LAPORAN RESMI

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

POLYMORPHISM



Fadilah Fahrul Hardiansyah S.ST., M. Kom

Ratri Maria Manik

3121600039

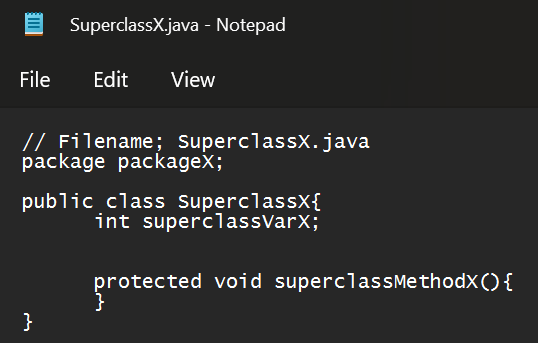
D4 TEKNIK INFORMATIKA – B

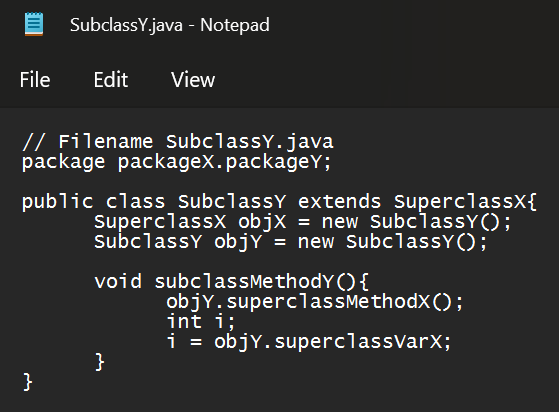
PROGRAM STUDI TEKNIK INFORMATIKA

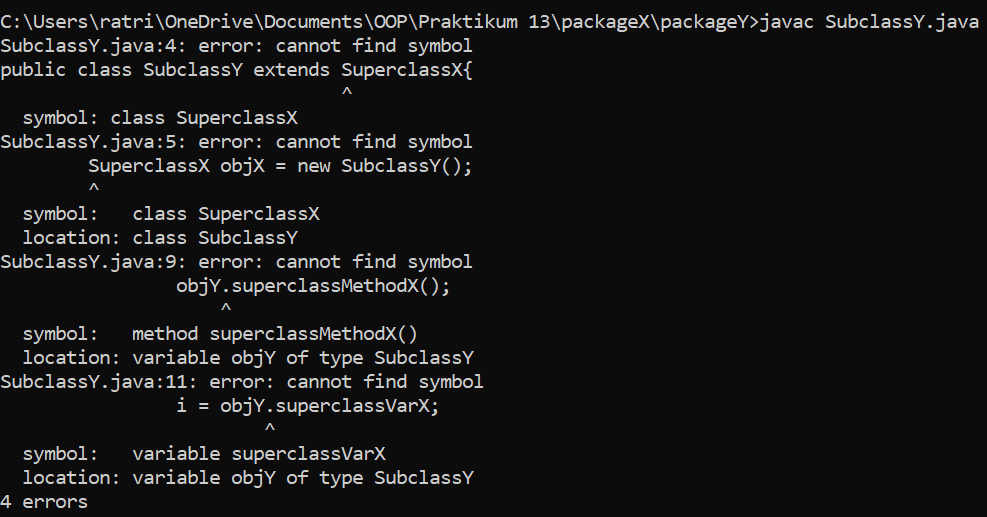
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

TA 2022/2023

1. **LATIHAN**
2. Apa yang terjadi bila kode dibawah ini dikompile dan dijalankan jika sebelumnya Base.java belum dikompile? Jelaskan !



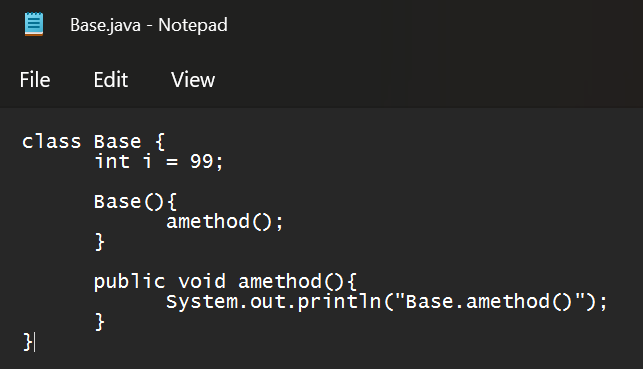


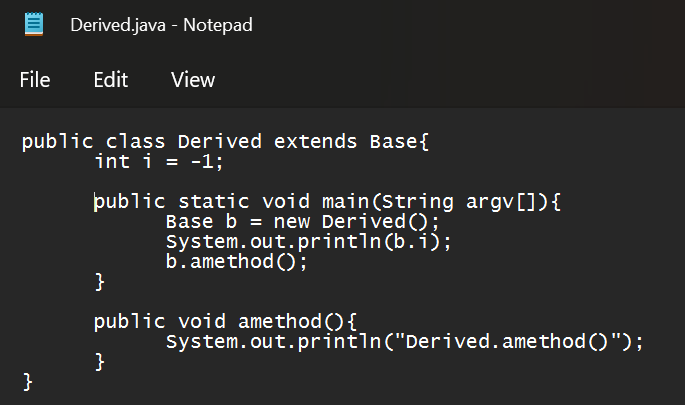


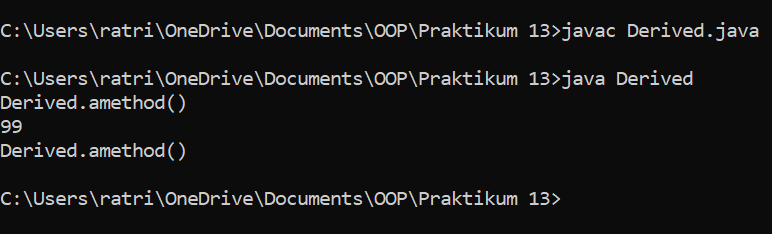
Analisa :

Terjadi error karena atribut pada pada parentclassnya memiliki modifier default, sehingga tidak bisa diakses oleh childclass.

1. Apa yang tampil di layar, jika kode dibawah ini dijalankan? Jelaskan!



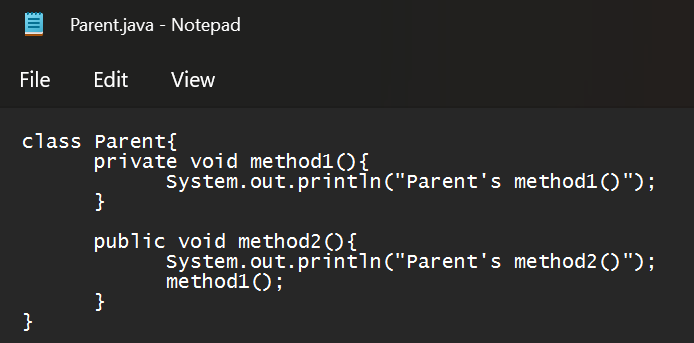


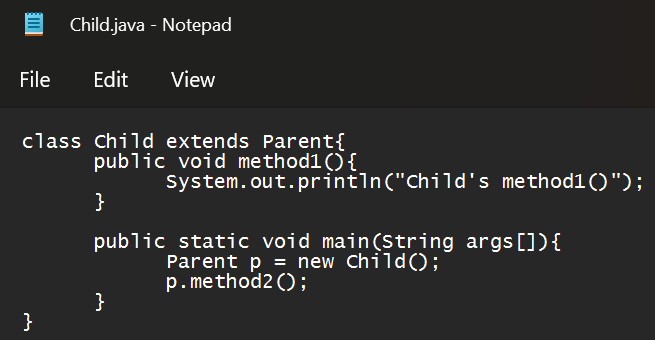


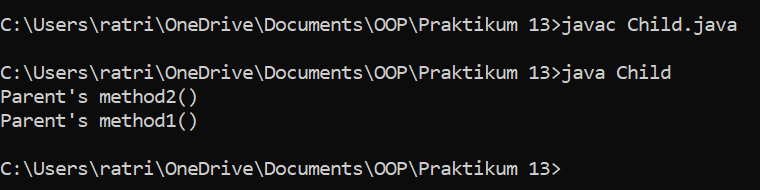
Analisa :

Polimorfisme pada program terjadi pada **Base b = new Derived()** dimana terjadi instance objek b bertipe Base.

1. Apa yang tampil di layar, jika kode dibawah ini dijalankan? Jelaskan!



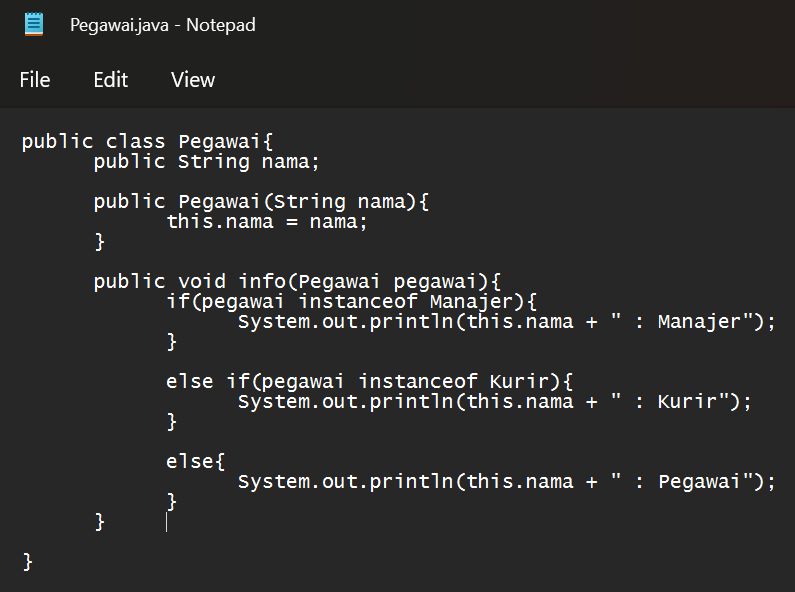


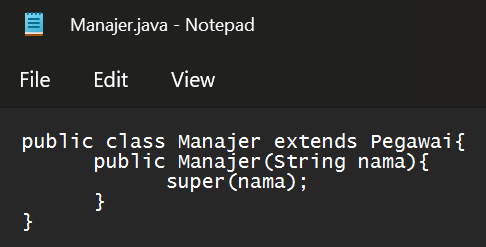


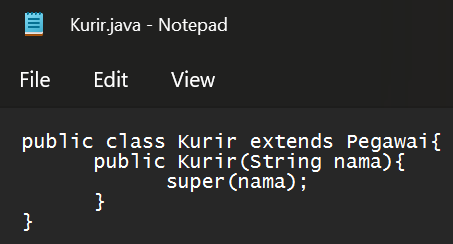
Analisa :

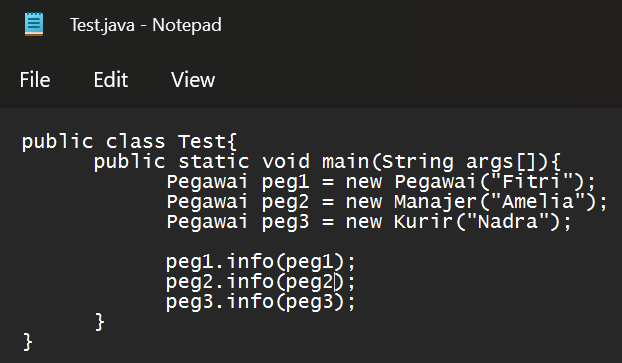
Terjadi polimorfisme pada class Child di method main, terjadi instance objek p bertipe Parent lalu menjalankan constructor Child.

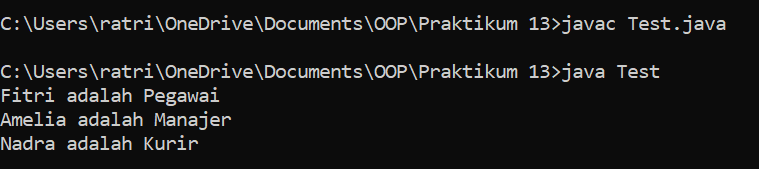
1. Suatu program terdiri dari class Pegawai sebagai parent class, class Manajer dan class Kurir sebagai subclass. Buatlah suatu program yang menerapkan konsep polymorphic argument sebagaimana yang telah disinggung dalam pembahasan sebelumnya.











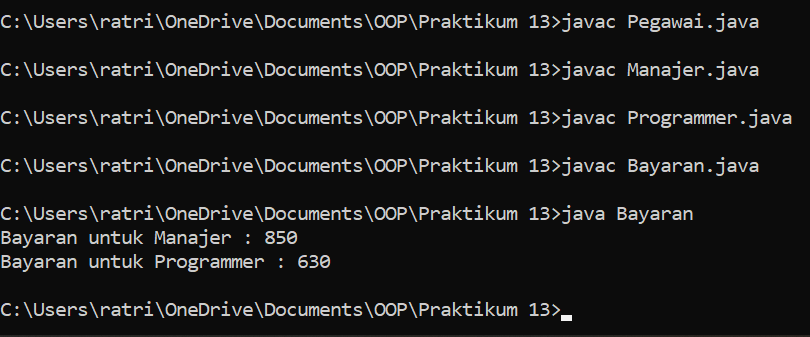
1. **TUGAS**
2. Pengimplementasian UML class Diagram

|  |
| --- |
| public class Pegawai{  protected String nama;  protected int gaji;    public Pegawai(String nama, int gaji){  this.nama = nama;  this.gaji = gaji;  }  public int infoGaji(){  return gaji;  }  } |

|  |
| --- |
| public class Manajer extends Pegawai{  private int tunjangan;  public Manajer(String nama, int gaji, int tunjangan){  super(nama, gaji);  this.tunjangan = tunjangan;  }  public int infoGaji(){  return super.infoGaji();  }  public int infoTunjangan(){  return tunjangan;  }  } |

|  |
| --- |
| public class Programmer extends Pegawai{  private int bonus;  public Programmer(String nama, int gaji, int bonus){  super(nama, gaji);  this.bonus = bonus;  }  public int infoGaji(){  return super.infoGaji();  }  public int infoBonus(){  return bonus;  }  } |

|  |
| --- |
| public class Bayaran{  public int hitungbayaran(Pegawai peg){  int uang = peg.infoGaji();  if (peg instanceof Manajer)  uang += ((Manajer) peg).infoTunjangan();  else if (peg instanceof Programmer)  uang += ((Programmer) peg).infoBonus();  return uang;  }    public static void main(String args[]){  Manajer man = new Manajer("Agus", 800, 50);  Programmer prog = new Programmer("Budi", 600, 30);  Bayaran hr = new Bayaran();  System.out.println("Bayaran untuk Manajer : " + hr.hitungbayaran(man));  System.out.println("Bayaran untuk Programmer : " + hr.hitungbayaran(prog));  }  } |



Analisa :

Terjadi polimorfisme pada class Bayaran. Terdapat instance dari class Manajer, Programmer, dan Bayaran. Terbentuk objek baru bernama hr lalu menjalankan method hitungBayaran yang mengirim instance dari class Manajer dan Programmer. Pada method ini terjadi pengecekan asal instance menggunakan instanceof lalu menjalankan sesuai instancenya. Pada method ini juga terdapat casting objek parameter dari tipe parameter ke tipe asal agar kita dapat menggunakan kedua method yang berasal dari subclassnya.