**I. Tujuan Instruksional Khusus :**

Setelah Menyelesaikan praktek ini mahasiswa dapat :

1. Menjelaskan pengertian konsep pemrograman berorientasi objek
2. Mahasiswa dapat menjelaskan karakteristik PBO
3. Mahasiswa dapat membuat program nerorientasi objek dengan java

# II. Persyaratan / Kebutuhan Sistem Komputer

# Persyaratan / kebutuhan 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:**

**1. Membuat Metode Dengan Parameter Dan Argumen**

**Parameter** adalah sebuah variable yang didefinisikan oleh metode yang menerima sebuah nilai. Sedangkan **Argumen** adalah sebuah nilai yang dilewatkan ke metode ketika metode dipanggil. Dengan menambahkan **parameter**, akan menjadikan metode **lebih bermanfaat**. Karena metode dapat dioperasikan secara *general* pada data yang bervariasi dan dengan situasi yang sedikit berbeda. Potongan program dibawah ini adalah metode yang tidak menggunakan parameter.

|  |
| --- |
| double hitung\_gaji()  {  return 1000\*10;  } |

Jika program ini dijalankan, akan mengembalikan nilai hasil perhitungan dari gapok \* masakerja, tetapi penggunaanya sangat terbatas. Agar metode lebih bermanfaat, maka metode harus menggunakan **parameter**.

|  |
| --- |
| double hitung\_gaji(double gapok, int masakerja)  {  Return gapok \* masakerja;  } |

Metode **hitung\_gaji** sekarang akan lebih bermanfaat karena dapat menghitung gaji berdasarkan nilai yang dilewatkan kedalam parameter **gapok** dan **masakerja** tidak terbatas pada nilai 10000 dan 10

Bentuk umum dari sintaks **metode** dengan **parameter** adalah sebagai berikut:

|  |
| --- |
| Nama metode(*parameter*)  {  // kode program  } |

Pada penerapan program di atas, nilai *instance variable* dari setiap objek diset secara terpisah, dengan menggunakan statement seperti berikut ini:

|  |
| --- |
| //potongan program  //nilai instance variabel-1  KaryawanKu.nama="Eko";  KaryawanKu.gapok=1000;  KaryawanKu.masakerja=10;  //nilai instance variabel-2  KaryawanMu.nama="Dwi";  KaryawanMu.gapok=2000;  KaryawanMu.masakerja=10; |

Namun sebenarnya hal tersebut, **kurang tepat**, karena dapat menyebabkan masalah. Dalam program Java, nilai ***instance variable*** sebaiknya diakses **hanya** melalui **metode** yang didefiniskan oleh kelas itu sendiri. Caranya yaitu dengan menggunakan metode yang mempunyai parameter dan nilai / ukuran (*dimension*) dari instance variable yang ditetapkan/ditentukan dengan menggunakan metode, seperti contoh di atas menggunakan metode **SetDim.** Untuk lebih jelasnya, perhatikan program dibawah ini, yang merupakan modifikasi dari program sebelumnya.

**2. Konstruktor**

Konstruktor merupakan sebuah fungsi (metode) yang digunakan untuk menginisialisasikan suatu objek baru. Fungsi ini dijalankan secara otomatis pada saat kelas diciptakan, dengan **nama yang sama dengan kelasnya**, **dan tidak mengembalikan nilai**, tetapi tidak menggunakan fungsi **Void**. Konstruktor dipanggil ketika suatu objek dibuat berdasarkan kelasnya. Dari bentuk pendeklarasian, konstruktor dapat dibedakan menjadi dua, yaitu:

* Konstruktor implicit: didefinisikan oleh Java, dan disebut: *default constructor*
* Konstruktor eksplisit: ditulis oleh programmer.

Dari penggunaan parameter konstruktor ada dua jenis:

* Konstruktor dengan parameter
* Konstruktor tanpa parameter

Seperti kita ketahui, ketika kita mendeklarasikan sebuah objek, kita menggunakan bentuk umum sebagai berikut:

*Namakelas var* = new *namakelas*( );

Parentheses ( ) diperlukan sesudah nama kelas, sesungguhnya menunjukan bahwa konstruktor sedang dipanggil. Sehingga statemen berikut ini

persegi persegipanjang = **new** persegi ( );

**new persegi( )** adalah sedang memanggil konstruktor **persegi( )**. Jika kita tidak mendefinisikan sebuah konstruktor secara eksplisit untuk sebuah kelas, maka Java membuat sebuah *default construktor* untuk kelas tersebut. Hal inilah yang membuat mengapa program yang terdahulu ‘program pertama **class NonOOP**’ yang tidak mendefinisikan konstruktor secara eksplisit tetap berjalan. *Default konstruktor* secara otomatis menginisialisasi semua instance variables bernilai **nol**. Namun demikian *default konstruktor* hanya cocok untuk kelas-kelas yang sederhana.

###### Konstruktor Tanpa Parameter

Konstruktor ini digunakan jika objeknya mempunyai dimensi/ukuran yang sama.

###### Konstruktor Menggunakan Parameter

Ketika metode konstruksi **Man()** dalam contoh di atas diinisialisasi sebuah Objek, hal ini **sangat tidak bermanfaat**, karena semua **Man** mempunyai dimensi yang sama. Bagaimana caranya jika obyeknya mempunyai dimensi yang berbeda-beda? Solusinya adalah dengan menerapkan parameter ke konstuktor. Parameter ini merupakan variable lokal yang digunakan pada konstruktor dan dapat menggunakan nama yang sama dengan instance variable, tetapi biasanya lebih mudah menggunakan nama yang berbeda. Jika menggunakan nama yang sama maka variable lokal akan menyembunyikan ’hide’ instance variable, dan hal ini dapat menyebabkan terjadinya *collision* antara instance variable dengan variable lokal. Untuk mengatasi hal ini maka dapat menggunakan kata kunci **this**.

**IV. Langkah Kerja**

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

1. Buat program menggunakan metode dan parameter.

|  |  |
| --- | --- |
| |  | | --- | | **//DENGAN MENGGUNAKAN METODE DAN PARAMETER**  class Man  { //claas hanya membuat objek template  //instance variabel  String nama;  double gapok;  int masakerja;  //metode pengembalian nilai  int hitung\_gaji()  {  return gapok\*masakerja;  }  // Sets dimension Man  void setDim(String nm, double g, int mk )  {  Nama=nm;  gapok=g;  masakerja=Mk;  }  }  class KaryawanMetodeParameter  {  public static void main(String[] args)  {  Man KaryawanKu = new Man();//membuat objek sesungguhnya  Man KaryawanMu = new Man(); //membuat objek sesungguhnya  double gaji;  //nilai instance variabel-1  // penetapan nilai awal  KaryawanKu.setDim("Eko,1000,10);  gaji=KaryawanKu.hitung\_gaji();  System.out.println("Karyawan Ku:"+KaryawanKu.nama);  System.out.println("Gajinya:"+gaji);  //nilai instance variabel-2  KaryawanMu.setDim("Dwi",2000,10);  gaji=KaryawanMu.hitung\_gaji();  System.out.println("Karyawan Mu:"+KaryawanMu.nama);  System.out.println("Gajinya:"+gaji);  }  } | |

1. Sebagai contoh di bawah ini diberikan sebuah program menggunakan konsruktor tanpa parameter.

|  |  |
| --- | --- |
| |  | | --- | | class Man  {  String nama;  double gapok;  int masakerja;  // konstruktor tanpa parameter  Man()  {  System.out.println("Konstruksi dari Man");  nama="Eko";  gapok=1000;  masakerja=10.5;  }  double hitung\_gaji();  {  return gapok\*masakerja;  }  }  class KonstruktorKu  {  public static void main(String args[])  {  Man KaryawanKu = new Man()  Man KaryawanMu = new Man()  double gaji;  gaji = KaryawanKu.hitung\_gaji();  System.out.println("Volume is " + gaji);  gaji = KaryawanMu.hitung\_gaji();  System.out.println("Volume is " + gaji);  }  } | |

1. Penerapan parameter ke konstruktor

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | | class Man  {  String nama;  double gapok;  int masakerja;  Man(String nm, double g, int mk) // konstruktur Man.  {  nama=nm;  gapok=g;  masakerja=mk;  }  // metode untuk menghitung gaji dan me-return  double hitung\_gaji()  {  return gapok\*masakerja;  }  }  class KonstruktorParameter  {  public static void main(String args[])  {  // deklarasi, allokasi, dan inisialisasi objek  Man KaryawanKu = new Man(“eko”,10000,10);  Man KaryawanMu = new Man(“Dwi”,20000,10);  double gaji;    //cetak gaji dari KaryawanKu  gaji = KaryawanKu.hitung\_gaji();  System.out.println("Gaji Karyawanku " + gaji);    //cetak gaji dari KaryawanMu  gaji = KaryawanMu.hitung\_gaji();  System.out.println("Gaji Karyawanmu" + gaji);  }  } | | |

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. Latihan**

1. Buat program untuk menghitung gaji karyawan, dimana gaji total= (gaji pokok \* masakerja) + bonus.Bonus dihitung dengan ketentuan, sebagai berikut:

* pegawai Gol III bonus= 10% \* gaji pokok
* pegawai Gol II bonus= 15% \* gaji pokok
* pegawai Gol I bonus= 20% \* gaji pokok

2. Buat menu program (5 objek karyawan)(

1.Tambahkan Bonus Karyawan ( Sesuai dengan latihan 1)

2.Ada Potongan ( Cari nama karyawan lalu kurangi gajinya dengan potongan dan keterangannya potongannya apa)

3.Cetak Gaji Berdasarkan nama (termasuk Potongan dan bonus)

4.Cetak Semua Gaji (termasuk Potongan dan bonus)

5.Exit