

BAB I PENDAHULUAN

I.1 Latar belakang

Dalam Pemrograman Berbasis Object (PBO), kita harus memperhatikan dan memahami konsep-konsep dasar pada program itu sendiri, khususnya pada program java yang akan kita pakai. Pada makalah ini ditekankan untuk lebih memahami konsep dasar suatu sifat pada Java, yaitu *polymorphisme*, bagaimana fungsinya dan cara penulisannya. Oleh karena itu kami membahasnya dalam makalah ini.

I.2 Tujuan

Dalam makalah yang kami bahas bertujuan untuk mengetahui, memahami konsep-konsep dasar *polymorphisme* pada java yang berbasis OOP(Object Oriented Programming), dan bagaimana penulisannya. Adapun konsep-konsep tersebut antara lain :

Polymorphism

- Definisi
- Fungsi
- Jenis, dan contoh :
 1. Overriding
 2. Overloading

BAB II PEMBAHASAN

II.1 Definisi

Polimorfisme adalah suatu kejadian ketika objek dapat mengungkap banyak hal melalui suatu cara yang sama, suatu object dapat memiliki berbagai bentuk, sebagai object dari class sendiri atau object dari superclassnya. Dalam PBO, konsep ini memungkinkan digunakannya suatu interface yang sama untuk memerintah objek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda.

Secara harfiah, poli berarti “banyak” dan morph berarti “bentuk”. Jadi, polimorfisme berarti “mempunyai banyak bentuk”.

Polimorfisme mengizinkan kelas induk untuk mendefinisikan sebuah method general (bersifat umum) untuk semua kelas turunannya, dan selanjutnya kelas-kelas turunan dapat memperbarui implementasi dari method tersebut secara lebih spesifik sesuai dengan karakteristiknya masing-masing.

Untuk mempermudah dalam memahami konsep polimorfisme, marilah kita ambil sebuah contoh. Misalnya, kita memiliki sebuah kelas induk dengan nama Penyanyi, yang akan diturunkan lagi menjadi kelas penyanyi dangdut, penyanyi pop.

Sebagai contoh, ada kelas A yang diturunkan menjadi kelas B,C, dan D. Dengan konsep Polimorfisme, anda dapat menjalankan method – method yang terdapat pada kelas B,C, dan D hanya dari objek yang diinstansiasi dengan kelas A. Polimorfisme memungkinkan anda mengenali kesamaan diantara kelas yang berbeda.

II.2 Keuntungan Pemograman dengan menggunakan Polymorphism adalah :

- Kita dapat menggunakan kelas-kelas yang kita buat (sebagai super kelas) dan membuat kelas kelas baru berdasar superkelas tersebut dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki superkelas.
- kita dapat membuat super kelas yang hanya mendefinisikan behaviour namun tidak memberikan implementasi dari metode-metode yang ada. Hal ini berguna jika kita ingin membuat semacam template kelas, kelas semacam ini disebut kelas abstrak karena behaviournya masih abstrak dan belum diimplementasikan. subkelas-subkelas dari kelas semacam ini yang disebut kelas konkret, mengimplementasikan behaviour abstrak tersebut sesuai dengan kebutuhan masing-masing.
- Menghindari duplikasi object, kita dapat menciptakan class baru dari class yang sudah ada, sehingga tidak perlu menuliskan code dari nol ataupun mengulangnya, namun tetap bisa menambahkan attribute dan atau method unik dari class itu sendiri. Dalam konsep yang lebih umum sering kali polymorphism disebut dalam istilah satu interface banyak aksi.

Overriding dan overloading merupakan bagian dari polymorphism. Mengapa ?

Karena, pada dasarnya ada 2 tipe polymorphism, yaitu:

1. Dynamic polimorfisme atau true Merupakan function overriding (sebuah fungsi dalam class turunan yang memiliki nama, return type argumen function yang sama dengan fungsi dalam class induk) / (terjadi ketika deklarasi method subclass dengan nama dan parameter yang sama dengan method dari superclassnya). Syarat Override yaitu nama metode, return type, dan parameter harus sama. Jika tidak sama maka bukan dianggap sebagai override tetapi metode yang baru pada subclass. Pada override method, diharuskan menggunakan inheritance, agar dapat digunakan pada tiap-tiap class.

- Dengan overriding, kita dapat memiliki pengenalan method yang sama persis dengan dengan pengenalan method yang ada di super class, tapi berbeda behavior.
- Constructor tidak dapat dioverriding. Sebab constructor tidak diturunkan ke subclassnya.

• **Memanggil method yang dioverride di super class**

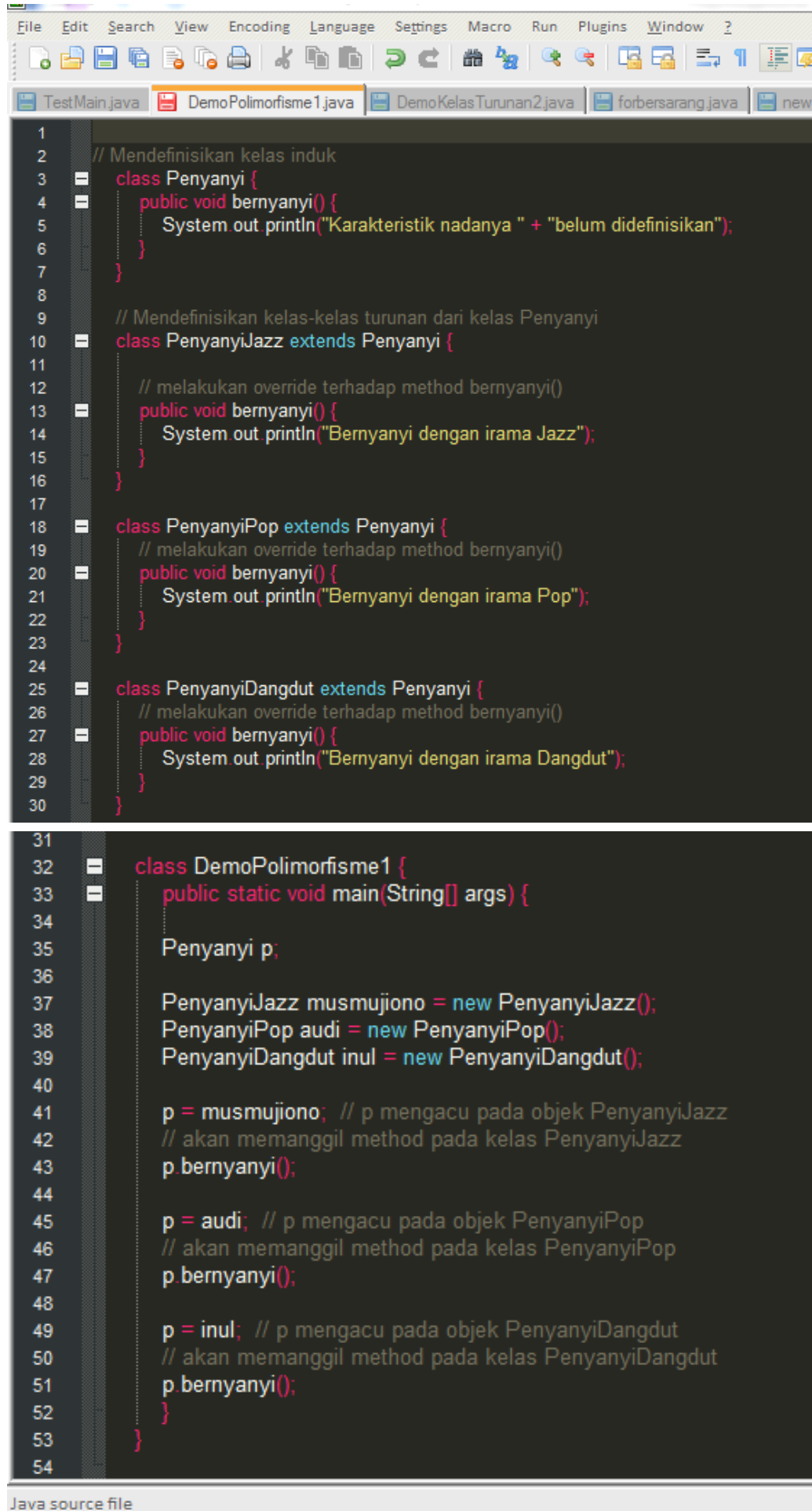
- Kita dapat memanggil method yang dioverride di superclass dengan keyword super.
- Penggunaan keyword super hanya dapat memanggil method di superclassnya langsung. Tidak dapat memanggil method di superclass dari superclass.

• **Syarat access modifier :**

Bila suatu method tidak dapat diturunkan, maka method tersebut tidak dapat dioverriding.

- Method beraccess modifier private adalah tidak diturunkan, oleh karena itu, bila di dalam superclass terdapat method private, dan kemudia pada subclassnya terdapat method yang namanya sama, maka method pada subclass tersebut sebenarnya adalah bukan overriding.
- Access modifier method yang mengoverride TIDAK BOLEH lebih restrictive bila dibandingkan dengan access modifier method yang dioverride.

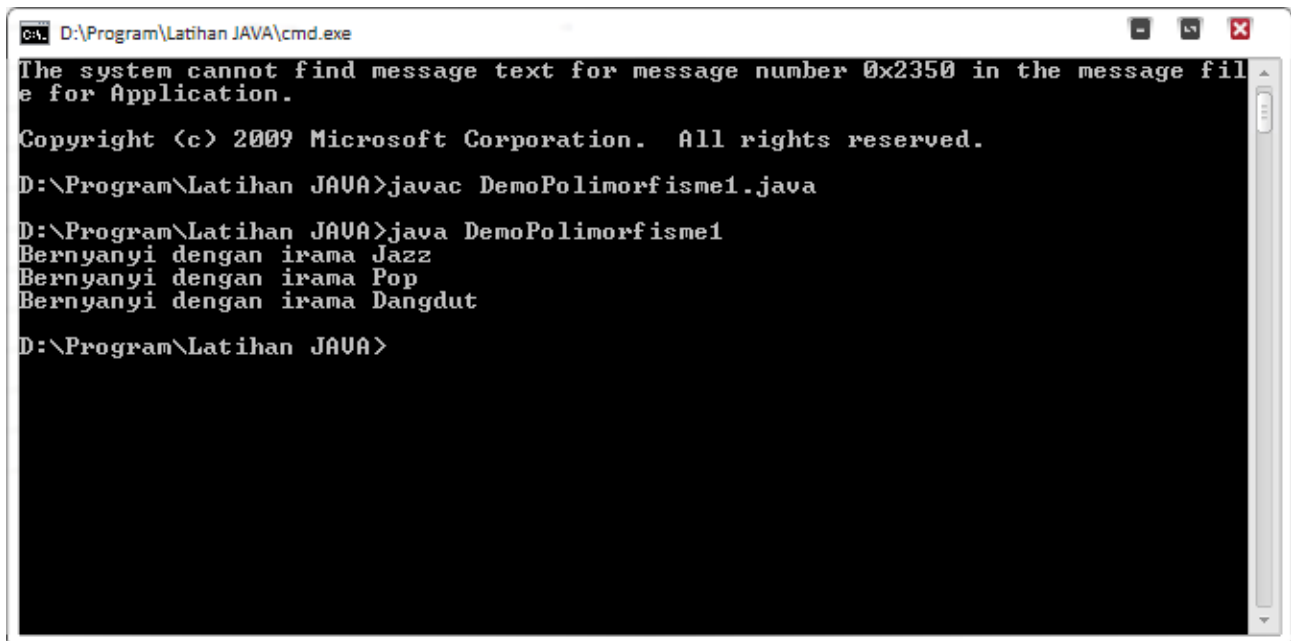
Contoh program :



```
1
2 // Mendefinisikan kelas induk
3 class Penyanyi {
4     public void bernyanyi() {
5         System.out.println("Karakteristik nadanya " + "belum didefinisikan");
6     }
7 }
8
9 // Mendefinisikan kelas-kelas turunan dari kelas Penyanyi
10 class PenyanyiJazz extends Penyanyi {
11
12     // melakukan override terhadap method bernyanyi()
13     public void bernyanyi() {
14         System.out.println("Bernyanyi dengan irama Jazz");
15     }
16 }
17
18 class PenyanyiPop extends Penyanyi {
19     // melakukan override terhadap method bernyanyi()
20     public void bernyanyi() {
21         System.out.println("Bernyanyi dengan irama Pop");
22     }
23 }
24
25 class PenyanyiDangdut extends Penyanyi {
26     // melakukan override terhadap method bernyanyi()
27     public void bernyanyi() {
28         System.out.println("Bernyanyi dengan irama Dangdut");
29     }
30 }
31
32 class DemoPolimorfisme1 {
33     public static void main(String[] args) {
34
35         Penyanyi p;
36
37         PenyanyiJazz musmujiono = new PenyanyiJazz();
38         PenyanyiPop audi = new PenyanyiPop();
39         PenyanyiDangdut inul = new PenyanyiDangdut();
40
41         p = musmujiono; // p mengacu pada objek PenyanyiJazz
42         // akan memanggil method pada kelas PenyanyiJazz
43         p.bernyanyi();
44
45         p = audi; // p mengacu pada objek PenyanyiPop
46         // akan memanggil method pada kelas PenyanyiPop
47         p.bernyanyi();
48
49         p = inul; // p mengacu pada objek PenyanyiDangdut
50         // akan memanggil method pada kelas PenyanyiDangdut
51         p.bernyanyi();
52     }
53 }
54
```

Java source file

Apabila di jalankan, program di atas akan memberikan hasil seperti berikut :



```
D:\Program\Latihan JAVA\cmd.exe
The system cannot find message text for message number 0x2350 in the message file for Application.

Copyright (c) 2009 Microsoft Corporation. All rights reserved.

D:\Program\Latihan JAVA>javac DemoPolimorfisme1.java
D:\Program\Latihan JAVA>java DemoPolimorfisme1
Bernyanyi dengan irama Jazz
Bernyanyi dengan irama Pop
Bernyanyi dengan irama Dangdut
D:\Program\Latihan JAVA>
```

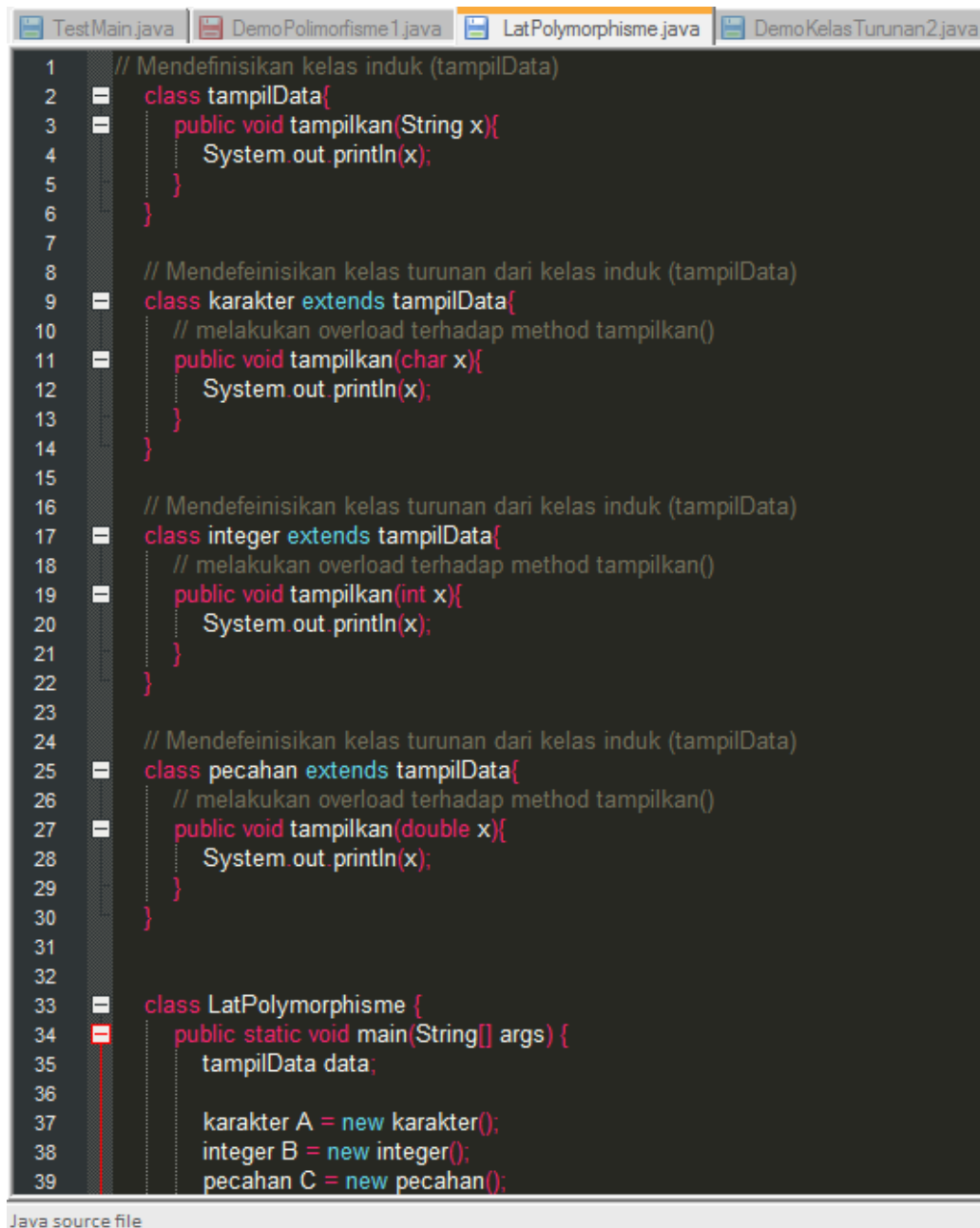
Seperti yang Anda lihat diatas, kita mendeklarasikan variable referensi ke tipe Penyanyi dengan nama p. Sampai disini, kita belum mengetahui apakah p merupakan penyanyi jazz, penyanyi pop, ataukah penyanyi dangdut. Namun, pada saat p mengacu ke objek dari kelas PenyanyiJazz, kemudian kita memanggil method bernyanyi() melalui referensi tersebut, maka method yang akan di eksekusi adalah method bernyanyi() yang terdapat pada kelas PenyanyiJazz. Ini artinya, method yang akan dipanggil oleh p akan tergantung dari abjek yang sedang ditunjuk atau diacu oleh p begitu pula apabila p sedang menunjuk ke objek dari kelas PenyanyiPop maupun PenyanyiDangdut, method yang di panggil pun akan disesuaikan dengan objek yang ada. Hal semacam inilah yang di namakan polimorfisme. Artinya, cara yang di lakukan sama, yaitu p.bernanyani(), akan tetapi implementasinya berbeda-beda sesuai dengan tipe objek yang sedang diacu. Penentuan objek mana yang kan di acu dilakukan pada saan run-time sehingga proses ini di kenal dengan istilah runtime polymorphism atau dynamic method dispath (pengiriman method secara dinamis).

2. Static polimorfisme atau trivial merupakan, function overloading (penggunaan kembali nama fungsi yang sama tapi dgn argumen yang berbeda) / (Penggunaan satu nama untuk beberapa method yang berbeda (beda parameter)). Syarat Overload yaitu nama return type, argument, dan parameter harus sama. Jika tidak sama maka bukan dianggap sebagai overload. Pada overload method, tidak diharuskan menggunakan inheritance, karena dapat digunakan secara public asalkan parameter dan argumennya sesuai.
- Dengan menggunakan kemampuan overload java, kita dapat memiliki lebih dari 1 method dengan nama yang sama di dalam suatu class.
 - Method yang dioverload dan yang mengoverload sebenarnya adalah method-method yang benar-benar berbeda. Mereka hanya memiliki kesamaan nama saja.
 - Constructor dapat dioverload.

Aturan overload :

1. Method yang mengoverload HARUS merubah argument list
 2. Method yang mengoverload BOLEH merubah return type
 3. Method yang mengoverload BOLEH merubah access modifier
 4. Method yang mengoverload BOLEH melempar checked exception yang lebih luas atau baru sama sekali dari method yang dioverload.
 5. Suatu method dapat dioverload di class tersebut atau di subclassnya.
- Method yang mengoverload HARUS merubah argument list. Aturan ini adalah satu-satunya syarat agar dapat terjadi overload
 - Suatu method dapat dioverload di classnya atau dapat pula dioverload di subclassnya.

Contoh program :



```
1 // Mendefinisikan kelas induk (tampilData)
2 class tampilData{
3     public void tampilkan(String x){
4         System.out.println(x);
5     }
6 }
7
8 // Mendefinisikan kelas turunan dari kelas induk (tampilData)
9 class karakter extends tampilData{
10     // melakukan overload terhadap method tampilkan()
11     public void tampilkan(char x){
12         System.out.println(x);
13     }
14 }
15
16 // Mendefinisikan kelas turunan dari kelas induk (tampilData)
17 class integer extends tampilData{
18     // melakukan overload terhadap method tampilkan()
19     public void tampilkan(int x){
20         System.out.println(x);
21     }
22 }
23
24 // Mendefinisikan kelas turunan dari kelas induk (tampilData)
25 class pecahan extends tampilData{
26     // melakukan overload terhadap method tampilkan()
27     public void tampilkan(double x){
28         System.out.println(x);
29     }
30 }
31
32
33 class LatPolymorphisme {
34     public static void main(String[] args) {
35         tampilData data;
36
37         karakter A = new karakter();
38         integer B = new integer();
39         pecahan C = new pecahan();
```

Java source file

```

32
33 = class LatPolymorphisme {
34 =     public static void main(String[] args) {
35         tampilData data;
36
37         karakter A = new karakter();
38         integer B = new integer();
39         pecahan C = new pecahan();
40
41
42         A.tampilkan("A"); // akan memanggil method tampilkan() pada kelas A
43
44         B.tampilkan(1234); // akan memanggil method tampilkan() pada kelas B
45
46         C.tampilkan(11.5); // akan memanggil method tampilkan() pada kelas C
47     }
48

```

Java source file

Maka akan menghasilkan output sebagai berikut :

```

G:\College\PBO\Training>javac LatPolymorphisme.java
G:\College\PBO\Training>java LatPolymorphisme
A
1234
11.5
G:\College\PBO\Training>

```

Berdasarkan program diatas, kita mendeklarasikan variable referensi ke tipe tampilData dengan nama data. Sampai disini, kita belum mengetahui apakah data merupakan karakter, integer atau pecahan. Namun, pada saat data mengacu ke objek dari kelas karakter, kemudian kita memanggil method bernyanyi() melalui referensi tersebut, maka method yang akan di eksekusi adalah method tampilkan() yang terdapat pada kelas karakter. Ini artinya, method yang akan dipanggil oleh data akan tergantung dari abjek yang sedang ditunjuk atau diacu oleh data begitu pula apabila data sedang menunjuk ke objek dari kelas integer maupun pecahan, method yang di panggil pun akan disesuaikan dengan objek yang ada. Hal semacam ini juga disebut sebagai polimorfisme. Artinya, cara yang di lakukan sama, yaitu data.tampilkan(), akan tetapi implementasinya berbeda-beda setiap objek akan memproses tiap-tiap method. Penentuan setiap objek yang diacu dilakukan pada saat compile-time sehingga proses ini di kenal dengan istilah compile polymorphism atau static method dispath (pengiriman method secara statis).

BAB III PENUTUP

III.1 Kesimpulan

JAVA termasuk kedalam bahasa pemrograman yang berbasis objek(OOP) dimana didalamnya terdapat berbagai sifat-sifat yang unik dalam pemrogramannya, seperti : *encapsulation*, *inheritance*, dan *polymorphism*. Polymorphisme merupakan salah satu sifat yang unik dalam bahasa pemrograman yang berbasis objek khususnya java, sifat ini sangat membantu programmer dalam memanipulasi program agar lebih mudah dan efisien. Sifat ini juga sangat terkait dengan kedua sifat lainnya yaitu *encapsulation* dan *inheritance*. Misalnya hubungan dengan *inheritance*, untuk mengoverride sebuah method yang ingin di-polymorphisme harus menggunakan sifat *inheritance*. Polimorfisme terdiri dari 2 jenis, yaitu dengan menggunakan fungsi override dan overload method, yang masing-masing memiliki kemampuan dan syarat yang berbeda. Dengan kedua fungsi tersebut memungkinkan terciptanya berbagai bentuk dari satu objek.

III.2 Saran

Jadi dalam penggunaan/pembuatan program dalam java yang berbasis OOP kita harus memahami konsep-konsep dasar dari OOP tersebut, sehingga dapat menciptakan program dengan baik. Khususnya ketiga sifat OOP yang dimiliki Java yaitu *encapsulation*, *inheritance*, dan *polymorphism*. Pada polimorfisme kita harus memahami kedua overriding dan overload agar dapat membuat program berbasis polimorfisme dengan baik.

DAFTAR PUSTAKA

Situs-situs terkait :

- <http://tentang-info-teknologi.blogspot.com/2013/05/makalah-java-tentang-overload-method.html>
- <http://fikriirizky.blogspot.com/2012/12/polimorfisme-dan-inheritance-pada.html>
- <http://sangwidy.wordpress.com/web-design/oop-2/3-polymorphism/>
- <http://marianus63.blogspot.com/2012/04/pewarisan-inheritance-dan-polimorfisme.html>
- http://en.wikibooks.org/wiki/Java_Programming/Overloading_Methods_and_Constructors
- <http://stackoverflow.com/questions/5099924/is-constructor-overriding-possible>

Buku-buku terkait :

- Java, A Beginner Guide, 2005, Herbert Schildt
- Head First Java 2nd Edition, 2005, Kathy Sierra