Template Lembar Kerja Individu

Nama & NPM	Topik:	Tanggal:
Renni Ren Rofa.Pgb **G1F021002**	**Kelas, Objek, Method**	**3 Oktober 2024**

[Latihan 1.] Identifikasi Masalah:

1) Uraikan permasalahan :
   1.1 Analisa ciri-ciri umum Kelas Manusia yang dapat menjadi, a. atribut variabel, dan b. perilaku/ behavior untuk method!

2) Rincikan sumber informasi yang relevan :
   Java Documentation: *https://docs.oracle.com/javase/tutorial/java/concepts/*

3) Uraikan rancangan solusi yang diusulkan :
   Solusi yang diusulkan adalah membuat daftar atribut dan method yang umum dimiliki oleh manusia, berdasarkan karakteristik fisik dan perilaku umum.

[Latihan 1.] Analisis dan Argumentasi

1) Uraikan rancangan solusi yang diusulkan: Solusi yang diusulkan adalah membagi karakteristik manusia menjadi dua kategori: a. Atribut variabel: karakteristik yang dapat direpresentasikan sebagai data b. Perilaku/behavior untuk method: aksi yang dapat dilakukan oleh manusia

2) Analisis solusi, kaitkan dengan permasalahan: Solusi ini memungkinkan kita untuk memodelkan manusia dalam konteks pemrograman berorientasi objek. Atribut variabel merepresentasikan state dari objek Manusia, sementara method merepresentasikan behavior. Ini sesuai dengan prinsip dasar OOP dimana objek memiliki state dan behavior.

[Latihan 1.] Penyusunan Algoritma dan Hasil Analisis

1) Rancang desain solusi atau algoritma :
- Identifikasi atribut fisik dan identitas manusia
- Identifikasi perilaku umum manusia
- Kategorikan atribut sebagai variabel
- Kategorikan perilaku sebagai method

2) Hasil Analisis :

a. Atribut variabel:
- Nama
- Umur
- Jenis kelamin
- Tinggi badan
- Berat badan
- Warna kulit
- Golongan darah

b. Perilaku/behavior untuk method:
- Berjalan()
- Berbicara()
- Makan()
- Tidur()
- Bekerja()

- Belajar()
- Berolahraga()

[Latihan 1.] Kesimpulan

1) Analisa
    a) Kesimpulan: Kelas Manusia dapat dimodelkan dengan atribut seperti nama, umur, jenis kelamin dll, serta method seperti berjalan, berbicara,makan dan lain-lain. Ini mencerminkan karakteristik dan perilaku umum manusia dalam konteks pemrograman berorientasi objek.
    b) Dasar pengambilan keputusan: Keputusan diambil berdasarkan karakteristik umum manusia dan prinsip-prinsip OOP.

Latihan 2

[Latihan 2.] Identifikasi Masalah:

1) **Uraikan permasalahan:**
   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?

2) **Rincikan sumber informasi yang relevan :**
   Java Documentation: *https://docs.oracle.com/javase/tutorial/java/concepts/*

3) **Uraikan rancangan Solusi yang diusulkan :**
   - Modifikasi constructor class Ortu dengan menambahkan parameter baru
   - Membuat daftar sifat, atribut, dan perilaku yang mungkin diturunkan

[Latihan 2.] Analisis dan Argumentasi

1) Solusi diatas memungkinkan representasi yang lebih detail dari objek Ortu, sekaligus memberikan gambaran tentang konsep pewarisan dalam OOP. Ini relevan dengan permasalahan karena menunjukkan bagaimana karakteristik orang tua dapat dimodelkan dan potensial diturunkan ke anak.

[Latihan 2.] Penyusunan Algoritma dan Hasil Analisa

1) **Algoritma:**
   - Modifikasi constructor class Ortu
   - parameter baru ke constructor
   - Tampilkan informasi lengkap di dalam constructor
   - Buat daftar atribut, constructor, dan perilaku yang dapat diturunkan

2) **Kode program dan luaran :**

```
public class Ortu {
  // Constructor yang dimodifikasi
  public Ortu(String nama, String rambut, int umur, double tinggi, String golonganDarah) {
    // Menampilkan informasi lengkap
    System.out.println("Nama saya : " + nama +
                       "\nWarna Rambut : " + rambut +
                       "\nUmur : " + umur + " tahun" +
                       "\nTinggi : " + tinggi + " cm" +
                       "\nGolongan Darah : " + golonganDarah);
  }

  public static void main(String[] args) {
    // Membuat objek Ortu dengan data lengkap
    Ortu satu = new Ortu("Renni", "hitam", 23, 160, "A");
  }
}
```

   a) **Screenshot/ Capture potongan kode dan hasil luaran**

```java
public class Ortu {
    // Constructor yang dimodifikasi
    public Ortu(String nama, String rambut, int umur, double tinggi, String golonganDarah) {
        // Menampilkan informasi lengkap
        System.out.println("Nama saya : " + nama +
                           "\nWarna Rambut : " + rambut +
                           "\nUmur : " + umur + " tahun" +
                           "\nTinggi : " + tinggi + " cm" +
                           "\nGolongan Darah : " + golonganDarah);
    }

    public static void main(String[] args) {
        // Membuat objek Ortu dengan data lengkap
        Ortu satu = new Ortu("Renni", "hitam", 23, 160, "A");
    }
}
```

Gambar 2.1 Kode pemrograman tentang *constructor*

Gambar 2.2 Luaran program diatas

b) Analisa sifat (atribut), constructor, dan perilaku positif (behavior) yang mungkin akan diturunkan:

1. Atribut:
    - Warna rambut
    - Warna mata
    - Struktur tulang
    - Predisposisi genetik tertentu
2. Constructor:
    - Nama
    - Tanggal lahir
3. Perilaku positif (behavior):
    - Kejujuran()
    - Kerja keras()
    - Sopan santun()
    - Kreativitas()
    - Kecerdasan()

[Latihan 2.] Kesimpulan

1) Analisa :

a) Kesimpulan:
    - Modifikasi constructor memungkinkan representasi yang lebih detail dari objek Ortu.
    - Atribut fisik seperti warna rambut dan golongan darah, serta karakteristik seperti nama dapat diturunkan.
    - Constructor dapat digunakan untuk menginisialisasi atribut-atribut ini.
    - Perilaku positif seperti kejujuran, kerja keras, dan sopan santun dapat dimodelkan sebagai method yang dapat diturunkan.
b) Dasar alasan pengambilan keputusan: Keputusan diambil berdasarkan prinsip-prinsip OOP, terutama konsep inheritance dan encapsulation.

Latihan 3

[Latihan 3.] Identifikasi Masalah:

1) Uraikan permasalahan :
   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:
   - constructor overloading dan overriding
   - method overloading, dan method overriding
   - method yang mengembalikan nilai dan method tidak mengembalikan nilai.
2) Uraikan rancangan solusi yang diusulkan:
   Solusi yang diusulkan adalah melakukan analisis komparatif antara constructor, method, dan method utama, serta menjelaskan konsep overloading dan overriding.

[Latihan 3.] Analisis dan Argumentasi:

1) Solusi yang diusulkan adalah membuat analisis terstruktur yang membandingkan dan membedakan berbagai aspek dari constructor, method, dan method utama, serta konsep terkait seperti overloading dan overriding**.** Solusi ini secara langsung menjawab permasalahan dengan memberikan pemahaman mendalam tentang komponen-komponen kunci dalam pemrograman berorientasi objek Java. Ini membantu dalam memahami kapan dan bagaimana menggunakan berbagai elemen OOP dengan tepat.

[Latihan 3.] Pembahasan

1) Analisa perbedaan deklarasi constructor, method, dan method utama!
   - Constructor:
     1. Nama sama dengan nama kelas
     2. Tidak memiliki tipe pengembalian
     3. Digunakan untuk inisialisasi objek
   - Method:
     1. Memiliki nama berbeda dari kelas
     2. Memiliki tipe pengembalian (void jika tidak mengembalikan nilai)
     3. Digunakan untuk menentukan perilaku objek
   - Method utama (main):
     1. Selalu dideklarasikan sebagai "public static void main(String[] args)"
     2. Merupakan titik masuk eksekusi program
     3. Tidak dipanggil secara eksplisit, dijalankan oleh JVM

2) Tentukan kapan Anda perlu menggunakan constructor dan method?
   Constructor digunakan saat membuat objek baru dan inisialisasi nilai awal atribut objek.

3) Uraikan perbedaan berikut:
   a) constructor overloading dan overriding:
      - Overloading: Membuat beberapa constructor dengan parameter berbeda dalam satu kelas
      - Overriding: Tidak berlaku untuk constructor, karena constructor tidak diwariskan
   b) method overloading, dan method overriding

- Overloading: Membuat beberapa method dengan nama sama tapi parameter berbeda dalam satu kelas
- Overriding: Membuat method di kelas anak dengan nama dan parameter sama seperti di kelas induk

c) method yang mengembalikan nilai dan method tidak mengembalikan nilai:
   - Mengembalikan nilai: Menggunakan tipe data spesifik (bukan void) dan menggunakan keyword "return" .
   - Tidak mengembalikan nilai: Menggunakan tipe void dan tidak menggunakan "return"

[Latihan 3.] Kesimpulan

1) Analisa:
   - Constructor digunakan untuk inisialisasi objek, tidak memiliki tipe pengembalian, dan namanya sama dengan nama kelas.
   - Method digunakan untuk mendefinisikan perilaku objek, dapat memiliki tipe pengembalian, dan namanya berbeda dari nama kelas.
   - Method utama (main) adalah titik masuk program Java, selalu dideklarasikan sebagai public static void main(String[] args).
   - Overloading terjadi ketika ada beberapa method dengan nama sama tapi parameter berbeda dalam satu kelas.
   - Overriding terjadi ketika method di kelas anak menggantikan method di kelas induk dengan nama dan parameter yang sama.
   - Method dapat mengembalikan nilai atau tidak (void).

Latihan 4

[Latihan 4.] Identifikasi Masalah:

1) Uraikan permasalahan :

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!

2) Rincikan sumber informasi yang relevan :

Website: Baeldung - Java Inheritance: *https://www.baeldung.com/java-inheritance*

3) Uraikan rancangan Solusi yang diusulkan :

Melakukan analisis perbandingan method antara class Ortu dan class Anak dan modifikasi kode dengan penambahan method baru di class Anak

[Latihan 4.] Analisis dan Argumentasi:

Solusi di atas memungkinkan pemahaman yang lebih dalam tentang inheritance dan polymorphism dalam Java. Dengan membandingkan method dan menambahkan method baru, kita dapat melihat bagaimana class anak dapat mewarisi, mengubah, dan menambah fungsionalitas dari class induknya.

[Latihan 4.] Penyusunan Algoritma dan Kode Program

1) Algoritma:

- Analisis method di class Ortu
- Analisis method di class Anak
- Bandingkan method-method tersebut
- Tambahkan method baru di class Anak
- Uji fungsionalitas dengan membuat objek dan memanggil method

2) Kode program dan luaran

a) Screenshot/ Capture potongan kode dan hasil luaran

```java
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");
        objekA.bermainGame("Mobile Legends"); // Method baru
    }
}

class Anak extends Ortu {
    void sukaMenonton(int a, String b) {
        System.out.println("Nonton Jam " + a + " Malam " + b);
    }
    void sukaMenonton(String a) {
        System.out.println("Nonton " + a);
    }
    void sukaMembaca(String a) {
        System.out.println("Suka Baca " + a);
    }
    void bermainGame(String game) { // Method baru
        System.out.println("Suka Bermain " + game);
    }

    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");
        objekA.bermainGame("Mobile Legends"); // Memanggil method baru
    }
}
```

Gambar 4.1 Kode program dengan tambahan method

Gambar 4.2 Luaran kode program diatas

Penjelasan :

Dari gamar 4.1 diatas merupakan sebuah modifikasi dengan menambahkan method baru bermainGame(String game) di kelas Anak untuk menunjukkan perbedaan tambahan antara kelas Anak dan Ortu.

b) Analisa Perbandingan method di class anak dan class ortu :

Perbandingan method di class Anak dan class Ortu:

- Class Ortu:
    1. sukaMenonton(String a)
    2. sukaMembaca(String a)
- Class Anak:
    1. sukaMenonton(int a, String b) – overloading
    2. sukaMenonton(String a) – overriding
    3. sukaMembaca(String a) - overriding

[Latihan 4.] Kesimpulan

1. Analisa

a) Kesimpulan:

- Class Anak mewarisi method dari class Ortu
- Class Anak dapat melakukan overriding pada method yang diwarisi
- Class Anak dapat menambahkan method baru yang tidak ada di class Ortu
- Overloading memungkinkan variasi parameter pada method dengan nama yang sama

b) Dasar alasan pengambilan keputusan: Keputusan diambil berdasarkan prinsip-prinsip inheritance dan polymorphism dalam OOP Java.