|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Randi Satria Anggarra**  **G1F024048** | **Kelas Java** | **5 oktober 2024** |
| **Unit 1 kelas (Class)** | | |
| Latihan 1.1  Code:  **Sebuah gambar berisi teks, cuplikan layar, software, tampilan  Deskripsi dibuat secara otomatis**  **Output:**  **Sebuah gambar berisi teks, cuplikan layar, Font  Deskripsi dibuat secara otomatis**  **Penjelasan :**  **String nama**: Ini merepresentasikan nama dari objek Manusia. **String rambut**: Ini merepresentasikan jenis atau warna rambut dari objek Manusia.**void berbicara(String kata)**: Metode ini dapat digunakan untuk mengeluarkan suara atau berbicara dengan kata-kata tertentu. **void berjalan()**: Metode ini dapat menggambarkan tindakan berjalan. **void memperkenalkanDiri()**: Metode ini dapat mencetak nama dan informasi lainnya tentang individu tersebut. **void merubahRambut(String jenisRambut)**: Metode ini bisa digunakan untuk mengganti atribut rambut. | | |
| **Unit 2 Objek** | | |
| Latihan 2.1  **Sebuah gambar berisi teks, cuplikan layar, tampilan, Font  Deskripsi dibuat secara otomatis**  Penjelasan :  Sebuah gambar berisi teks, cuplikan layar, Font, kartu bisnis  Deskripsi dibuat secara otomatisKode di atas mendefinisikan kelas Ortu yang memiliki konstruktor dengan dua parameter: nama dan rambut. Ketika objek baru dari kelas Ortu dibuat, seperti pada Ortu satu = new Ortu("Randi", "hitam");, konstruktor ini dipanggil dan mencetak pesan yang menampilkan nama dan warna rambut yang diberikan sebagai argument. Dan output yang dihasilkan adalah :  2.2  Sifat atribut :   * **Nama**: Nama dari keturunan bisa diturunkan. * **Rambut**: Ciri fisik seperti warna atau tekstur rambut.   perilaku positif (behavior) :   * **Berbicara**: Kemampuan untuk berbicara dengan cara yang sopan dan baik. * **Bermain**: Mengembangkan hobi yang diwarisi, seperti bermain musik atau olahraga. * **Belajar**: Memiliki rasa ingin tahu dan semangat untuk belajar hal-hal baru. * **Berbagi**: Sifat sosial seperti berbagi dengan teman dan lingkungan. | | |
| **Unit 3 Method** | | |
| 3.1 Analisa perbedaan deklarasi constructor, method, dan method utama!  Constructor digunakan untuk menginisialisasi objek baru ketika kelas diinstansiasi, memiliki nama yang sama dengan kelas, dan tidak memiliki tipe kembalian. Sebaliknya, method digunakan untuk mendefinisikan perilaku objek dan dapat memiliki berbagai nama serta tipe kembalian, termasuk void. Sementara itu, method utama adalah titik awal eksekusi program, selalu bernama main, memiliki tipe void, dan menerima parameter berupa array string untuk mengambil argumen dari command line. Dengan memahami perbedaan ini, kita dapat dengan jelas mendefinisikan dan mengorganisir kode dalam program Java.  3.2 Tentukan kapan Anda perlu menggunakan constructor dan method?  menggunakan constructor ketika ingin menginisialisasi atribut objek saat objek tersebut dibuat, memastikan bahwa objek mulai dalam keadaan yang valid dan konsisten. Sedangkan , method digunakan untuk mendefinisikan perilaku dan fungsi yang dapat dilakukan oleh objek setelah inisialisasi, memungkinkan mengimplementasikan logika yang dapat dipanggil berulang kali, memodifikasi status objek, atau melakukan operasi tertentu, sehingga meningkatkan keterbacaan kode.  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  a) **Constructor Overloading dan Overriding**  Tterjadi ketika sebuah kelas memiliki lebih dari satu constructor dengan nama yang sama tetapi parameter yang berbeda, memungkinkan instansiasi objek dengan cara yang bervariasi sesuai kebutuhan  b) **Method Overloading dan Method Overriding**  Terjadi ketika sebuah kelas memiliki beberapa method dengan nama yang sama tetapi dengan parameter yang berbeda (jumlah atau tipe parameter), memungkinkan penggunaan yang fleksibel dalam konteks yang berbeda.  c) **Method yang Mengembalikan Nilai dan Method Tidak Mengembalikan Nilai**  Memiliki tipe kembalian yang ditentukan dan menggunakan kata kunci `return` untuk mengembalikan nilai kepada pemanggil, sehingga dapat digunakan dalam ekspresi atau disimpan dalam variabel. Sebaliknya, **method tidak mengembalikan nilai** dideklarasikan dengan tipe `void`, tidak mengembalikan nilai kepada pemanggil, dan umumnya digunakan untuk melakukan tindakan tertentu, seperti menampilkan output atau mengubah keadaan objek tanpa memberikan nilai kembali.  Code :  Sebuah gambar berisi teks, cuplikan layar, tampilan, Font  Deskripsi dibuat secara otomatis  Penjelasan :  Kode tersebut mendefinisikan kelas Manusia yang memiliki dua atribut, yaitu nama dan rambut, serta satu constructor yang menginisialisasi objek dengan nilai-nilai untuk kedua atribut tersebut dan mencetaknya. Selain itu, terdapat method sukaNonton yang menerima parameter berupa judul film dan mencetak hobi menonton film tersebut. Dalam method main, sebuah objek Manusia bernama satu diinstansiasi dengan nama "Randi Satria Anggara" dan warna rambut "Hitam", lalu memanggil method sukaNonton untuk mencetak hobi menontonnya, yaitu "mOTOGP". Dan menghasilkan output berikut :  Sebuah gambar berisi teks, cuplikan layar, Font  Deskripsi dibuat secara otomatis | | |
| **Unit 4 Extends** | | |
| 4.1. Bandingkan method yang dimiliki class Anak extends Ortu dengan method di class Ortu!  Kelas `Ortu` biasanya memiliki method yang berfokus pada perilaku dan atribut yang berkaitan dengan orang tua, seperti mengatur dan menampilkan informasi dasar. Sebaliknya, kelas `Anak` dapat memiliki method yang lebih spesifik atau diperluas, seperti method yang menangani aktivitas atau ciri-ciri unik yang relevan untuk anak, serta dapat melakukan overriding pada method yang ada di kelas `Ortu` untuk memberikan perilaku yang lebih sesuai dengan konteks anak. Dengan cara ini, kelas `Anak` dapat memanfaatkan kembali logika dari kelas `Ortu` sekaligus menambahkan atau mengubah fungsionalitas sesuai kebutuhan.  4.2. Ubahlah Contoh 4 dengan menambahkan objek anak dengan method yang berbeda!  Source code :  public class Extends {  // 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 :");  Extends objekO = new Extends(); // 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 motoGP"); //memanggil sifat spesifik anak yang diturunkan induk  objekA.sukaMembaca("NOVEL"); //memanggil method ke induk yang otomatis diturunkan tanpa deklarasi ulang di anak  } }  class Anak extends Extends {  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 :");  Extends objekO = new Extends(); // 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 motoGP"); //memanggil sifat spesifik anak yang diturunkan induk  objekA.sukaMembaca("NOVEL"); //memanggil method ke induk yang otomatis diturunkan tanpa deklarasi ulang di anak  }  }  Penjelasan kode :  Kode di atas mendefinisikan dua kelas, yaitu Extends sebagai kelas induk dan Anak sebagai kelas anak yang mewarisi dari Extends. Kelas Extends memiliki dua method: sukaMenonton, yang mencetak jenis tontonan, dan sukaMembaca, yang mencetak jenis bacaan. Di dalam main, objek Extends dibuat untuk memanggil kedua method ini. Kelas Anak mengoverride method sukaMenonton dengan dua versi: satu dengan dua parameter (jam dan jenis film) dan satu dengan satu parameter, yang berfungsi sebagai method spesifik untuk anak. Saat objek Anak dibuat, method dari kelas induk tetap dapat dipanggil tanpa perlu mendeklarasikan ulang, menunjukkan kemampuan pewarisan dan overriding. Dan menghasilkan output : | | |
| **KESIMPULAN**  Kesimpulan dari teks tersebut adalah bahwa pemrograman berorientasi objek (OOP) menggunakan konsep kelas, objek, dan method untuk menyusun kode secara efisien dan terstruktur. Kelas berfungsi sebagai cetak biru untuk menciptakan objek yang memiliki atribut dan perilaku tertentu, di mana constructor digunakan untuk menginisialisasi objek saat dibuat, sedangkan method mendefinisikan fungsi yang dapat dilakukan objek tersebut. Melalui pewarisan, seperti yang ditunjukkan dalam contoh antara kelas `Ortu` dan kelas `Anak`, program dapat memanfaatkan kembali logika dari kelas induk dan menambahkan atau memodifikasi perilaku sesuai kebutuhan. Selain itu, konsep overloading dan overriding pada method memberikan fleksibilitas dalam mendefinisikan dan menyesuaikan fungsionalitas, sehingga menciptakan kode yang lebih modular dan mudah dipelihara. | | |