LAPORAN RESMI

Praktikum 9 Inheritance 1

Mata Kuliah: Praktek Pemrograman Berbasis Objek

Disusun oleh:

M. Ainur Ramadhan (3122500047)

2 D3 Teknik Informatika B

Dosen Pengampu: Yanuar Risah Prayogi S.Kom., M.Kom.

PROGRAM STUDI D3 TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

2023/2024

A. TUGAS PENDAHULUAN

1. Apa yang dimaksud dengan inheritance?
   Jawab:
   Inheritance adalah metode untuk membuat hierarki antar kelas dengan cara mewarisi kelas lain.

2. Buatlah contoh kasus yang menerapkan konsep inheritance !
   Jawab:
   Contoh kasus dalam system E learning, terdapat berbagai jenis pengguna seperti Mahasiswa, Dosen dan Admin. Masing – masing memiliki peran dan fungsionalitas yang berbeda. Di sini Pengguna sebagai Class Induk, sedangkan Mahasiswa, Dosen, dan Admin sebagai Class anak. Oleh karena itu, Mahasiswa, Dosen, dan Admin akan mewarisi atribut dan methode dari kelas pengguna, tetapi juga memiliki atribut dan methode tambahan yang sesuai dengan peran dan fungsionalitas masing-masing.

3. Adakah perbedaan cara mengakses member class milik parent dan member class milik sendiri? Jelaskan melalui contoh ! (Silahkan memanfaatkan jawaban soal nomor 2.)
   Jawab:
   Terdapat perbedaan dalam bagaimana cara mengakses member class milik parent dan member class milik sendiri, yaitu terletak pada penggunaan kata kunci saat mengakses atribut atau methode dari class parent. Jika ingin mengakses atribut atau methode milik class parent harus menggunakan kata kunci super. Sedangkan mengakses class miliki sendiri tidak diperlukan kata kunci super. Berikut contohnya dalam kode program java yang berkaitan dengan contoh kasus diatas.

   Class Pengguna

```
class Pengguna {
    String nama;
    String email;

    Pengguna(String nama, String email) {
        this.nama = nama;
        this.email = email;
    }

    void displayInfo() {
        System.out.println("Nama: " + nama);
        System.out.println("Email: " + email);
    }
}
```

   Class Mahasiswa

```
class Mahasiswa extends Pengguna {
    String nim;

    Mahasiswa(String nama, String email, String nim) {
```

```
        super(nama, email);
        this.nim = nim;
    }

    void displayInfo() {
        super.displayInfo();
        System.out.println("NIM: " + nim);
    }

    void ambilKelas(String namaMataKuliah) {
        System.out.println("Mahasiswa " + nama + " mengambil kelas " +
namaMataKuliah);
    }
}
```

Pada kode program tersebut terdapat pengaksesan atribut atau method class parent (pengguna) pada member class (Mahasiswa) yaitu nama, email serta methode DisplayInfo yang mana ketiganya menggunakan kata kunci **super** dalam Class Mahasiswa namun berbeda apabila di dalam class mahasiswa, saat memanggil atribut atau methodenya sendiri yaitu tanpa menggunakan kata kunci **super**.

4. Apa yang dimaksud dengan konsep single inheritance ?
   Jawab:
   Konsep single inheritance adalah suatu konsep di mana sebuah kelas anak (member class) hanya dapat mewarisi sifat atau perilaku dari satu kelas induk (parent class) saja. Dalam konsep ini, sebuah kelas anak hanya memiliki satu kelas parent. Artinya, sebuah kelas tidak bisa mewarisi sifat dari lebih dari satu kelas parent.

5. Apa yang dimaksud dengan konsep multi level inheritance ?
   Jawab:
   Konsep multi-level inheritance adalah sebuah konsep di mana sebuah kelas anak dapat mewarisi sifat atau perilaku dari kelas induk (parent class) dan juga dapat menjadi parent class bagi kelas lainnya. Dalam konsep ini, ketika sebuah kelas anak mewarisi sifat dari kelas parent, kelas tersebut juga dapat menjadi parent bagi kelas lainnya, membentuk rantai atau hierarki lebih dari dua tingkat.

B. PERCOBAAN
   Penyembunyian informasi

```
public class Pegawai {
    private String nama;
    public double gaji;
}
```

```
    public class Manajer extends Pegawai {
         public String departemen;

         public void IsiData(String n, String d) {
              nama=n;
              departemen=d;
          }
    }
```

Output

```
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\percobaan> javac *.java
Manajer.java:4: error: cannot find symbol
        nama=n;
        ^
  symbol:   variable nama
  location: class Manajer
1 error
```

C. LATIHAN

Latihan 1

Tempatkan class Base dan class Class1 di direktori yang sama. Apa yang terjadi ketika Class1.java dikompile dan dijalankan jika sebelumnya Base.java belum dikompile? Jelaskan !

```
//Base.java
package Base;
class Base{
    protected void amethod(){
        System.out.println("amethod");
    }//End of amethod
}//End of class base
```

```
package Class1;
//Class1.java
public class Class1 extends Base{
    public static void main(String argv[]){
      Base b = new Base();
      b.amethod();
    }//End of main
}//End of Class1
```

Output

```
Class1.java:4: error: cannot find symbol
public class Class1 extends Base {
                            ^
  symbol: class Base
Class1.java:6: error: cannot find symbol
        Base b = new Base();
        ^
  symbol:   class Base
  location: class Class1
Class1.java:6: error: cannot find symbol
        Base b = new Base();
                     ^
  symbol:   class Base
  location: class Class1
3 errors
```

Penjelasan:
Error tersebut terjadi disebabkan class Class1 tidak menemukan class Base yang berjalan, oleh karena itu terjadi errror cannot find symbol yang mengindikasikan bahwa compiler tidak dapat menemukan deklarasi atau definisi dari symbol Base.

Latihan 2
Aturan overriding

a. Berdasarkan kode di bawah ini, akses modifier (public, protected atau private) apa yang diijinkan di tambahkan sebelum myMethod() baris 3?
b. Jika baris 3 seperti kode di bawah (apa adanya tanpa perubahan) keywords apa yang diijinkan ditambahkan sebelum myMethod baris 8?

```
1.      class HumptyDumpty
2.      {
3.              void myMethod() {}
4.      }
5.
6.      class HankyPanky extends HumptyDumpty
7.      {
8.              void myMethod() {}
9.      }
```

Jawab:

a. Dalam kasus kode yang diberikan, metode `myMethod()` pada kelas `HumptyDumpty` memiliki akses package-private, yang berarti metode ini hanya dapat diakses oleh kelas-kelas dalam package yang sama. Ketika kelas `HankyPanky` diturunkan dari `HumptyDumpty`, metode `myMethod()` di kelas `HankyPanky` dapat memiliki akses yang sama (package-private) atau lebih longgar, yaitu protected atau public. Dengan kata lain, jika metode pada kelas induk memiliki akses package-private, kelas anak dapat mempertahankan tingkat akses tersebut atau memperluasnya menjadi protected atau public. Namun, penggunaan akses modifier private tidak diizinkan di kelas anak,

karena metode private tidak dapat diwariskan dan tidak dapat diakses oleh kelas anak. Dalam hal ini, kelas `HankyPanky` memiliki fleksibilitas untuk memilih tingkat akses yang sesuai untuk metodenya, dengan batasan bahwa tingkat akses tidak boleh lebih ketat (private) dari tingkat akses metode pada kelas induk.

b. Dalam konteks kode yang diberikan di mana metode `myMethod()` pada kelas `HumptyDumpty` memiliki akses package-private (tanpa akses modifier), kelas turunan `HankyPanky` dapat mengakses dan meng-override metode tersebut dengan tingkat akses yang lebih longgar, yaitu `protected` atau `public`. Akses modifier `protected` memungkinkan metode `myMethod()` di kelas `HankyPanky` dapat diakses oleh kelas-kelas dalam package yang sama dan oleh kelas-kelas turunannya. Sementara itu, akses modifier `public` memungkinkan metode tersebut dapat diakses oleh semua kelas, termasuk kelas di luar package. Dengan menggunakan akses modifier `protected` atau `public` pada `myMethod()` di kelas `HankyPanky`, kita memperbolehkan penggunaan metode tersebut oleh kelas turunan lain dan memperluas aksesibilitas metode di luar package sesuai dengan kebutuhan aplikasi yang bersangkutan.

Latihan 3

a. Apa yang terjadi bila kedua kode dibawah ini dikompile dan dijalankan dalam satu direktori? Jelaskan !

b. Bagaimana solusi supaya tidak terjadi error?

```
//File P1.java
package MyPackage;

class P1{
      void aFancyMethod(){
           System.out.println("What a fancy method");
      }
}
```

```
//File P2.java
public class P2 extends P1{
    public static void main(String argv[]){
        P2 p2 = new P2();
        p2.aFancyMethod();
    }
}
```

a. Terdapat error saat menjalankan kode program yang kedua atau class P2, berikut errornya.

```
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3\MyPackage> javac P2.java
P2.java:1: error: cannot access P1
public class P2 extends P1 {
                        ^
  bad class file: .\P1.class
    class file contains wrong class: MyPackage.P1
    Please remove or make sure it appears in the correct subdirectory of the classpath.
```

Meski dijalankan dalam satu direktori namun, untuk deklarasi MyPackage pada class P2 belum ada, sehingga program untuk P1 tidak bisa dikenali.

b. Agar error tersebut hilang, menurut saya dapat diatasi dengan dua acara yaitu kedua class tanpa deklarasi package dan dengan deklarasi package.

Berikut output apabila tanpa deklarasi package

```
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3\MyPackage> javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3\MyPackage> java P2
What a fancy method
```

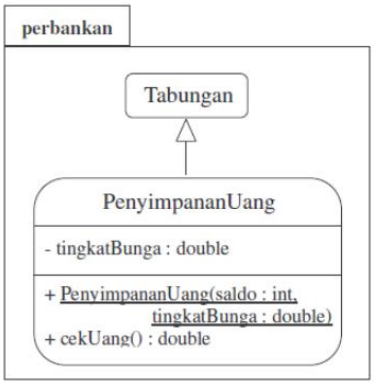
Berikut output jika dengan deklarasi package

```
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3> javac Mypackage/P1.java
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3> javac Mypackage/P2.java
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3> java MyPackage.P2
What a fancy method
```

Namun apabila di jalankan dalam direktori MyPackage terdapat error seperti berikut ini.

```
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3\MyPackage> javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3\MyPackage> java P2
Error: Could not find or load main class P2
PS C:\Users\USER\OOP\Praktikum\Praktikum 9\latihan\latihan3\MyPackage>
```

Latihan 4
Mengimplementasikan UML class diagram dalam program untuk package perbankan



Ubahlah mode akses atribut saldo pada Tabungan menjadi protected. Lalu Transformasikan class diagram diatas ke dalam bentuk program! Tulislah listing program berikut ini sebagai pengetesan.

```
import perbankan.*;

public class TesLatihan{
    public static void main(String args[]){
        PenyimpananUang tabungan = new PenyimpananUang(5000, 8.5/100);
        System.out.println("Uang yang ditabung : 5000");
        System.out.println("Tingkat bunga sekarang : 8.5%");
        System.out.println("Total uang anda sekarang : " +
                           tabungan.cekUang());
    }
}
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
Uang yang ditabung : 5000
Tingkat bunga sekarang : 8.5%
Total uang anda sekarang : 5425.0
```

Jawab:
Berikut kode program untuk class diagram tersebut:

PenyimpananUang.java

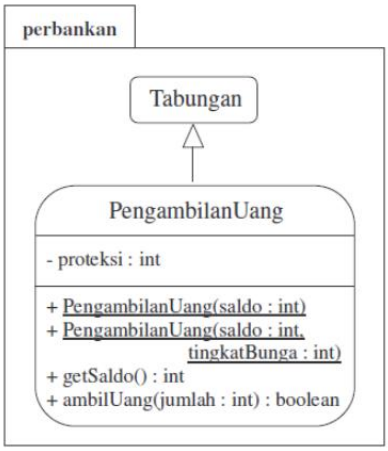
```java
package perbankan;

public class PenyimpananUang extends Tabungan {
    public double tingkatBunga;

    // Konstruktor dengan parameter saldo awal dan tingkat bunga
    public PenyimpananUang(int saldo, double tingkatBunga) {
        super(saldo); // Memanggil konstruktor kelas Tabungan dengan saldo
awal
        this.tingkatBunga = tingkatBunga;
    }

    // Metode untuk mengambil nilai saldo saat ini
    public double cekUang() {
        saldo += saldo * tingkatBunga;
        return saldo; // Menggunakan metode getSaldo() dari kelas Tabungan
    }
}
```

Output

```
PS C:\Users\USER\OOP\Praktikum\Praktikum9\latihan\latihan4> javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum9\latihan\latihan4> java TesLatihan
Uang yang ditabung : 5000
Tingkat bunga sekarang : 8.5%
Total uang anda sekarang : 5425.0
PS C:\Users\USER\OOP\Praktikum\Praktikum9\latihan\latihan4>
```

D. TUGAS

Mengimplementasikan UML class diagram dalam program untuk package perbankan



Transformasikan class diagram diatas ke dalam bentuk program! Tulislah listing program berikut ini sebagai pengetesan.

```
import perbankan.*
public class TesTugas{
   public static void main(String args[]){
   PengambilanUang tabungan = new PengambilanUang(5000, 1000);
   System.out.println("Uang yang ditabung : 5000");
   System.out.println("Uang yang diproteksi : 1000");
   System.out.println("-----------------------------");
   System.out.println("Uang yang akan diambil : 4500 " +
   tabungan.ambilUang(4500));
   System.out.println("Saldo sekarang : " + tabungan.getSaldo());
   System.out.println("-----------------------------");
   System.out.println("Uang yang akan diambil : 2500 " +
   tabungan.ambilUang(2500));
   System.out.println("Saldo sekarang : " + tabungan.getSaldo());
   }
}
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
Uang yang ditabung : 5000
Uang yang diproteksi : 1000
-----------------
Uang yang akan diambil : 4500 false
Saldo sekarang : 5000
-----------------
Uang yang akan diambil : 2500 true
Saldo sekarang : 2500
```

Jawab:

Berikut Kode program untuk clas diagram tersebut.

PengambilanUang.java

```java
package perbankan;

public class PengambilanUang extends Tabungan {
    private int proteksi;

    public PengambilanUang(int saldo) {
        super(saldo);
        this.proteksi = 0;
    }

    public PengambilanUang(int saldo, int proteksi) {
        super(saldo);
        this.proteksi = proteksi;
    }

    public int getSaldo() {
        return super.getSaldo();
    }

    public boolean ambilUang(int jumlah) {
        int saldoSetelahProteksi = super.getSaldo() - proteksi;
        if (saldoSetelahProteksi >= jumlah) {
            return super.ambilUang(jumlah);
        } else {
            return false;
        }
    }
}
```

Analisa Kode Program

Inheritance (Pewarisan): Class PengambilanUang mewarisi semua atribut dan metode dari class Tabungan. Dengan menggunakan kata kunci extends, class PengambilanUang dapat mengakses atribut dan metode yang ada di class Tabungan.

Private Variable (Variabel Private): Terdapat sebuah variabel private dengan tipe data int yang bernama proteksi. Variabel ini hanya dapat diakses oleh class PengambilanUang dan tidak dapat diakses oleh class lain.

Constructor (Konstruktor): Terdapat dua konstruktor dalam class PengambilanUang. Konstruktor pertama menerima satu parameter saldo dan digunakan untuk menginisialisasi atribut saldo dari class Tabungan menggunakan konstruktor kelas induk (super(saldo)). Konstruktor kedua menerima dua parameter saldo dan proteksi dan juga menginisialisasi atribut saldo dan proteksi.

Getter Method (Metode Getter): Terdapat sebuah metode getSaldo() yang mengembalikan nilai dari atribut saldo menggunakan metode getSaldo() dari class Tabungan.

Method (Metode) ambilUang(): Terdapat sebuah metode ambilUang() yang menerima satu parameter jumlah dan mengembalikan nilai boolean. Metode ini digunakan untuk mengambil uang dari saldo dengan memeriksa apakah saldo yang tersedia cukup untuk melakukan penarikan. Jika saldo yang tersedia dikurangi dengan proteksi lebih besar atau sama dengan jumlah, maka metode ambilUang() dari class Tabungan akan dipanggil dan metode ini akan mengembalikan nilai yang sama dengan hasil pemanggilan metode tersebut. Jika saldo tidak mencukupi, metode akan mengembalikan nilai false.

Output

```
PS C:\Users\USER\OOP\Praktikum\Praktikum9\tugas> javac *.java
PS C:\Users\USER\OOP\Praktikum\Praktikum9\tugas> java TesTugas
Uang yang ditabung : 5000
Uang yang diproteksi : 1000
----------------------------
Uang yang akan diambil : 4500 false
Saldo sekarang : 5000
----------------------------
Uang yang akan diambil : 2500 true
Saldo sekarang : 2500
```

E. KESIMPULAN

Inheritance, atau pewarisan, adalah konsep kunci dalam pemrograman berorientasi objek yang memungkinkan kelas baru untuk mewarisi properti dan metode dari kelas yang sudah ada. Dengan inheritance, kode dapat digunakan kembali secara efisien, mengurangi duplikasi dan memfasilitasi polimorfisme, di mana objek dari kelas yang berbeda dapat merespons metode yang sama dengan cara yang berbeda. Selain itu, inheritance memungkinkan pembentukan hierarki kelas yang logis, memisahkan tanggung jawab dan memungkinkan pengorganisasian kode yang lebih mudah dibaca dan dipahami. Namun, perlu diingat bahwa pemilihan dan perancangan hierarki kelas yang bijak adalah kunci untuk memanfaatkan inheritance dengan efektif, karena penyalahgunaannya dapat menghasilkan struktur yang kompleks dan sulit dipelihara. Dengan memahami konsep ini, programmer dapat meningkatkan efisiensi, keterbacaan, dan keberlanjutan kode dalam pengembangan aplikasi berorientasi objek.