasi penyebab kesalahan dan perbaiki kode tersebut!

Jawab:

Terdapat beberapa kesalahan dalam kode Java yang Anda berikan. Berikut adalah perbaikan yang diperlukan untuk mengatasi masalah tersebut:

- Nama kelas dalam constructor harus sesuai dengan nama kelas.
- Method sukaNonton harus dideklarasikan dengan benar, baik dari segi parameter maupun tipe kembalian.
- Ada beberapa variabel yang tidak dideklarasikan, seperti film, episode, dan durasi.

Yang telah diperbaiki

- **Constructor**: Ganti public Manusia1 menjadi public Manusia untuk mencocokkan dengan nama kelas.
- **Method sukaNonton**: Tambahkan parameter String film untuk method pertama dan int episode, int durasi untuk method kedua.
- **Return Value**: Tambahkan return pada method sukaNonton(int episode, int durasi) untuk mengembalikan hasil perhitungan.

3.2 a.) Method Overloading:

- perkenalan(): Metode pertama hanya memperkenalkan nama.
- perkenalan(String hobi): Metode kedua memperkenalkan nama dan juga menyebutkan hobi.

b.) Method dengan Return Value:

- getDeskripsi(): Metode ini mengembalikan string yang berisi deskripsi tentang nama dan warna rambut.

c.) Method tanpa Return Value:

- tampilkanHobi(String hobi): Metode ini mencetak hobi yang diberikan sebagai parameter.

[Nomor Soal] Penyusunan Algoritma dan Kode Program

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

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

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

    // method untuk menghitung jam nonton
    int sukaNonton(int episode, int durasi) {
        return 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:

```
 java -cp /tmp/qysq1tyquO/Manusia
Nama saya: Putri
Warna Rambut: hitam
Hobi Menonton: Drakor
Jam nonton = 4 jam

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

3.2 a,b,c.

```
public class Ortu {
    private String nama;
    private String rambut;

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

    // a) Method Overloading
    public void perkenalan() {
        System.out.println("Halo, saya " + this.nama + ".");
    }

    public void perkenalan(String hobi) {
```

```
        System.out.println("Halo, saya " + this.nama + " dan hobi saya adalah " + hobi
+ ".");
    }

    // b) Method dengan Return Value
    public String getDeskripsi() {
        return "Nama: " + this.nama + ", Warna Rambut: " + this.rambut;
    }

    // c) Method tanpa Return Value
    public void tampilkanHobi(String hobi) {
        System.out.println("Hobi saya adalah: " + hobi);
    }

    public static void main(String[] args) {
        Ortu orangTua = new Ortu("Putri", "hitam");

        // Memanggil method overloading
        orangTua.perkenalan();
        orangTua.perkenalan("menggambar");

        // Memanggil method dengan return value
        String deskripsi = orangTua.getDeskripsi();
        System.out.println(deskripsi);

        // Memanggil method tanpa return value
        orangTua.tampilkanHobi("membaca");
    }
}
```

Luaran:

```
 java -cp /tmp/uHX7TmmgKg/Ortu
Nama saya: Putri
Warna Rambut: hitam
Halo, saya Putri.
Halo, saya Putri dan hobi saya adalah menggambar.
Nama: Putri, Warna Rambut: hitam
Hobi saya adalah: membaca

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

[3.1, 3.2] Kesimpulan

1. **Pentingnya Deklarasi yang Benar**: Dalam pemrograman, sangat penting untuk mendeklarasikan kelas, constructor, dan metode dengan nama yang konsisten agar tidak menimbulkan kebingungan dan kesalahan.
2. **Method Overloading**: Dengan menggunakan metode yang sama dengan nama yang berbeda tetapi dengan parameter yang berbeda.
3. **Penggunaan Parameter dan Return Value**: Menggunakan parameter dalam metode memungkinkan kita untuk membuat metode yang lebih dinamis dan dapat digunakan kembali
4. **Debugging**: Ketika menghadapi kesalahan, penting untuk membaca pesan kesalahan dengan cermat dan menganalisis bagian mana dari kode yang menyebabkan masalah. Ini membantu dalam menemukan dan memperbaiki kesalahan dengan lebih cepat.

Dengan mengikuti prinsip-prinsip ini, Anda dapat menulis kode yang lebih efisien, terstruktur, dan mudah dipahami.
Refleksi Mendapatkan pemahaman yang baru

Tugas 4

Nama & NPM	**Topik:**	**Tanggal:**
M. Jenyfer Aprilino G1F024057	**Kelas Java**	**18/09/2024**

[Nomor Soal] 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

4.1 Evaluasi method yang dimiliki Contoh 4 pada class Anak extends Ortu dengan method di class Ortu.
Simpulkan hasil evaluasi Anda agar method ini menjadi efisien
Jawab:

1. **Overriding vs Overloading:**
   - **sukaMenonton(String a) di kelas Anak mengoverride metode yang sama dari kelas Ortu. Namun, kelas Anak juga memiliki metode overloading sukaMenonton(int a, String b). Ini memberikan fleksibilitas tetapi juga dapat membingungkan jika tidak digunakan dengan jelas.**
   - **Untuk menyederhanakan, jika sukaMenonton(String a) di kelas Anak tidak ditujukan untuk memperluas fungsionalitas, maka bisa dihapus. Anda dapat langsung menggunakan metode dari kelas Ortu.**
2. **Redundansi:**
   - **sukaMembaca(String a) di kelas Anak juga mengoverride metode dari kelas Ortu. Jika fungsionalitas yang diinginkan tidak berubah, Anda tidak perlu mendeklarasikannya ulang di kelas Anak.**
   - **Anda cukup memanggil metode induk, sehingga lebih jelas dan mengurangi redundansi.**
3. **Keterbacaan dan Kejelasan:**
   - **Terlalu banyak metode yang overload dapat membuat kode sulit dipahami. Sebaiknya jaga agar setiap metode memiliki satu tanggung jawab yang jelas.**

Kesimpulan
Untuk membuat metode di kelas Anak lebih efisien, Anda dapat melakukan hal berikut:

1. **Hapus Metode yang Tidak Perlu:**
   - **Hapus sukaMenonton(String a) dan sukaMembaca(String a) dari kelas Anak jika mereka tidak melakukan hal yang berbeda dari metode induk.**
2. **Panggil Metode Induk:**
   - **Jika ingin menggunakan metode induk yang sama, panggil langsung super.sukaMembaca(a) di kelas Anak ketika diperlukan.**

4.2 . Setelah dirunning di JDoodle, catat waktu eksekusinya.
Susun kembali kode program yang dapat mengefisienkan waktu eksekusi!
Jawab:

waktu : 04,09

[4.1, 4.2] Penyusunan Algoritma dan Kode Program

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

   // Hapus overriding jika tidak ada perubahan yang signifikan
}

4.2 public class Ortu {      // kelas induk
   void sukaMenonton(String a) {    // method induk spesifik
      System.out.println("Nonton " + a);
   }
   void sukaMembaca(String a) {     // method induk umum
      System.out.println("Suka Baca " + a);
   }

   public static void main(String[] args) {
      Ortu objekO = new Ortu();     // objek induk
      System.out.println("Sifat Orang Tua :");
      objekO.sukaMenonton("Berita");
      objekO.sukaMembaca("Koran");

      Anak objekA = new Anak();    // objek anak
      System.out.println("\nSifat Anak :");
      objekA.sukaMenonton(9, "Film Drakor");
      objekA.sukaMembaca("Komik One Piece");
   }
}

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

Luaran:
java -cp /tmp/jcW1APrU8F/Ortu
Sifat Orang Tua :
Nonton Berita
Suka Baca Koran

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

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

[4.1, 4.2] Kesimpulan

- **Pengurangan Redundansi:**
  Menghapus metode yang tidak perlu di kelas Anak yang mengoverride metode dari kelas Ortu meningkatkan efisiensi dan mengurangi kebingungan dalam kode.
- **Sederhanakan Struktur Kode:**

Menggunakan satu main method di kelas Ortu membantu menghindari duplikasi dan mengurangi overhead, menjadikan program lebih ringkas dan mudah dipahami.

- **Keterbacaan yang Lebih Baik:**
  Dengan menjaga kode tetap sederhana dan jelas, pemeliharaan dan pengembangan di masa depan menjadi lebih mudah.
- **Peningkatan Efisiensi:**
  Meskipun dampak pada waktu eksekusi mungkin tidak signifikan dalam program kecil, prinsip-prinsip yang diterapkan dapat memberikan manfaat besar dalam aplikasi yang lebih kompleks.

Refleksi
Mendapatkan pemahaman yang baru