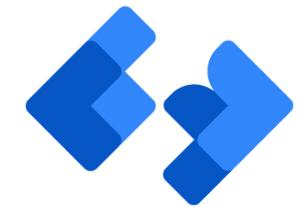


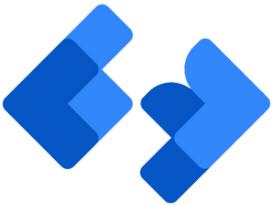
Bab 8

Interface

Definisi Interface.



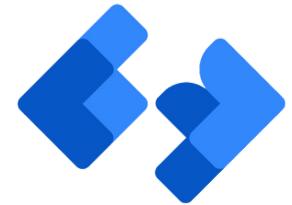
Interface merupakan sebuah antarmuka ,yang secara umum interface berfungsi sebagai penghubung antar sesuatu yang 'Abstract' dengan sesuatu yang nyata. Terdapat istilah 'able' atau 'mampu' yaitu istilah yang sering digunakan untuk menunjukkan bahwa , objek yang mengimplementasikan interface tersebut memiliki sesuatu atau memiliki kemampuan tertentu. Dalam OOP, sebuah interface dapat dianggap sebagai prototipe atau templat untuk sebuah kelas. Analoginya, jika sebuah class abstrak adalah kerangka dasar untuk kelas-kelas lain, maka interface adalah templat yang memberikan struktur khusus untuk kelas tertentu.



Definisi Interface.

Ketika kita mendefinisikan metode di dalam sebuah interface, kita hanya memberikan gambaran atau "prototipe" dari metode tersebut tanpa memberikan implementasi konkret(abstract method). Kelas-kelas yang mengimplementasikan interface tersebut wajib meng-Overide method untuk mendefinisikan implementasi metode – metode ini. Jadi Interface ini ,digunakan sebagai protokol untuk kelas yang mengimplementasi interface tersebut memiliki method yang ada di interfce .

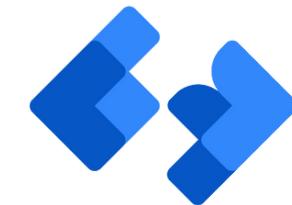
Deklarasi Interface



Interface secara struktur mirip dengan class, tetapi dengan perbedaan penting. Isi dari interface terdiri dari deklarasi method yang bersifat abstrak, yaitu metode yang hanya didefinisikan tanpa isi (badan metode). Deklarasi metode dalam interface mirip dengan deklarasi metode pada kelas abstrak. Ketika kita membuat sebuah method dalam interface maka otomatis sama dengan kita membuat class abstract.

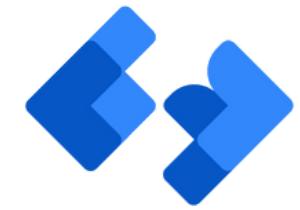
Variabel yang dideklarasikan dalam interface otomatis dianggap sebagai static (artinya, terkait dengan kelas itu sendiri, bukan instance) dan final (artinya, nilainya tidak dapat diubah setelah diinisialisasi). Sedangkan Metode dalam interface secara otomatis dianggap sebagai public dan abstract.

Perbedaan Interface dan Abstract



Abstract	Interface
Bisa berisi abstract dan non-abstract method.	Hanya boleh berisi abstract method.
Modifier harus dituliskan sendiri.	Tidak perlu menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method di interface adalah public dan abstract .
Suatu abstact class hanya bisa meng-extend satu abstract class lainnya.	Suatu interface bisa meng-extend satu atau lebih interface lainnya.
Kata kunci abstract digunakan untuk mendeklarasikan kelas abstract. Juga, kelas abstract dapat diperluas dengan menggunakan kata kunci "extend"	Kata kunci interface yang digunakan untuk mendeklarasikan interface. Juga, interface dapat diimplementasikan dengan menggunakan kata kunci "implement"

Deklarasi Interface



Sintak untuk menciptakan interface serupa dengan cara menciptakan sebuah class tetapi terdapat beberapa pengecualian, yaitu :

1. Seluruh method yang di deklarasikan pada Interface pasti bersifat public dan abstract.
2. Variable selalu bersifat public, final dan static.
3. Bentuk Deklarasi Interface:

```
● ● ●  
1  public interface NamaInterface {  
2      // Variabel konstan (static, final)  
3      type_data NAMA_VARIABEL = nilai;  
4  
5      // Metode abstrak (public, abstract)  
6      type_method namaMetode(parameter);  
7  }
```

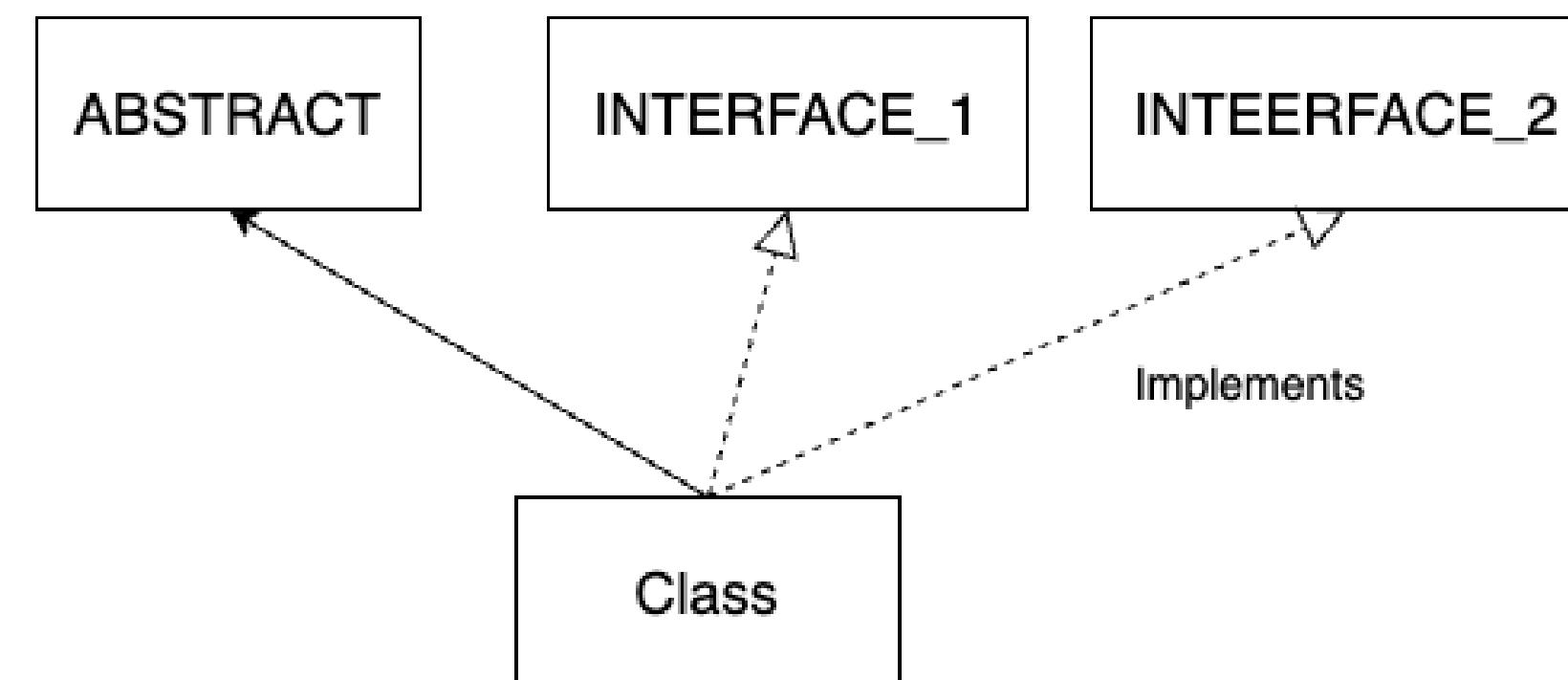
Struktur multiple implements



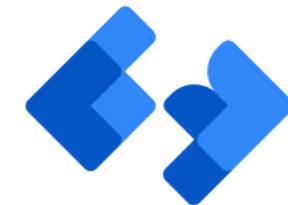
1. Sebuah class dapat meng extends dan implements interface secara bersamaan:

Java mendukung pewarisan tunggal dari kelas, tetapi sebuah kelas dapat mengimplementasikan banyak interface.

```
● ● ●  
1 public class nama_class extends abstract_class implements implements_1 , implements_2
```



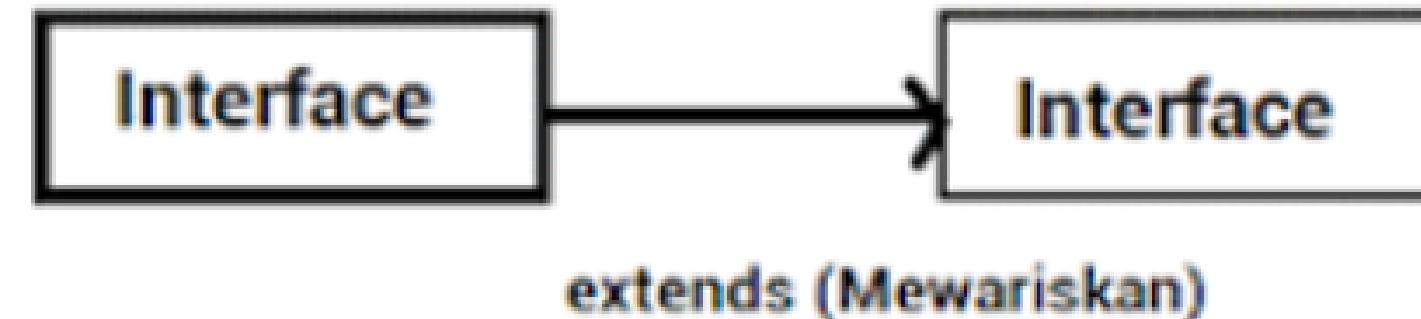
Struktur multiple implements



2. Sebuah Interface dapat extends dari interface lain :

Memungkinkan fungsionalitas dan mendefinisikan hubungan antara berbagai interface. Interface yang meng-extend interface lainnya mewarisi deklarasi metode dari interface yang di-extend.

```
● ● ●  
1 public interface nama_inteface extends interface_1
```

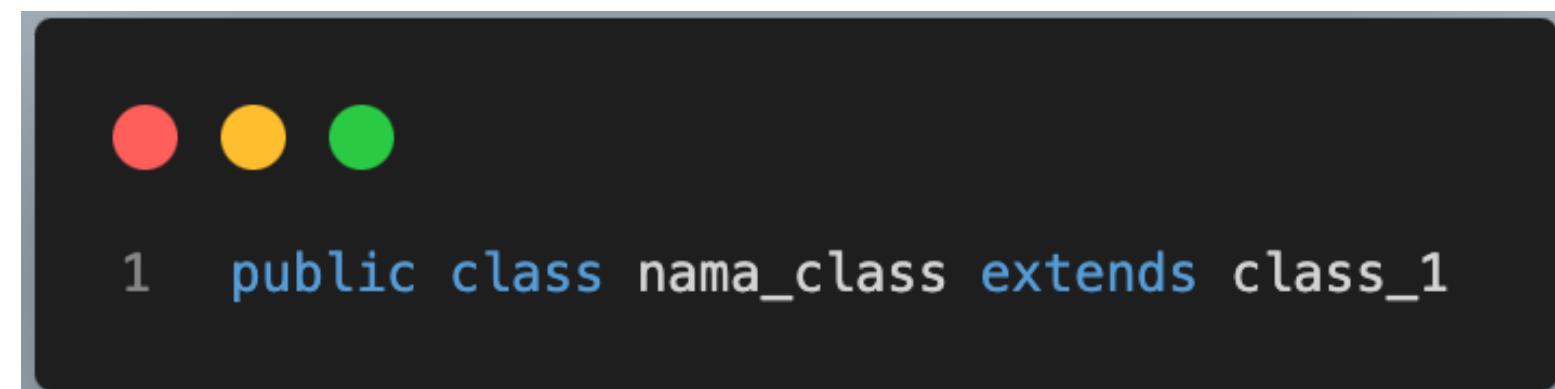


Struktur multiple implements

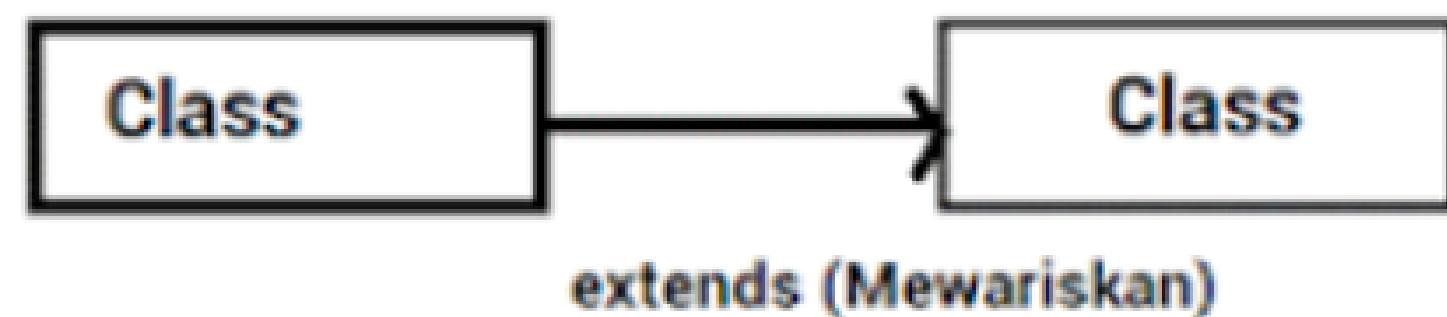


3. Sebuah Class dapat extends dari kelas lain :

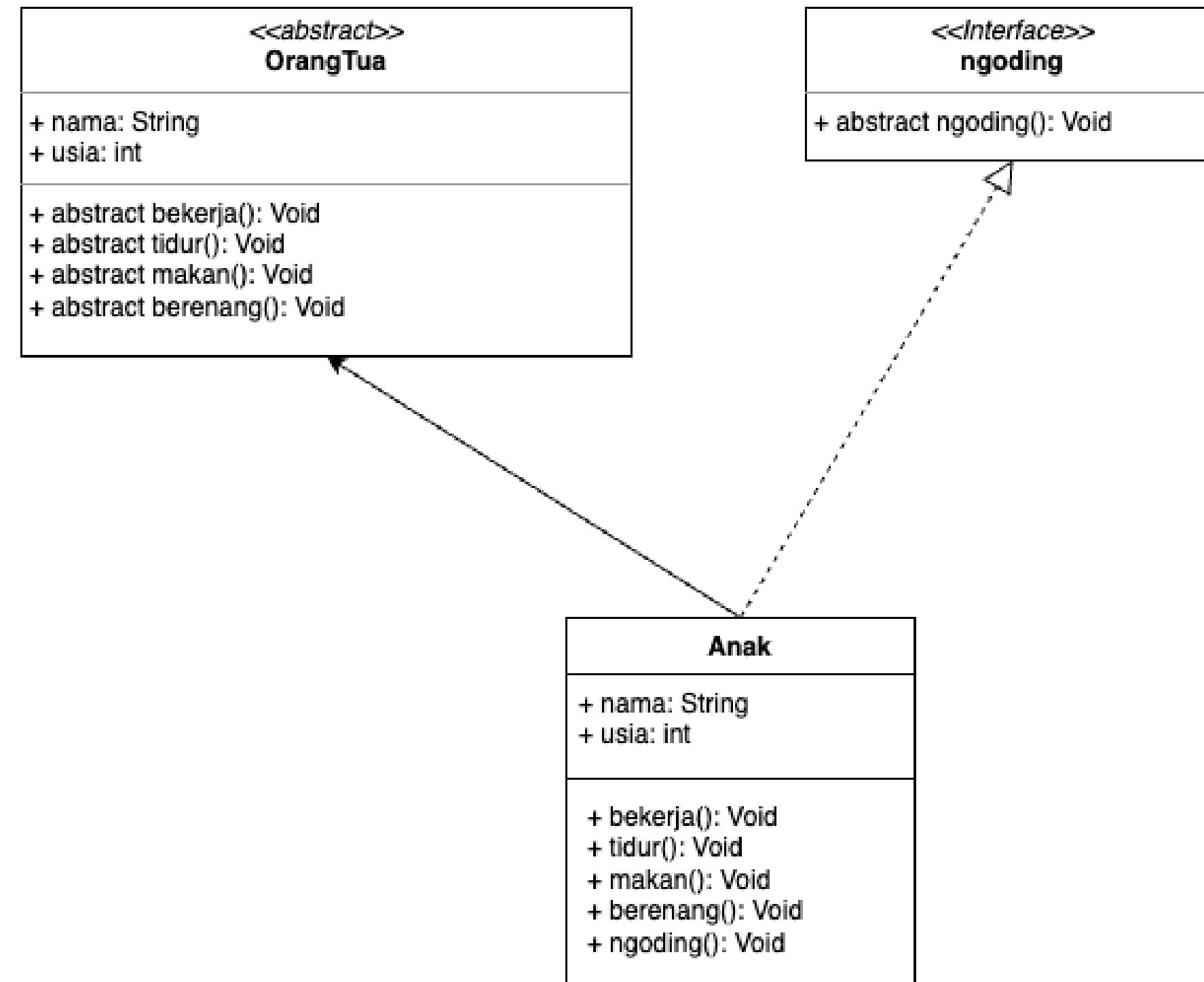
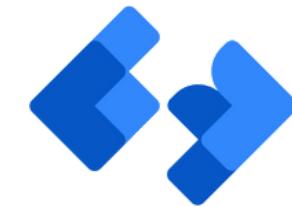
Pewarisan kelas memungkinkan kelas turunan (subclass) untuk mewarisi atribut dan metode dari kelas induk (superclass), dan juga untuk menambahkan atau mengubah perilaku tersebut jika diperlukan.



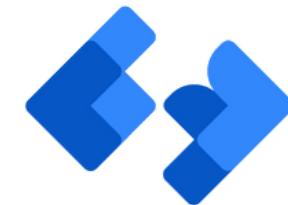
```
1 public class nama_class extends class_1
```



Ilustrasi



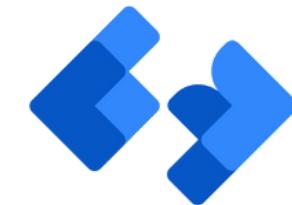
Ilustrasi



```
① public abstract class orangtua {  
12  
13     // Atribut  
14     String nama;  
15     int usia;  
16  
17     // Konstruktor  
18     public orangtua(String nama, int usia) {  
19         this.nama = nama;  
20         this.usia = usia;  
21     }  
22     // Metode yang dapat diwariskan  
23      abstract void tidur();  
24      abstract void makan();  
25      abstract void bekerja();  
26  
27      abstract void berenang();  
28 }
```

```
①  
②  
13  
public interface Ngoding {  
    abstract void ngoding();  
}
```

Ilustrasi

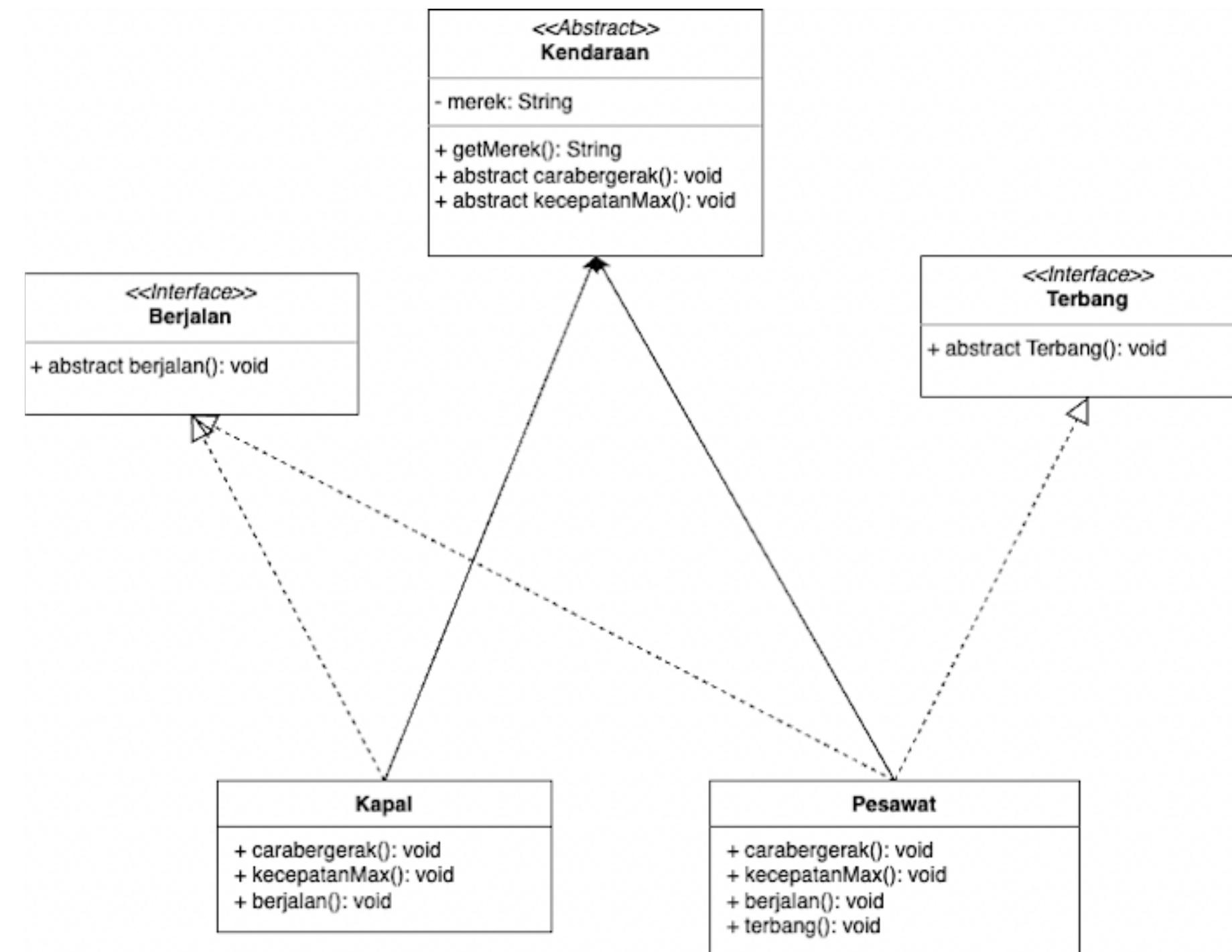


```
public class main {  
    public static void main(String[] args) {  
        Anak anak = new Anak( nama:"Bilbert", usia:20);  
        System.out.println("Object anak bernama "+anak.nama+" Berusia "+anak.usia);  
        anak.tidur();  
        anak.makan();  
        anak.bekerja();  
        anak.berenang();  
        anak.ngoding();  
    }  
}
```

Output - PraktikumOOP_2118112 (run) ×

run:
Object anak bernama Bilbert Berusia 20
Bilbert sedang tidur.
Bilbert sedang makan.
Bilbert sedang bekerja.
Bilbert karena diajari orang tua berenang maka ia pintar berenang
Bilbert memiliki kemampuan ngoding
BUILD SUCCESSFUL (total time: 0 seconds)

Contoh Implementasi



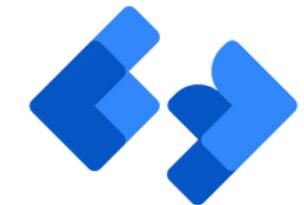
Contoh Implementasi



```
① public abstract class kendaraan {  
12  
②     private String merek;  
14  
15     public kendaraan(String merek) {  
16         this.merek = merek;  
17     }  
18  
19     public String getMerek() {  
20         return merek;  
21     }  
22  
③     abstract void carabegerak();  
24  
④     abstract void kecepatanMax();  
26  
27 }
```

```
① public interface Terbang {  
②     abstract void terbang();  
③ }  
  
① public interface Berjalan {  
②     abstract void berjalan();  
③ }
```

Contoh Implementasi



```
11  public class Mobil extends Kendaraan implements Berjalan{  
12  
13      public Mobil(String merek) {  
14          super(merek);  
15      }  
16  
17      @Override  
18      ① void carabegerak() {  
19          System.out.println("bergerak maju melalui roda yang bersentuhan langsung dengan permukaan tanah.");  
20      }  
21  
22      @Override  
23      ① void kecepatanMax() {  
24          System.out.println("500km/h");  
25      }  
26  
27      @Override  
28      ① public void berjalan() {  
29          System.out.println("Mobil " + getMerek() + " sedang berjalan di jalan raya");  
30      }  
31  
32  }
```

Contoh Implementasi



```
11 public class Pesawat extends Kendaraan implements Berjalan, Terbang {  
12       
13     public Pesawat(String merek) {  
14         super(merek);  
15     }  
16       
17     @Override  
18     void carabegerak() {  
19         System.out.println("Pesawat bergerak dalam udara dengan menggunakan mesin-mesin yang memungkinkan untuk terbang");  
20     }  
21       
22     @Override  
23     void kecepatanMax() {  
24         System.out.println("900 km/h");  
25     }  
26       
27     @Override  
28     public void berjalan() {  
29         System.out.println("Pesawat " + getMerek() + " berjalan maju melalui roda yang bersentuhan langsung dengan permukaan tanah.");  
30     }  
31       
32     @Override  
33     public void terbang() {  
34         System.out.println("Psawat " + getMerek() + " mesin yang berkecepatan tinggi, baling2 dan sayap aerodinamic memungkinkannya pesawat untuk lepas landas, terbang");  
35     }  
36       
37 }
```

Contoh Implementasi



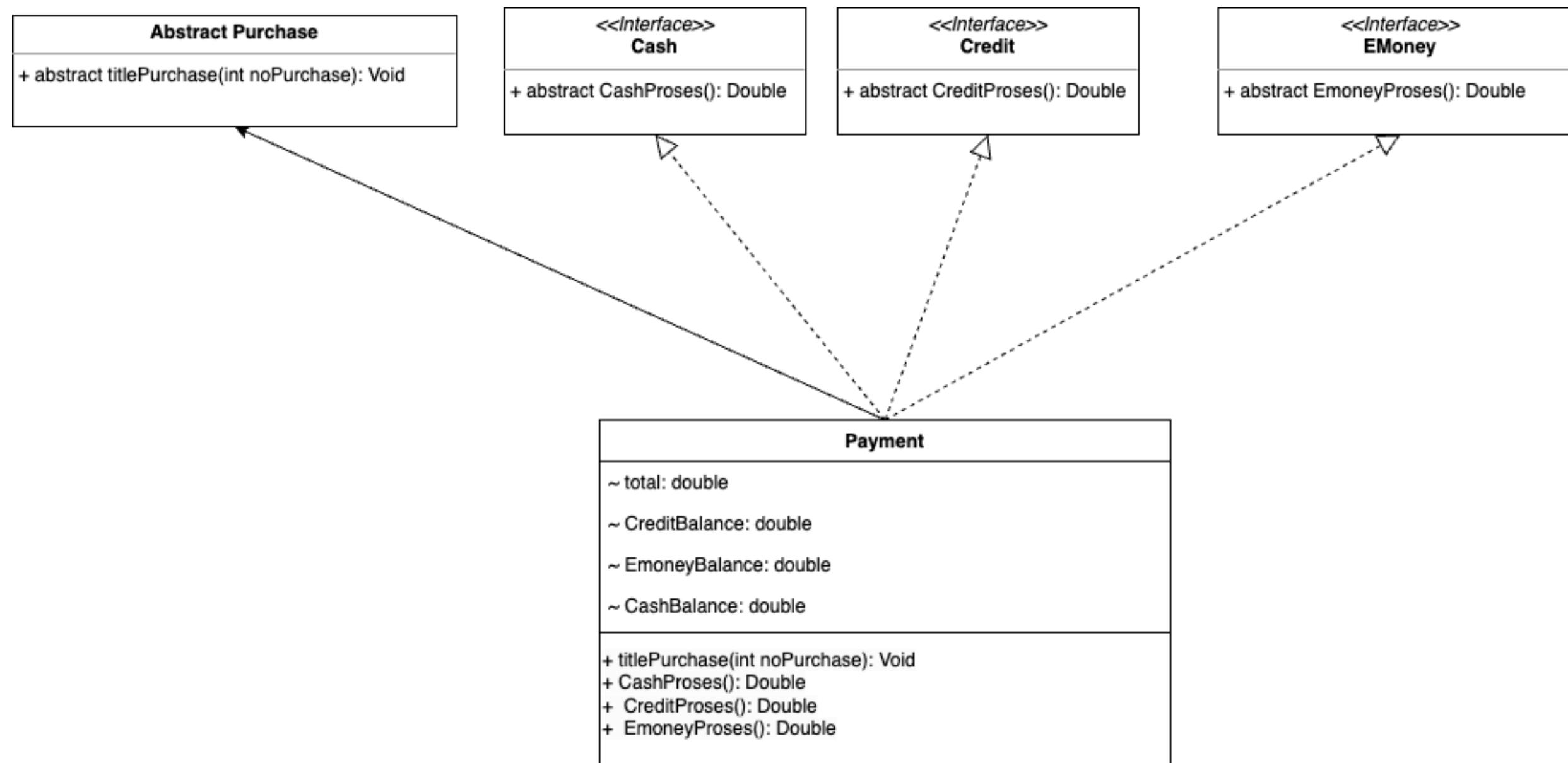
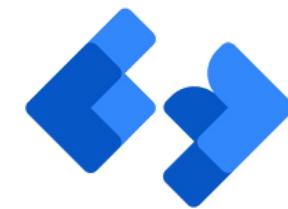
```
11 public class main {  
12  
13     public static void main(String[] args) {  
14         // Contoh penggunaan dengan objek Mobil  
15         Mobil mobil = new Mobil(merek: "Toyota");  
16         mobil.berjalan();  
17         mobil.carabegerak();  
18         mobil.kecepatanMax();  
19         System.out.println("-----");  
20         Pesawat pesawat = new Pesawat(merek: "Boeing 737");  
21         pesawat.carabegerak();  
22         pesawat.kecepatanMax();  
23         pesawat.berjalan();  
24         pesawat.terbang();  
25     }  
26 }
```

Contoh Implementasi

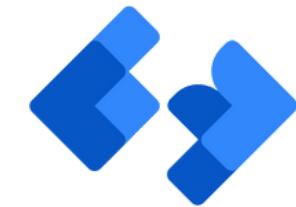


```
Output - PraktikumOOP_2118112 (run) ×
run:
Mobil Toyota sedang berjalan di jalan raya
bergerak maju melalui roda yang bersentuhan langsung dengan permukaan tanah.
500km/h
=====
Pesawat bergerak dalam udara dengan menggunakan mesin-mesin yang memungkinkan untuk terbang
900 km/h
Pesawat Boeing 737 berjalan maju melalui roda yang bersentuhan langsung dengan permukaan tanah.
Psawat Boeing 737 mesin yang berkecepatan tinggi, baling2 dan sayap aerodinamic memungkinkannya pesawat untuk lepas landas,
BUILD SUCCESSFUL (total time: 0 seconds)
```

Latihan



Latihan



(I)
(I)

13

```
public interface EMoney {  
    abstract double emoneyProses();  
}
```

(I)
(I)

13

```
public interface Cash {  
    abstract double cashProses();  
}
```

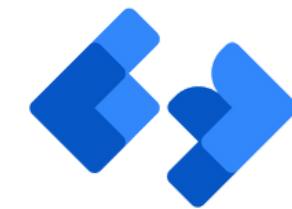
(I)
(I)

13

14

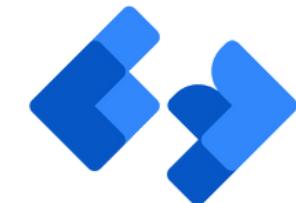
```
public interface Credit {  
    abstract double creditProses();  
}
```

Latihan



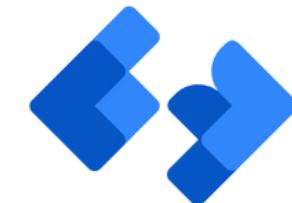
```
①  
12  
💡⬇️  
14  
abstract public class Purchase {  
    // int noPurchase;  
    abstract void titlePurchase(int noPurchase);  
}
```

Latihan



```
11  public class Payment extends Purchase implements Cash, Credit, EMoney{  
12      double total, creditBalance, emoneyBalance, cashBalance;  
13      public Payment(double setTotal) {  
14          this.creditBalance = 1000000;  
15          this.emoneyBalance = 500000;  
16          this.cashBalance = 1500000;  
17          this.total = setTotal;  
18      }  
19      //dari induk abstract Purchase  
20      @Override  
21      void titlePurchase(int numberPurchase) {  
22          System.out.println("Pembelian ke- " + numberPurchase);  
23      }  
24      // dari interface cash  
25      @Override  
26      public double cashProses() {  
27          total = cashBalance - this.total;  
28          return total;  
29      }  
30      //dari interface credit  
31      @Override  
32      public double creditProses() {  
33          total = creditBalance - this.total;  
34          return total;  
35      }  
36      //dari emoney  
37      @Override  
38      public double emoneyProses() {  
39          total = emoneyBalance - this.total;  
40          return total;  
41      }  
42  }
```

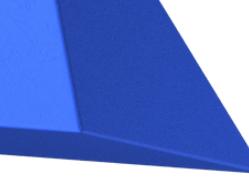
Latihan



```
11 public class Main {
12     public static void main(String[] args) {
13         Payment pay1 = new Payment( setTotal:750000); // atur jumlah uang total pembelian di sini
14
15         pay1.titlePurchase( numberPurchase:1);
16         System.out.println("credit Balance : " + pay1.creditBalance);
17         System.out.println("Emoney Balance : " + pay1.emoneyBalance);
18         System.out.println("Cash Balance : " + pay1.cashBalance);
19         System.out.println( x:=====);
20         System.out.println("total payment : " + pay1.total);
21         System.out.println( x:=====);
22         //menampilkan sisa saldo dengan berbagai macam metode pemayaran
23         System.out.println("payment With credit, Ending Balance : " + pay1.creditProses());
24         System.out.println("payment With E-Money, Ending Balance : " + pay1.emoneyProses());
25         System.out.println("payment With Cash, Ending Balance : " + pay1.cashProses());
26     }
27 }
```

Output - PraktikumOOP_2118112 (run) ×

- ▶ run:
- ▶ Pembelian ke- 1
- ▶ credit Balance : 1000000.0
- ▶ Emoney Balance : 500000.0
- ▶ Cash Balance : 1500000.0
- =====
- ▶ total payment : 750000.0
- =====
- ▶ payment With credit, Ending Balance : 250000.0
- ▶ payment With E-Money, Ending Balance : 250000.0
- ▶ payment With Cash, Ending Balance : 1250000.0
- ▶ BUILD SUCCESSFUL (total time: 0 seconds)



"Open NetBeans, and let's write code Java"