|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Sevia Ayu Wandinda**  **G1F021005** | **Kelas (class)** | **12 September 2024** |
| **1.a 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!   1. Rincikan sumber informasi yang relevan (buku / webpage)   **https://www.youtube.com/watch?v=60IdOc8m8Es** | | |
| **1.b Analisis dan Argumentasi** | | |
| 1. Uraikan rancangan solusi yang diusulkan.   Solusinya adalah dengan mencari atribut variabel yang mana merupakan karakteristik yang dimiliki suatu objek, sedangkan perilaku/behavior untuk method merujuk ke tindakan atau fungsi yang dapat dilakukan oleh objek.   1. Analisis solusi, kaitkan dengan permasalahan.   Untuk atribut variabel pada kelas manusia adalah String nama;, dan String rambut, String tinggi badan;, String berat badan;  Untuk perilaku/ behavior pada kelas manusia adalah berjalan, bernapas, mengetik, berlari. | | |
| **1.c Penyusunan Algoritma dan Kode Program** | | |
| 1. Algoritma  * Mulai * Mencari ciri-ciri umum kelas Manusia yang dapat menjadi atribut variabel untuk method * Mencari ciri-ciri umum kelas Manusia yang dapat menjadi perilaku/ behavior untuk method * Selesai  1. Tuliskan kode program dan luaran   Screenshot/ Capture potongan kode dan hasil luaran  Source scode nya:  public class Manusia {  // Atribut variabel  private String nama;  private String rambut;  private String tinggiBadan; // dalam cm  private String beratBadan; // dalam kg  // Konstruktor  public Manusia(String nama, String rambut, String tinggiBadan, String beratBadan) {  this.nama = nama;  this.rambut = rambut;  this.tinggiBadan = tinggiBadan;  this.beratBadan = beratBadan;  }  // Metode untuk perilaku manusia  public void berjalan() {  System.out.println(nama + " sedang berjalan.");  }  public void bernapas() {  System.out.println(nama + " sedang bernapas.");  }  public void mengetik() {  System.out.println(nama + " sedang mengetik.");  }  public void berlari() {  System.out.println(nama + " sedang berlari.");  }  // Metode untuk menampilkan informasi  public void tampilInfo() {  System.out.println("Nama: " + nama);  System.out.println("Rambut: " + rambut);  System.out.println("Tinggi Badan: " + tinggiBadan + " cm");  System.out.println("Berat Badan: " + beratBadan + " kg");  }  public static void main(String[] args) {  // Membuat objek dari kelas Manusia  Manusia manusia1 = new Manusia("Sevia Ayu Wandinda", "Coklat", "150", "57");  // Menampilkan informasi  manusia1.tampilInfo();  manusia1.berjalan();  manusia1.bernapas();  manusia1.mengetik();  manusia1.berlari();  }  }   * Kode program | | |
| **1.d Kesimpulan** | | |
| Untuk String nama, rambut, tinggi badan, berat badan termasuk dalam atribut variable karena merupakan ciri-ciri manusia. Sedangkan berjalan, bernapas, mengetik, dan berlari termasuk dalam perilaku manusia dan masuk ke bagian perilaku/behavior. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Sevia Ayu Wandinda**  **G1F021005** | **Objek** | **12 September 2024** |
| **2.a Identifikasi Masalah:** | | |
| 1. Uraikan permasalahan dan variabel   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?   1. Rincikan sumber informasi yang relevan (buku / webpage)   **https://www.youtube.com/watch?v=60IdOc8m8Es** | | |
| **2.b Analisis dan Argumentasi** | | |
| 1. Analisis   Apabila nanti Anda akan memiliki keturunan, analisa sifat (atribut), constructor, dan perilaku positif (behavior) apa yang akan diturunkan?  ketika saya memiliki keturunan, ada beberapa hal yang dapat diturunkan pada keturunan saya, yaitu sebagai berikut:   1. Sifat(Atribut)   Atribut yang biasanya dapat diwariskan dapat berupa warna mata, tinggi badan, jenis rambut, dan kondisi kesehatan tertentu.   1. Construktor   Constructor digunakan saat membuat objek dari anak untuk menganalisis atribut yang akan diwariskan.   1. Perilaku positif(Behavior)   Hal ini merujuk pada perilaku positif yang dimilki saya yaitu kelas induk yang akan diturunkan ke perilaku keturunan atau kelas anak. | | |
| **2.c Penyusunan Algoritma dan Kode Program** | | |
| 1. Algoritma  * Mulai * Membuat constructor yaitu String nama, String rambut, String tinggi badan dan berat badan * Menampilkan informasi dari constructor * Membuat objek * Selesai  1. Tuliskan kode program dan luaran   Screenshot/ Capture potongan kode dan hasil luaran   1. Kode program | | |
| **2.d Kesimpulan** | | |
| Pada pemrograman objek diatas dapat disimpulkan bahwa pemrograman berorientasi objek dapat dipakai untuk mempresentasikan pewarisan sifat dan perilaku seperti kehidupan nyata. | | |
|  | | |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Sevia Ayu Wandinda**  **G1F021005** | **Method** | **12 September 2024** |
| **3.a Identifikasi Masalah:** | | |
| 1. Uraikan permasalahan dan variabel   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     1. Rincikan sumber informasi yang relevan (buku / webpage)   **https://www.youtube.com/watch?v=60IdOc8m8Es** | | |
| **3.b Analisis dan Argumentasi** | | |
| 1) Analisis  a) Analisis perbedaan deklarasi constructor, method, dan method utama  • Konstruktor adalah metode khusus yang dipanggil saat Anda membuat sesuatu baru dari suatu kelas. Tujuannya adalah untuk menganalisis objek dengan nilai awal.  • Method adalah fungsi atau prosedur yang dideklarasikan dalam sebuah kelas dan dapat dipanggil untuk melakukan operasi tertentu pada objek kelas tersebut.  • Metode utama adalah cara utama aplikasi Java memulai, dan itulah yang memulai JVM atau lingkungan runtime.  b) Kapan perlu menggunakan constructor dan method  • Constructor.  Mulai sesuatu atau kelas, dan persiapkan dengan pengaturan pertama. Konstruktor akan mulai bekerja saat Anda membuat objek baru dengan kata kunci new  • Metode  Digunakan untuk menjalankan operasi tertentu pada objek atau kelas. Metode dipanggil secara eksplisit berdasarkan namanya.  Uraikan perbedaan berikut:  • Construktor overloading dan overriding  Construktor overloading terjadi saat kelas memiliki beberapa dengan parameter yang berbeda, memungkinkan objek dibuat dengan cara yang berbeda. Namun, constructor tidak dapat di-override karena tidak diwariskan oleh subclass, sehingga konsep overriding tidak berlaku untuk constructor.  • Method overloading dan overriding  Method overloading adalah ketika kita membuat beberapa method dengan nama yang sama tetapi berbeda parameter dalam satu kelas constructor. Ini memungkinkan pemanggilan metode dengan berbagai input. Di sisi lain, overriding method terjadi saat sebuah subclass menggunakan nama dan parameter yang sama, namun dengan perilaku yang berbeda.  • Method yang mengembalikan nilai menghasilkan output berupa tipe data tertentu setelah dieksekusi dan menggunakan return untuk mengirimkan nilai tersebut. Sementara itu,  metode yang tidak mengembalikan nilai menggunakan tipe void  hanya melakukan aksi tertentu tanpa memberikan nilai balik. | | |
| **3.c Penyusunan Algoritma dan Kode Program** | | |
| 1. Screenshot/ Capture potongan kode dan hasil luaran 2. Kode program | | |
| **3.d Kesimpulan** | | |
| * Pada constructor dan method terdapat perbedaan yaitu pada constructor saat membuat dan berfungsinya objek constructor secara otomatis berjalan, sedangkan method tidak seperti itu. Method dapat dipanggil kapan saja untuk menjalankan tindakan tertentu pada objek. * Overloading memungkinkan baik constructor dan method memiliki versi berbeda berdasarkan parameter yang diterima, sedangkan overriding hanya berlaku untuk method dan memungkinkan subclass mengganti implementasi dari superclass. Selain itu, method dapat dikelompokan menjadi yang mengembalikan nilai dan yang hanya menjalankan aksi tanpa mengembalikan hasil. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Nama & NPM** | **Topik:** | **Tanggal:** |
| **Sevia Ayu Wandinda**  **G1F021005** | **Extends** | **12 September 2024** |
| **4.a Identifikasi Masalah:** | | |
| 1. Uraikan permasalahan dan variabel   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!   1. Rincikan sumber informasi yang relevan (buku / webpage)   **https://www.youtube.com/watch?v=6qULMlcv-eg** | | |
| **4.b Analisis dan Argumentasi** | | |
| * **Solusi yang diusulkan**   Solusi yang saya usulkan adalah dengan menambahkan method pada kelas anak dari method kelas ortu. Sebagai contoh saya menambahkan method pada kode program yaitu objekA.sukaMakan(“Burger”); | | |
| **4.c Penyusunan Algoritma dan Kode Program** | | |
| 1. Algoritma  * Mulai * Membuat kelas induk * Membuat method induk spesifik dan method umum bisa diubah ke kelas anak * Membuat objek untuk memanggil objek induk * Memanggil sifat induk dan method dengan variabel * Membuat kelas anak * Membuat objek anak * Membuat method anak * Memanggil method anak * Selesai  1. Tuliskan kode program dan luaran   Screenshot/ Capture potongan kode dan hasil luaran  Source code:  public class Ortu { // Membuat kelas induk  void sukaMenonton(String a) { // Metode induk spesifik  System.out.println("Nonton " + a);  }  void sukaMembaca(String a) { // Metode 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("Sepak Bola"); // Memanggil method baru yang ditambahkan  }  }  class Anak extends Ortu {  void sukaMenonton(int a, String b) {  System.out.println("Nonton Jam " + a + " Malam " + b);  }  void sukaMenonton(String a) { // Metode induk spesifik  System.out.println("Nonton " + a);  }  void sukaMembaca(String a) { // Metode induk umum bisa diubah anak  System.out.println("Suka Baca " + a);  }  void sukaOlahraga(String olahraga) { // Metode baru  System.out.println("Suka Olahraga: " + olahraga);  }  }   1. Kode program | | |
| **4.d Kesimpulan** | | |
| Menambahkan method baru pada kelas anak tidak harus selalu sama dengan kelas induk, bisa berbeda jika melalui proses overriding dan penambahan method khusus. Pada kelas anak saya menambahkan method objekA.sukaMakan(“Burger”); sehingga kelas anak memiliki fungsional yang lain atau tambahan yang tidak dimilki oleh kelas induk, namun tetap dapat mewarisi perilaku method yang sama dari kelas induk. | | |
|  | | |

**Refleksi**

Setelah mengerjakan tugas ini saya mendapat ilmu baru, mulai dari memahami konsep kelas, objek, method, dan extends. Saya dapat memahami bagaimana cara membuat atribut dan perilaku yang dapat diwariskan dari orang tua(kelas induk) ke keturunan (kelas anak atau kelas turunan), dapat membedakan constructor, method, dan method utama. Mengetahui kapan penggunaan constructor dan method, mengetahui perbedaan constructor overloading dan overriding, serta bagaimana konsep-konsep ini diterapkan dalam berbagai scenario pemrograman.