Polimorfisme (bahasa inggris *polymorphism*) adalah sebuah prinsip dalam biologi di mana oraganisme atau spesias dapat memiliki banyak bentuk atau tahapan *(stages)*.

Prinsip ini juga diadopsi pada pemrograman berorientasikan objek.

Sehingga kita dapat definisikan sebagai berikut:

Polimorfisme dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak **"bentuk"** method yang berbeda-beda meskipun namanya sama.

"Bentuk" di sini dapat kita artikan: isinya berbeda, parameternya berbeda, dan tipe datanya berbeda.

Interface yang dibahas disini bukan interface pada UI/UX dan tidak ada kaitannya sama sekali. meski sama-sama memiliki istilah interface tapi interface yang dibahas disini adalah keyword dalam pemrograman java. Sedangkan User Interface/ User Experience (UI/UX) berhubungan tentang desain estetika dan bagaimana pengguna menggunakannya.

Interface adalah class yang tidak memiliki tubuh pada method-methodnya. Method interface tersebut harus diimplementasikan dalam kelas turunannya tidak boleh tidak. Di dalam interface, deklarasi variable memiliki atribut final sehingga bersifat absolut. Keyword final inilah yang menjadi keunikan sendiri bagi interface bahwa ouput dari bagian interface berupa final yang tidak diganti pada saat implementasi kecuali di override.

Keuntungan membuat interface sendiri adalah menutupi kekurangan pada java yang hanya memperbolehkan satu kelas saja yang berhak mendapatkan warisan kelas induk (extends). Sehingga satu kelas hanya dapat menggunakan satu kelas induk, sebaliknya pada interface dapat di implementasi lebih dari satu. Ciri-ciri interface adalah interface tidak dapat di instansiasi, tidak terdapat konstruktor dan semua method interface adalah abstrak.

Program polimorfisme.

```
abstract class Bentuk {

protected int panjang;

protected int lebar;

public String getBentuk() {

return "Bentuk Dasar";

}

public abstract int hitungLuas();

}
```

```
class BujurSangkar extends Bentuk {
public BujurSangkar(int panjang1, int lebar1) {
this.panjang = panjang1;
this.lebar  = lebar1;
}
public String getBentuk() {
return “Bentuk Bujur Sangkar”;
}
public int hitungLuas() {
return panjang*lebar;
}
}
class SegiTiga extends Bentuk {
public SegiTiga(int panjang2, int lebar2) {
this.panjang = panjang2;
this.lebar  = lebar2;
}
//public String getBentuk() {
//return “Bentuk Segi Tiga”;
//return “”;
//}
public int hitungLuas() {
return this.panjang*this.lebar/2;
}
}
class Polimorfisme {
```

```
public static void cetakLuasBentuk(Bentuk btk) {
System.out.println(btk.getBentuk() + ” dengan luas ” +
btk.hitungLuas());
}
public static void main(String[] args) {
BujurSangkar bs  = new BujurSangkar(10,20);
BujurSangkar bs1 = new BujurSangkar(10,20);
SegiTiga st  = new SegiTiga(5,10);
SegiTiga st1 = new SegiTiga(50,100);
cetakLuasBentuk(bs);
cetakLuasBentuk(bs1);
cetakLuasBentuk(st);
cetakLuasBentuk(st1);
}
}
```

Programing interface.

```
interface MyInterface {
void iMethod();
}
class MyClass1 implements MyInterface {
public void iMethod() {
System.out.println(“Interface method.”);
}
void myMethod() {
System.out.println(“Another method.”);
```

```
}
}
class MyClass2 implements MyInterface {
public void iMethod() {
System.out.println(“Another implementation.”);
}
}
class InterfaceDemo {
public static void main(String args[]) {
MyClass1 mc1 = new MyClass1();
MyClass2 mc2 = new MyClass2();
mc1.iMethod();
mc1.myMethod();
mc2.iMethod();
}
}
```