|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariiq Ashar Sofyan**  **G1F024052** | **Class** | **28 september 2024** |
| **[Nomor 1] Identifikasi Masalah:** | | |
| **Contoh 1:**  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! | | |
| **[Nomor 1] Analisis dan Argumentasi** | | |
| * 1. **Perbaiki pesan kesalahan Contoh 1!**   Dalam kode yang diberikan, ada beberapa kesalahan:   * Nama kelas yang dideklarasikan adalah Manusia, tetapi pada constructor dan pemanggilannya di dalam method main, nama kelasnya disebut sebagai Manusia1. Harus ada keselarasan antara nama kelas dan constructor. * Constructor yang dideklarasikan hanya menerima satu parameter (String nama), tetapi di method main, Anda mencoba memanggilnya dengan dua argumen ("Putri", "hitam"). Constructor perlu diperbarui agar dapat menerima dua parameter.   Berikut contoh perbaikannya :  public class Manusia { // deklarasi kelas  // Deklarasi atribut Manusia dalam variabel  String nama, rambut;    // Deklarasi constructor  public Manusia(String nama, String rambut) {  this.nama = nama; // Mengisi variabel nama  this.rambut = rambut; // Mengisi variabel rambut  System.out.println("Nama saya : " + nama +  "\nWarna Rambut : " + rambut);  }    // Deklarasi method utama  public static void main(String[] args) {  Manusia satu = new Manusia("Putri", "hitam");  }  }  Ganti nama constructor dari Manusia1 menjadi Manusia agar sesuai dengan nama kelas.  Constructor sekarang menerima dua parameter (String nama, String rambut) sehingga cocok dengan pemanggilan dalam method main.  **1.2. Analisa ciri-ciri lain Kelas Manusia yang dapat menjadi**  **a. Atribut variabel yang bisa ditambahkan**  Atribut-atribut lain yang dapat menjadi ciri khas dari Manusia termasuk:   * umur: Menyimpan informasi usia. * jenisKelamin: Menyimpan informasi tentang gender. * tinggiBadan: Menyimpan tinggi badan dalam satuan cm atau meter. * beratBadan: Menyimpan berat badan dalam kg.   contoh deklarasi atribut :  int umur;  String jenisKelamin;  double tinggiBadan, beratBadan;  **b. Perilaku/ behavior yang bisa ditambahkan**  Perilaku atau method yang relevan untuk manusia mungkin termasuk:   * berbicara(): Method yang menggambarkan manusia sedang berbicara. * berjalan(): Method untuk menggambarkan aksi berjalan. * makan(): Method untuk menunjukkan aktivitas makan. * perkenalkanDiri(): Method untuk memperkenalkan diri, menampilkan nama, umur, dan atribut lain.   Contoh deklarasi atribut :  public void berbicara(String kata) {  System.out.println(nama + " berkata: " + kata);  }  public void berjalan() {  System.out.println(nama + " sedang berjalan.");  } | | |
| **[Nomor Soal] Penyusunan Algoritma dan Kode Program** | | |
| * 1. **Perbaiki pesan kesalahan Contoh 1!**   Algoritma   1. Mulai 2. Deklarasikan kelas Manusia dengan atribut nama dan rambut. 3. Buat constructor yang menerima dua parameter nama dan rambut. 4. Inisialisasi atribut nama dan rambut di dalam constructor. 5. Cetak informasi nama dan warna rambut di dalam constructor. 6. Dalam method main(), buat objek Manusia dengan memberikan nilai untuk nama dan rambut. 7. Akhiri program.     Input & Output  **1.2. Analisa ciri-ciri lain Kelas Manusia yang dapat menjadi**  **a. atribut variabel, dan**  **b. perilaku/ behavior!**  Algoritma   1. Tambahkan atribut lain yang relevan seperti umur, jenisKelamin, tinggiBadan, dan beratBadan. 2. Buat method untuk perilaku manusia, seperti:  * perkenalkanDiri() untuk mencetak atribut nama, umur, dll. * berbicara() untuk mencetak ucapan dari manusia. * berjalan() untuk menampilkan manusia sedang berjalan.  1. Dalam method main(), buat objek Manusia dengan memberikan nilai untuk semua atribut. 2. Panggil method perkenalkanDiri(), berbicara(), dan berjalan() pada objek manusia. 3. Akhiri program.     Input    Output | | |
| **[Nomor 1] Kesimpulan** | | |
| 1. Analisa   **Soal 1.1** berfokus pada perbaikan kesalahan dalam kode:   * **Masalah utama** adalah constructor yang tidak sesuai dengan nama kelas dan jumlah parameter yang tidak cocok. * **Solusi**: Nama constructor harus sama dengan nama kelas, dan jumlah parameter harus sesuai dengan yang diberikan saat pembuatan objek. Setelah diperbaiki, program dapat dijalankan dengan benar, mencetak nama dan warna rambut sesuai input.   **Soal 1.2** menambahkan atribut dan perilaku untuk kelas Manusia:   * **Atribut tambahan** seperti umur, jenisKelamin, tinggiBadan, dan beratBadan memperkaya informasi yang dapat disimpan oleh objek manusia. * **Perilaku tambahan** seperti method perkenalkanDiri(), berbicara(), dan berjalan() memberikan kemampuan untuk berinteraksi dan menampilkan informasi tentang objek manusia. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariiq Ashar Sofyan**  **G1F024052** | **Objek** | **28 september 2024** |
| **[Nomor 1] Identifikasi Masalah:** | | |
| **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! | | |
| **[Nomor 1] Analisis dan Argumentasi** | | |
| **2.1. Evaluasi penyebab kesalahan dan perbaiki kode tersebut!**  Dalam kode yang diberikan Kesalahan terjadi karena constructor yang seharusnya menerima parameter String nama dan String rambut tidak dideklarasikan dengan benar. Constructor dalam kode tersebut tidak memiliki parameter apapun, padahal di dalam method main, sebuah object Ortu dibuat dengan dua parameter ("Putri", "hitam"). Untuk memperbaikinya, constructor perlu didefinisikan dengan parameter yang sesuai.Berikut contoh perbaikannya :  public class Ortu {  // Variabel instance untuk menyimpan nama dan warna rambut  String nama;  String rambut;  // Deklarasi constructor dengan dua parameter  public Ortu(String nama, String rambut) {  // Inisialisasi variabel instance  this.nama = nama;  this.rambut = rambut;  // Menampilkan informasi  System.out.println("Nama saya: " + nama +  "\nWarna Rambut: " + rambut);  }  public static void main(String[] args) {  Ortu satu = new Ortu("Putri", "hitam");  }  }  **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!**  Untuk menambahkan variabel umur dan jenis kelamin, kita harus memperbarui constructor agar menerima dua parameter tambahan, yaitu int umur dan char jenisKelamin. Berikut adalah versi yang diperbarui:Contoh :  public class Ortu {  // Variabel instance untuk menyimpan nama, rambut, umur, dan jenis kelamin  String nama;  String rambut;  int umur;  char jenisKelamin;  // Deklarasi constructor dengan empat parameter  public Ortu(String nama, String rambut, int umur, char jenisKelamin) {  // Inisialisasi variabel instance  this.nama = nama;  this.rambut = rambut;  this.umur = umur;  this.jenisKelamin = jenisKelamin;  // Menampilkan informasi  System.out.println("Nama saya: " + nama +  "\nWarna Rambut: " + rambut +  "\nUmur: " + umur +  "\nJenis Kelamin: " + jenisKelamin);  }  public static void main(String[] args) {  Ortu satu = new Ortu("Putri", "hitam", 25, 'P');  }  }  Dengan perubahan ini, program akan menerima dan menampilkan informasi tambahan tentang umur dan jenis kelamin. | | |
| **[Nomor Soal] Penyusunan Algoritma dan Kode Program** | | |
| **2.1. Evaluasi penyebab kesalahan dan perbaiki kode tersebut!**  Algoritma:   1. **Mulai Program**. 2. Deklarasikan kelas Ortu dengan variabel instance: nama dan rambut. 3. Buat constructor yang menerima dua parameter: String nama dan String rambut.  * Inisialisasi variabel instance dengan nilai parameter yang diterima. * Tampilkan informasi: nama dan rambut.  1. Dalam method main, buat objek dari kelas Ortu dengan mengirimkan argumen untuk nama dan rambut. 2. **Akhiri Program**     Input & Output  **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!**  Algoritma   1. **Mulai Program**. 2. Deklarasikan kelas Ortu dengan variabel instance: nama, rambut, umur, dan jenisKelamin. 3. Buat constructor yang menerima empat parameter: String nama, String rambut, int umur, dan char jenisKelamin. 4. Inisialisasi variabel instance dengan nilai parameter yang diterima. 5. Tampilkan informasi: nama, rambut, umur, dan jenisKelamin. 6. Dalam method main, buat objek dari kelas Ortu dengan mengirimkan argumen sesuai parameter constructor: nama, rambut, umur, dan jenisKelamin. 7. **Akhiri Program**.   input    Output | | |
| **[Nomor 1] Kesimpulan** | | |
| 1. Analisa   2.1. Memperbaiki Kesalahan Kode:   * Awalnya, program mengalami error karena constructor tidak dibuat dengan benar. Constructor adalah metode khusus yang digunakan untuk mengatur nilai awal objek saat dibuat. * Setelah kita menambahkan parameter yang benar (seperti nama dan rambut), error hilang, dan program bisa berjalan dengan baik.   2.2. Menambahkan Variabel Lain:   * Setelah memperbaiki constructor, kita menambahkan lebih banyak informasi pada objek, seperti umur dan jenisKelamin. Untuk itu, kita menambah dua parameter lagi pada constructor. * Dengan tambahan ini, kita bisa membuat objek yang menyimpan informasi lebih lengkap. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Ariiq Ashar Sofyan**  **G1F024052** | **Method** | **28 september 2024** |
| **[Nomor 3] Identifikasi Masalah:** | | |
| **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");     } }  **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 | | |
| **[Nomor 3] Analisis dan Argumentasi** | | |
| **3.1. Evaluasi penyebab kesalahan dan perbaiki kode tersebut!**  Masalah pada constructor: Nama constructor harus sama dengan nama kelas. Dalam kode, constructor dinamakan Manusia1, tetapi nama kelas adalah Manusia. Seharusnya, nama constructor adalah Manusia.  Method sukaNonton: Ada dua masalah di sini:   * Method pertama sukaNonton yang tidak menerima parameter tidak memiliki tanda kurung kosong. * Method kedua sukaNonton bertipe int harus menerima parameter dan mengembalikan nilai, tapi ada kesalahan di logika perhitungan.   Perbaikan kode :  public class Manusia {  // Deklarasi atribut Manusia  String nama, rambut;    // Deklarasi constructor  public Manusia(String nama, String rambut) {  System.out.println("Nama saya: " + nama +  "\nWarna Rambut: " + rambut);  }    // Method yang tidak mengembalikan nilai  void sukaNonton(String film) {  System.out.println("Hobi Menonton: " + film);  }    // Method yang mengembalikan nilai  int sukaNonton(int episode, int durasi) {  return episode \* durasi;  }    // 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");  }  }  **3.2. Ubahlah method dan constructor Contoh 3 sesuai dengan perilaku/ behavior anda**  Misalnya, jika ingin membuat constructor dan method lebih fleksibel:   * Tambahkan constructor overloading agar bisa membuat objek tanpa warna rambut. * Tambahkan method sukaNonton untuk menghitung jam dari berbagai jenis acara.   Contoh :  public class Manusia {  String nama, rambut;    // Constructor 1 (dengan warna rambut)  public Manusia(String nama, String rambut) {  this.nama = nama;  this.rambut = rambut;  System.out.println("Nama saya: " + nama +  "\nWarna Rambut: " + rambut);  }    // Constructor 2 (tanpa warna rambut)  public Manusia(String nama) {  this.nama = nama;  System.out.println("Nama saya: " + nama);  }    // Method yang tidak mengembalikan nilai  void sukaNonton(String film) {  System.out.println("Hobi Menonton: " + film);  }    // Method yang mengembalikan nilai  int sukaNonton(int episode, int durasi) {  return episode \* durasi;  }    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");    // Objek tanpa warna rambut  Manusia dua = new Manusia("Andi");  dua.sukaNonton("Anime");  int totalJam = dua.sukaNonton(3, 1);  System.out.println("Jam nonton = " + totalJam + " jam");  }  }  **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**   1. **Constructor Overloading vs Constructor Overriding**:   **Overloading**: Membuat beberapa constructor dengan nama yang sama tetapi parameter yang berbeda.  **Overriding**: Tidak berlaku pada constructor karena constructor tidak dapat diwarisi. Overriding hanya berlaku untuk method dalam pewarisan.   1. **Method Overloading vs Method Overriding**:   **Overloading**: Membuat beberapa method dengan nama yang sama tetapi parameter yang berbeda.  **Overriding**: Mengubah implementasi method dari superclass dalam subclass, biasanya dalam pewarisan (inheritance).   1. **Method Mengembalikan Nilai vs Tidak Mengembalikan Nilai**:   **Mengembalikan Nilai**: Method mengembalikan hasil menggunakan keyword return dan memiliki tipe data yang ditentukan (int, String, dll.).  **Tidak Mengembalikan Nilai**: Method bertipe void dan hanya melakukan tindakan tanpa mengembalikan apapun. | | |
| **[Nomor 3l] Penyusunan Algoritma dan Kode Program** | | |
| **3.1 Evaluasi Penyebab Kesalahan dan Perbaikan Kode**  Algoritma:   1. Deklarasikan kelas Manusia dengan atribut nama dan rambut. 2. Buat constructor untuk menginisialisasi nama dan rambut. 3. Buat method sukaNonton:   Method pertama menampilkan jenis film yang disukai.  Method kedua menghitung waktu menonton berdasarkan episode dan durasi.   1. Inisialisasi objek Manusia di method main. 2. Panggil method sukaNonton untuk menampilkan hobi dan menghitung total jam nonton. 3. Cetak hasil waktu menonton.       Input & Output  **3.2 Mengubah Constructor dan Method Sesuai Behavior Anda**  Algoritma   1. Deklarasikan kelas Manusia dengan atribut nama dan rambut. 2. Buat constructor overloading:  * Constructor pertama menerima parameter nama dan rambut. * Constructor kedua hanya menerima parameter nama.  1. Buat method sukaNonton:  * Method pertama menampilkan jenis film yang disukai. * Method kedua menghitung waktu menonton berdasarkan episode dan durasi.  1. Inisialisasi objek Manusia dengan atau tanpa warna rambut di method main. 2. Panggil method sukaNonton untuk menampilkan hobi dan menghitung total jam nonton. 3. Cetak hasil waktu menonton untuk kedua objek.   .  input    Output  **3.3 Menyimpulkan Perbedaan**  Algoritma untuk Menjelaskan Perbedaan:   1. Constructor Overloading:  * Buat lebih dari satu constructor dalam satu kelas dengan nama yang sama tetapi parameter berbeda.  1. Constructor Overriding:  * Constructor overriding tidak mungkin, karena constructor tidak diwarisi.  1. Method Overloading:  * Buat lebih dari satu method dengan nama yang sama tetapi parameter berbeda.  1. Method Overriding:  * Buat subclass yang mewarisi superclass, lalu ubah implementasi method dari superclass di subclass.  1. Method Mengembalikan Nilai:  * Method memiliki tipe data selain void dan menggunakan return.  1. Method Tidak Mengembalikan Nilai:  * Method bertipe void dan tidak menggunakan return. | | |
| **[Nomor 3] Kesimpulan** | | |
| **Analisa**  **Berikut kesimpulannya dengan bahasa yang lebih sederhana:**   1. **Perbaikan Kode:**  * **Masalah awal ada pada nama constructor yang tidak sesuai dengan nama kelas dan method yang tidak lengkap. Setelah diperbaiki, kode bisa bekerja dengan baik untuk menampilkan informasi dan menghitung total jam menonton.**  1. **Perbedaan Constructor Overloading dan Overriding:**  * **Constructor Overloading: Kita bisa membuat beberapa constructor (fungsi khusus untuk membuat objek) dengan nama yang sama, tetapi dengan jumlah atau jenis parameter berbeda, sehingga bisa digunakan dalam situasi berbeda.** * **Constructor Overriding: Tidak berlaku untuk constructor karena constructor tidak diwariskan ke kelas turunan.**  1. **Perbedaan Method Overloading dan Overriding:**  * **Method Overloading: Membuat beberapa method dengan nama yang sama, tapi parameternya berbeda. Ini memberi kita fleksibilitas untuk memanggil method yang sama dengan input berbeda.** * **Method Overriding: Mengubah isi (implementasi) method dari kelas induk di kelas turunan. Jadi, kelas turunan bisa memiliki cara kerja sendiri yang berbeda dari kelas induknya.**  1. **Method Mengembalikan Nilai vs Tidak Mengembalikan Nilai:**  * **Method yang Mengembalikan Nilai: Method ini menghasilkan sesuatu (seperti angka) yang bisa digunakan atau disimpan. Contohnya, method menghitung total jam menonton.** * **Method yang Tidak Mengembalikan Nilai: Method ini hanya melakukan sesuatu tanpa mengembalikan hasil apapun, seperti menampilkan pesan ke layar.**  1. **Manfaat Overloading dan Overriding:**  * **Overloading membuat kita bisa menggunakan satu nama method atau constructor untuk berbagai situasi.** * **Overriding membuat kelas turunan bisa menggunakan ulang method dari kelas induk, tapi dengan cara yang lebih sesuai dengan kebutuhannya.**   **Semua konsep ini membuat program lebih rapi, fleksibel, dan lebih mudah dipahami dan diperbaiki di masa depan.** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Nama & NPM | Topik: | Tanggal: |
| Ariiq Ashar Sofyan  G1F024052 | Extends | 28 september 2024 |
| [Nomor 4] Identifikasi Masalah: | | |
| **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 }   }  **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! | | |
| **[Nomor 4] Analisis dan Argumentasi** | | |
| **4.1. Evaluasi method yang dimiliki class Anak extends Ortu dengan method di class Ortu! Apakah penulisan method ini sudah efisien?**   * Kelas Ortu:   Memiliki dua method:   * + sukaMenonton(String a): Menampilkan informasi tentang menonton film dengan satu parameter (nama film).   + sukaMembaca(String a): Menampilkan informasi tentang membaca buku dengan satu parameter (nama buku). * Kelas Anak:   Memiliki tiga method:   * + sukaMenonton(int a, String b): Menampilkan informasi menonton dengan dua parameter (jam dan jenis film).   + sukaMenonton(String a): Menyalin method dari kelas Ortu untuk menampilkan informasi menonton.   + sukaMembaca(String a): Menyalin method dari kelas Ortu untuk menampilkan informasi membaca.   Evaluasi:   * Di kelas Anak, method sukaMenonton(String a) tidak perlu ditulis ulang karena fungsinya sama dengan yang ada di kelas Ortu. Ini membuat kode menjadi lebih rumit. * Jadi, lebih baik menghapus method yang sama dan hanya menyimpan method yang benar-benar diperlukan.   **4.2. Setelah dirunning di JDoodle, catat waktu eksekusinya. Rekomendasikan perbaikan penulisan kode method untuk dapat mengefisienkan waktu eksekusi!**  Perbaikan Kode:   1. Hapus method yang tidak perlu di kelas Anak. 2. Gunakan method dari kelas induk jika tidak perlu mengubahnya.   Kode Program yang Diperbaiki:  public class Ortu {  void sukaMenonton(String a) {  System.out.println("Nonton " + a);  }    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();  objekO.sukaMenonton("Berita");  objekO.sukaMembaca("Koran");    System.out.println("\nSifat Anak :");  Anak objekA = new 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);  }    @Override  void sukaMembaca(String a) {  System.out.println("Suka Baca " + a);  }  }  Dengan menghapus method sukaMenonton(String a) dari kelas Anak, kode menjadi lebih bersih dan mudah dimengerti. Hasil eksekusi akan tetap sama, tetapi kode lebih efisien dan lebih mudah dikelola.  **Waktu Eksekusi**  Untuk mencatat waktu eksekusi, jalankan kode di JDoodle dan lihat hasilnya. Biasanya, waktu eksekusi sangat cepat (dalam milidetik). Meskipun perbaikan ini mungkin tidak mengubah waktu eksekusi secara signifikan, kode yang lebih rapi akan lebih mudah dibaca dan dipahami. | | |
| **[Nomor 4] Penyusunan Algoritma dan Kode Program** | | |
| **4.1. Evaluasi method yang dimiliki class Anak extends Ortu dengan method di class Ortu! Apakah penulisan method ini sudah efisien?**  Algoritma:   1. Identifikasi method di class Anak yang mengoverride method di class Ortu. 2. Periksa apakah method yang di-override memiliki signature yang sama dengan method di Ortu. 3. Evaluasi efisiensi method overriding.       Input & Output  **4.2. Setelah dirunning di JDoodle, catat waktu eksekusinya. Rekomendasikan perbaikan penulisan kode method untuk dapat mengefisienkan waktu eksekusi!**  Algoritma   1. Mulai 2. Definisikan kelas Ortu:    * + Buat method sukaMenonton(String a).      + Buat method sukaMembaca(String a). 3. Definisikan kelas Anak yang mewarisi dari kelas Ortu:    * + Buat method sukaMenonton(int a, String b).      + Hapus method sukaMenonton(String a) dari kelas Anak karena sudah ada di kelas Ortu.      + Hapus method sukaMembaca(String a) dari kelas Anak karena sudah ada di kelas Ortu. 4. Buat objek dari kelas Ortu dan panggil method-nya. 5. Buat objek dari kelas Anak dan panggil method-nya. 6. Selesai.   .  input    Output | | |
| **[Nomor 4] Kesimpulan** | | |
| Analisa  Kesimpulan 4.1:   1. Evaluasi method di kelas Anak dan Ortu:    * Kelas Anak mewarisi method dari kelas Ortu, yang berarti semua method di kelas Ortu dapat digunakan di kelas Anak tanpa perlu mendeklarasikan ulang.    * Method sukaMenonton(String a) dan sukaMembaca(String a) yang ada di kelas Anak tidak perlu dideklarasikan ulang karena sudah ada di kelas Ortu. Ini menyebabkan redundansi (duplikasi) yang membuat kode kurang efisien. 2. Penulisan method bisa lebih efisien:    * Di kelas Anak, hanya method yang berbeda dari Ortu yang perlu dideklarasikan, seperti sukaMenonton(int a, String b).    * Tidak perlu mendeklarasikan ulang method yang sama dengan kelas induk kecuali ingin melakukan perubahan perilaku (overriding).   Kesimpulan 4.2:   1. Waktu eksekusi bisa diperbaiki dengan menghapus method yang tidak diperlukan:    * Dengan menghapus method yang menduplikasi method di kelas Ortu, kita dapat memperbaiki efisiensi kode.    * Penghapusan method yang tidak diperlukan mengurangi waktu kompilasi dan membuat kode lebih ringan. 2. Rekomendasi perbaikan kode:    * Hanya tambahkan method di kelas Anak yang benar-benar berbeda dari Ortu.    * Gunakan konsep inheritance (pewarisan) dengan baik, sehingga kelas anak bisa memanfaatkan method dari kelas induk tanpa duplikasi yang tidak perlu.   Secara keseluruhan, kita belajar bahwa konsep pewarisan (inheritance) dalam OOP (Object-Oriented Programming) bertujuan untuk menghindari duplikasi kode dan membuat program lebih efisien. | | |
|  | | |