LAPORAN RESMI

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

INHERITANCE 2



Fadilah Fahrul Hardiansyah S.ST., M. Kom

Ratri Maria Manik

3121600039

D4 TEKNIK INFORMATIKA – B

PROGRAM STUDI TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

TA 2022/2023

1. **TUGAS PENDAHULUAN**
2. Ada berapa modifer untuk pengontrolan akses? Jelaskan masing – masing!
3. Public

Sebuah kelas, method, ataupun property yang mempunyai akses modifier public artinya bahwa kelas, method, ataupun property tersebut dapat diakses oleh kelas manapun

1. Protected

Sebuah method, ataupun property yang mempunyai akses modifier protected artinya bahwa method, ataupun property tersebut dapat diakses hanya oleh kelas turunannya (subclass) dan hanya dapat diakses oleh kelas yang satu package

1. Private

Access modifier private bersifat tertutup. Sesuai dengan konsep OOP Encapsulation, maka setiap variabel wajib untuk dilindungi hak aksesnya secara langsung dari luar. Oleh karena itu, variabel diberikan hak akses private dan untuk melakukan pengaksesan/perubahan data digunakan setter getter.

1. Default

Access modifier default bisa diakses oleh file di class yang sama atau

package yang sama.

1. Apakah kegunaan kata kunci super? Jelaskan!

Kata kunci super dipakai untuk merujuk pada member dari parent class, sebagaimana kata kunci this yang dipakai untuk merujuk pada member dari class itu sendiri. Adapun format penulisannya yaitu

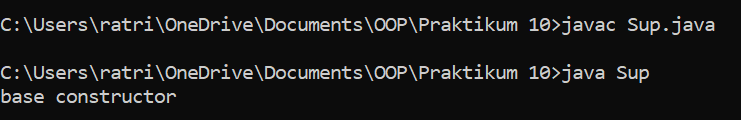
1. super.data\_member 🡪 Data member pada parent class
2. super.function\_member 🡪 Function member pada parent class
3. super() 🡪 konstruktor pada parent class
4. Apakah yang dimaksud dengan konstruktor tidak diwariskan?

konstruktorparent class tidak diwariskan ke sub class. Tetapi ketika suatu obyek anak dibuat dalam artian ketika konstruktor anak dijalankan maka konstruktor parent class dijalankan terlebih dahulu dan selanjutnya menyelesaikan konstruktor anak.

1. **LATIHAN**
2. Konstruktor tidak diwariskan

|  |  |
| --- | --- |
| * Class Base | * Class Sup |

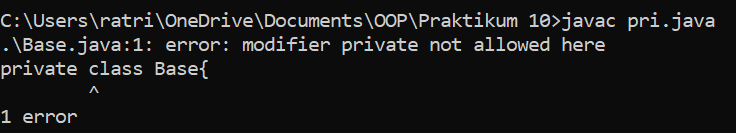
* Hasil kompile



1. Konstruktor tidak diwariskan

|  |  |
| --- | --- |
| * Class Base | * Class Pri |

* Hasil kompile



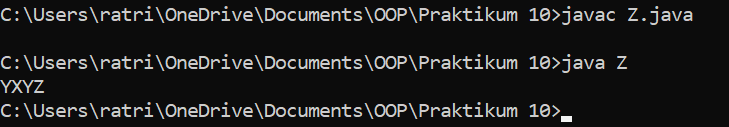
Analisa :

Program tersebut tejadi error dikarenakan pada parent classnya(class Base) memiliki modifier private, sehingga tidak dapat diakses oleh class manapun, modifier private hanya dapat diakses oleh class yang sama.

1. Apa yang tampil di layar bila kode dibawah ini dijalankan?

|  |  |
| --- | --- |
| * Class Y | * Class X |
| * Class Z |  |

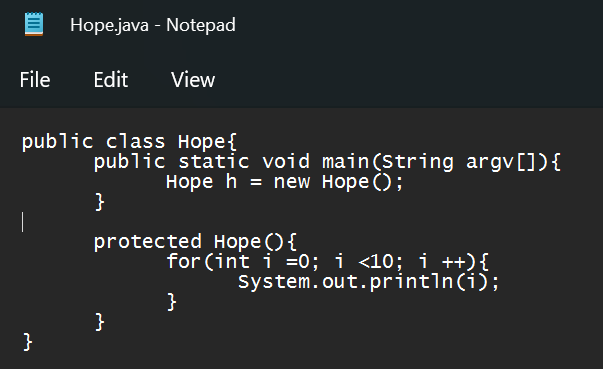
* Hasil kompile



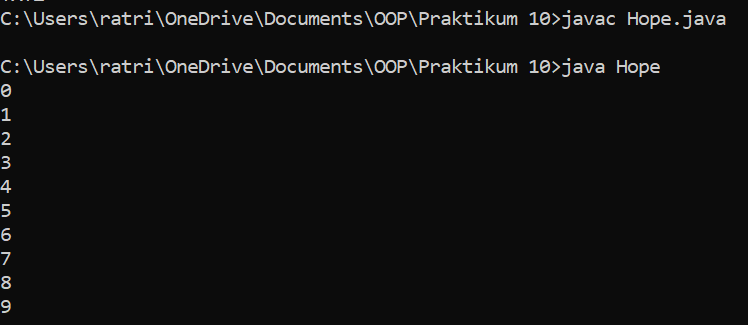
Analisa :

Konstruktor yang pertama kali diproses yaitu Y, karena class Y merupakan konstruktor parent, sebelum subclass menjalankan konstruktornya sendiri, subclass akan menjalankan konstruktor parent terlebih dahulu (aturan tentang konstruktor tidak diwariskan). Selanjutnya akan menjalankan konstruktor X, dan terakhir menjalankan konstruktor Z.

1. Kompile dan jalankan program berikut! Apa yang terjadi? Jelaskan!



* Hasil kompile



Analisa :

Program tersebut digunakan untuk menampilkan bilangan dari 0 hingga 9. Dapat dilihat pada baris pertama memiliki modifier public, sedangkan pada bagian konstruktornya memiliki modifier protected.

1. **TUGAS**
2. Mengimplementasikan UML class diagram dalam program
3. Kendaraan.java

|  |
| --- |
| public class Kendaraan{  protected int jmlRoda;  protected String warna;  public void setJmlRoda(int jmlRoda){  this.jmlRoda = jmlRoda;  }  public int getJmlRoda(){  return jmlRoda;  }  public void setWarna(String warna){  this.warna = warna;  }  public String getWarna(){  return warna;  }  } |

1. Mobil.java

|  |
| --- |
| public class Mobil extends Kendaraan{  protected String bahanBakar;  protected int kapasitasMesin;  public void setBahanBakar(String bahanBakar){  this.bahanBakar = bahanBakar;  }  public String getBahanBakar(){  return bahanBakar;  }  public void setKapasitasMesin(int kapasitasMesin){  this.kapasitasMesin = kapasitasMesin;  }  public int getKapasitasMesin(){  return kapasitasMesin;  }  } |

1. Sepeda.java

|  |
| --- |
| public class Sepeda extends Kendaraan{  protected int jmlSadel;  protected int jmlGir;  public void setJmlSadel(int jmlSadel){  this.jmlSadel = jmlSadel;  }  public int getJmlSadel(){  return jmlSadel;  }  public void setJmlGir(int jmlGir){  this.jmlGir = jmlGir;  }  public int getJmlGir(){  return jmlGir;  }  } |

1. Truk.java

|  |
| --- |
| public class Truk extends Mobil{  protected int muatanMaks;  public void setMuatanMaks(int muatanMaks){  this.muatanMaks = muatanMaks;  }  public int getMuatanMaks(){  return muatanMaks;  }  } |

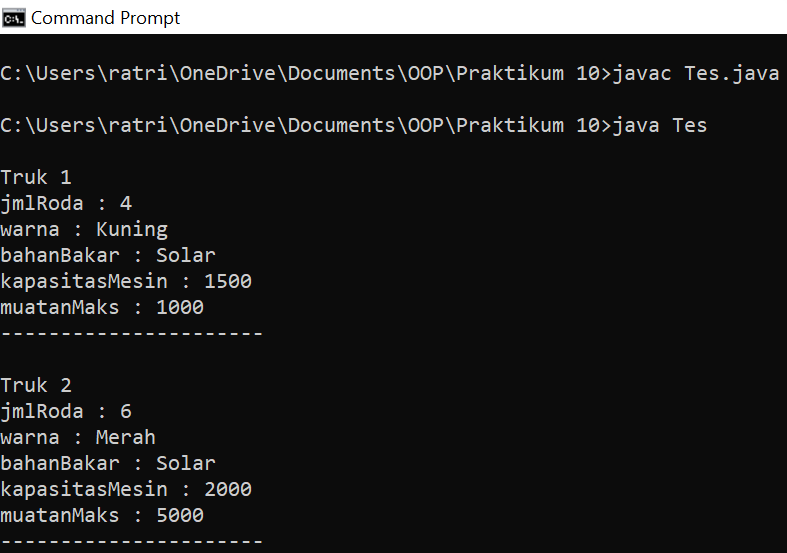
1. Taksi.java

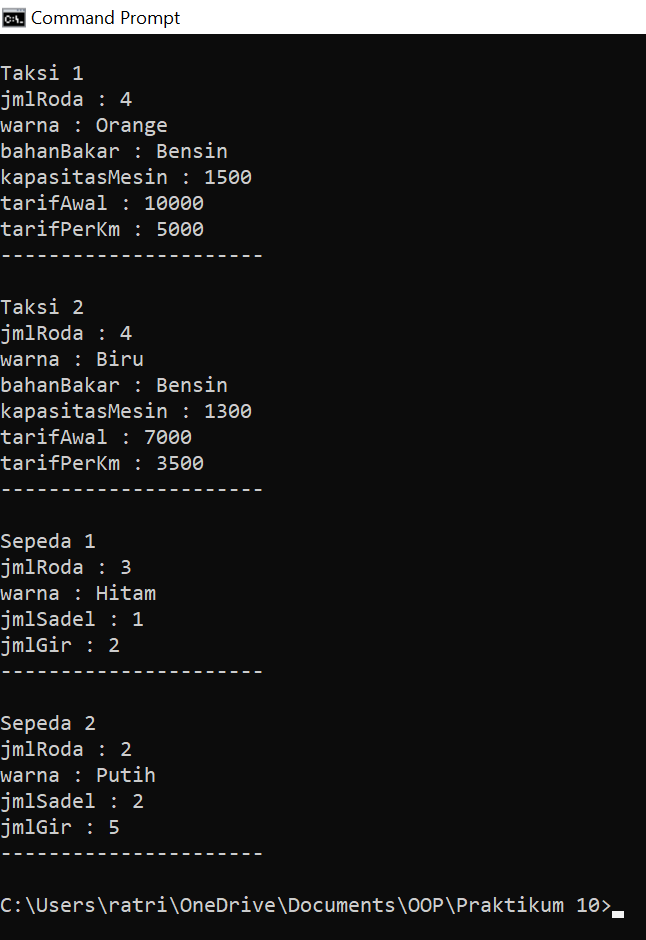
|  |
| --- |
| public class Taksi extends Mobil{  protected int tarifAwal;  protected int tarifPerKm;  public void setTarifAwal(int tarifAwal){  this.tarifAwal = tarifAwal;  }  public int getTarifAwal(){  return tarifAwal;  }  public void setTarifPerKm(int tarifPerKm){  this.tarifPerKm = tarifPerKm;  }  public int getTarifPerKm(){  return tarifPerKm;  }  } |

1. Tes.java

|  |
| --- |
| public class Tes{  public static void main(String args[]){  Truk truk1 = new Truk();  truk1.setJmlRoda(4);  truk1.setWarna("Kuning");  truk1.setBahanBakar("Solar");  truk1.setKapasitasMesin(1500);  truk1.setMuatanMaks(1000);  Truk truk2 = new Truk();  truk2.setJmlRoda(6);  truk2.setWarna("Merah");  truk2.setBahanBakar("Solar");  truk2.setKapasitasMesin(2000);  truk2.setMuatanMaks(5000);  Taksi taksi1 = new Taksi();  taksi1.setJmlRoda(4);  taksi1.setWarna("Orange");  taksi1.setBahanBakar("Bensin");  taksi1.setKapasitasMesin(1500);  taksi1.setTarifAwal(10000);  taksi1.setTarifPerKm(5000);  Taksi taksi2 = new Taksi();  taksi2.setJmlRoda(4);  taksi2.setWarna("Biru");  taksi2.setBahanBakar("Bensin");  taksi2.setKapasitasMesin(1300);  taksi2.setTarifAwal(7000);  taksi2.setTarifPerKm(3500);  Sepeda sepeda1 = new Sepeda();  sepeda1.setJmlRoda(3);  sepeda1.setWarna("Hitam");  sepeda1.setJmlSadel(1);  sepeda1.setJmlGir(2);  Sepeda sepeda2 = new Sepeda();  sepeda2.setJmlRoda(2);  sepeda2.setWarna("Putih");  sepeda2.setJmlSadel(2);  sepeda2.setJmlGir(5);  System.out.println("\nTruk 1");  System.out.println("jmlRoda : " + truk1.getJmlRoda());  System.out.println("warna : " + truk1.getWarna());  System.out.println("bahanBakar : " + truk1.getBahanBakar());  System.out.println("kapasitasMesin : " + truk1.getKapasitasMesin());  System.out.println("muatanMaks : " + truk1.getMuatanMaks());  System.out.println("----------------------");  System.out.println("\nTruk 2");  System.out.println("jmlRoda : " + truk2.getJmlRoda());  System.out.println("warna : " + truk2.getWarna());  System.out.println("bahanBakar : " + truk2.getBahanBakar());  System.out.println("kapasitasMesin : " + truk2.getKapasitasMesin());  System.out.println("muatanMaks : " + truk2.getMuatanMaks());  System.out.println("----------------------");  System.out.println("\nTaksi 1");  System.out.println("jmlRoda : " + taksi1.getJmlRoda());  System.out.println("warna : " + taksi1.getWarna());  System.out.println("bahanBakar : " + taksi1.getBahanBakar());  System.out.println("kapasitasMesin : " + taksi1.getKapasitasMesin());  System.out.println("tarifAwal : " + taksi1.getTarifAwal());  System.out.println("tarifPerKm : " + taksi1.getTarifPerKm());  System.out.println("----------------------");  System.out.println("\nTaksi 2");  System.out.println("jmlRoda : " + taksi2.getJmlRoda());  System.out.println("warna : " + taksi2.getWarna());  System.out.println("bahanBakar : " + taksi2.getBahanBakar());  System.out.println("kapasitasMesin : " + taksi2.getKapasitasMesin());  System.out.println("tarifAwal : " + taksi2.getTarifAwal());  System.out.println("tarifPerKm : " + taksi2.getTarifPerKm());  System.out.println("----------------------");  System.out.println("\nSepeda 1");  System.out.println("jmlRoda : " + sepeda1.getJmlRoda());  System.out.println("warna : " + sepeda1.getWarna());  System.out.println("jmlSadel : " + sepeda1.getJmlSadel());  System.out.println("jmlGir : " + sepeda1.getJmlGir());  System.out.println("----------------------");  System.out.println("\nSepeda 2");  System.out.println("jmlRoda : " + sepeda2.getJmlRoda());  System.out.println("warna : " + sepeda2.getWarna());  System.out.println("jmlSadel : " + sepeda2.getJmlSadel());  System.out.println("jmlGir : " + sepeda2.getJmlGir());  System.out.println("----------------------");  }  } |

1. Hasil kompile





Analisa :

Apabila dilihat dari UML class diagram pada soal, terdapat single inheritance dan multilevel inheritance. Single inheritance ketika suatu subclass memiliki satu parent class, adapun parent class pada UML class diagram yaitu class “Kendaraan”. Multilevel inheritance ketika suatu subclass memiliki subclass lainnya seperti subclass “Mobil” memiliki subclass “Truk” dan subclass “Taksi”. Metode inheritance yang diterapkan pada soal ini yaitu setiap subclass memiliki suatu atribut yang sama yaitu “jmlRoda(int)” dan “warna(String)”, serta subclass tersebut juga memiliki metode setter dan getter yang sama terhadap atribut yang sama tersebut.