

BAB 8

INTERFACE

Tujuan

1. Mampu memahami konsep Interface dalam pemrograman Java
2. Mampu mengimplementasikan konsep Interface dalam menyelesaikan permasalahan

Ringkasan Materi

A. Pengertian Interface

Pada bab sebelumnya telah dijelaskan bahwa polymorphism hanya memungkinkan untuk mewarisi sifat dari satu kelas abstrak saja, namun tidak jarang juga diperlukan untuk mewarisi sifat lebih dari satu class abstrak, atau biasa disebut juga dengan multiple inheritance. Dalam Java, tidak mendukung multiple inheritance, artinya sebuah class hanya bisa mewarisi sifat dari sebuah superclass atau yang biasa disebut dengan single inheritance. Sebuah Java class tersebut tidak mendukung prinsip multiple inheritance. Dalam memenuhi prinsip kebutuhan multiple inheritance, pada Java terdapat Interfaces yang memiliki aspek seperti multiple inheritance namun berbeda dengan abstract class. Sebuah interface hanya memiliki konstanta dan abstract method. Syntax untuk mendeklarasikan interface adalah sebagai berikut:

```
Modifier interface Interface_name {
    /** Konstanta */
    /** Abstract Method */
}
```

B. Implementasi Interface

Sebuah kelas dapat mengimplementasikan sebuah interface dengan menggunakan kata kunci *implements* sebagai berikut.

```
Modifier class class_name implements Interface_name {
    /** Attribute */
    /** Method */
}
```

Penggunaan kata kunci implements dapat lebih dari 2

```
Modifier    class    class_name    implements    Interface_name1,
interface_name2 {
    /** Attribute */
    /** Method */
}
```

Pelaksanaan Percobaan

A. Interface

Identitas.java	
1	package latihaninterface;
2	public interface Identitas {
3	public void tampilkanNama();
4	public void tampilkanUmur();
5	}

Manusia.java

```

1 package latihaninterface;
2 public class Manusia implements MakhlukHidup, Identitas {
3
4     private String nama;
5     private int umur;
6
7     @Override
8     public void makan() {
9         System.out.println("Makan pakai sendok garpu");}
10
11     @Override
12     public void berjalan() {
13         System.out.println("Jalan pakai dua kaki");}
14
15     @Override
16     public void bersuara() {
17         System.out.println("Suaranya merdu");}
18
19     @Override
20     public void tampilkanNama() {
21         System.out.println("Nama saya: " + this.nama);}
22
23     @Override
24     public void tampilkanUmur() {
25         System.out.println("Umur saya: " + this.umur);}
26
27 }

```

Hewan.java

```

1 package latihaninterface;
2 public class Hewan implements MakhlukHidup, Identitas {
3
4     @Override
5     public void makan() {
6         System.out.println("Makan pakai tangan dan mulut");
7     }
8
9     @Override
10    public void berjalan() {
11        System.out.println("Jalan pakai 4 kaki");
12    }
13
14    @Override
15    public void bersuara() {
16        System.out.println("Suaranya nggak jelas");
17    }
18
19    public void tampilkanNama (){}
20
21    public void tampilkanUmur () {}
22 }

```

MakhlukHidup.java

```

1 package latihaninterface;
2 public interface MakhlukHidup {
3     public void makan();
4     public void berjalan();
5     public void bersuara();
6 }

```

Data dan Analisis hasil percobaan

Pertanyaan

```

1 package praktikumpl;
2
3 public interface Colorable {
4     public void howToColor();
5 }
6 public interface Comparable
7
8 {
9     public void compareTo(Object obj);
10 }
11 public class Rectangle implements Colorable, Comparable{ // lass
12     rectanggle
13     private String warna;
14     private int kategori;
15
16     public Rectangle() {
17     }
18
19     public Rectangle(String warna) {
20         this.warna = warna;
21     }
22
23     public void howToColor() {
24         if(this.warna == null){
25             System.out.println("tidak ada warna, warna bangun kotak
26 masih polos");
27         }
28         else{
29             System.out.println("bangun kotak sudah diwarnai dengan
30 warna "+this.warna);
31         }
32     }
33
34     public void compareTo(Object obj) {
35         this.kategori = (int) obj;
36         if(this.kategori == 0){
37             System.out.println("ukuran cat yang cocok untuk bangun
38 kotak dengan ukuran kategori " +this.kategori+" yaitu 2.5L" );
39         }
40         else{
41             System.out.println("ukuran cat yang cocok untuk bangun kotak
42 dengan ukuran kategori " +this.kategori+" yaitu 6.5L" );
43         }
44     }
45 }
46 public static void main(String[] args) {
47
48     Rectangle kotak1 = new Rectangle("merah");
49     Rectangle kotak2= new Rectangle();
50     Rectangle kotak3 = new Rectangle();
51     kotak1.howToColor();
52     kotak2.howToColor();
53     kotak3.compareTo(4);
54
55 }

```

1. Lakukan percobaan diatas dan benahi jika menemukan kesalahan serta jelaskan!

Cuma terdapat kesalahan pada class main nya =

```

J Main.java > Main
Run | Debug
1 public static void main(String[] args) {
2     Rectangle kotak1 = new Rectangle(warna:"merah");
3     Rectangle kotak2 = new Rectangle();
4     Rectangle kotak3 = new Rectangle();
5     kotak1.howToColor();
6     kotak2.howToColor();
7     kotak3.compareTo(obj:4);
8 }
9

```

Perbaikan yang saya lakukan adalah dengan menambahkan =

```

J Main.java > Main
Run | Debug
1 public class Main{ //tambahkan
2     public static void main(String[] args) {
3         Rectangle kotak1 = new Rectangle(warna:"merah");
4         Rectangle kotak2 = new Rectangle();
5         Rectangle kotak3 = new Rectangle();
6         kotak1.howToColor();
7         kotak2.howToColor();
8         kotak3.compareTo(obj:4);
9     }
10 }

```

Outputannya =

```

bangun kotak sudah diwarnai dengan warna merah
tidak ada warna, warna bangun kotak masih polos
ukuran cat yang cocok untuk bangun kotak dengan ukuran kategori 4 yaitu 6.5L

```

2. Apakah class yang berbentuk Interface bisa diinstansiasi menjadi sebuah objek? Jelaskan alasannya!

Class yang berbentuk Interface tidak bisa diinstansiasi menjadi sebuah objek karena Interface hanya mendefinisikan metode-metode tanpa memberikan implementasi konkrit. Interface berfungsi sebagai kontrak yang harus dipatuhi oleh kelas-kelas yang mengimplementasinya, memastikan bahwa mereka menyediakan implementasi spesifik untuk metode-metode yang didefinisikan dalam Interface.

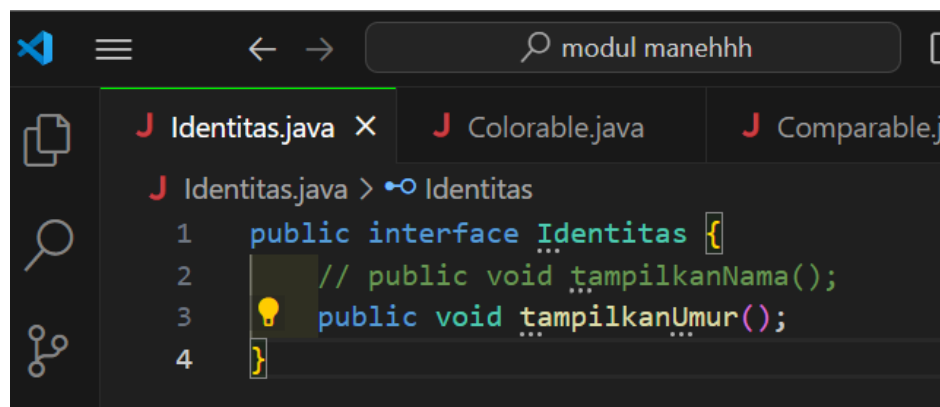
Alasan utama mengapa sebuah interface tidak bisa diinstansiasi menjadi sebuah objek adalah karena interface hanya mendefinisikan kontrak metode tanpa menyediakan implementasi konkrit untuk metode-metode tersebut.

3. Apakah suatu class dapat mengimplementasi class interface yang jumlahnya lebih dari satu? Jelaskan alasannya!

Bisa, sebuah class dapat mengimplementasikan lebih dari satu interface. Alasan utamanya adalah karena interface mendefinisikan kontrak, bukan implementasi, sehingga memungkinkan sebuah class untuk mematuhi beberapa kontrak sekaligus.

4. Pada interface Identitas.java hapus method tampilkan nama, amati apa yang terjadi dan mengapa demikian?

Jika kita menghapus =

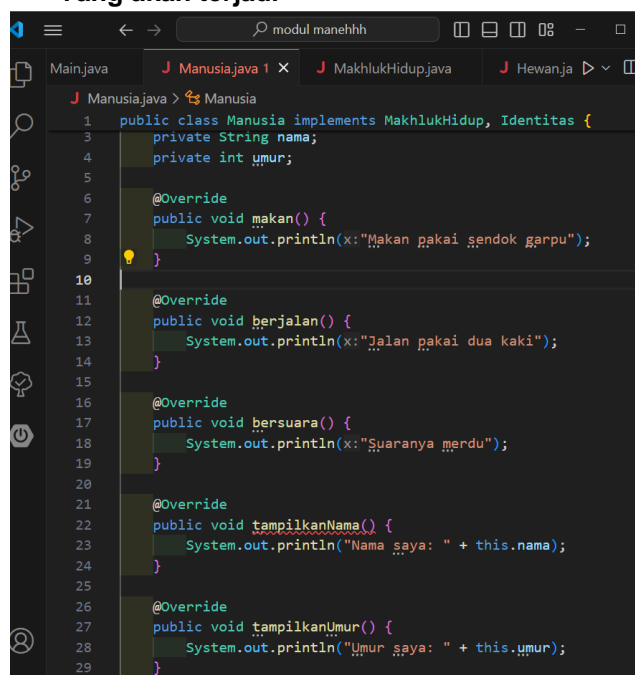


```

1 public interface Identitas {
2     // public void tampilkanNama();
3     public void tampilkanUmur();
4 }

```

Yang akan terjadi =



```

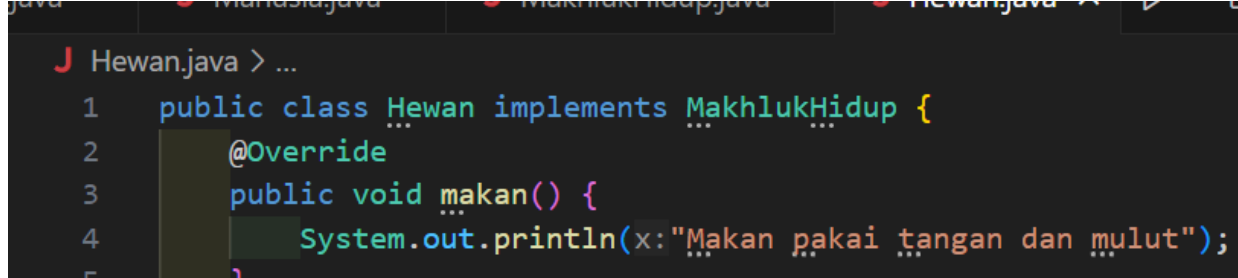
1 public class Manusia implements MakhlukHidup, Identitas {
2     private String nama;
3     private int umur;
4
5
6     @Override
7     public void makan() {
8         System.out.println(x: "Makan pakai sendok garpu");
9     }
10
11     @Override
12     public void berjalan() {
13         System.out.println(x: "Jalan pakai dua kaki");
14     }
15
16     @Override
17     public void bersuara() {
18         System.out.println(x: "Suaranya merdu");
19     }
20
21     @Override
22     public void tampilkanNama() {
23         System.out.println("Nama saya: " + this.nama);
24     }
25
26     @Override
27     public void tampilkanUmur() {
28         System.out.println("Umur saya: " + this.umur);
29     }

```

Akan terjadi error pada class manusia pada bagian tampilkanNama. Hal ini terjadi karena kelas Manusia mewarisi dari interface Identitas dan secara implisit menjanjikan untuk mengimplementasikan semua metode yang dideklarasikan dalam antarmuka tersebut.

Jadi, ketika Anda menghapus metode `tampilkanNama()` dari antarmuka `Identitas`, kelas `Manusia` tidak lagi memenuhi kontrak tersebut, yang menghasilkan kesalahan kompilasi.

5. Jika pada class hewan kita hanya ingin mengimplements interface `MakhlukHidup` saja apa yang terjadi? Jelaskan



```

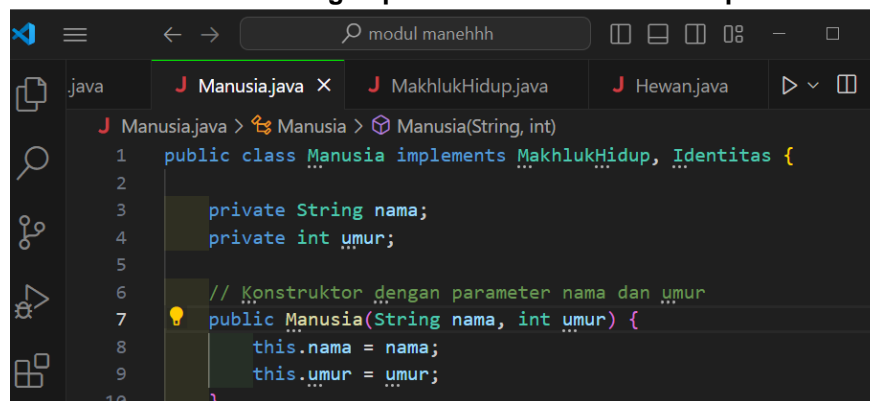
1 public class Hewan implements MakhlukHidup {
2     @Override
3     public void makan() {
4         System.out.println(x:"Makan pakai tangan dan mulut");
5     }

```

Tidak akan ada kesalahan kompilasi karena kelas `Hewan` hanya perlu mengimplementasikan metode yang dideklarasikan dalam antarmuka `MakhlukHidup`.

6. Buatlah konstruktor pada manusia dengan parameter umur dan nama kemudian panggil pada Class Main dengan menginstan objek bernama nama anda!

Menambahkan konstruktor dengan parameter nama dan umur pada class manusia =

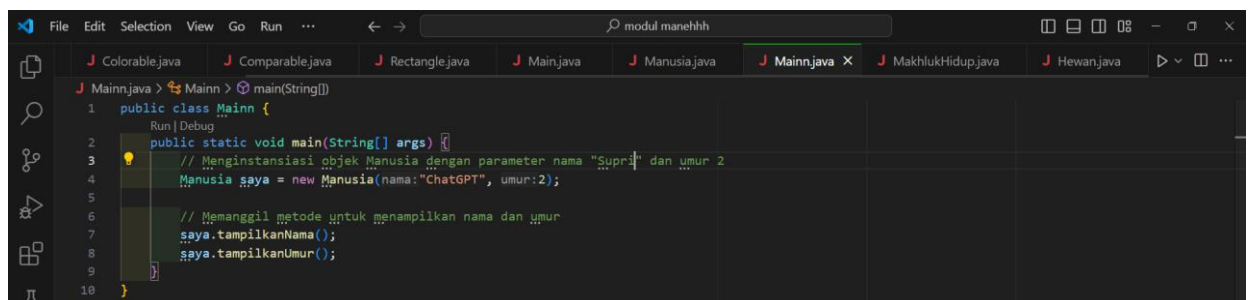


```

1 public class Manusia implements MakhlukHidup, Identitas {
2
3     private String nama;
4     private int umur;
5
6     // Konstruktor dengan parameter nama dan umur
7     public Manusia(String nama, int umur) {
8         this.nama = nama;
9         this.umur = umur;
10    }

```

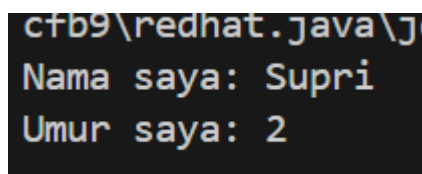
Membuat class main untuk menginstansiasi objek manusia=



```

1 public class Mainn {
2     public static void main(String[] args) {
3         // Menginstansiasi objek Manusia dengan parameter nama "Supri" dan umur 2
4         Manusia saya = new Manusia(nama:"ChatGPT", umur:2);
5
6         // Memanggil metode untuk menampilkan nama dan umur
7         saya.tampilkanNama();
8         saya.tampilkanUmur();
9     }
10 }

```



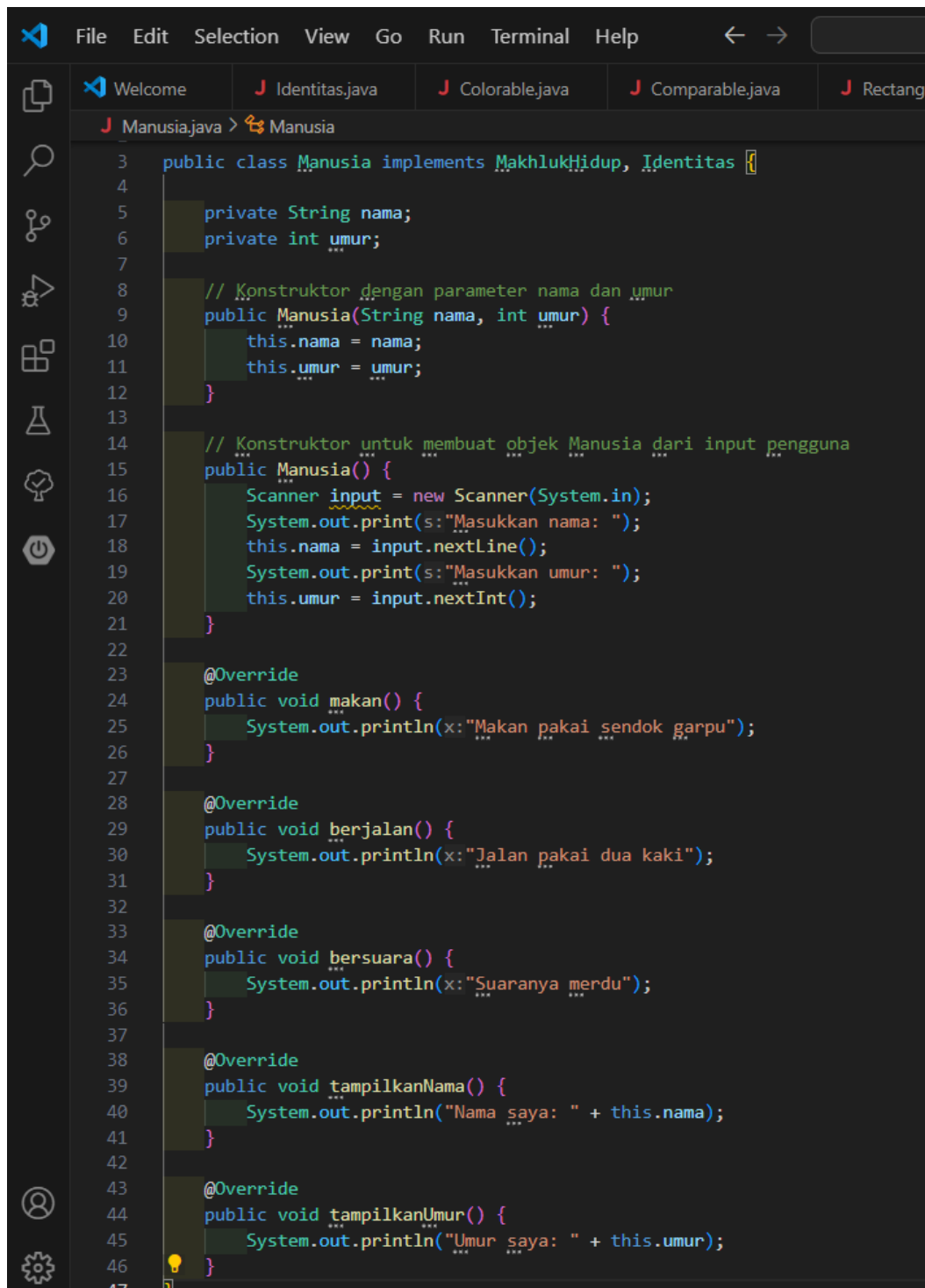
```

c:\b9\redhat.java\J
Nama saya: Supri
Umur saya: 2

```

- Ubah source code diatas menjadi proses meminta inputan dari user dan buat menjadi interaktif!

Class manusia =



```

3 public class Manusia implements MakhlukHidup, Identitas {
4
5     private String nama;
6     private int umur;
7
8     // Konstruktor dengan parameter nama dan umur
9     public Manusia(String nama, int umur) {
10         this.nama = nama;
11         this.umur = umur;
12     }
13
14     // Konstruktor untuk membuat objek Manusia dari input pengguna
15     public Manusia() {
16         Scanner input = new Scanner(System.in);
17         System.out.print(s:"Masukkan nama: ");
18         this.nama = input.nextLine();
19         System.out.print(s:"Masukkan umur: ");
20         this.umur = input.nextInt();
21     }
22
23     @Override
24     public void makan() {
25         System.out.println(x:"Makan pakai sendok garpu");
26     }
27
28     @Override
29     public void berjalan() {
30         System.out.println(x:"Jalan pakai dua kaki");
31     }
32
33     @Override
34     public void bersuara() {
35         System.out.println(x:"Suaranya merdu");
36     }
37
38     @Override
39     public void tampilkanNama() {
40         System.out.println("Nama saya: " + this.nama);
41     }
42
43     @Override
44     public void tampilkanUmur() {
45         System.out.println("Umur saya: " + this.umur);
46     }
47

```

Class mainn =

```

1  import java.util.Scanner;
2
3  public class Mainn {
4      public static void main(String[] args) {
5          Scanner input = new Scanner(System.in);
6
7          // Membuat objek Manusia dari input pengguna
8          Manusia saya = new Manusia();
9
10         // Memanggil metode untuk menampilkan nama dan umur
11         saya.tampilkanNama();
12         saya.tampilkanUmur();
13
14         input.close();
15     }
16 }

```

Output=

```

Masukkan nama: Eka
Masukkan umur: 1035
Nama saya: Eka
Umur saya: 1035

```

8. Buat objek selain objek diatas dengan menggunakan method yang berbeda dengan yang diatas! (min.1 contoh)

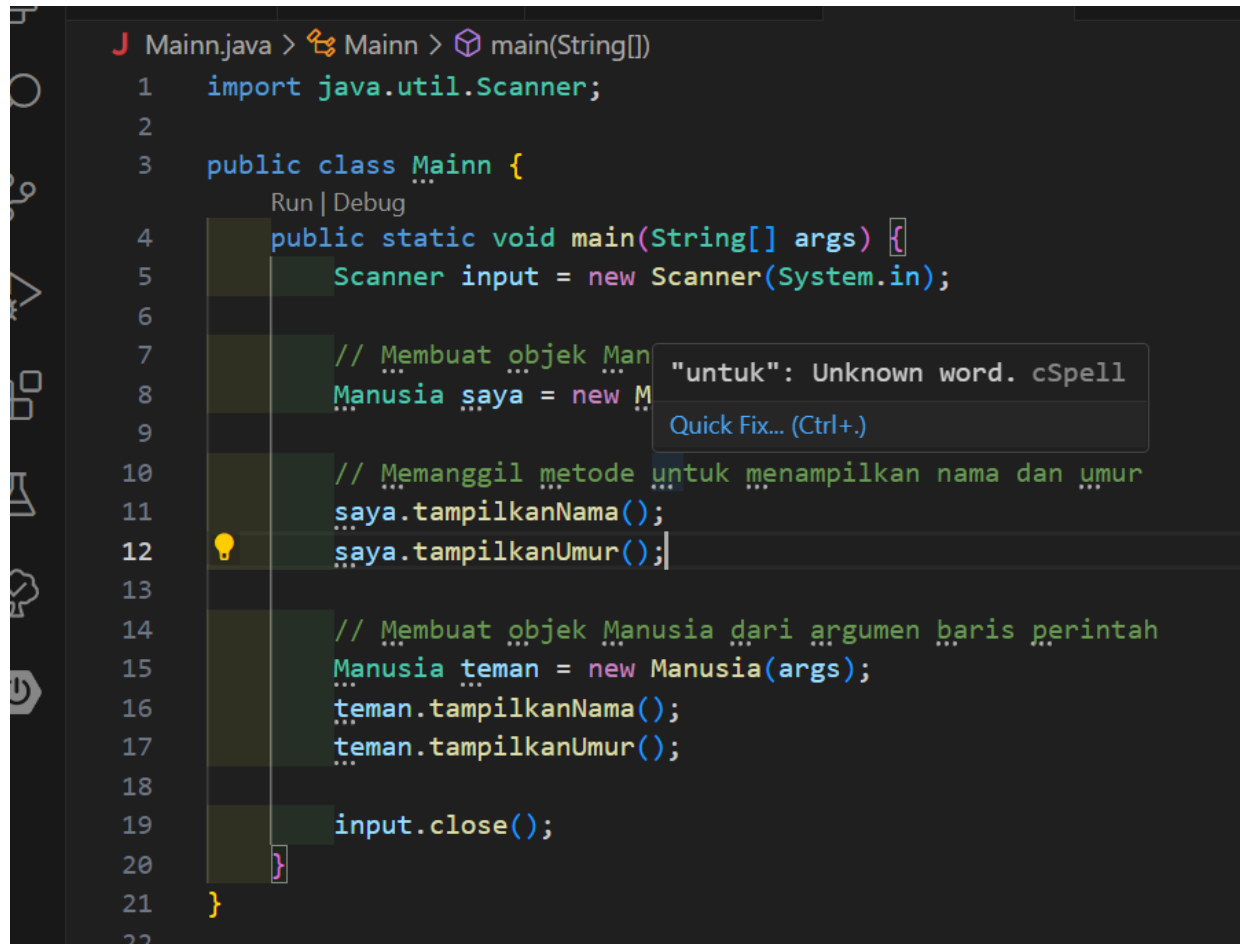
Tambahan pada class Manusia =

```

//Konstruktor untuk membuat objek Manusia
public Manusia(String[] args) {
    if (args.length >= 2) {
        this.nama = args[0];
        this.umur = Integer.parseInt(args[1]);
    } else {
        System.out.println(x: "Nama Teman");
        this.nama = "Agos";
        this.umur = 20;
    }
}

```


Pada class Mainn =



```

J Mainn.java > Mainn > main(String[])
1  import java.util.Scanner;
2
3  public class Mainn {
4      public static void main(String[] args) {
5          Scanner input = new Scanner(System.in);
6
7          // Membuat objek Manusia
8          Manusia saya = new Manusia();
9
10         // Memanggil metode untuk menampilkan nama dan umur
11         saya.tampilkanNama();
12         saya.tampilkanUmur();
13
14         // Membuat objek Manusia dari argumen baris perintah
15         Manusia teman = new Manusia(args);
16         teman.tampilkanNama();
17         teman.tampilkanUmur();
18
19         input.close();
20     }
21 }
22

```

Output =

```

Masukkan nama: Eka
Masukkan umur: 19
Nama saya: Eka
Umur saya: 19
Nama Teman
Nama saya: Agos
Umur saya: 20

```

Dalam kode di atas, memiliki dua konstruktor di kelas Manusia. Satu untuk membuat objek Manusia dari input pengguna dan satu lagi untuk membuat objek Manusia dari argumen baris perintah. Di dalam metode main, kita membuat dua objek Manusia, satu menggunakan input pengguna dan satu lagi menggunakan argumen baris perintah. Objek-objek tersebut kemudian dipanggil untuk menampilkan nama dan umur.

Tugas Praktikum

Perusahaan NV. Meneer memiliki koperasi karyawan yang memungkinkan karyawannya berbelanja di koperasi tersebut. Tentunya, karyawan tersebut bisa membayar belanjanya tersebut di akhir bulan melalui pemotongan gaji. Ada 2 kelas yang terlibat disini, Invoice dan Employee. Kedua class tadi mengimplementasikan **interface Payable** yang mana ia hanya memiliki satu method yang harus diimplementasikan di kedua class, yaitu **getPayableAmount()**. Program harus bisa mengolah gaji karyawan di akhir bulan beserta invoice belanjaan karyawan yang nantinya gaji karyawan perbulannya dikurang total harga belanjanya secara polimorfis. Tampilkan informasi dari karyawan tersebut beserta total gaji setelah dipotong hutang belanjaan di koperasi dan tampilkan pula detail belanjanya secara polimorfis pula.

1. **Attribut dari Invoice:**

String productName, Integer quantity, Integer pricePerItem

2. **Attribut dari Employee:**

Integer registrationNumber, String name, Integer salaryPerMonth, Invoice[] invoices