

Jobsheet 9 – Abstract Class
Praktikum Pemrograman Berbasis Objek



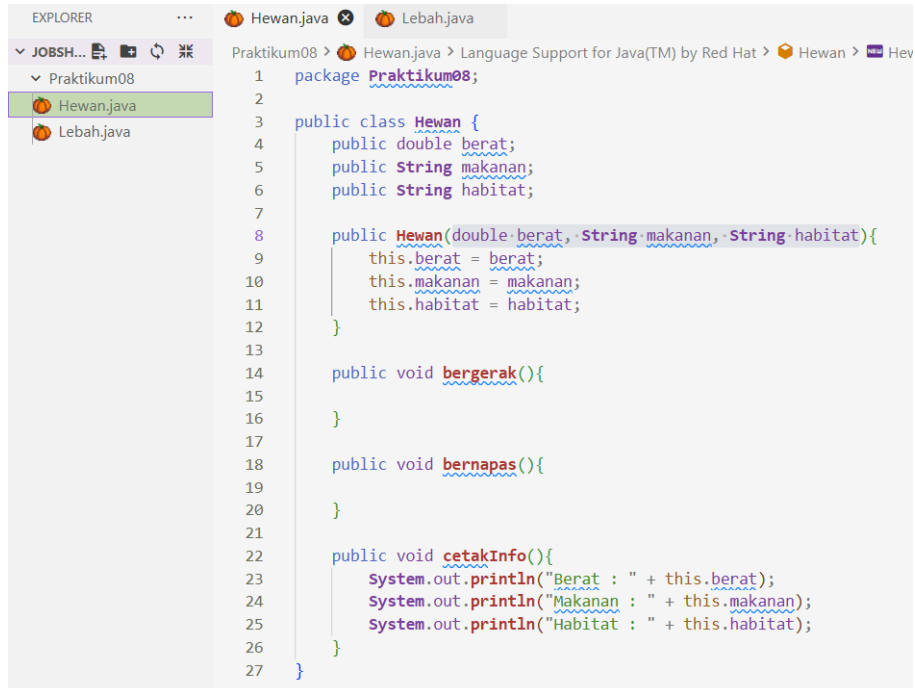
244107060144

Maria Savira

D-IV Sistem Informasi Bisnis / 2C

Percobaan 1


1. Membuat Class Hewan dalam package Praktikum08 :



The screenshot shows an IDE with the Explorer on the left and the editor on the right. The Explorer shows a project named 'JOBSH...' with a package 'Praktikum08' containing two files: 'Hewan.java' and 'Lebah.java'. The editor shows the code for 'Hewan.java' in the package 'Praktikum08'. The code defines a public class 'Hewan' with three attributes: 'berat' (double), 'makanan' (String), and 'habitat' (String). It includes a constructor 'Hewan(double berat, String makanan, String habitat)' that initializes these attributes, and three methods: 'bergerak()', 'bernapas()', and 'cetakInfo()' which prints the attributes.

```
1 package Praktikum08;
2
3 public class Hewan {
4     public double berat;
5     public String makanan;
6     public String habitat;
7
8     public Hewan(double berat, String makanan, String habitat){
9         this.berat = berat;
10        this.makanan = makanan;
11        this.habitat = habitat;
12    }
13
14    public void bergerak(){
15
16    }
17
18    public void bernapas(){
19
20    }
21
22    public void cetakInfo(){
23        System.out.println("Berat : " + this.berat);
24        System.out.println("Makanan : " + this.makanan);
25        System.out.println("Habitat : " + this.habitat);
26    }
27 }
```

2. Membuat Class Lebah :



The screenshot shows the IDE with the Explorer on the left and the editor on the right. The Explorer shows the 'Praktikum08' package with 'Hewan.java' and 'Lebah.java'. The editor shows the code for 'Lebah.java' in the package 'Praktikum08'. The code defines a public class 'Lebah' that extends 'Hewan'. It has a new attribute 'kasta' (String) and a constructor 'Lebah(double berat, String makanan, String habitat, String kasta)' that calls the superclass constructor and initializes 'kasta'.

```
1 package Praktikum08;
2
3 public class Lebah extends Hewan {
4     public String kasta;
5
6     public Lebah(double berat, String makanan, String habitat, String kasta){
7         super(berat, makanan, habitat);
8         this.kasta = kasta;
9     }
10 }
11
```

3. Membuat Class Main lalu menginstansiasi objek dari class Hewan dan class Lebah :



The screenshot shows the IDE with the Explorer on the left and the editor on the right. The Explorer shows a project named 'JOBSHEET 9 - ABSTRACT C...' with a package 'Praktikum08' containing three files: 'AbstractClassDemo.java', 'Hewan.java', and 'Lebah.java'. The editor shows the code for 'AbstractClassDemo.java' in the package 'Praktikum08'. The code defines a public class 'AbstractClassDemo' with a static method 'main' that creates two objects: 'hewan1' of type 'Hewan' and 'lebah1' of type 'Lebah', and calls their 'cetakInfo()', 'bergerak()', and 'bernapas()' methods.

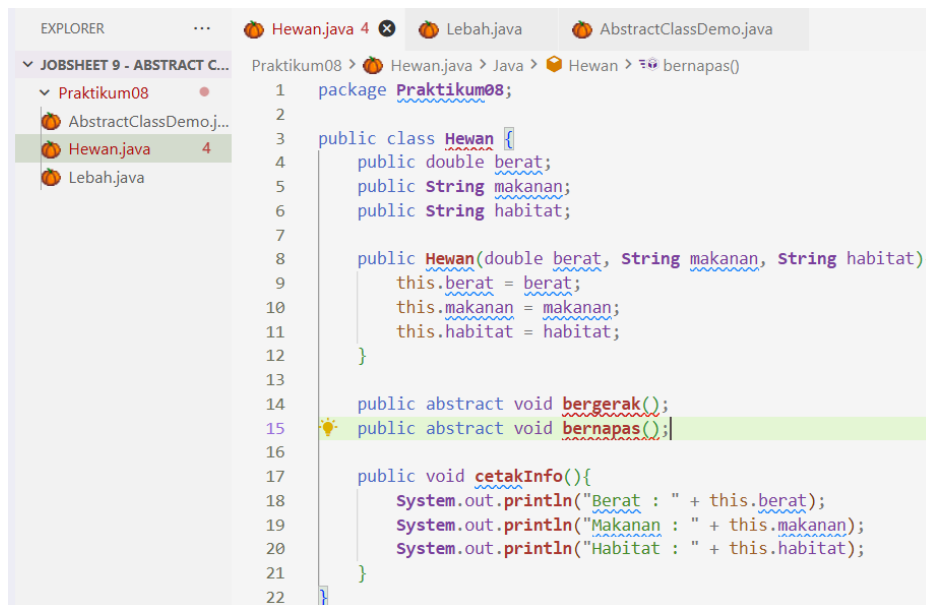
```
1 package Praktikum08;
2
3 public class AbstractClassDemo {
4
5     public static void main(String[] args) {
6         Hewan hewan1 = new Hewan(10, "Rumput", "Savana");
7         hewan1.cetakInfo();
8         hewan1.bergerak();
9         hewan1.bernapas();
10
11         Lebah lebah1 = new Lebah(0.05, "Nektar", "Hutan", "Ratu");
12         lebah1.cetakInfo();
13         lebah1.bergerak();
14         lebah1.bernapas();
15     }
16 }
```

Pertanyaan

1. Pada langkah 3, objek hewan1 dan lebah1 dapat diinstansiasi karena parent Hewan belum merupakan abstract class.
2. Karena cara bergerak hewan berbeda-beda dan tidak umum, sehingga method bergerak() dan bernapas() dibiarkan kosong.
3. Karena child Lebah secara otomatis menurunkan class milik parent Hewan, maka dari itu method apapun yang dimiliki parent Hewan dapat diakses dan digunakan oleh child Lebah.

Percobaan 2

1. Mengubah method bergerak dan bernapas menjadi abstract method :



```
1 package Praktikum08;
2
3 public class Hewan {
4     public double berat;
5     public String makanan;
6     public String habitat;
7
8     public Hewan(double berat, String makanan, String habitat){
9         this.berat = berat;
10        this.makanan = makanan;
11        this.habitat = habitat;
12    }
13
14    public abstract void bergerak();
15    public abstract void bernapas();
16
17    public void cetakInfo(){
18        System.out.println("Berat : " + this.berat);
19        System.out.println("Makanan : " + this.makanan);
20        System.out.println("Habitat : " + this.habitat);
21    }
22 }
```

2. Muncul error :



```
5 public String makanan;
6 public String habitat;
7
8 public Hewan(double b
9     this.berat = bera
10    this.makanan = ma
11    this.habitat = ha
12 }
13
14 public abstract void bergerak();
15 public abstract void bernapas();
```

"bergerak": Unknown word. cSpell

The abstract method bergerak in type Hewan can only be defined by an abstract class Java(67109227)

void Praktikum08.Hewan.bergerak()

Praktikum08.Hewan

public abstract void bergerak()

View Problem (Alt+F8) Quick Fix... (Ctrl+.)

Karena method abstract hanya dapat dibuat jika classnya juga merupakan class abstract.

3. Mengubah class Hewan menjadi class abstract :

```
1 package Praktikum08;
2
3 public abstract class Hewan {
4     public double berat;
5     public String makanan;
6     public String habitat;
7
8     public Hewan(double berat, String makanan, String habitat){
9         this.berat = berat;
10        this.makanan = makanan;
11        this.habitat = habitat;
12    }
13
14    public abstract void bergerak();
15    public abstract void bernapas();
16
17    public void cetakInfo(){
18        System.out.println("Berat : " + this.berat);
19        System.out.println("Makanan : " + this.makanan);
20        System.out.println("Habitat : " + this.habitat);
21    }
22 }
```

4. Mengubah class Demo :

```
1 package Praktikum08;
2
3 public class AbstractClassDemo {
4     Run | Debug | Run main | Debug main
5     public static void main(String[] args) {
6         Hewan hewan1 = new Hewan(10, "Rumput", "Savana");
7         hewan1.cetakInfo();
8         hewan1.bergerak();
9         hewan1.bernapas();
10    }
```

Outputnya :

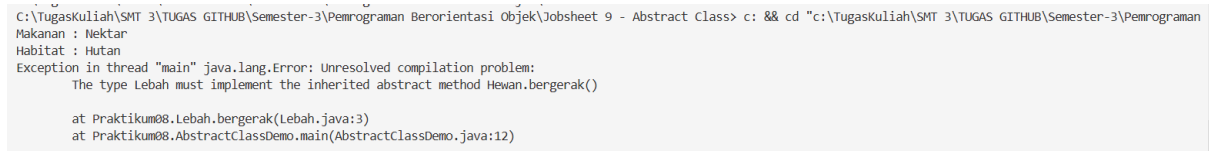
```
C:\TugasKuliah\SMT 3\TUGAS GITHUB\Semester-3\Pemrograman Berorientasi Objek\Jobsheet 9 - Abstract Class cmd /C ""C:\Program Files\Java\jdk21.0.1\bin\java.exe" -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\Asus\AppData\Roaming\Code\User\workspaceStorage\7aadb5ece3d2cee5a75a876a69db40fa\redhat.java\jdt_ws\Jobsheet 9 - Abstract Class_bf0aea1a\bin" Praktikum08.AbstractClassDemo
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Cannot instantiate the type Hewan
    at Praktikum08.AbstractClassDemo.main(AbstractClassDemo.java:5)
```

5. Mengubah class Demo lagi :



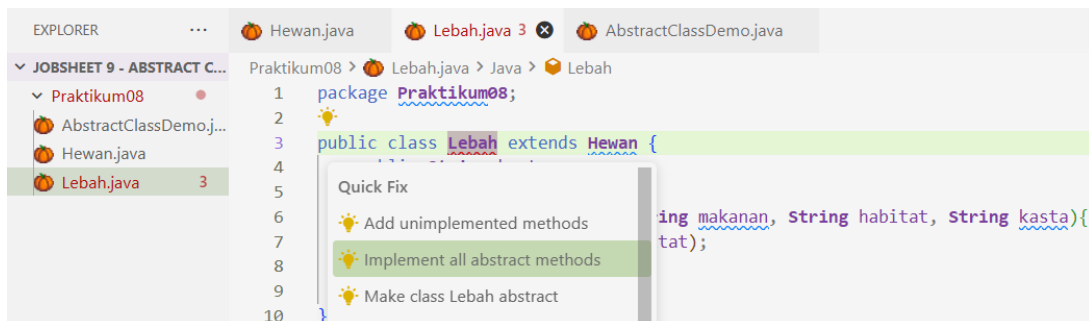
```
1 package Praktikum08;
2
3 public class AbstractClassDemo {
4     public static void main(String[] args) {
5         //Hewan hewan1 = new Hewan(10, "Rumput", "Savana");
6         //hewan1.cetakInfo();
7         //hewan1.bergerak();
8         //hewan1.bernapas();
9
10        Lebah lebah1 = new Lebah(berat:0.05, makanan:"Nektar", habitat:"Hutan", kasta:"Ratu");
11        lebah1.cetakInfo();
12        lebah1.bergerak();
13        lebah1.bernapas();
14    }
15 }
```

Outputnya :



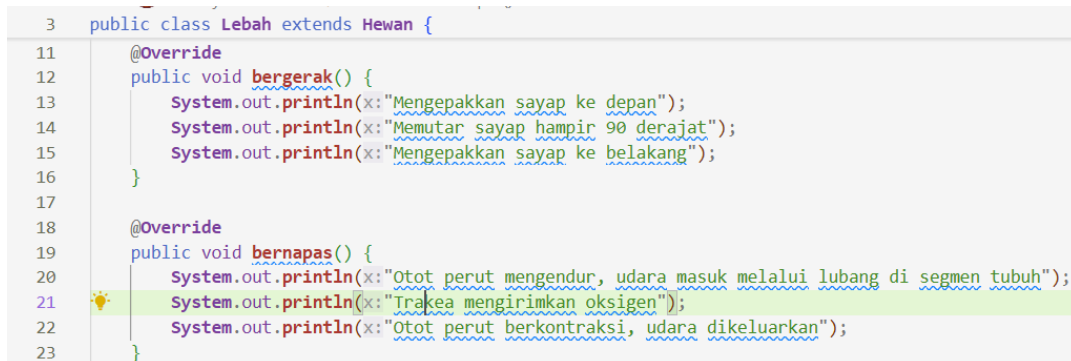
```
C:\Tugaskuliah\SMT 3\TUGAS GITHUB\Semester-3\Pemrograman Berorientasi Objek\Jobsheet 9 - Abstract Class> c: && cd "c:\Tugaskuliah\SMT 3\TUGAS GITHUB\Semester-3\Pemrograman
Makanan : Nektar
Habitat : Hutan
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
The type Lebah must implement the inherited abstract method Hewan.bergerak()
at Praktikum08.Lebah.bergerak(Lebah.java:3)
at Praktikum08.AbstractClassDemo.main(AbstractClassDemo.java:12)
```

6. Mengklik icon lampu pada class Lebah :



```
1 package Praktikum08;
2
3 public class Lebah extends Hewan {
4
5
6
7
8
9
10 }
```

7. Melakukan override dalam beberapa method pada class Lebah :



```
3 public class Lebah extends Hewan {
11     @Override
12     public void bergerak() {
13         System.out.println(x:"Mengepakkan sayap ke depan");
14         System.out.println(x:"Memutar sayap hampir 90 derajat");
15         System.out.println(x:"Mengepakkan sayap ke belakang");
16     }
17
18     @Override
19     public void bernapas() {
20         System.out.println(x:"Otot perut mengendur, udara masuk melalui lubang di segmen tubuh");
21         System.out.println(x:"Trakea mengirimkan oksigen");
22         System.out.println(x:"Otot perut berkontraksi, udara dikeluarkan");
23     }
24 }
```

8. Melakukan override pada method cetakInfo :

```
3 public class Lebah extends Hewan {
25     @Override
26     public void cetakInfo(){
27         super.cetakInfo();
28         System.out.println("Kasta : " + kasta);
29     }
30 }
```

9. Membuat class baru bernama Ular :

```
1 package Praktikum08;
2
3 public class Ular extends Hewan {
4     public boolean isBerbisa;
5
6     public Ular(double berat, String makanan, String habitat, boolean isBerbisa){
7         super(berat, makanan, habitat);
8         this.isBerbisa = isBerbisa;
9     }
10
11     @Override
12     public void bergerak(){
13         System.out.println(x:"Mengerutkan otot dari segala sisi hingga membentuk lengkungan");
14         System.out.println(x:"Menemukan titik penahan seperti batu atau pohon");
15         System.out.println(x:"Menggunakan sisik untuk mendorong tubuh ke depan");
16     }
17
18     @Override
19     public void bernapas(){
20         System.out.println(x:"Otot tulang rusuk kontraksi, udara masuk lewat hidung");
21         System.out.println(x:"Trakea mengirimkan udara ke paru-paru");
22         System.out.println(x:"Otot tulang rusuk relaksasi, udara dikeluarkan lewat hidung");
23     }
24
25     @Override
26     public void cetakInfo(){
27         super.cetakInfo();
28         System.out.println("Berbisa : " + (this.isBerbisa ? "Ya" : "Tidak"));
29     }
30 }
```

Pada Class Main :

```
3 public class AbstractClassDemo {
4     public static void main(String[] args) {
5         // Hewan hewan1 = new Hewan(10, "Rumput", "Savana");
6         // hewan1.cetakInfo();
7         // hewan1.bergerak();
8         // hewan1.bernapas();
9
10        Lebah lebah1 = new Lebah(berat:0.05, makanan:"Nektar", habitat:"Hutan", kasta:"Ratu");
11        lebah1.cetakInfo();
12        System.out.println();
13        System.out.println(x:"Cara Lebah bergerak : ");
14        lebah1.bergerak();
15        System.out.println();
16        System.out.println(x:"Cara Lebah bernapas : ");
17        lebah1.bernapas();
18        System.out.println();
19    }
20 }
```

```

20     Ular ular1 = new Ular(berat:1, makanan:"Daging", habitat:"Sawah", isBerbisa:true);
21     ular1.cetakInfo();
22     System.out.println();
23     System.out.println(x:"Cara Ular bergerak : ");
24     ular1.bergerak();
25     System.out.println();
26     System.out.println(x:"Cara Lebah bernapas : ");
27     ular1.bernapas();
28 }
29 }

```

Output :

```

Berat : 0.05
Makanan : Nektar
Habitat : Hutan
Kasta : Ratu

Cara Lebah bergerak :
Mengepakkan sayap ke depan
Memutar sayap hampir 90 derajat
Mengepakkan sayap ke belakang

Cara Lebah bernapas :
Otot perut mengendur, udara masuk melalui lubang di segmen tubuh
Trakea mengirimkan oksigen
Otot perut berkontraksi, udara dikeluarkan

Berat : 1.0
Makanan : Daging
Habitat : Sawah
Berbisa : Ya

Cara Ular bergerak :
Mengerutkan otot dari segala sisi hingga membentuk lengkungan
Menemukan titik penahan seperti batu atau pohon
Menggunakan sisik untuk mendorong tubuh ke depan

Cara Lebah bernapas :
Otot tulang rusuk kontraksi, udara masuk lewat hidung
Trakea mengirimkan udara ke paru-paru
Otot tulang rusuk relaksasi, udara dikeluarkan lewat hidung

```

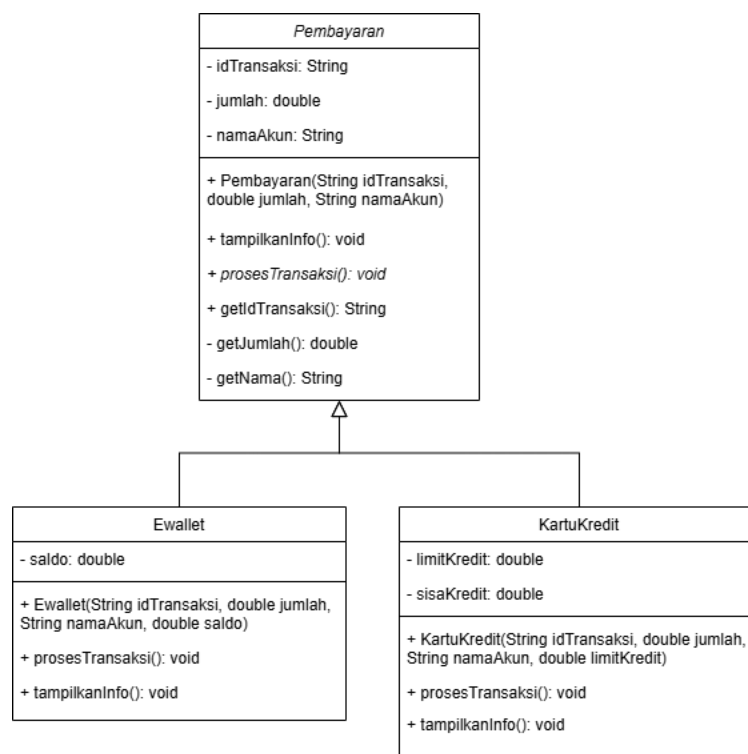
Pertanyaan

1. Method bergerak() dan bernapas() sebaiknya dideklarasikan sebagai abstract karena setiap hewan memiliki cara bernapas dan bergerak yang berbeda-beda, tidak ada bentuk yang umum.
2. Karena method abstract hanya dapat dibuat jika classnya juga merupakan class abstract.
3. Iya, agar tidak error.
4. Class abstract dapat memiliki concrete maupun method abstract, jadi tidak harus.
5. Karena yang diinstansiasikan merupakan superclass bertipe abstract, sedangkan superclass abstract tidak bisa langsung diinstansiasikan.
6. Abstract class dapat memiliki constructor berparameter maupun default.

7. Konstruktor abstract dari superclass dapat dipanggil dalam subclass, namun tidak dapat langsung diinstansiasikan di dalam main.
8. Karena method abstract yang dimiliki parent harus dioverride atau diisi melalui masing-masing child agar bisa digunakan, sedangkan cetakInfo tidak harus di override karena bukan merupakan abstract method.
9. Kegunaan abstract method
10. Kegunaan abstract class ialah agar dapat mencegah pembuatan objek yang terlalu umum, ia juga berfungsi sebagai kerangka umum, dan untuk memaksa subclass agar punya perilaku tertentu.

Tugas

Class Diagram



Class Pembayaran

```

Tugas > Pembayaran.java > Language Support for Java(TM) by Red Hat > Pembayaran > getIdTi
3  public abstract class Pembayaran {
4      private String idTransaksi;
5      private double jumlah;
6      private String nama;
7
8      public Pembayaran(String idTransaksi, double jumlah, String nama) {
9          this.idTransaksi = idTransaksi;
10         this.jumlah = jumlah;
11         this.nama = nama;
12     }

```



```

14     public String getIdTransaksi() {
15         return idTransaksi;
16     }
17
18     public double getJumlah() {
19         return jumlah;
20     }
21
22     public String getName() {
23         return nama;
24     }
25
26     public void tampilkanInfo() {
27         System.out.println("Nama : " + nama);
28         System.out.println("ID Transaksi : " + idTransaksi);
29         System.out.println("Jumlah : Rp" + jumlah);
30     }
31
32     public abstract void prosesTransaksi();
33 }

```

Class Ewallet

```

3     public class Ewallet extends Pembayaran {
4         private double saldo;
5
6         public Ewallet(String idTransaksi, double jumlah, String nama, double saldo) {
7             super(idTransaksi, jumlah, nama);
8             this.saldo = saldo;
9         }
10
11         @Override
12         public void prosesTransaksi() {
13             System.out.println(x:"\n--- Proses Transaksi E-Wallet ---");
14             if (saldo >= getJumlah()) {
15                 saldo -= getJumlah();
16                 System.out.println(x:"Transaksi berhasil!");
17                 System.out.println("Saldo tersisa : Rp" + saldo);
18             } else {
19                 System.out.println(x:"Transaksi gagal. Saldo tidak cukup!");
20             }
21         }
22
23         @Override
24         public void tampilkanInfo() {
25             super.tampilkanInfo();
26             System.out.println(x:"Metode : E-Wallet");
27             System.out.println("Saldo : Rp" + saldo);
28         }
29     }

```

Class KartuKredit

```

3     public class KartuKredit extends Pembayaran {
4         private double limitKredit;
5         private double sisalimit;
6
7         public KartuKredit(String idTransaksi, double jumlah, String nama, double limitKredit, double sisalimit) {
8             super(idTransaksi, jumlah, nama);
9             this.limitKredit = limitKredit;
10            this.sisalimit = sisalimit;
11        }

```

```

13     @Override
14     public void prosesTransaksi() {
15         System.out.println(x:"\n--- Proses Transaksi Kartu Kredit ---");
16         if (sisalimit >= getJumlah()) {
17             sisalimit -= getJumlah();
18             System.out.println(x:"Transaksi berhasil!");
19             System.out.println("Sisa limit : Rp" + sisalimit);
20         } else {
21             System.out.println(x:"Transaksi gagal. Limit tidak mencukupi!");
22         }
23     }
24
25     @Override
26     public void tampilkanInfo() {
27         super.tampilkanInfo();
28         System.out.println(x:"Metode : Kartu Kredit");
29         System.out.println("Limit Kredit : Rp" + limitKredit);
30         System.out.println("Sisa Limit : Rp" + sisalimit);
31     }
32 }

```

Class PembayaranMain

```

3 public class PembayaranMain {
4     public static void main(String[] args) {
5         Ewallet e1 = new Ewallet(idTransaksi:"TRX001", jumlah:50000, nama:"Maria Savira", saldo:120000);
6         Kartukredit k1 = new Kartukredit(idTransaksi:"TRX002", jumlah:150000, nama:"Maria Savira", limitKredit:100000, sisaLim400000);
7
8         System.out.println(x:"=== INFORMASI E-WALLET ===");
9         e1.tampilkanInfo();
10        e1.prosesTransaksi();
11
12        System.out.println(x:"\n=== INFORMASI KARTU KREDIT ===");
13        k1.tampilkanInfo();
14        k1.prosesTransaksi();
15    }
16 }

```

Output

```

=== INFORMASI E-WALLET ===
Nama : Maria Savira
ID Transaksi : TRX001
Jumlah : Rp50000.0
Metode : E-Wallet
Saldo : Rp120000.0

--- Proses Transaksi E-Wallet ---
Transaksi berhasil!
Saldo tersisa : Rp70000.0

=== INFORMASI KARTU KREDIT ===
Nama : Maria Savira
ID Transaksi : TRX002
Jumlah : Rp150000.0
Metode : Kartu Kredit
Limit Kredit : Rp1000000.0
Sisa Limit : Rp400000.0

--- Proses Transaksi Kartu Kredit ---
Transaksi berhasil!
Sisa limit : Rp250000.0

```

Link GitHub : [Semester-3/Pemrograman Berorientasi Objek/Jobsheet 9 - Abstract Class at main · MariaSavira/Semester-3](#)