

Lattihan 1

Nama & NPM	Topik:	Tanggal:
Dini Ramadona G1F024050	Kelas, Objek dan Method	17 September 2024
[No.1] Identifikasi Masalah:		
<p>Latihan 1:</p> <ol style="list-style-type: none">1.1. Perbaiki pesan kesalahan Contoh 1!1.2. Analisa ciri-ciri lain Kelas Manusia yang dapat menjadi<ol style="list-style-type: none">a. atribut variabel, danb. perilaku/ behavior! <p>Contoh 1:</p> <pre>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"); } }</pre> <p>Pada soal diatas kita disuruh untuk memperbaiki kode memperbaiki nama konstruktor agar sesuai dengan nama kelas dan menyesuaikan parameter konstruktor untuk menerima dua argumen (nama dan warna rambut) selain itu juga kita disuruh untuk menganalisis perilaku atau method dari kelas Manusia. Perilaku adalah fungsi atau tindakan yang dapat dilakukan oleh objek dari kelas tersebut.</p>		
[No.1] Analisis dan Argumentasi		
<ol style="list-style-type: none">1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara kode yang ada terdapat kesalahan dalam nama konstruktor. Nama konstruktor harus sama dengan nama kelas. Di dalam kode mendeklarasikan konstruktor dengan nama Manusia1, tetapi nama kelasnya adalah Manusia. Selain itu, konstruktor hanya memiliki satu parameter, tetapi di dalam main, mencoba untuk memanggil konstruktor dengan dua parameter. Perbaiki dengan cara ubah nama konstruktor menjadi Manusia dan tambahkan parameter yang sesuai.2) Nama konstruktor diubah menjadi Manusia agar sesuai dengan nama kelas.Konstruktor sekarang memiliki dua parameter (String nama dan String rambut) untuk menginisialisasi atribut nama dan rambut.Kata kunci this digunakan untuk membedakan antara parameter konstruktor dan atribut kelas yang memiliki nama yang sama.Di dalam metode main, pemanggilan konstruktor sekarang benar karena mencocokkan jumlah dan jenis parameter yang didefinisikan.		
[No.1] Penyusunan Algoritma dan Kode Program		
<ol style="list-style-type: none">1) Algoritma<ol style="list-style-type: none">(a) Membuka aplikasi eclipse(b) Salin kode yang ada di E-learning lalu paste ke eclipse(c) Mencari Penyebab Kesalahan kode(d) Mencari tutorial di youtube Rumah Raflesia(e) Mencari tipe data yang cocok lalu perbaiki kesalahan(f) Ubah nama konstruktor menjadi Manusia dan tambahkan parameter yang sesuai.(g) Tambahkan perilaku atau method dalam kelas Manusia		

- (h) Periksa hasil output untuk memastikan bahwa data ditampilkan dengan benar sesuai dengan tipe data yang digunakan
- (i) Selesai.

2) Kode program dan luaran

```
Online Java Compiler IDE

1 public class Manusia {
2     // Deklarasi atribut variabel
3     String nama;
4     String rambut;
5     int umur;
6     String alamat;
7     String pekerjaan;
8
9     // Deklarasi constructor
10    public Manusia(String nama, String rambut, int umur, String alamat, String pekerjaan) {
11        this.nama = nama;
12        this.rambut = rambut;
13        this.umur = umur;
14        this.alamat = alamat;
15        this.pekerjaan = pekerjaan;
16
17        System.out.println("Nama saya : " + nama +
18                           "\nWarna Rambut : " + rambut +
19                           "\nUmur : " + umur +
20                           "\nAlamat : " + alamat +
21                           "\nPekerjaan : " + pekerjaan);
22    }
23
24    // Method perilaku
25    public void makan() {
26        System.out.println(nama + " sedang bermain.");
27    }
28
29    public void tidur() {
30        System.out.println(nama + " sedang olahraga.");
31    }
32
33    public void berinteraksi() {
34        System.out.println(nama + " sedang berinteraksi dengan teman-teman.");
35    }
36
37    public void belajar() {
```

```
Online Java Compiler IDE

19         "\nUmur : " + umur +
20         "\nAlamat : " + alamat +
21         "\nPekerjaan : " + pekerjaan);
22    }
23
24    // Method perilaku
25    public void makan() {
26        System.out.println(nama + " sedang bermain.");
27    }
28
29    public void tidur() {
30        System.out.println(nama + " sedang olahraga.");
31    }
32
33    public void berinteraksi() {
34        System.out.println(nama + " sedang berinteraksi dengan teman-teman.");
35    }
36
37    public void belajar() {
38        System.out.println(nama + " sedang belajar java.");
39    }
40
41    public void menggunakanTeknologi() {
42        System.out.println(nama + " sedang menggunakan smartphone.");
43    }
44
45    // Deklarasi method utama
46    public static void main(String[] args) {
47        Manusia satu = new Manusia("Putri", "hitam", 25, "Jl. Raya No. 123", "Mahasiswa");
48
49        satu.makan(); // Menampilkan perilaku bermain
50        satu.tidur(); // Menampilkan perilaku olahraga
51        satu.berinteraksi(); // Menampilkan perilaku berinteraksi
52        satu.belajar(); // Menampilkan perilaku belajar java
53        satu.menggunakanTeknologi(); // Menampilkan perilaku menggunakan teknologi
54    }
55 }
```

```
Nama saya : Putri
Warna Rambut : hitam
Umur : 25
Alamat : Jl. Raya No. 123
Pekerjaan : Mahasiswa
Putri sedang bermain.
Putri sedang olahraga.
Putri sedang berinteraksi dengan teman-teman.
Putri sedang belajar java.
Putri sedang menggunakan smartphone.
|
```

[No.1] Kesimpulan

1) Analisa

a) Kesimpulan Berdasarkan Permasalahan, Algoritma, dan Kode Program

Kode program yang diberikan mengalami kesalahan kompilasi karena nama konstruktor tidak sesuai dengan nama kelas. Dalam Java, nama konstruktor harus Konstruktor hanya memiliki satu parameter, tetapi dalam metode main, dua parameter diharapkan saat membuat objek dari kelas tersebut. Periksa kode untuk menemukan kesalahan kompilasi. Temukan penyebab kesalahan, yaitu nama konstruktor dan jumlah parameter yang tidak sesuai. Ubah nama konstruktor dari Manusia1 menjadi Manusia. Sesuaikan parameter konstruktor untuk menerima dua argumen: String nama dan String rambut. Gunakan kata kunci this untuk menginisialisasi atribut kelas dengan nilai parameter yang diterima. Jalankan kode setelah perbaikan untuk memastikan tidak ada kesalahan kompilasi dan program berjalan sesuai harapan.

b) Dasar Alasan Pengambilan Keputusan untuk Kasus Ini

Dalam pemrograman Java, sangat penting untuk mengikuti aturan sintaksis yang berlaku. Salah satu aturan utama adalah bahwa nama konstruktor harus sama dengan nama kelas. Dengan memperbaiki nama konstruktor, kita memastikan bahwa kode mematuhi aturan ini. Mengubah jumlah dan jenis parameter pada konstruktor agar sesuai dengan cara objek dibuat di dalam metode main adalah langkah penting untuk memastikan bahwa objek dapat diinisialisasi dengan benar. Ini membantu dalam menghindari kesalahan runtime dan menjamin bahwa semua atribut memiliki nilai yang valid saat objek dibuat. Dalam konteks pemrograman berorientasi objek, penting untuk mendefinisikan atribut dan perilaku (metode) yang relevan untuk sebuah kelas. Dengan menambahkan atribut seperti nama dan rambut, serta memberikan perilaku melalui metode, kita menciptakan representasi yang lebih realistis dari objek tersebut. Menguji kode setelah perbaikan adalah langkah penting dalam proses pengembangan perangkat lunak. Ini memastikan bahwa perbaikan yang dilakukan efektif dan bahwa program berfungsi sesuai harapan tanpa menghasilkan kesalahan.

Refleksi

Pengalaman belajar dalam kelas ini sangat berharga karena memberikan saya pemahaman yang lebih dalam tentang konsep dasar pemrograman berorientasi objek (OOP) di Java. Saya telah belajar tentang cara mendeklarasikan kelas, objek, dan method dalam Java, serta bagaimana mengintegrasikan atribut dan perilaku dalam sebuah kelas.

Latihan 2

Nama & NPM	Topik:	Tanggal:
Dini Ramadona G1F024050	Kelas,Objek dan Method	17 September 2024

[No.2] Identifikasi Masalah:

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!

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");  
    }  
}
```

Pada soal diatas diminta untuk menganalisis penyebab kesalahan tersebut dan memperbaiki kode agar dapat dikompilasi dan dijalankan tanpa masalah, kita juga diminta untuk menambahkan fitur baru ke dalam kelas Ortu dengan menambahkan atribut baru (umur dan jenis kelamin) dan merekomendasikan konstruktor baru yang dapat menerima parameter ini.

[No.2] Analisis dan Argumentasi

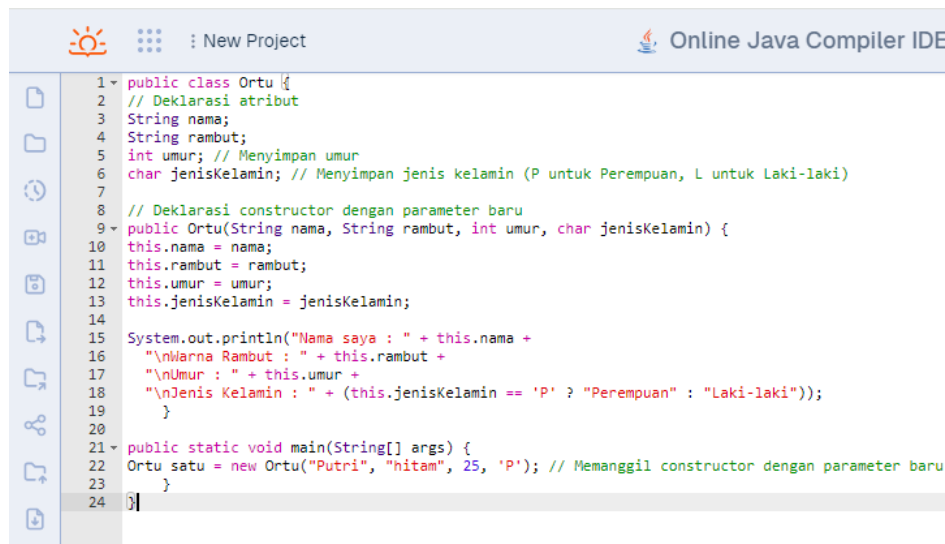
- 1) Saya mengusulkan permasalahan ini dapat diatasi dengan cara ubah nama konstruktor menjadi Ortu agar sesuai dengan nama kelas.Tambahkan parameter ke dalam konstruktor untuk menerima nilai nama dan rambut.Deklarasikan atribut nama dan rambut sebagai variabel instance dalam kelas.Perbaiki kode main untuk memanggil konstruktor dengan dua parameter yang sesuai.
- 2) Permasalahan nama konstruktor yang salah (ortu bukan Ortu) menyebabkan kesalahan kompilasi karena Java tidak menemukan konstruktor yang sesuai dengan nama yang diberikan.Kurangnya parameter dalam konstruktor menyebabkan kesalahan saat mencoba memanggil konstruktor dengan dua parameter.Menggunakan kata kunci this untuk menginisialisasi atribut kelas dengan nilai parameter yang diterima memastikan bahwa atribut dapat diakses dan diinisialisasi dengan benar.Mengubah kode main untuk memanggil konstruktor dengan dua parameter yang sesuai ("Putri" dan "hitam") memastikan bahwa konstruktor dapat dijalankan dengan benar dan menghasilkan output yang diharapkan.
Dengan demikian, rancangan solusi yang diusulkan telah memperbaiki kesalahan kompilasi dan memastikan bahwa program dapat berjalan dengan benar.

[No.2] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
 - (a) Membuka aplikasi eclipse
 - (b) Salin kode yang ada di E-learning lalu paste ke eclipse
 - (c) Mencari Penyebab Kesalahan kode
 - (d) Mencari tutorial di youtube Rumah Raflesia
 - (e) Mencari tipe data yang cocok lalu perbaiki kesalahan
 - (f) Ubah nama konstruktor menjadi Manusia dan tambahkan parameter yang sesuai.
 - (g) Ubah nama konstruktor menjadi Ortu agar sesuai dengan nama kelas
 - (h) Tambahkan parameter ke dalam konstruktor untuk menerima nilai nama dan rambut.

- (i) Deklarasikan atribut nama dan rambut sebagai variabel instance dalam kelas.
- (j) Perbaiki kode main untuk memanggil konstruktor dengan dua parameter yang sesuai.
- (k) Periksa hasil output untuk memastikan bahwa data ditampilkan dengan benar sesuai dengan tipe data yang digunakan
- (l) Selesai.

2) Tuliskan kode program dan luaran



```

1 public class Ortu {
2     // Deklarasi atribut
3     String nama;
4     String rambut;
5     int umur; // Menyimpan umur
6     char jenisKelamin; // Menyimpan jenis kelamin (P untuk Perempuan, L untuk Laki-laki)
7
8     // Deklarasi constructor dengan parameter baru
9     public Ortu(String nama, String rambut, int umur, char jenisKelamin) {
10        this.nama = nama;
11        this.ambut = rambut;
12        this.umur = umur;
13        this.jenisKelamin = jenisKelamin;
14
15        System.out.println("Nama saya : " + this.nama +
16            "\nWarna Rambut : " + this.ambut +
17            "\nUmur : " + this.umur +
18            "\nJenis Kelamin : " + (this.jenisKelamin == 'P' ? "Perempuan" : "Laki-laki"));
19    }
20
21    public static void main(String[] args) {
22        Ortu satu = new Ortu("Putri", "hitam", 25, 'P'); // Memanggil constructor dengan parameter baru
23    }
24 }
  
```

```

Nama saya : Putri
Warna Rambut : hitam
Umur : 25
Jenis Kelamin : Perempuan
  
```

[No.2] Kesimpulan

- 1) Evaluasi
 - a) Konsekuensi dari Skenario Pemrograman Ini

Jika kode tidak diperbaiki, kesalahan kompilasi akan menghalangi program dari eksekusi. Ini menunjukkan pentingnya mengikuti aturan sintaksis dalam pemrograman, seperti kesesuaian nama konstruktor dengan nama kelas. Skenario ini memberikan kesempatan untuk memahami konsep dasar pemrograman berorientasi objek (OOP), seperti kelas, objek, konstruktor, dan atribut. Memperbaiki kesalahan dalam kode membantu memperkuat pemahaman tentang bagaimana elemen-elemen ini berinteraksi. Proses menganalisis dan memperbaiki kesalahan dalam kode meningkatkan keterampilan debugging. Kemampuan untuk menemukan dan memperbaiki kesalahan menjadi keterampilan yang sangat berharga dalam pengembangan perangkat lunak. Dengan menambahkan parameter baru (umur dan jenis kelamin), program menjadi lebih informatif dan fungsional. Ini menunjukkan bagaimana pengembangan perangkat lunak dapat beradaptasi dengan kebutuhan baru. Meskipun tidak dibahas dalam skenario ini, penting untuk mempertimbangkan validasi input saat menerima data dari pengguna. Hal ini dapat mencegah kesalahan di kemudian hari dan meningkatkan keandalan program.
 - b) Evaluasi Input, Proses, dan Luaran yang Dihasilkan

Input yang diterima oleh konstruktor adalah dua string (nama dan rambut) dalam versi awal kode, dan setelah perbaikan, ditambahkan dua parameter baru (umur dan jenisKelamin).
 Contoh input:
 Nama: "Putri"
 Warna Rambut: "hitam"

Umur: 25

Jenis Kelamin: 'P'

Ketika objek Ortu dibuat, konstruktor dipanggil dengan parameter yang diberikan. Atribut nama, rambut, umur, dan jenisKelamin diinisialisasi menggunakan nilai parameter. Informasi tentang objek dicetak ke konsol menggunakan `System.out.println()`. Luaran dari program adalah output yang dicetak ke konsol yang memberikan informasi tentang objek Ortu yang baru dibuat. Luaran dari program adalah output yang dicetak ke konsol yang memberikan informasi tentang objek Ortu yang baru dibuat.

Refleksi

Pengalaman belajar dalam kelas ini sangat berharga karena memberikan saya pemahaman yang lebih dalam tentang konsep dasar pemrograman berorientasi objek (OOP) di Java. Saya telah belajar tentang cara mendeklarasikan kelas, objek, dan method dalam Java, serta bagaimana mengintegrasikan atribut dan perilaku dalam sebuah kelas

Latihan 3

Nama & NPM	Topik:	Tanggal:
Dini Ramadona G1F024050	Kelas,Objek dan Method	17 September 2024

[No.3] Identifikasi Masalah:

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

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 {  
        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");  
    }  
}
```

Soal diatas membahas tentang analisis dan perbaikan kode program Java yang mengalami kesalahan kompilasi serta memperkenalkan konsep konstruktor overloading dan overriding, method overloading dan overriding, serta perbedaan antara method yang mengembalikan nilai dan tidak mengembalikan nilai.

[No.3] Analisis dan Argumentasi

- 1) Saya mengusulkan permasalahan ini dapat diatas dengan cara ubah nama konstruktor dari Manusia1 menjadi Manusia agar sesuai dengan nama kelas. Ini adalah langkah penting untuk memastikan bahwa Java dapat mengenali konstruktor ketika objek dari kelas tersebut dibuat.Tambahkan atribut yang diperlukan dalam kelas, seperti film, untuk menyimpan informasi tentang film yang disukai.Ubah metode sukaNonton menjadi dua metode terpisah.Satu metode dengan parameter int episode dan int durasi untuk menghitung total jam menonton berdasarkan jumlah episode dan durasi per episode.Implementasikan semua perubahan di atas dalam kode program, sehingga kode dapat dikompilasi dan dijalankan tanpa kesalahan.
- 2) Dengan mengubah nama konstruktor menjadi Manusia, solusi ini langsung menyelesaikan masalah kesalahan kompilasi yang muncul karena ketidakcocokan **nama**. Ini adalah langkah

[No.3] Penyusunan Algoritma dan Kode Program

2) Kode program dan luaran

Nama saya : Dini Ramadana
Warna Rambut : hitam
Hobi Menonton : Drakor
Jam nonton = 4 jam

[No.3] Kesimpulan

a) Constructor Overloading dan Overriding

Constructor Overloading:

Merupakan kemampuan untuk membuat beberapa konstruktor dengan parameter yang berbeda dalam satu kelas.

Contoh: `public Manusia(String nama, String rambut)` dan `public Manusia(String nama, String rambut, int umur)`.

Constructor Overriding:

Merupakan kemampuan untuk memberikan implementasi yang berbeda dari konstruktor yang sudah ada di kelas induk.

Contoh: Dalam kelas anak, Anda dapat membuat konstruktor yang sama dengan kelas induk tetapi dengan implementasi yang berbeda.

b) Method Overloading dan Overriding

Method Overloading:

Merupakan kemampuan untuk membuat beberapa metode dengan nama yang sama tetapi dengan parameter yang berbeda dalam satu kelas.

Contoh: `void sukaNonton(String film)` dan `int sukaNonton(int episode, int durasi)`.

Method Overriding:

Merupakan kemampuan untuk memberikan implementasi yang berbeda dari metode yang sudah ada di kelas induk.

Contoh: Dalam kelas anak, Anda dapat membuat metode yang sama dengan kelas induk tetapi dengan implementasi yang berbeda.

c) Method yang Mengembalikan Nilai dan Method Tidak Mengembalikan Nilai

Method yang Mengembalikan Nilai:

Merupakan metode yang memiliki return type dan mengembalikan nilai setelah dieksekusi.

Contoh: `int sukaNonton(int episode, int durasi)`.

Method Tidak Mengembalikan Nilai:

Merupakan metode yang tidak memiliki return type dan tidak mengembalikan nilai setelah dieksekusi.

Contoh: `void sukaNonton(String film)`.

a) Kesimpulan Berdasarkan Permasalahan, Algoritma, dan Kode Program

Nama konstruktor `Manusia1` tidak sesuai dengan nama kelas `Manusia`. Metode `sukaNonton` tidak dideklarasikan dengan benar, baik dari segi parameter maupun tipe pengembalian. Ada duplikasi dalam nama metode `sukaNonton`, yang menyebabkan kebingungan dalam pemanggilan. Menganalisis kode untuk menemukan kesalahan kompilasi dan sintaksis. Menentukan penyebab kesalahan, seperti ketidakcocokan nama konstruktor dan metode. Mengubah nama konstruktor menjadi `Manusia` agar sesuai dengan nama kelas. Satu metode untuk mencetak hobi menonton berdasarkan film. Satu metode untuk menghitung jam nonton berdasarkan episode dan durasi. Mengimplementasikan perubahan dalam kode program. Menguji kode di dalam metode `main` untuk memastikan bahwa semua perubahan berfungsi dengan baik dan menghasilkan output yang diharapkan.

1) Analisa

b) Dasar Alasan Pengambilan Keputusan untuk Kasus Ini

Nama konstruktor harus sama dengan nama kelas untuk memastikan bahwa Java dapat mengenali konstruktor saat objek dibuat. Mengubah nama konstruktor menjadi `Manusia` adalah langkah krusial untuk menghindari kesalahan kompilasi. Memisahkan perilaku dalam dua metode `sukaNonton` mengikuti prinsip OOP yang baik. Setiap metode memiliki tujuan yang jelas dan spesifik, sehingga meningkatkan keterbacaan dan pemeliharaan kode. Dengan menambahkan atribut dan memperbaiki metode, kelas `Manusia` menjadi lebih informatif dan fungsional. Ini memberikan pengguna fleksibilitas lebih dalam menggunakan kelas sesuai kebutuhan mereka. Menguji kode setelah perbaikan adalah

langkah penting dalam pengembangan perangkat lunak. Hal ini memastikan bahwa semua perubahan yang dilakukan efektif dan bahwa program berjalan sesuai harapan tanpa menghasilkan kesalahan. Proses menganalisis kesalahan dan memperbaiki kode membantu meningkatkan keterampilan debugging, yang merupakan keterampilan penting bagi seorang programmer.

Refleksi

Pengalaman belajar dalam kelas ini sangat berharga karena memberikan saya pemahaman yang lebih dalam tentang konsep dasar pemrograman berorientasi objek (OOP) di Java. Saya telah belajar tentang cara mendeklarasikan kelas, objek, dan method dalam Java, serta bagaimana mengintegrasikan atribut dan perilaku dalam sebuah kelas

Latihan 4

Nama & NPM	Topik:	Tanggal:
Dini Ramadona G1F024050	Kelas,Objek dan Method	17 September 2024

[No.4] Identifikasi Masalah:

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!).

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
}
```

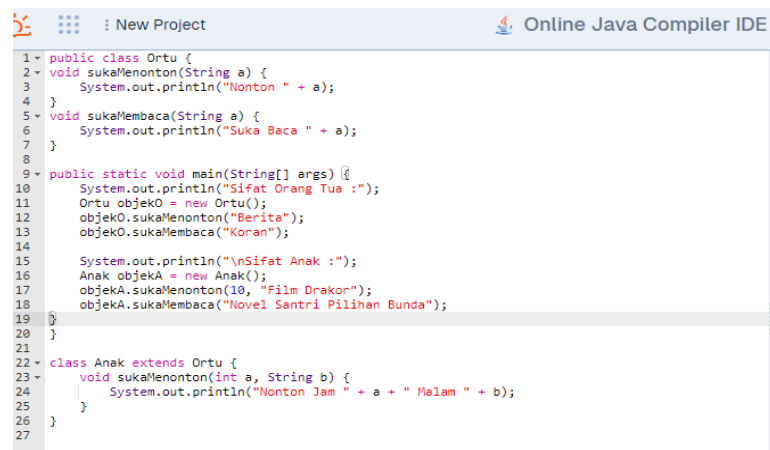
Soal di atas meminta untuk melakukan beberapa analisis dan evaluasi terkait kode program Java yang menggunakan konsep pewarisan (inheritance) dalam pemrograman berorientasi objek.

[No.4] Analisis dan Argumentasi

- 1) Rekomendasi Perbaikan untuk Meningkatkan Efisiensi Waktu Eksekusi.
yaitu dengan menghapus metode yang tidak perlu atau mengganti metode yang sama dengan menggunakan super, Anda dapat mengurangi kompleksitas dan meningkatkan efisiensi. Pastikan bahwa setiap metode memiliki logika yang diperlukan dan tidak ada redundansi dalam kode. Pastikan bahwa parameter yang digunakan dalam metode adalah yang paling relevan dan membantu dalam menjaga performa program. Dengan menerapkan rekomendasi tersebut, Anda dapat meningkatkan efisiensi kode program dan mengurangi waktu eksekusi secara keseluruhan.
- 2) Saya juga mengusulkan permasalahan ini dapat diatasi dengan cara method sukaMembaca dari kelas Anak jika tidak memiliki logika tambahan. Dengan demikian, akan mengurangi duplikasi dan menjaga kode tetap bersih. Jika ingin memanggil method induk dari kelas anak, menggunakan kata kunci super dapat meningkatkan kejelasan kode dan menunjukkan bahwa method tersebut berasal dari kelas induk. Dengan mempertahankan method sukaMenonton di kelas Anak, Anda memanfaatkan polymorphism, di mana metode yang sama dapat memiliki implementasi berbeda tergantung pada objek yang memanggilnya. Mengoptimalkan kode dengan menghindari duplikasi dan menjaga hanya metode yang relevan akan membantu dalam mengurangi waktu eksekusi secara keseluruhan.
Dengan menghapus method yang tidak perlu dan menghindari duplikasi, solusi ini langsung mengatasi masalah efisiensi dalam penulisan kode. Kode yang lebih bersih dan terstruktur dengan baik akan lebih mudah dibaca dan dipelihara.

[No.4] Penyusunan Algoritma dan Kode Program

- 1) Algoritma
 - (a) Membuka aplikasi eclipse
 - (b) Salin kode yang ada di E-learning lalu paste ke eclipse
 - (c) Mencari Penyebab Kesalahan kode
 - (d) Mencari tutorial di youtube Rumah Raflesia
 - (e) Mencari tipe data yang cocok lalu perbaiki kesalahan
 - (f) Deklarasi Kelas Induk (Ortu)
 - (g) Deklarasi Kelas Anak (Anak)
 - (h) Deklarasi Method Utama (main)
 - (i) Periksa hasil output untuk memastikan bahwa data ditampilkan dengan benar sesuai dengan tipe data yang digunakan
 - (j) Selesai.
- 2) Tuliskan kode program dan luaran



```
1 public class Ortu {
2     void sukaMenonton(String a) {
3         System.out.println("Nonton " + a);
4     }
5     void sukaMembaca(String a) {
6         System.out.println("Suka Baca " + a);
7     }
8 }
9 public static void main(String[] args) {
10     System.out.println("Sifat Orang Tua :");
11     Ortu objekO = new Ortu();
12     objekO.sukaMenonton("Berita");
13     objekO.sukaMembaca("Koran");
14
15     System.out.println("\nSifat Anak :");
16     Anak objekA = new Anak();
17     objekA.sukaMenonton(10, "Film Drakor");
18     objekA.sukaMembaca("Novel Santri Pilihan Bunda");
19 }
20
21
22 class Anak extends Ortu {
23     void sukaMenonton(int a, String b) {
24         System.out.println("Nonton Jam " + a + " Malam " + b);
25     }
26 }
27
```

```
Sifat Orang Tua :  
Nonton Berita  
Suka Baca Koran  
  
Sifat Anak :  
Nonton Jam 10 Malam Film Drakor  
Suka Baca Novel Santri Pilihan Bunda
```

[No.4] Kesimpulan

1) Kreasi

a) Pengetahuan Baru dan Konsep Baru sebagai Usulan Solusi

Pengetahuan Baru:

Dari analisis kode program yang menggunakan konsep pewarisan dalam pemrograman berorientasi objek, beberapa pengetahuan baru yang dapat dikembangkan meliputi:

Pewarisan dan Polymorphism:

Memahami bagaimana kelas anak dapat mewarisi sifat dari kelas induk dan bagaimana method dapat di-overriding untuk memberikan perilaku spesifik.

Mengetahui cara menggunakan method overloading untuk memberikan beberapa implementasi dari method yang sama dengan parameter berbeda.

Efisiensi Kode:

Memahami pentingnya menghindari duplikasi kode dan menjaga kode tetap bersih dan terstruktur untuk meningkatkan keterbacaan dan pemeliharaan.

Penggunaan Kata Kunci super:

Mempelajari cara menggunakan kata kunci super untuk memanggil method dari kelas induk, yang membantu dalam memperjelas asal method yang dipanggil.

Konsep Baru sebagai Usulan Solusi:

Mengusulkan pendekatan untuk mengurangi duplikasi dalam kode dengan hanya mendeklarasikan method yang diperlukan, serta menggunakan inheritance secara lebih efisien.

Mengimplementasikan prinsip-prinsip desain perangkat lunak yang baik, seperti DRY (Don't Repeat Yourself), untuk meningkatkan kualitas kode.

b) Konstruksi Hubungan antara Variabel yang Berbeda dengan Konsep yang Diketahui

Variabel dalam Kode:

Kelas Ortu:

sukaMenonton(String a): Method untuk menampilkan aktivitas menonton.

sukaMembaca(String a): Method untuk menampilkan aktivitas membaca.

Kelas Anak:

sukaMenonton(int a, String b): Method untuk menampilkan aktivitas menonton dengan detail waktu.

sukaMenonton(String a): Overriding method dari kelas induk.

sukaMembaca(String a): Overriding method dari kelas induk.

Hubungan antara Variabel dan Konsep:

Inheritance (Pewarisan):

Kelas Anak mewarisi method dari kelas Ortu, memungkinkan penggunaan kembali kode tanpa perlu mendeklarasikan ulang.

Polymorphism:

Dengan mendeklarasikan method dengan nama yang sama di kelas anak (overriding), Anda dapat memanggil method yang sesuai berdasarkan objek yang digunakan (objek dari kelas Anak atau Ortu).

Overloading:

Method sukaMenonton di kelas Anak memiliki dua versi (satu dengan parameter string dan satu dengan parameter int dan string), menunjukkan kemampuan Java untuk membedakan method berdasarkan jumlah dan tipe parameter.

Efisiensi Kode:

Dengan mengurangi redundansi dalam deklarasi method, kode menjadi lebih efisien dan mudah dipahami, menciptakan hubungan antara variabel dan prinsip desain perangkat lunak yang baik.

Refleksi

Pengalaman belajar dalam kelas ini sangat berharga karena memberikan saya pemahaman yang lebih dalam tentang konsep dasar pemrograman berorientasi objek (OOP) di Java. Saya telah belajar tentang cara mendeklarasikan kelas, objek, dan method dalam Java, serta bagaimana mengintegrasikan atribut dan perilaku dalam sebuah kelas