PRAKTIKUM 13

POLIMORFISME

A. TUJUAN PEMBELAJARAN

1. Memahami dan menerapkan konsep polimorfisme dalam pemrograman
2. Memahami proses terjadinya Virtual Method Invocation
3. Memahami dan menerapkan polymorphic arguments dalam pemrograman
4. Memahami penggunaan instanceof dan cara melakukan casting object

B. 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…
}
…
```

C. TUGAS PENDAHULUAN

1. Apakah yang dimaksud dengan polimorfisme ?
2. Jelaskan proses terjadinya Virtual Method Invocation !
3. Apakah yang dimaksud dengan polymorphic arguments ?
4. Apakah kegunaan kata kunci instanceof ?

D. 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();
         }
}
```

Apakah output program diatas? Jelaskan !

E. LATIHAN

Latihan 1. Apa yang terjadi bila kode dibawah ini dikompile dan dijalankan jika sebelumnya Base.java belum dikompile? Jelaskan !

```
// Filename; SuperclassX.java
package packageX;

public class SuperclassX{
      int superclassVarX;

      protected void superclassMethodX(){
      }
}
```

```
// Filename SubclassY.java
package packageX.packageY;

public class SubclassY extends SuperclassX{
      SuperclassX objX = new SubclassY();
      SubclassY objY = new SubclassY();

      void subclassMethodY(){
             objY.superclassMethodX();
             int i;
             i = objY.superclassVarX;
      }
}
```

Latihan 2. Apa yang tampil di layar jika kode dibawah ini dijalankan? Jelaskan !

```
class Base {
       int i = 99;

      Base(){
            amethod();
       }

      public void amethod(){
            System.out.println("Base.amethod()");
      }
}
```

```
public class Derived extends Base{
      int i = -1;

      public static void main(String argv[]){
            Base b = new Derived();
            System.out.println(b.i);
            b.amethod();
      }

      public void amethod(){
            System.out.println("Derived.amethod()");
      }
}
```

Latihan 3. Apa yang tampil di layar jika kode dibawah ini dijalankan? Jelaskan !

```
class Parent{
      private void method1(){
            System.out.println("Parent's method1()");
      }

      public void method2(){
            System.out.println("Parent's method2()");
            method1();
      }
}
```

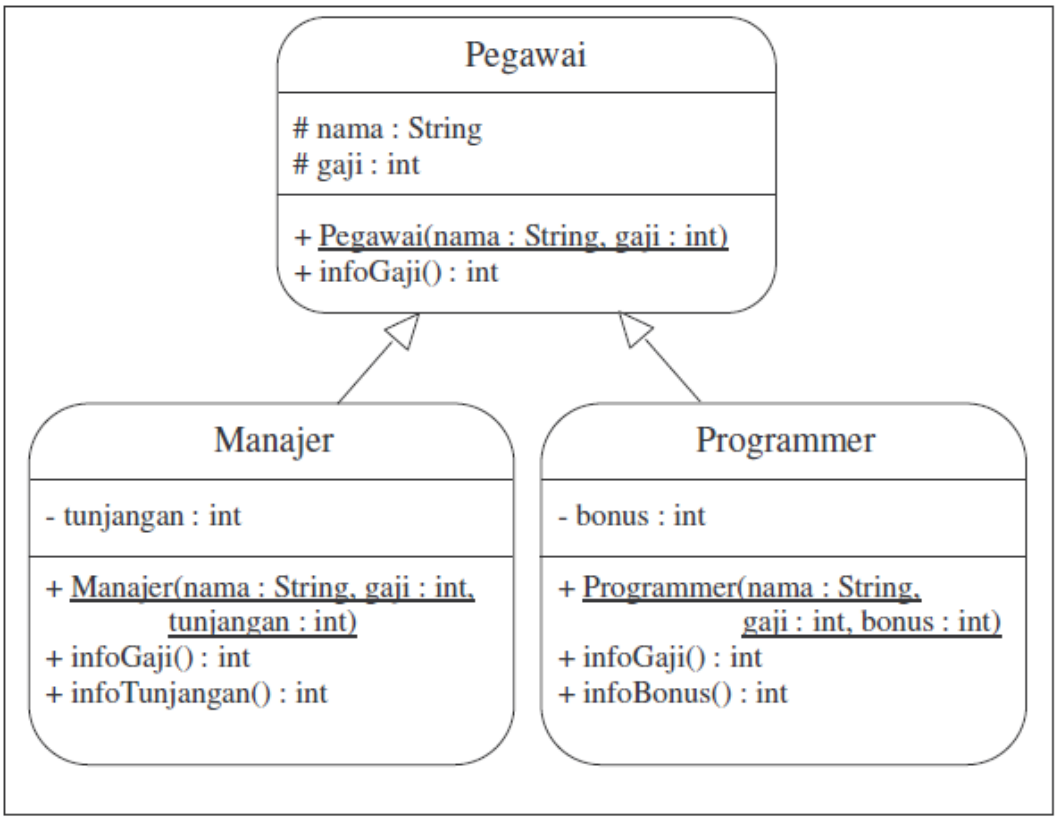
```
class Child extends Parent{
      public void method1()
            System.out.println("Child's method1()");

      }

      public static void main(String args[]){
            Parent p = new Child();
            p.method2();
      }
}
```

Latihan 4. Mengimplementasikan UML class diagram dalam program

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

F. TUGAS

Mengimplementasikan UML class Diagram dalam program



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 di bawah 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
```

G. LAPORAN RESMI

Kumpulkan hasil latihan dan tugas di atas. Tambahkan analisa dalam laporan resmi.