**LAPORAN TUGAS PRAKTIKUM PROGRAM BERORIENTASI OBJEK**

“Polymorphism”



Nama : Fauzan Afif Lutfiansah

NIM : 432022611016

PRODI ; Teknik Informatika/3

Dosen : Ustadzah Hanifatus Sa’diah Widihasaniputri S.kom,M.kom

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS DARUSSALAM GONTOR SIMAN**

**1444 H / 2023 M**

Daftar Isi

BAB I : Pendahuluan

1.1 Latar Belakang…………………………………………………………… 1

1.2 Tujuan……………………………………………………………………. 1

BAB II : Teori Dasar

2.1 Polymorphism…………………………………………………………… 2

2.2 Overloading……………………………………………………………… 3

2.3 Overriding……………………………………………………………….. 3

BAB III : Pembahasan

2.1 Deklarasi Polymorphism………………………………………………... 4

2.2 Implementasi Polymorphism dalam Java……………………………….. 5

2.3 Implementasi Overloading dalam java………………………………….. 9

2.4 Iplementasi overriding dalam java……………………………………… 11

BAB IV : Penutup

2.1 Kesimpulan…………………………………………………………….. 12

2.2 Saran……………………………………………………………………. 12

**BAB I**

**Pendahuluan**

1. Latar belakang

Dengan berkembangnya era digital saat ini dan semakin majunya peradaban dunia akan system informasi manusia tak bisa jauh dengan hal hal yang berbau digital/system informasi karna dalam system informasi seperti aplikasi dan lain-lain di buat untuk mempermudah pekerjaan manusia seperti penerapan ilmu Object Oriented Programing yang di rancang untuk pengonsepan model di dunia nyata dan membuat system yang mempermudah pekerjaan manusia seperti pembuatan aplikasi penghitung dan lain-lain.

Objek Oriented Programming (OOP) adalah paradigma pemrograman yang fokus pada Bahasa pemerograman yang mendukung OOP yang berfokuskan pada konsep objek, yang memungkinkan pemodelan dunia nyata dalam bentuk program computer. Salah satu konsep penting OOP adalah “Polymorphism”, yang mengacu pada kemampuan suatu objek untuk berperilaku dengan cara yang berbeda beda tergantung pada konteks yang di inginkan oleh programmer tersebut.

Polymorphism adalah salah satu pilar mundamental penting dalam konsep OOP, Bersama dengan yang telah kita pelajari sebelumnya seperti encapsulation,inheritance,dan abstract. Adapun Polymorphism memungkinkan pengembang untuk menulis kode yang lebih fleksibel dan dapat di ubah dengan mudah, serta memungkinkan pengguna polimorfisme, kita dapat menghindari kode yang berlebihan dan sulit diatur,serta memungkinkan pengembangan aplikasi yang lebih modular.

1. Tujuan

Adapun tujuan dalam laporan pembelajarn Tugas Objek Oriented Programing dengan bertemakan Polymorphism sebagai berikut:

2.1. Memahami konsep Polymorphism.

2.2. Memahami jenis-jenis Polymorphism

2.3. Menggunakan method Overloading dan Overriding.

2.4. Penerapan Polymoriphism dalam aplikasi nyata.

**BAB II**

**Teori Dasar**

2.1 Polymorphism

Polymorphism dujelskan secara bahasa, memiliki makna “banyak bentuk” atau “bermacam-macam”. Dalam beberapa textbook, Polymorphism adalah konsep dimana suatu interface tunggal digunakan pada entity yang berbeda-beda, atau penggunaan suatu symbol tunggal untuk mewakili beberapa jenis tipe entity. Jika dilihat dari sisi teknis pada bahasa pemrograman, Polymorphism adalah menggunakan suatu fungsi atau attribute tertentu dari suatu base class untuk diimplementasikan oleh children class baik secara default maupun dimodifikasi sesuai dengan relevansi yang digunakan oleh masing-masing children class. Dari penjelasan-penjelasan tersebut dapat disimpulkan Polymorphism secara umum adalah penggunaan suatu item baik interface, fungsi, dan lain-lain pada berbagai maca jenis objek maupun entity yang berbeda-beda dengan syarat suatu objek atau entity tersebut memiliki relasi yang menjembatani agar akses ke item tersebut dapat didapatkan.

Adapun Contoh sederhana konsep Polymorphism adalah bagaimana seekor hewan bersuara. Kita tahu bahwa setiap hewan mengeluarkan suara baik secara jelas maupun tidak, namun pada intinya hewan pasti bersuara. Jika kita konsepkan hewan bersuara sebagai fungsi, tentu fungsi ini akan diimplementasikan oleh makhluk hidup yang berada dalam kategori hewan secara berbeda-beda. Misalnya, kucing mengimplementasikan fungsi ini dengan suara “Meong”, anjing mengimplementasikan dengan suara “Gug gug”, ular mengimplmentasikan dengan suara “Hshshhss”, sapi mengimplementasikan dengan suara “Mooo”, dan lain-lain. Perbedaan dalam implmentasi dari suatu fungsi yang sama inilah yang menjadi salah satu contoh penerapan Polymorphism, ada dua jenis polymorphism utama dalam OOP, Polimorfise statis(compile=time-polimorphism) dan polimorfise dinamis(runtime polymorphism).

2.2 Overloading

Di dalam bahasa Java, method overloading adalah membuat beberapa method dengan nama yang sama, tapi dibedakan dari jumlah dan/atau tipe parameter. Seharusnya, kita tidak bisa membuat method dengan nama yang sama. Mirip seperti penamaan variabel, compiler Java akan error jika menemukan 2 atau lebih method dengan nama yang sama. Akan tetapi jika jumlah argument dan/atau tipe data argument berbeda, maka akan dianggap sebagai method yang berbeda pula. Method overloading bisa saja terjadi dalam satu class yang sama, dan bisa juga dari class turunan.

2.3 Overriding

Method overriding adalah sebuah situasi dimana method class turunan menimpa method milik parent class. Ini bisa terjadi jika terdapat nama method yang sama baik di child class dan juga parent class, ada beberapa hal yang perlu di perhatikan dalam overrreding:

* **Nama dan Tanda Tangan Sama:** Metode yang di-override di kelas turunan harus memiliki nama, tipe pengembalian (return type), dan daftar parameter yang sama dengan metode di kelas dasar. Ini penting agar compiler dan runtime bisa memahami bahwa metode tersebut merupakan penggantian (override) dari metode di kelas dasar.
* **Aksesibilitas:** Metode di kelas turunan tidak boleh lebih terbatas dalam hal aksesibilitas (visibility) dibandingkan dengan metode di kelas dasar. Sebagai contoh, jika metode di kelas dasar memiliki aksesibilitas **public**, maka metode yang di-override di kelas turunan juga harus memiliki aksesibilitas **public** atau lebih luas.
* **Pewarisan dan Polimorfisme:** Overriding merupakan salah satu mekanisme pewarisan (inheritance) yang memungkinkan kelas turunan memperluas atau mengubah perilaku metode yang sudah ada di kelas dasar. Hal ini juga berkaitan dengan konsep polimorfisme di mana objek dari kelas turunan dapat memanggil metode yang di-override, dan pemanggilan metode ini akan merujuk pada implementasi yang sesuai dalam kelas turunan.

**BAB III**

**Pembahasan**

2.1 Deklarasi Plymorphism

Polymorphism merupakan termasuk kategori konsep penting dalam object oriented programming (OOP) di Java, setelah abstraction dan inheritance yang telah kita pelajari . Arti Polymorphism berarti banyak bentuk ada beberapa definisi berbeda tentang polymorphism. Sedangkan polymorphism sendiri sulit didefinisikan sejalan dengan contoh program Java sederhana yang diberikan, Polimorfisme sering melibatkan penggunaan banyak metode dengan nama yang sama. Menggunakan metode dengan nama yang sama dapat dilakukan dengan metode overloading dan metode overriding. Peran polimorfisme tidak terbatas pada ini ada hubungan yang erat antara polimorfisme dan pewarisan (derivasi).

Dalam konsep turunan dari Java, ketika objek dari subclass dibangun, objek dari superclass juga dibangun jadi setiap objek (instance) dari subclass juga merupakan instance dari superclass. Jika Anda mendeklarasikan metode yang mengambil parameter tipe dari superclass, maka Anda diizinkan untuk meneruskan objek dari subclass yang mewarisi dari superclass ini sebagai argument, dan berikut merupakan contoh study kasus Polymorphism yang saya pelajari dengan menggunakan objek sepeda :

* SEPEDA ( Super Class)

Yang dimana kita ketahui bahwa sepeda memiliki banyak jenis (class) dari sepeda tersebut memiliki perbedaan masing- masing seperti merk rantai , jumlah roda, merk roda , jenis rem, merk rem)

* Turunan kelas (sepeda)

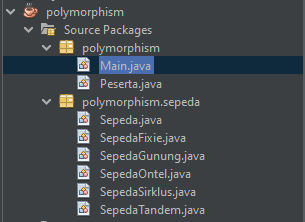
Terdiri dari( Sepeda Gunung, Sepeda Fixie, Sepeda Tandem, Sepeda SIrkus, Sepeda Ontel)

* Overreding

Dalam pembelajaran kali ini kita mengenal bahwa setiap sepeda memiliki Khass masing2 yang berbeda yang akan kita pangil dalam kelas-kelas tersebut.

2.2 Implementasi Polymorphism dalam Java

Di dalam polymorphism ini terdapat dua package yang di mana package yang pertama berisi main.java sebagai oproses running hasil akhir class yang telah kita buat atau bisa di sebut pemanggilan dan peserta.java berisikan study kasus peserta dengan cara / syarat mengikuti menggunakan pemanggilan class yang ada.



1. Main Java.

Fungsi main adalah fungsi khusus dalam bahasa pemrograman Java yang merupakan titik masuk utama ke program Java. Fungsi ini harus ada di setiap program Java karena bertanggung jawab untuk menjalankan program tersebut. Fungsi main memiliki sintaks yang sama untuk semua program Java dan harus didefinisikan sebagai berikut:

package polymorphism;

import polymorphism.sepeda.SepedaFixie;

import polymorphism.sepeda.SepedaGunung;

public class Main {

public static void main(String[] args) {

Peserta[] daftarpeserta = new Peserta[10];

Peserta peserta1 = new Peserta("Fauzan", new SepedaGunung("Polygon","Rantaigon",2,"Roda","tuas","Polu"));

Peserta peserta2 = new Peserta("Rudi", new SepedaFixie("wimcycle","shadow",2,"shadowban","tuas","wicicler"));

daftarpeserta[0]= peserta1;

daftarpeserta[1]= peserta2;

for(int i =0; i< daftarpeserta.length; i++) {

if (daftarpeserta[i] != null) {

System.out.println(daftarpeserta[i].getNamapeserta());

daftarpeserta[i].getSepedapeserta().mengerem();

}

}

}

}

1. Peserta

package polymorphism;

import polymorphism.sepeda.Sepeda;

/\*\*

\*

\* @author Apipz

\*/

public class Peserta {

private String namapeserta;

private Sepeda sepedapeserta;

public Peserta(String namapeserta, Sepeda sepedapeserta) {

this.namapeserta = namapeserta;

this.sepedapeserta = sepedapeserta;

}

public String getNamapeserta() {

return namapeserta;

}

public void setNamapeserta(String namapeserta) {

this.namapeserta = namapeserta;

}

public Sepeda getSepedapeserta() {

return sepedapeserta;

}

public void setSepedapeserta(Sepeda sepedapeserta) {

this.sepedapeserta = sepedapeserta;

}

}

1. Class Sepeda (Super Class)

package polymorphism.sepeda;

public class Sepeda {

public String merksepeda;

public String merkrantai;

public int jumlahroda;

public String merkroda;

public String jenisrem;

public String merkrem;

public Sepeda(String merksepeda, String merkrantai,int jumlahroda,String merkroda,String jenisrem,String merkrem){

this.merkrem = merksepeda;

this.merkrantai = merkrantai;

this.jumlahroda = jumlahroda;

this.merkroda = merkroda;

this.jenisrem = jenisrem;

this.merkrem = merkrem;

}

public void berjalan(){

System.out.println(merksepeda+"Sepeda ini berjalan");

}

public void mengerem(){

System.out.println("tuas rem tangan di tarik"+ merksepeda+"berhenti");

}

public void berbelok(){

System.out.println(merksepeda+"ini berbelok");

}

}

1. Sepeda fixie (Class Turunan/inheritance)

package polymorphism.sepeda;

/\*\*

\*

\* @author Apipz

\*/

public class SepedaFixie extends Sepeda{

public SepedaFixie(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

@Override

public void mengerem(){

System.out.println("Pedal di putar beralawan arah,"+merksepeda+" berhenti");

}

}

1. Sepeda Gunung(Class Turunan)

package polymorphism.sepeda;

/\*\*

\*

\* @author Apipz

\*/

public class SepedaGunung extends Sepeda {

public SepedaGunung(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

}

1. Sepeda Ontel (Class Turunan)

package polymorphism.sepeda;

/\*\*

\*

\* @author Apipz

\*/

public class SepedaOntel extends Sepeda{

public SepedaOntel(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

}

1. Sepeda Sirklus(Class Turunan)

package polymorphism.sepeda;

/\*\*

\*

\* @author Apipz

\*/

public class SepedaSirklus extends Sepeda{

public SepedaSirklus(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

@Override

public void mengerem(){

System.out.println("Pedal di putar beralawan arah,"+merksepeda+" berjalan mundur");

}

}

1. Sepeda Tandem (Class Turunan)

package polymorphism.sepeda;

/\*\*

\*

\* @author Apipz

\*/

public class SepedaTandem extends Sepeda{

public SepedaTandem(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

}

2.3 Implementasi Overloading

Dalam bahasa Java, kelebihan metode melibatkan pembuatan beberapa metode dengan nama yang sama, tetapi jumlah dan/atau jenis parameter berbeda.Katakanlah, kita tidak dapat membuat metode dengan nama yang sama. Mirip dengan penamaan variabel, kompiler Java akan melontarkan kesalahan jika menemukan 2 atau lebih metode dengan nama yang sama. Namun, jika jumlah argumen dan/atau tipe data argumen berbeda, maka akan diperlakukan sebagai metode yang berbeda. Kelebihan metode dapat terjadi dalam kelas yang sama dan juga dapat berasal dari kelas turunan seperti mana dari kode yang saya buat di bawah ini :

package polymorphism.sepeda;

/\*\*

\*

\* @author Apipz

\*/

public class Sepeda {

public String merksepeda;

public String merkrantai;

public int jumlahroda;

public String merkroda;

public String jenisrem;

public String merkrem;

public Sepeda(String merksepeda, String merkrantai,int jumlahroda,String merkroda,String jenisrem,String merkrem){

this.merkrem = merksepeda;

this.merkrantai = merkrantai;

this.jumlahroda = jumlahroda;

this.merkroda = merkroda;

this.jenisrem = jenisrem;

this.merkrem = merkrem;

}

public void berjalan(){

System.out.println(merksepeda+"Sepeda ini berjalan");

}

public void mengerem(){

System.out.println("tuas rem tangan di tarik"+ merksepeda+"berhenti");

}

public void berbelok(){

System.out.println(merksepeda+"ini berbelok");

}

}

2.4 Iplementasi Overreading

Method overriding adalah sebuah situasi dimana method class turunan menimpa method milik parent class. Ini bisa terjadi jika terdapat nama method yang sama baik di child class dan juga parent class, Seperti yang mana terdapat di kode yang saya buat di bawah ini:

public class SepedaFixie extends Sepeda{

public SepedaFixie(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

@Override

public void mengerem(){

System.out.println("Pedal di putar beralawan arah,"+merksepeda+" berhenti");

}

}

public class SepedaGunung extends Sepeda {

public SepedaGunung(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

}

public class SepedaOntel extends Sepeda{

public SepedaOntel(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

}

public class SepedaSirklus extends Sepeda{

public SepedaSirklus(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

@Override

public void mengerem(){

System.out.println("Pedal di putar beralawan arah,"+merksepeda+" berjalan mundur");

}

}

public class SepedaTandem extends Sepeda{

public SepedaTandem(String merksepeda, String merkrantai, int jumlahroda, String merkroda, String jenisrem, String merkrem) {

super(merksepeda, merkrantai, jumlahroda, merkroda, jenisrem, merkrem);

}

}

**BAB IV**

**Penutup**

2.1 Kesimpulan

Dalam pembelajaran materi polymorphism dalam bahasa pemrograman Java, telah dijelaskan bagaimana konsep ini dapat diaplikasikan melalui contoh objek "sepeda" sebagai kelas utama (super class) dan turunannya (sub class). Polymorphism memungkinkan kita untuk mengakses objek-objek dengan tipe yang berbeda melalui antarmuka yang seragam, sehingga mempermudah pengelolaan dan penggunaan kode. Dalam konteks sepeda, telah diperlihatkan bahwa dengan menggunakan konsep inheritance, kita dapat membuat kelas utama "sepeda" yang memiliki sifat-sifat dan metode umum. Selanjutnya, kita bisa membuat kelas-kelas turunan seperti "sepeda gunung" dan "sepeda balap" yang mewarisi sifat-sifat dari kelas utama, namun juga memiliki perbedaan karakteristik dan implementasi tertentu.

Polymorphism diterapkan melalui overriding, yang memungkinkan kelas turunan menggantikan implementasi metode di kelas utama sesuai dengan kebutuhan. Ketika kita menggunakan referensi kelas utama untuk mengakses objek dari kelas turunan, metode yang akan dieksekusi akan sesuai dengan tipe objek yang sebenarnya, sehingga menghasilkan output yang tepat. Melalui studi kasus ini, telah terbukti bahwa polymorphism merupakan konsep penting dalam pemrograman berorientasi objek yang memungkinkan kita untuk membuat kode yang lebih fleksibel, modular, dan mudah di-maintain. Dengan memahami konsep ini, kita dapat memanfaatkannya untuk membuat hierarki kelas yang lebih terstruktur dan mengoptimalkan reusabilitas kode dalam pengembangan aplikasi berbasis Java.

2.2 Saran

Dan pastinya saya selaku penulis mengakui masih banyak kesalahan dan kekurangan dalam segi penulisan makalah laporan tugas OOP(Object Oriented Programing) Polymorphism kali ini maka dari itu saya selaku penulis berharap keritik dan Saranya yang bersifat membangun saya selaku penulis dan mungkin pembaca-pembacanya karna manusia tak tuput dari salah dan lupa, perlu kita ketahui keilmuan itu sangatlah luas sekali seperti mana contohnya dalam laporan tugas saya kali ini yang membahas tentang Polymorphim dan terdapat beberapa percabangan di dalamnya, maka dari itu kami siap menerima kritik dan saran yang bersifat membangun dalam proses pambelajaran kami , cukup sekian dan trimakasih.

*Wassalamualaikum warahmatullahi wabarakatuhu*