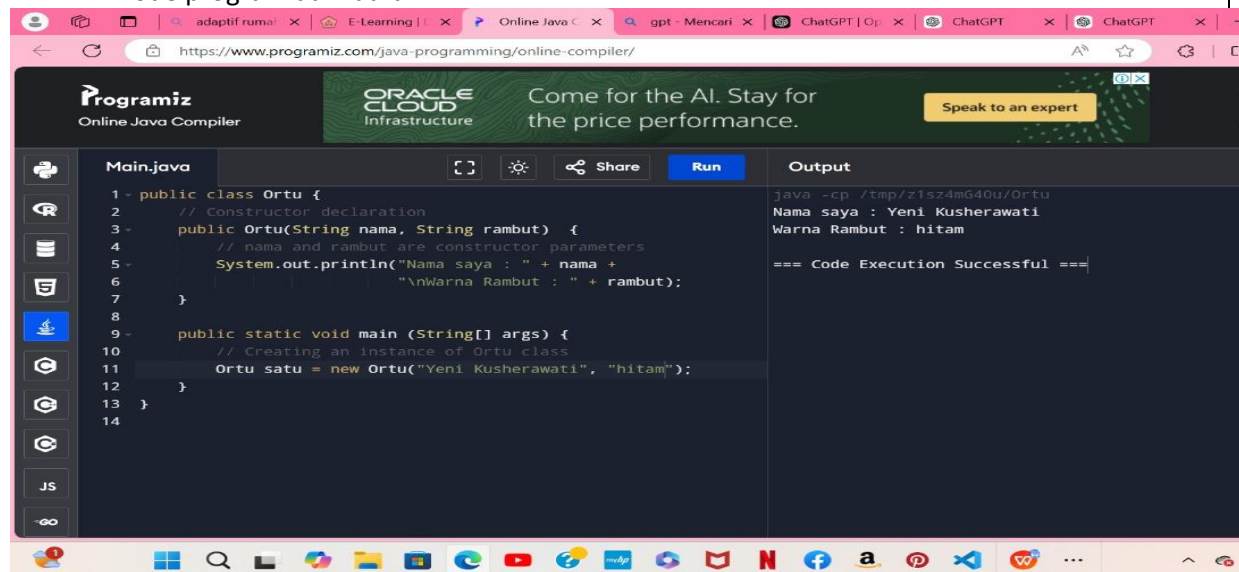


Nama & NPM	Topik:	Tanggal:
Yeni kusherawati G1F024013	Method pemrograman	17 september 2024
[Nomor 1] Identifikasi Masalah:		
<pre>public class Manusia { // deklarasi kelas // deklarasi variabel String nama; String rambut; // deklarasi constructor tanpa parameter public Manusia() { System.out.println("Kelas Manusia tanpa nama"); } }</pre> <p>Kode ini mendefinisikan sebuah kelas manusia dalam Java dengan variabel nama dan rambut, serta sebuah konstruktor tanpa parameter. Namun, terdapat beberapa potensi masalah dalam kode tersebut, seperti konstruktor tanpa parameter tidak menginisialisasi variabel, meskipun ada konstruktor yang mencetak pesan, tidak ada inisialisasi untuk variabel nama dan rambut. Sehingga, ketika objek manusia dibuat, kedua variabel tersebut akan berisi null secara default, kecuali diinisialisasi secara eksplisit di luar konstruktor. Solusinya adalah dengan inisialisasi variabel variabel nama dan rambut harus diinisialisasi, baik di dalam konstruktor tanpa parameter atau melalui konstruktor dengan parameter, agar tidak bernilai null.</p>		
[Nomor 1] Analisis dan Argumentasi		
<p>Latihan 1:</p> <p>1.2 Analisa ciri-ciri umum Kelas Manusia yang dapat menjadi</p> <ol style="list-style-type: none"> atribut variabel, dan perilaku/ behavior untuk method! <p>jawab:Analisa ciri-ciri umum dari kelas manusia yang dapat menjadi atribut variabel dan perilaku (behavior) untuk method.</p> <p>Atribut variabel</p> <ol style="list-style-type: none"> 1. Nama (`String nama`):Menyimpan nama individu. Ini adalah atribut dasar yang umum dimiliki oleh manusia. 2. Rambut (`String rambut`): Menyimpan informasi tentang warna atau jenis rambut individu. Ini bisa digunakan untuk menggambarkan karakteristik fisik. <p>Perilaku (Behavior) untuk method</p> <ol style="list-style-type: none"> 1.Menampilkan informasi:Metode ini bertugas untuk menampilkan informasi dasar tentang objek <code>Manusia</code>, seperti nama dan warna rambut. Ini berguna untuk memberikan gambaran tentang individu kepada pengguna atau sistem. 2.Mengubah nama:Metode ini memungkinkan pengguna untuk mengatur atau mengubah nama individu. Ini memberikan fleksibilitas untuk memperbarui atribut objek sesuai kebutuhan. 3.Mengubah rambut:Serupa dengan metode mengubah nama, metode ini digunakan untuk mengubah informasi tentang rambut, seperti warna atau jenis. Ini membantu dalam mendeskripsikan karakteristik fisik yang mungkin berubah. 		

[Nomor 1] Penyusunan Algoritma dan Kode Program

- 1 Algoritma
 - a. Mulai program
 - b. Buat kelas manusia dengan atribut nama dan rambut
 - c. Implementasikan constructor dan metode yang diperlukan
 - d. Buat objek manusia
 - e. Set manusia nama menjadi Yeni kusherwati
 - f. Set manusia rambut menjadi hitam
 - g. Tampilkan informasi
 - f. Akhiri program
- 2 kode program dan luaran



The screenshot shows the Programiz Online Java Compiler interface. The code editor on the left contains the following Java code:

```
1- public class Ortu {
2-     // Constructor declaration
3-     public Ortu(String nama, String rambut) {
4-         // nama and rambut are constructor parameters
5-         System.out.println("Nama saya : " + nama +
6-                             "\nWarna Rambut : " + rambut);
7-     }
8-
9-     public static void main (String[] args) {
10-        // Creating an instance of Ortu class
11-        Ortu satu = new Ortu("Yeni Kuserawati", "hitam");
12-    }
13- }
14
```

The output window on the right shows the execution results:

```
java -cp /tmp/z1sz4mG40u/Ortu
Nama saya : Yeni Kuserawati
Warna Rambut : hitam
=== Code Execution Successful ===
```

Luaran dari kode ini menunjukkan bagaimana objek diinisialisasi dan bagaimana informasi tentang objek tersebut dapat diubah dan ditampilkan.

[Nomor 1] Kesimpulan

- 1) Analisa

Dengan struktur yang sederhana dan jelas, kelas ini dapat diperluas dan dikembangkan lebih lanjut untuk memenuhi kebutuhan aplikasi yang lebih kompleks. Melalui penambahan atribut, metode, dan implementasi prinsip pemrograman berorientasi objek, kelas ini dapat berfungsi sebagai komponen integral dalam berbagai jenis aplikasi, merepresentasikan interaksi dan karakteristik manusia dengan lebih realistis dan efektif.

Nama & NPM	Topik:	Tanggal:
Yeni kusherawati G1F024013	Method pemrograman	17 september 2024
[Nomor 2] Identifikasi Masalah:		
<pre> 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"); } } </pre> <p>Luaran 2: Nama saya : Putri Warna Rambut : hitam</p> <p>jumlah parameter constructor <code>ortu</code> hanya menerima dua parameter (<code>nama</code> dan <code>rambut</code>), tetapi saat pembuatan objek, lima argumen disediakan. Ini menyebabkan ketidakcocokan. Dan tipe data tidak konstiten salah satu argumen (156) adalah tipe <code>int</code>, sementara semua parameter dalam constructor diharapkan bertipe <code>String</code>. Ini akan menyebabkan kesalahan saat kompilasi.</p> <p>Solusinya ada</p> <p>Perbarui constructor tambahkan parameter tambahan dalam constructor untuk mencakup semua informasi yang ingin disertakan, seperti warna kulit dan usia. Dan sesuaikan pemanggilan constructor Setelah memperbaiki constructor, pastikan pemanggilan objek menggunakan jumlah dan tipe parameter yang sesuai.</p>		
[Nomor 2] Analisis dan Argumentasi		
<p>Latihan 2: 2.1. Susun kembali kode di contoh 2 dengan menambahkan data ciri-ciri Anda di dalam variabel constructor!</p> <p>jawab: Kelas <code>ortu</code> dideklarasikan sebagai kelas publik, yang berarti dapat diakses dari kelas lain dalam program. Kelas ini merepresentasikan diri, dengan atribut yang bisa mendeskripsikan ciri-ciri fisik dan preferensi.</p> <p>Kode yang diperbarui</p> <ul style="list-style-type: none"> • Nama: Yeni kusherawati (nama saya). • Rambut: Hitam (warna rambut). • Warna Kulit: Sawo matang (warna kulit). • Tinggi badan: 156(tinggi badan) • Kebiasaan:Drakor(kebiasaan) <p>Kode ini mencetak ciri-ciri saya .</p>		

2.2. Apabila nanti Anda akan memiliki keturunan, analisa sifat (atribut), constructor, dan perilaku positif (behavior) apa yang akan diturunkan?

jawab: analisis mengenai sifat (atribut), constructor, dan perilaku positif (behavior) yang dapat diturunkan:

Sifat atribut yang diturunkan

1. Nama: Nama keturunan dapat diambil dari nama orang tua.
2. Warna Rambut: Warna atau jenis rambut bisa diturunkan, misalnya coklat atau hitam.
3. Warna Kulit: Ciri fisik seperti warna kulit, yang dapat dipengaruhi oleh gen dari kedua orang tua.
4. Usia: Meskipun usia bukan atribut yang diturunkan, anak-anak akan memiliki usia yang berkembang seiring waktu.
5. Minat atau Hobi: Keturunan mungkin mewarisi minat atau hobi tertentu dari orang tua.

Constructor untuk keturunan

a.Pendidikan: Pendidikan yang diharapkan atau diinginkan.

b.Hobi: Minat yang dapat diturunkan atau dibentuk dari pengalaman orang tua

[Nomor 2] Penyusunan Algoritma dan Kode Program

1.Algoritma

a.Mulai

b.Definisikan kelas ortu

c.Definisikan kelas keturunan

e.Dalam main

f.Selesai

2.Kode program dan luaran

The screenshot shows the Programiz Online Java Compiler interface. On the left, the 'Main.java' file contains the following code:

```
1 int tinggiBadan;
2
3 // Deklarasi constructor
4 public Ortu(String nama, String rambut, String warnaKulit,
5     int tinggiBadan, String kebiasaan) {
6
7     // Menampilkan informasi ciri-ciri
8     System.out.println("Nama saya: " + nama +
9         "\nWarna Rambut: " + rambut +
10        "\nWarna Kulit: " + warnaKulit +
11        "\nTinggi Badan: " + tinggiBadan + " cm" +
12        "\nKebiasaan: " + kebiasaan);
13
14 }
15
16 // Method utama
17 public static void main(String[] args) {
18     // Membuat objek Ortu dengan ciri-ciri lengkap
19     Ortu satu = new Ortu("Yeni Kuserawati ", "hitam",
20         "sawo matang", 156, "drakor");
21 }
22
```

On the right, the 'Output' panel shows the result of the code execution:

```
java -cp /tmp/9SyEAUy2wS/Ortu
Nama saya: Yeni Kuserawati
Warna Rambut: hitam
Warna Kulit: sawo matang
Tinggi Badan: 156 cm
Kebiasaan: drakor

=== Code Execution Successful ===
```

Luaran dari kodingan ini memberikan gambaran yang jelas tentang karakteristik orang tua dan keturunannya.

[Nomor 2] Kesimpulan

1)Analisa

codingan yang disusun untuk kelas ortu dan potensi keturunan mencerminkan prinsip-prinsip dasar pemrograman berorientasi objek (OOP) dengan cara yang terstruktur dan terorganisir. Kelas ortu didefinisikan dengan atribut yang mencerminkan ciri-ciri individu, seperti:

Nama: Mewakili identitas individu.

Warna Rambut: Menyediakan informasi fisik yang spesifik

Warna Kulit: Menunjukkan ciri fisik yang juga relevan dalam konteks sosial dan budaya.

Usia: Memberikan konteks temporal, yang penting untuk memahami fase kehidupan individu.

Genre Favorit: Mencerminkan minat dan preferensi pribadi.

Representasi ini memungkinkan untuk menciptakan objek yang mudah dipahami, serta memberikan gambaran yang komprehensif mengenai karakteristik individu. Penggunaan atribut yang relevan ini mencerminkan kekuatan OOP dalam memodelkan objek dunia nyata secara realistis.

Nama & NPM	Topik:	Tanggal:
Yeni kusherawati G1F024013	Method pemrograman	17 september 2024
[Nomor 3] Identifikasi Masalah:		
<pre>public class Manusia { //deklarasi atribut Manusia dalam variabel String nama, rambut; //deklarasi constructor public Manusi1(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"); } }</pre> <p>Luaran 3: Nama saya : Putri Warna Rambut : hitam Hobi Menonton : Drakor</p> <p>Kesalahan Nama Constructor:Constructor dalam kodingan diberi nama manusia , sementara nama kelasnya adalah manusia . Dalam Java, nama constructor harus sesuai dengan nama kelas. Ini akan menyebabkan kesalahan saat mencoba membuat objek dari kelas ini.</p> <p>Validasi Input di Metode:Metode suka nonton berfungsi untuk mencetak hobi menonton, tetapi tidak ada validasi terhadap input film . Jika nilai yang diberikan kosong atau tidak valid, akan tetap mencetak output yang mungkin tidak diinginkan.</p> <p>Solusinya adalah</p> <ol style="list-style-type: none">1. Perbaiki Nama Constructor:Ganti nama constructor dari Manusi1 menjadi Manusia untuk mencocokkan dengan nama kelas. Hal ini akan memungkinkan objek kelas Manusia dibuat dengan benar.2. (Opsional) Penambahan Validasi:Tambahkan logika dalam metode sukaNonton untuk memeriksa apakah parameter film tidak kosong atau null. Jika input tidak valid, program bisa mencetak pesan yang menyatakan bahwa hobi menonton tidak ditentukan.		
[Nomor 3] Analisis dan Argumentasi		
<p>Latihan 3:</p> <p>3.1. Analisa perbedaan deklarasi constructor, method, dan method utama! jawab: Constructor berfungsi untuk menginisialisasi objek saat dibuat, tanpa tipe pengembalian dan harus sesuai dengan nama kelas. Method adalah fungsi yang melakukan operasi tertentu, memiliki nama fleksibel, tipe pengembalian, dan bisa menerima parameter. Method utama adalah titik masuk program, dengan nama yang tetap (main), tidak mengembalikan nilai, dan menerima argumen untuk eksekusi.Masing-masing memiliki peran yang berbeda dalam pengembangan kelas dan objek, dan pemahaman yang jelas mengenai perbedaan ini penting untuk pengembangan aplikasi yang efektif.</p>		

3.2. Tentukan kapan Anda perlu menggunakan constructor dan method?

jawab: Constructor diperlukan untuk inisialisasi atribut objek saat objek dibuat, memastikan bahwa objek dalam keadaan yang valid dan siap digunakan.

Method digunakan untuk melakukan tindakan, memproses data, dan mengubah status objek, memberikan fleksibilitas dan modularitas dalam desain kode.

Kedua elemen ini penting dalam pengembangan perangkat lunak yang efisien dan terstruktur, dan pemahaman kapan harus menggunakan masing-masing akan membantu dalam merancang kelas yang baik.

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

jawab: Constructor overloading memungkinkan variasi dalam cara objek diinisialisasi, sedangkan constructor overriding tidak ada.

Method overloading memungkinkan variasi dalam cara metode dipanggil berdasarkan parameter, sementara method overriding memberikan implementasi spesifik di subclass.

Method yang mengembalikan nilai memberikan hasil yang dapat digunakan lebih lanjut, sedangkan method yang tidak mengembalikan nilai hanya melakukan tindakan tanpa memberikan hasil.

Pemahaman tentang konsep-konsep ini penting untuk mendesain dan mengimplementasikan kode yang bersih dan efektif dalam pemrograman berorientasi objek.

[Nomor 3] Penyusunan Algoritma dan Kode Program

1. Algoritma

a. Mulai

b. Deklarasi kelas

c. Deklarasi atribut

e. Deklarasi constructor

f. Deklarasi method

g. Method utama

h. Selesai

2. Kode program dan luaran

The screenshot shows the Programiz Online Java Compiler interface. The left pane displays the source code for 'Main.java'. The right pane shows the output of the program execution.

```
3 String nama, rambut;
4
5 // deklarasi constructor
6 public Manusia(String nama, String rambut) {
7     System.out.println("Nama saya: " + nama +
8         "\nWarna Rambut: " + rambut);
9 }
10
11 // deklarasi method
12 void sukaNonton(String film) {
13     System.out.println("Hobi Menonton: " + film);
14 }
15
16 // deklarasi method utama
17 public static void main(String[] args) {
18     // Membuat objek Manusia dan memanggil constructor
19     Manusia satu = new Manusia("Yeni Kusherawati", "hitam");
20     satu.sukaNonton("nonton drakor");
21 }
```

The output pane shows the following text:

```
java -cp /tmp/f476T3B1hA/Manusia
Nama saya: Yeni Kusherawati
Warna Rambut: hitam
Hobi Menonton: nonton drakor
=== Code Execution Successful ===
```

Luaran ini memberikan gambaran yang jelas mengenai identitas objek manusia , termasuk nama, warna rambut, dan hobi menontonnya. Codingan ini berhasil mengilustrasikan cara kerja constructor untuk menginisialisasi objek dan method untuk menampilkan informasi tambahan.

[Nomor 3] Kesimpulan

1)Analisa

Codingan ini berhasil mengilustrasikan prinsip-prinsip dasar pemrograman berorientasi objek, penggunaan constructor untuk inisialisasi, dan penggunaan method untuk memberikan fungsionalitas tambahan. Codingan ini juga memberikan contoh yang jelas tentang bagaimana informasi dapat disampaikan melalui output, menjadikannya efektif untuk mendemonstrasikan konsep-konsep kunci dalam pemrograman Java. Luaran dari kodingan ini memberikan informasi yang jelas dan terstruktur tentang objek yang dibuat. Penggunaan output dari constructor dan method menciptakan narasi yang informatif tentang objek manusia.

Nama & NPM	Topik:	Tanggal:
Yeni kusherawati G1F024013	Method pemrograman	17 september 2024
[Nomor 4] Identifikasi Masalah:		
<p>Codingan pada nomor 4 mencakup kelas Ortu dan kelas Anak yang mewarisi dari kelas Ortu. Berikut adalah identifikasi masalah yang terdapat dalam codingan ini beserta solusinya:</p> <p>Masalah 1: Duplikasi Method sukaMenonton di Kelas Anak</p> <p>Identifikasi: Dalam kelas `Anak`, terdapat dua deklarasi method `sukaMenonton`. Satu adalah overload dengan parameter `int a, String b`, dan yang lainnya sama persis dengan method di kelas `Ortu`, yaitu `void sukaMenonton(String a)`. Ini menciptakan kebingungan, karena Java tidak bisa membedakan antara kedua method ini tanpa konteks yang jelas.</p> <p>Solusi: Hapus salah satu dari deklarasi method `sukaMenonton`. Jika ingin tetap memiliki method yang sama dengan kelas induk, cukup override method tersebut dengan implementasi yang baru.</p> <p>Masalah 2: Penulisan `main` Method di Kelas `Anak`</p> <p>Identifikasi: Kelas `Anak` mendeklarasikan method `main`, tetapi hal ini tidak perlu karena method `main` sudah didefinisikan di kelas `Ortu`. Menjalankan program akan menyebabkan duplikasi dan bisa membingungkan dalam konteks eksekusi program.</p> <p>Solusi: Hapus method `main` dari kelas `Anak` sehingga hanya ada satu titik masuk (method `main`) yang didefinisikan dalam kelas `Ortu`.</p> <p>Masalah 3: Tidak Ada Panggilan untuk Method yang Di-overridden</p> <p>Identifikasi: Di dalam kelas `Anak`, meskipun ada method `sukaMembaca` yang di-overridden, tidak ada penggunaan method tersebut yang berbeda dari induk. Hal ini mungkin tidak memanfaatkan fitur overriding secara maksimal.</p> <p>Solusi: Jika ada perubahan logika atau output yang diinginkan, tambahkan implementasi baru dalam method `sukaMembaca` di kelas `Anak`.</p> <p>Kesimpulan</p> <p>Codingan ini mengilustrasikan konsep pewarisan dalam pemrograman berorientasi objek, di mana kelas `Anak` mewarisi metode dari kelas `Ortu`. Dengan memperbaiki masalah yang diidentifikasi, program dapat berjalan dengan lancar dan memberikan output yang jelas dan informatif mengenai sifat dari orang tua dan anak. Ini menekankan pentingnya pengelolaan method dan pemahaman yang baik mengenai prinsip pewarisan untuk menjaga kode tetap bersih dan terstruktur.</p>		

[Nomor 4] Analisis dan Argumentasi

Latihan 4:

4.1. Bandingkan method yang dimiliki class `Anak` extends `Ortu` dengan method di class `Ortu`! jawab: Overloading dan Overriding: Kelas `Anak` memiliki kemampuan untuk melakukan overloading method `sukamenonton`, memperkenalkan varian baru yang tidak ada di kelas `Ortu`. Selain itu, kelas `Anak` juga dapat mengoverride method dari kelas `Ortu`, memberikan fleksibilitas dalam mendefinisikan perilaku yang berbeda..Fleksibilitas: Method di kelas `Anak` menunjukkan bagaimana subclass dapat mengubah atau memperluas perilaku dari superclass, yang merupakan inti dari prinsip pewarisan dalam pemrograman berorientasi objek..Implementasi: Untuk memanfaatkan sepenuhnya potensi overriding, kelas `Anak` dapat diubah untuk memberikan logika yang lebih unik atau berbeda dari kelas `Ortu`.Dengan demikian, perbandingan ini menunjukkan bagaimana subclass dapat memperluas dan mengubah perilaku yang diwarisi. Ubahlah dari superclass untuk memenuhi kebutuhan spesifik.

4.2.Contoh 4 dengan menambahkan objek anak dengan method yang berbeda!

jawab: Modifikasi ini memperlihatkan pentingnya pewarisan dan polymorphism dalam pemrograman berorientasi objek. Kelas `Anak` dapat memperluas atau memodifikasi perilaku yang diwarisi dari kelas `Ortu`, sekaligus menambahkan method baru yang relevan dengan karakteristiknya sendiri. Dengan demikian, kode ini mencerminkan bagaimana subclass dapat berinteraksi dengan superclass dan menambah kompleksitas serta variasi pada objek yang didefinisikan.

[Nomor 4] Penyusunan Algoritma dan Kode Program

1.Algoritma

- a.Mulai
- b.Buat kelas `Ortu`
- c.Buat kelas `Anak`(extends `Ortu`)
- d.Dalam main
- e.Cetak sifat `Anak`
- f.Selesai

2.Kode program dan luaran

```
11 System.out.println("Sifat Orang Tua :");
12 Ortu objek0 = new Ortu(); // Creating an instance of Ortu
13 objek0.sukaMenonton("Berita"); // Calls the method from Ortu
14 objek0.sukaMembaca("Koran"); // Calls the method from Ortu
15
16 System.out.println("\nSifat Anak :");
17 Anak objekA = new Anak(); // Creating an instance of Anak
18 objekA.sukaMenonton(9, "Film Drakor"); // Calls the overloaded method in Anak
19 objekA.sukaMembaca("Komik One Piece"); // Calls the inherited method from Ortu
20 objekA.hobi("bersepeda"); // Calls the new method in Anak
21 }
22 }
23
24 class Anak extends Ortu {
25
```

Output

```
java -cp /tmp/FwJjHhXHA3/Ortu
Sifat Orang Tua :
Nonton Berita
Suka Baca Koran

Sifat Anak :
Nonton Jam 9 Malam Film Drakor
Suka Baca Komik One Piece
Hobi saya adalah bersepeda

=== Code Execution Successful ===
```

Luaran dari kode ini tidak hanya memberikan informasi tentang kebiasaan menonton dan membaca, tetapi juga mencerminkan bagaimana struktur kelas dapat mendukung variasi dalam perilaku objek yang berbeda.

[Nomor 4] Kesimpulan

1)Analisa

Kode ini secara efektif menggambarkan konsep-konsep utama dalam pemrograman berorientasi objek, seperti pewarisan, overriding, dan polymorphism. Melalui interaksi antara kelas `Ortu` dan `Anak`, program ini menunjukkan bagaimana perilaku dapat diwariskan dan dimodifikasi, serta bagaimana objek dapat memiliki karakteristik unik meskipun berasal dari kelas yang sama. Ini menekankan pentingnya struktur yang baik dalam desain kelas untuk memungkinkan fleksibilitas dan ekspansi dalam pengembangan perangkat lunak.