**Modul 11 PBO Teori\_Ridhatul Nafsi**

**Pewarisan, Polimorfisme, dan Interface**

**Pewaris**

Student juga sebagai Person,hanya meng-extend class Person, sehingga kita dapat mewariskan semua properti dan method dari setiap class Person yang ada.

public class Student extends Person

{

public Student(){

System.out.println(“Inside Student:Constructor”);

//beberapa kode di sini

}

// beberapa kode di sini

}

Ketika object Student di-instantiate, default constructor dari superclass secara mutlak meminta untuk melakukan inisialisasi yang seharusnya. Setelah itu, pernyataan di dalam subclass dieksekusi.

public static void main( String[] args ){

Student anna = new Student();

}

Dalam kode ini, kita membuat sebuah object dari class Student. Keluaran dari program adalah

Inside Person:Constructor

Inside Student:Constructor

**Polimorfisme**

Kemampuan dari referensi untuk mengubah sifat menurut object apa yang dijadikan acuan dinamakan polimorfisme. Polimorfisme menyediakan multiobject dari subclasses yang berbeda untuk diperlakukan sebagai object dari superclass tunggal, secara otomatis menunjuk method yang tepat untuk menggunakannya ke particular object berdasar subclass yang termasuk di dalamnya.

public static main( String[] args )

{

Student studentObject = new Student();

Employee employeeObject = new Employee();

printInformation( studentObject );

printInformation( employeeObject );

}

public static printInformation( Person p ){

. . . .

}

**Interface**

Interface adalah jenis khusus dari blok yang hanya berisi method signature(atau constant ). Interface mendefinisikan sebuah(signature) dari sebuah kumpulan method tanpa tubuh. penggunaan interface digunakan kata kunci implements. Contohnya,

/\*\*

\* Class ini mendefinisikan segmen garis

\*/

public class Line implements Relation

{

private double x1;

private double x2;

private double y1;

private double y2;

public Line(double x1, double x2, double y1, double y2){

this.x1 = x1;

this.x2 = x2;

this.y1 = y1;

this.y2 = y2;

}

public double getLength(){

double length = Math.sqrt((x2-x1)\*(x2-x1) +

(y2-y1)\* (y2-y1));

return length;

}

public boolean isGreater( Object a, Object b){

double aLen = ((Line)a).getLength();

double bLen = ((Line)b).getLength();

return (aLen > bLen);

}

public boolean isLess( Object a, Object b){

double aLen = ((Line)a).getLength();

double bLen = ((Line)b).getLength();

return (aLen < bLen);

}

public boolean isEqual( Object a, Object b){

double aLen = ((Line)a).getLength();

double bLen = ((Line)b).getLength();

return (aLen == bLen);

}

}

Ketika class Anda mencoba mengimplementasikan sebuah interface, selalu pastikan bahwa Anda mengimplementasikan semua method dari interface, jika tidak, Anda akan menemukan kesalahan ini,

Line.java:4: Line is not abstract and does not override abstract

method isGreater(java.lang.Object,java.lang.Object) in

Relation

public class Line implements Relation

^

1 error

Gunakan interface untuk mendefinisikan method standar yang sama dalam class�class berbeda yang memungkinkan. Sekali Anda telah membuat kumpulan definisi method standar, Anda dapat menulis method tunggal untuk memanipulasi semua class�class yang mengimplementasikan interface.

*Thank you !*

*Pembuat*

Ridhatul Nafsi