Tugas 1

Nama & NPM	**Topik:**	**Tanggal:**
Abdullah Hasyim Syauqi G1F024019	**Kelas (Class)**	**16-09-2024**

[1.1,1.2,1.3] Identifikasi Masalah:

1.1
- Nama konstruktor diubah dari Manusia1 menjadi Manusia untuk mencocokkan nama kelas.
- Konstruktor Manusia menerima dua parameter untuk menginisialisasi atribut nama dan rambut.
- Atribut nama dan rambut diinisialisasi menggunakan this di dalam konstruktor.
- Di dalam main, objek dari kelas Manusia dibuat dengan menggunakan nama yang benar dan memberikan dua parameter sesuai dengan konstruktor.

1.2 susun kode menggunakan constructor dengan parameter data pribadi anda

Penyusunan Algoritma dan Kode Program

1.1

```
public class Manusia {
   String nama;
   String warnaRambut;
   int umur;

   public Manusia(String nama, String warnaRambut, int umur) {
      this.nama = nama;
      this.warnaRambut = warnaRambut;
      this.umur = umur;
   }

   // menampilkan data pribadi
   public void tampilkanData() {
      System.out.println("Nama saya: " + nama);
      System.out.println("Warna rambut: " + warnaRambut);
      System.out.println("Umur saya: " + umur);
   }

   public static void main(String[] args) {
      Manusia pribadi = new Manusia("Putri", "hitam", 25);

      // Menampilkan data pribadi
      pribadi.tampilkanData();
   }
}
```

1.2

```
public class Manusia {
    String nama;
    String warnaRambut;
    int umur;

    public Manusia(String nama, String warnaRambut, int umur) {
        this.nama = nama;
        this.warnaRambut = warnaRambut;
        this.umur = umur;
    }

    public void tampilkanData() {
        System.out.println("Nama saya: " + nama);
        System.out.println("Warna rambut: " + warnaRambut);
        System.out.println("Umur saya: " + umur);
    }

    public static void main(String[] args) {
        Manusia pribadi = new Manusia("Hasyim", "Hitam", 18);
        pribadi.tampilkanData();
    }
}
```

Kesimpulan

Kesimpulannya kode ini mendefinisikan kelas Manusia dengan atribut data pribadi (nama, warna rambut, umur), menginisialisasi atribut melalui konstruktor, dan menampilkan informasi menggunakan method tampilkanData dalam metode main.

Refleksi

kode ini menunjukkan cara mendefinisikan kelas dengan atribut dan metode di Java, serta bagaimana menggunakan konstruktor untuk menginisialisasi data. Implementasi ini memberikan pemahaman dasar tentang objek dan metode dalam pemrograman berorientasi objek, serta cara menyusun kode agar lebih terstruktur dan fungsional.

Tugas 2

Nama & NPM	**Topik:**	**Tanggal:**
Abdullah Hasyim Syauqi G1F024019	**Objek**	**16-09-2024**

[2.1,2.2] Identifikasi Masalah:

2.1 Nama class harus diawali huruf besar, jadi seharusnya Ortu, bukan ortu. Constructor tidak memiliki nama dan harus dinamai sesuai dengan nama class, yaitu Ortu. variabel nama dan rambut belum dideklarasikan, dan constructor harus memiliki parameter untuk menerima nilai nama dan rambut.

2.2 Nama, warna rambut, tinggi badan, kreativitas

2.3 Rancanglah kode program untuk sifat (atribut) dan constructor overloaded dari

Penyusunan Algoritma dan Kode Program

2.1

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

   // Constructor
   public Ortu(String nama, String rambut) {
      // Inisialisasi variabel instance dengan parameter constructor
      this.nama = nama;
      this.rambut = rambut;

      // Menampilkan informasi
      System.out.println("Nama saya: " + this.nama +
                  "\nWarna Rambut: " + this.rambut);
   }

   public static void main(String[] args) {
      // Membuat objek baru dengan constructor
      Ortu satu = new Ortu("Putri", "hitam");
   }
}
```

2.2

```
public class Ortu {
   // Deklarasi atribut
   private String nama;
   private String warnaRambut;
   private double tinggiBadan;
   private String kreativitas;

   // Constructor pertama (lengkap)
   public Ortu(String nama, String warnaRambut, double tinggiBadan, String kreativitas) {
      this.nama = nama;
      this.warnaRambut = warnaRambut;
      this.tinggiBadan = tinggiBadan;
      this.kreativitas = kreativitas;
```

```
        System.out.println("Constructor 1 (Lengkap)");
        tampilkanInformasi();
    }

    // Constructor kedua (tanpa kreativitas, default nilai)
    public Ortu(String nama, String warnaRambut, double tinggiBadan) {
        this(nama, warnaRambut, tinggiBadan, "Tidak Diketahui");
        System.out.println("Constructor 2 (Tanpa Kreativitas)");
    }

    // Constructor ketiga (nama dan warna rambut saja, default nilai)
    public Ortu(String nama, String warnaRambut) {
        this(nama, warnaRambut, 0.0, "Tidak Diketahui");
        System.out.println("Constructor 3 (Partial)");
    }

    // Method untuk menampilkan informasi
    private void tampilkanInformasi() {
        System.out.println("Nama: " + nama);
        System.out.println("Warna Rambut: " + warnaRambut);
        System.out.println("Tinggi Badan: " + tinggiBadan + " cm");
        System.out.println("Kreativitas: " + kreativitas);
    }

    // Main method sebagai entry point
    public static void main(String[] args) {
        // Membuat objek dengan constructor lengkap
        Ortu ortu1 = new Ortu("Hasyim", "Hitam", 168.0, "Tinggi");

        // Membuat objek dengan constructor tanpa kreativitas
        Ortu ortu2 = new Ortu("Hisyam", "Hitam", 157.0);

        // Membuat objek dengan constructor hanya nama dan warna rambut
        Ortu ortu3 = new Ortu("Hafizah", "hitam");
    }
}
```

Kesimpulan

Kesimpulannya adalah dalam Java, constructor overloading memungkinkan kita untuk membuat beberapa constructor dengan parameter yang berbeda dalam satu class. Dengan demikian, kita dapat menginisialisasi objek dengan berbagai cara sesuai kebutuhan. Kode di atas menunjukkan penggunaan berbagai constructor untuk mengatur atribut objek dan menampilkan informasi yang relevan.

Refleksi

Penggunaan constructor overloading meningkatkan fleksibilitas dan keterbacaan kode dengan memungkinkan berbagai cara untuk membuat objek, menjadikannya lebih mudah untuk menangani berbagai skenario inisialisasi tanpa menulis kode berulang.

Tugas 3

Nama & NPM	Topik:	Tanggal:
Abdullah Hasyim Syauqi G1F024019	**Method**	**16-09-2024**

[2.1,2.2] Identifikasi Masalah:

3.1 Ubah constructor menjadi Manusia dan pastikan constructor menyimpan nilai, bukan mencetak. Tambahkan method tampilInfo untuk menampilkan informasi nama dan rambut. Pisahkan method sukaNonton menjadi dua: satu untuk menampilkan film dan satu lagi untuk menghitung jam nonton. Jangan lupa untuk mendeklarasikan variabel dan parameter yang diperlukan.

3.2 Kembangkanlah kode program untuk method dari Ortu dengan data perilaku pribadi Anda yang menggunakan:
a) method overloading,
b) method dengan return value
c) method tanpa return value

Penyusunan Algoritma dan Kode Program

3.1

```
public class Manusia {
   // Deklarasi atribut Manusia
   String nama, rambut;

   // Deklarasi constructor
   public Manusia(String nama, String rambut) {
      this.nama = nama;
      this.rambut = rambut;
   }

   // Deklarasi method untuk menampilkan informasi
   void tampilInfo() {
      System.out.println("Nama saya: " + nama + "\nWarna Rambut: " + rambut);
   }

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

   // Method untuk menghitung jam nonton
   int sukaNonton(int episode, int durasi) {
      return episode * durasi;
   }

   // Method utama
   public static void main(String[] args) {
      Manusia satu = new Manusia("Putri", "hitam");
      satu.tampilInfo();
      satu.sukaNonton("Drakor");
      int jumlahJam = satu.sukaNonton(2, 2);
      System.out.println("Jam nonton = " + jumlahJam + " jam");
   }
}
```

3.2

```
public class Ortu {
    // Deklarasi atribut
    private String nama;
    private String warnaRambut;
    private double tinggiBadan;
    private String kreativitas;

    // Constructor pertama (lengkap)
    public Ortu(String nama, String warnaRambut, double tinggiBadan, String kreativitas) {
        this.nama = nama;
        this.warnaRambut = warnaRambut;
        this.tinggiBadan = tinggiBadan;
        this.kreativitas = kreativitas;

        System.out.println("Constructor 1 (Lengkap)");
        tampilkanInformasi();
    }

    // Constructor kedua (tanpa kreativitas, default nilai)
    public Ortu(String nama, String warnaRambut, double tinggiBadan) {
        this(nama, warnaRambut, tinggiBadan, "Tidak Diketahui");
        System.out.println("Constructor 2 (Tanpa Kreativitas)");
    }

    // Constructor ketiga (nama dan warna rambut saja, default nilai)
    public Ortu(String nama, String warnaRambut) {
        this(nama, warnaRambut, 0.0, "Tidak Diketahui");
        System.out.println("Constructor 3 (Partial)");
    }

    // Method untuk menampilkan informasi
    private void tampilkanInformasi() {
        System.out.println("Nama: " + nama);
        System.out.println("Warna Rambut: " + warnaRambut);
        System.out.println("Tinggi Badan: " + tinggiBadan + " cm");
        System.out.println("Kreativitas: " + kreativitas);
    }

    // Method overloading: berbicara dengan satu parameter
    public void berbicara(String pesan) {
        System.out.println("Pesan: " + pesan);
    }

    // Method overloading: berbicara dengan dua parameter
    public void berbicara(String pesan, int volume) {
        System.out.println("Pesan: " + pesan + " dengan volume " + volume);
    }

    // Method dengan return value: menghitung tinggi badan ideal
    public double hitungTinggiBadanIdeal(double usia) {
        double tinggiIdeal = 0.0;
        if (usia < 18) {
            tinggiIdeal = 150 + (usia * 2.5);
        } else {
```

```
            tinggiIdeal = 170 + (usia - 18) * 1.2;
        }
        return tinggiIdeal;
    }

    // Method tanpa return value: memberikan motivasi
    public void berikanMotivasi() {
        System.out.println("Mahasiswa Sistem informasi, teruslah berusaha!");
    }

    // Main method sebagai entry point
    public static void main(String[] args) {
        // Membuat objek dengan constructor lengkap
        Ortu ortu1 = new Ortu("Hasyim", "Hitam", 168.0, "Tinggi");

        // Membuat objek dengan constructor tanpa kreativitas
        Ortu ortu2 = new Ortu("Hisyam", "Hitam", 157.0);

        // Membuat objek dengan constructor hanya nama dan warna rambut
        Ortu ortu3 = new Ortu("Hafizah", "hitam");

        // Menggunakan method overloading
        ortu1.berbicara("Selamat pagi!");
        ortu1.berbicara("Selamat pagi!", 5);

        // Menggunakan method dengan return value
        double tinggiIdeal = ortu1.hitungTinggiBadanIdeal(25);
        System.out.println("Tinggi badan ideal untuk usia 25 tahun: " + tinggiIdeal + " cm");

        // Menggunakan method tanpa return value
        ortu1.berikanMotivasi();
    }
}
```

Kesimpulan

Kesimpulannya kode Ortu menunjukkan penerapan berbagai teknik pemrograman Java, termasuk method overloading untuk fleksibilitas fungsi, method dengan return value untuk kalkulasi, dan method tanpa return value untuk aksi sederhana. Ini memberikan struktur yang jelas dan fungsionalitas yang terorganisir.

Refleksi

Dengan menerapkan berbagai jenis metode, kita dapat meningkatkan fungsionalitas program dan membuat kode lebih modular dan mudah digunakan, serta memudahkan pemeliharaan dan pengembangan di masa depan.

Tugas 4

Nama & NPM	Topik:	Tanggal:
Abdullah Hasyim Syauqi G1F024019	**Extends**	**16-09-2024**

[4.1] Identifikasi Masalah:

4.1
- Method Overloading: Pertahankan metode sukaMenonton(int a, String b) di Anak dan hapus versi satu parameter dari Anak jika tidak diperlukan.
- Method Override: Jika method sukaMembaca(String a) di Anak tidak mengubah fungsionalitas, hapus method tersebut dari Anak untuk menghindari duplikasi yang tidak perlu.

4.2 kode pertama : CPU Time: 0.07 sec(s), Memory: 39760 kilobyte(s), Compiled and executed in 1.541 sec(s)
kode kedua : CPU Time: 0.07 sec(s), Memory: 39136 kilobyte(s), Compiled and executed in 1.43 sec(s)

Penyusunan Algoritma dan Kode Program

4.1

```
//kode yang belum efisien
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
}    }


4.2
//kode yang hasyim sarankan
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);
   }
   // Tidak perlu mendeklarasikan kembali method sukaMembaca jika tidak mengubah
fungsionalitas
}
```

Kesimpulan

Kesimpulannya metode di Anak memanfaatkan overloading untuk menambahkan fungsionalitas baru (sukaMenonton(int a, String b)) dan overriding untuk menyesuaikan fungsionalitas yang diwarisi (sukaMenonton(String a)). Method sukaMembaca(String a) di Anak mengulangi method dari Ortu tanpa perubahan, yang mungkin tidak perlu jika tidak ada modifikasi yang diperlukan.

Refleksi

Dengan menggunakan overloading dan overriding secara bijaksana, kita dapat memperluas fungsionalitas dan memodifikasi perilaku yang diwarisi dengan cara yang efisien, menghindari redundansi dan mempermudah pemeliharaan kode.