OVERLOADING & OVERRIDING

**I. Tujuan Instruksional Khusus**

Setelah menyelesaikan praktek ini mahasiswa dapat :

* 1. menjelaskan pengertian konsep *polimorfisme,overloading* dan *overriding,*
  2. menjelaskan jenis metode *overloadin,*
  3. dapat membuat program dengan mengimplementasikan konsep *polimorfisme.*

# II. Persyaratan / Kebutuhan Sistem Komputer

# Persyaratan sistem untuk menjalankan program Java adalah sebagai berikut :

# processor P-II atau yang lebih tinggi,

# ram minimal 32 mbyte,

# free space harddisk minimal 200 mbyte,

# mouse ( serial / ps2 ),

# sistem operasi :

# minimal Windows 98 ( disarankan versi Windows yang lebih tinggi),

# linux.

**III. Dasar Teori**

***Polimorfisme***

*Polymorphism* (berasal dari bahasa yunani (*Greek*) yang artinya adalah macam-macam bentuk (*many forms*) adalah sebuah konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku yang berbeda. *overloading* adalah salah satu cara penerapan dalam konsep polimorfisme. *Overloading* ada dua jenis, yaitu:

1. *Overloading methode*
2. *Overloading constructor*

***Overloading methode***

Dalam Java memungkinkan untuk mendefinisikan dua atau lebih metode dalam kelas yang sama, menggunakan nama yang sama, tetapi parameter yang dideklarasikan harus berbeda (jumlah atau tipe parameternya). Jika jumlah parameternya sama maka Tipe harus berbeda semua, tetapi jika jumlah parameternya berbeda-beda maka boleh tipenya sama. Jenis-jenis metode overloading:

1. Jumlah parameter beda, tipe beda
2. Jumlah parameter sama, tipe beda
3. Jumlah parameter beda, tipe sama

Metode overloading ini, terjadi pada objek yang sama, tapi nilai argumennya berbeda-beda.

Ketika metode **overloaded** dipanggil, Java mencari kesamaan antara argumen yang digunakan untuk memanggil metode dan parameter dari metode. Tetapi, kesamaan ini tidak mutlak. Hal ini karena pada java adanya otomatisasi konversi tipe yang berbeda.

|  |
| --- |
| **//PENERAPAN KONVERSI OTOMATIS PADA METODE OVERLOADING**  class OverloadMetode  {  //konversi tujuan bertipe integer  void test(int a)  {  System.out.println("Hasil konversi byte ke integer : " + a);  }  }  class OverloadKonversi  {  public static void main(String args[])  {  OverloadMetode ob = new OverloadMetode();  //variabel asal yang akan dikonversi bertipe byte  byte i= 88;  ob.test(i); // memanggil test(integer)  }  } |

Konversi otomatis pada java seperti program diatas, hanya dapat dilakukan jika kedua aturan ini dipenuhi:

1. Kedua tipe data (data asal dan tujuan) memiliki kompatibilitas(*compatible*).

2. Tipe tujuan mempunyai range lebih besar daripada tipe asal.

Selain itu Java juga mengerjakan konversi secara otomatis ketika menyimpan konstanta integer literal menjadi variable bertipe **byte**, **short**, atau **long**.

###### **Overloading constructors**

*Overloading* selain diterapkan pada **metode** juga diterapkan pada **konstuktor**. Bedanya *overloading constructor* terjadi pada objek yang berbeda-beda, nilai argument berbeda, sehingga metode bisa lebih dari satu. Penerapan *overloading* pada kelas Java akan menjadi suatu norma tanpa perkecualian.

###### **Method Overriding**

Metode overriding adalah metode pada subclass yang mempunyai nama dan tipe identik/sama dengan metode superclass. Jika kita ingin memanggil metode superclass pada subclass dapat menggunakan statemen berikut:

**Super.metode();**

Metode() merupakan nama metode superclass yang dipanggil.

**IV. Langkah Kerja**

Jalankan Edit Plus, buat file baru java dan ketikkan kode program berikut :

1. Buat program OVERLOAD JUMLAH PARAMETER BEDA, TIPE BEDA

|  |
| --- |
| class OverLoadMetode  {  //overload tes pertama  void tes()  {  System.out.println("Tanpa Parameter");  }  //overload kedua  void tes(int a)  {  System.out.println("nilai a:"+a);  }  //overload ketiga  void tes(int a,int b)  {  System.out.println("Nilai a dan b = "+a+" dan "+b);  }  //overload ke empat  double tes(double c)  {  System.out.println("Nilai c="+c);  return c\*c;  }  }  class OverLoad1  {  public static void main(String[] args)  {  OverLoadMetode demo = new OverLoadMetode();  double kali;  //Memanggil semua versi tes  demo.tes(); demo.tes(10); demo.tes(5,8);  kali=demo.tes(10.0);  System.out.println("Hasil (c\*c) ="+kali);  }  } |

1. Sebagai contoh di bawah ini diberikan sebuah program menggunakan OVERLOAD DENGAN JUMLAH PARAMETR SAMA, TIPE BEDA .

|  |
| --- |
| class OverLoadMetode  {  //overload kesatu  void tes(double a)  {  System.out.println("Nilai a bertipe double ="+a);  }  //overload ke dua  int tes(int b)  {  System.out.println("Nilai b bertipe integer="+b);  return b+b;  }  }  class OverLoad2  {  public static void main(String[] args)  {  OverLoadMetode demo = new OverLoadMetode();  int jum;  //memanggil versi tes 1  demo.tes(2.5);  //Memanggil versi tes kedua  jum=demo.tes(10);  System.out.println("Hasil penjumlahan b + b ="+jum);  }  } |

1. Progaram menggunakan JUMLAH PARAMETER BERBEDA, TIPE SAMA

|  |
| --- |
| class OverloadMetode  {  // Overload test for one integer parameter.  void test(int a)  {  System.out.println("Nilai a: " + a);  }  // Overload test for two integer parameters.  int test(int a, int b)  {  return a+b;  }  // overload test for tree integer parameter  int test(int a,int b, int c)  {  return a\*b\*c;  }  }  class Overload3  {  public static void main(String args[])  {  OverloadMetode ob = new OverloadMetode();  int jumlah,kali;  // call all versions of test()  ob.test(10);  jumlah=ob.test(10,20) ;  kali = ob.test(10,20,30);  System.out.println("Hasil penjumlahan (10+20): " + jumlah);  System.out.println("Hasil perkalian (10\*20\*30): " + kali);  }  } |

1. Berikut ini program menggunakan overloading konstruktor

|  |  |  |
| --- | --- | --- |
| |  | | --- | | class Man  {  double gapok; double masakerja;  // konstruktor untuk Man.  Man(double g, double mk)  {  gapok=g; masakerja=mk;  }  // konstructor yang tidak ditentukan ukurannya  Man()  {  gapok = 0; masakerja = 0;  }  // konstructor digunakan ketika KaryawanKita dibuat  Man(double lembur)  {  gapok=masakerja=lembur;  }  double hitung\_gaji()  {  return gapok\*masakerja;  }  }  class OverloadingKonstruktor  {  public static void main(String args[])  {  Man KaryawanKu = new Man(10,15);  Man KaryawanMu = new Man();  Man KaryawanKita = new Man(5);  double gaji;  gaji = KaryawanKu.hitung\_gaji();  System.out.println("Gaji Karyawanku adalah = " + gaji);  gaji = KaryawanMu.hitung\_gaji();  System.out.println("Gaji Karyawanmu adalah = " + gaji);  gaji = KaryawanKita.hitung\_gaji();  System.out.println("Gaji KaryawanKita adalah = " + gaji);  }  } | |  | |

1. Berikut ini contoh program yang menerapkan konsep metode overriding menggunakan tipe pengembalian nilai.

class superclass

{

int p,l;

superclass(int panjang,int lebar)

{

p=panjang;

l=lebar;

}

int hitung()

{

return p\*l;

}

}

class subclass extends superclass

{

int t;

subclass(int panjang,int lebar,int tinggi)

{

super(panjang,lebar);

t=tinggi;

}

{

/\*CARA 1

int L;

L=super.hitung();//MEMANGGIL METODE SUPERCLASS

return L\*t;

\*/

//CARA 2

return p\*l\*t;

}

}

class AplikasiMetodeOverriding

{

public static void main(String[] args)

{

subclass Objek=new subclass(3,4,6);

System.out.println("LUAS "+Objek.hitung());

}

1. Program Simpan program Java di atas.
2. Eksekusi program Java Pertama dengan mengklik ***Tool*** 🡪 ***Compile Program Java***. Lalu ***Tool*** 🡪 ***Menjalankan Program Java***.
3. Buat analisa terhadap proses program tersebut sebelum dijalankan dan pada saat dijalankan terhadap parameter dan argument yang digunakan..

**V. Pertanyaan**

1. Jelaskan apa yang dimaksud overloading & overriding
2. Apa perbedaan implementasi overloading pada metode dan konstruktor
3. Lakukan analisis dan buat kesimpulan terhadap penerapan overloading pada metode dan konstruktor.
4. Buat program penerapan overloading
5. Buat contoh program konversi otomatis pada java untuk semua variasi tipe. Berikan keterangan pada ouput untuk konversi yang tidak diijinkan.
6. Jelaskan perbedaan overriding dengan overloading