**Bab 7**

**Polimorfisme**

**POKOK BAHASAN**

• Konsep dasar polimorfisme

• Virtual Method Invocation

• Polymorphic arguments

• Pernyataan instanceof

• Casting object

**TUJUAN BELAJAR**

Dengan praktikum ini mahasiswa diharapkan dapat:

• Memahami dan menerapkan konsep polimorfisme dalam pemrograman

• Memahami proses terjadinya Virtual Method Invocation

• Memahami dan menerapkan polymorphic arguments dalam pemrograman

• Memahami penggunaan instanceof dan cara melakukan casting object

**Dasar Teori**

Polymorphism (polimorfisme) adalah kemampuan untuk mempunyai beberapa bentuk class yang berbeda. Polimorfisme ini terjadi pada saat suatu obyek bertipe parent class, akan tetapi pemanggilan constructornya melalui subclass. Misalnya deklarasi pernyataan berikut ini:

Employee employee=new Manager();

dimana Manager() adalah kontruktor pada class Manager yang merupakan subclass dari class Employee. Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan overriding. Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass, dimana yang seharusnya dipanggil adalah overridden method. Berikut contoh terjadinya VMI:

class Parent {

int x = 5;

public void Info() {

System.out.println(“Ini class Parent”);

}

}

class Child extends Parent {

int x = 10;

public void Info() {

System.out.println(“Ini class Child”);

}

}

public class Tes {

public static void main(String args[]) {

Parent tes=new Child();

System.out.println(“Nilai x = “ + tes.x);

tes.Info();

}

}

Hasil dari running program diatas adalah sebagai berikut:

Nilai x = 5

Ini class Child

Polymorphic arguments adalah tipe suatu parameter yang menerima suatu nilai yang bertipe subclass-nya. Berikut contoh dari polymorphics arguments:

class Pegawai {

…

}

class Manajer extends Pegawai {

…

}

public class Tes {

public static void Proses(Pegawai peg) {

…

}

public static void main(String args[]) { Manajer man = new Manajer(); Proses(man);

}

}

Pernyataan instanceof sangat berguna untuk mengetahui tipe asal dari suatu polymorphic arguments. Untuk lebih jelasnya, misalnya dari contoh program sebelumnya, kita sedikit membuat modifikasi pada class Tes dan ditambah sebuah class baru Kurir, seperti yang tampak dibawah ini:

…

class Kurir extends Pegawai {

…

}

public class Tes {

public static void Proses(Pegawai peg) {

if (peg instanceof Manajer) {

…lakukan tugas-tugas manajer…

} else if (peg instanceof Kurir) {

…lakukan tugas-tugas kurir…

} else {

…lakukan tugas-tugas lainnya…

}

}

public static void main(String args[]) { Manajer man = new Manajer();

Kurir kur = new Kurir(); Proses(man);

Proses(kur);

}

}

Seringkali pemakaian instanceof diikuti dengan casting object dari tipe parameter ke tipe asal. Misalkan saja program kita sebelumnya. Pada saat kita sudah melakukan instanceof dari tipe Manajer, kita dapat melakukan casting object ke tipe asalnya, yaitu Manajer. Caranya adalah seperti berikut:

…

if (peg instanceof Manajer) {

Manajer man = (Manajer) peg;

…lakukan tugas-tugas manajer…

}

…

**Percobaan**

**Memahami proses terjadinya Virtual Method Invocation**

Tulislah listing program berikut ini dan amati yang terjadi pada saat terjadinya

Virtual Method Invocation.

class Parent {

int x = 5;

public void Info() {

System.out.println(“Ini class Parent”);

}

}

class Child extends Parent {

int x = 10;

public void Info() {

System.out.println(“Ini class Child”);

}

}

public class Tes {

public static void main(String args[]) { Parent tes=new Child(); System.out.println(“Nilai x = “ + tes.x); tes.Info();

}

}

Ketika program tersebut dijalankan, akan tampak hasil seperti dibawah ini :

Nilai x = 5

Ini class Child

**Latihan**

**Latihan 1**

**Mengimplementasikan UML class diagram dalam program**

Suatu program terdiri dari class Pegawai sebagai parent class, dan class Manajer dan class Kurir sebagai subclass. Buatlah suatu program yang menerapkan konsep polymorphic argument sebagaimana yang telah disinggung dalam pembahasan sebelumnya.

**Latihan 2**

**Mengimplementasikan Polimorfisme**

Terdapat Class pengetesan sebuah program.

**static void cetakLuasBentuk(Bentuk btk) {**

**System.out.println(btk.getBentuk() + " dengan luas " + btk.hitungLuas());**

**}**

**public static void main(String[] args) {**

**BujurSangkar bs = new BujurSangkar(5,20);**

**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);**

**}**

Buatlah Class parent bernama Bentuk yang memiliki Class Child BujurSangkar dan SegiTiga. Output dari program pengetesan adalah sebagai berikut :

|  |
| --- |
| **Bentuk Bujur Sangkar dengan luas 100**  **Bentuk Bujur Sangkar dengan luas 200**  **Bentuk Segi Tiga dengan luas 25**  **Bentuk Segi Tiga dengan luas 250** |

**Tugas**

**Mengimplementasikan UML class diagram dalam program**

|  |  |
| --- | --- |
| Pegawai  # nama : String  # gaji : int  + Pegawai(nama : String, gaji : int)  + infoGaji() : int | |
| Manajer  - tunjangan : int  + Manajer(nama : String, gaji : int, tunjangan : int)  + infoGaji() : int  + infoTunjangan() : int | Programmer  - bonus : int  + Programmer(nama : String,  gaji : int, bonus : int)  + infoGaji() : int  + infoBonus() : int |

Transformasikan class diagram diatas ke dalam bentuk program?. Tulislah listing program berikut ini sebagai pengetesan.

public class Bayaran {

public int hitungBayaran(Pegawai peg) {

int uang=peg.infoGaji();

if (peg instanceof Manajer)

uang+=((Manajer) peg).infoTunjangan();

else if (peg instanceof Programmer)

uang+=((Programmer) peg).infoBonus();

return uang;

}

public static void main(String args[]) {

Manajer man=new Manajer("Agus",800,50);

Programmer prog=new Programmer("Budi",600,30);

Bayaran hr=new Bayaran();

System.out.println("Bayaran untuk Manajer : " + hr.hitungBayaran(man));

System.out.println("Bayaran untuk Programmer : " + hr.hitungBayaran(prog));

}

}

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

Bayaran untuk Manajer : 850

Bayaran untuk Programmer : 630