PRAKTIKUM 14

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);
         }
}
```

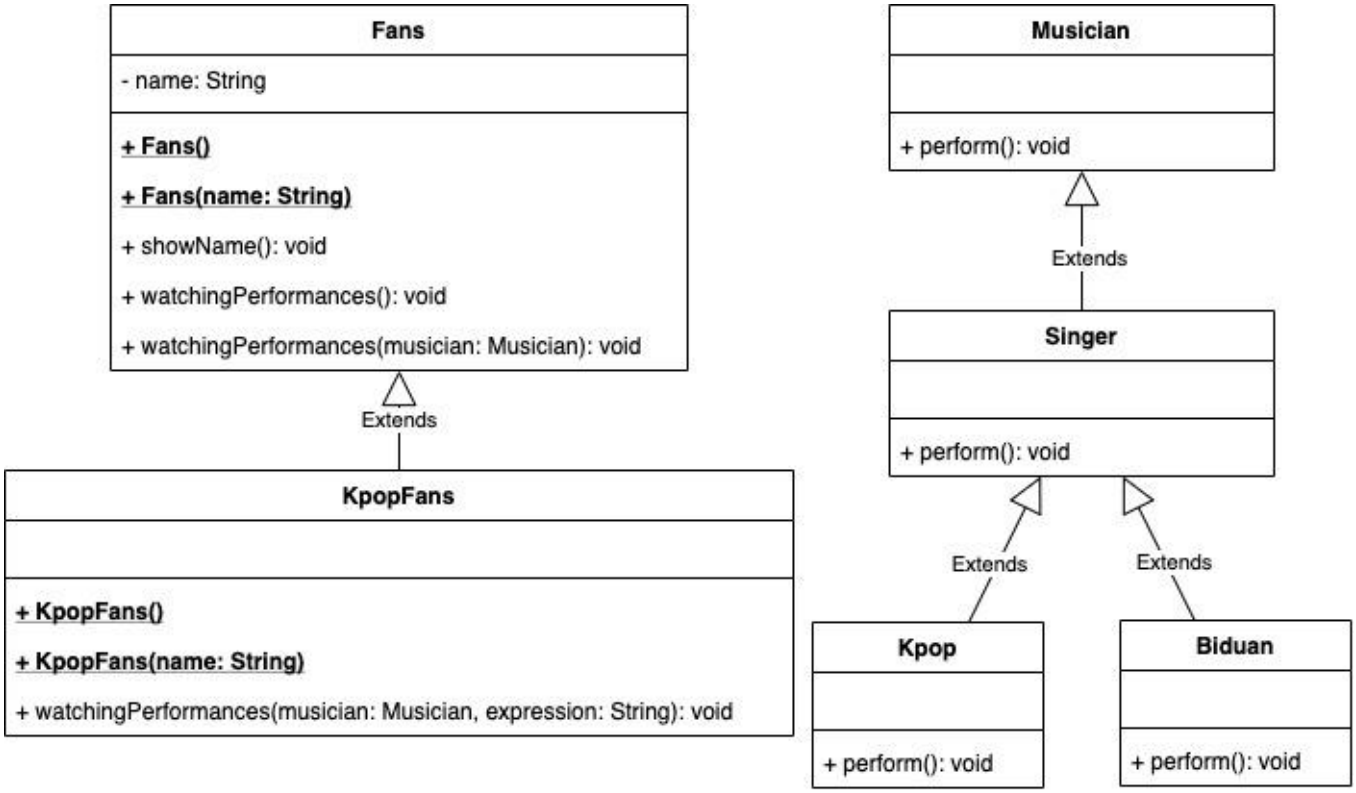
Seringkali pemakaian instanceof diikuti dengan casting object dari tipe

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…
}
…
```

D. PERCOBAAN

Impelemntasikan class diagram berikut dalam kode program. Hasil harus sesuai output yang diminta, dan class Test tidak boleh diubah/dimodifikasi!



saat method perform dijalankan

Tercetak kalimat berikut:
Performer: Beraksi di atas panggung
Singer: Beraksi diatas panggung, bernyanyi dengan merdu
Kpop: Beraksi diatas panggung, bernyanyi dengan merdu, dan ngedance
Biduan: Beraksi diatas panggung, bernyanyi dengan merdu, dengan cengkok melayu

```
public class Test {
    public static void main(String args[]){
        Fans fans1 = new Fans();
        Fans fans2 = new Fans("Mona");
        Fans fans3 = new KpopFans("Tomi");
        KpopFans fans4 = new KpopFans("Febi");

        fans1.watchingPerformance();
        fans2.watchingPerformance(new Musician());
        fans2.watchingPerformance(new Singer());
        fans3.watchingPerformance(new Biduan());
        fans4.watchingPerformance(new Kpop(), "teriak histeris");    }
}
```

Output:

```
noname: melihat idolanya dari youtube
Mona: melihat idolanya Beraksi di atas panggung
Mona: melihat idolanya Beraksi di atas panggung, bernyanyi dengan merdu
Tomi: melihat idolanya Beraksi di atas panggung, bernyanyi dengan merdu, dengan cengkok melayu
Febi: teriak histeris melihat idolanya Beraksi di atas panggung, bernyanyi dengan merdu, dan ngedance
```

E. LATIHAN

1. Tunjukkan contoh Overloading dalam percobaan diatas!
2. Tunjukkan contoh Overriding method dan overriden method pada percobaan diatas!
3. Tunjukkan contoh Overloding yang terjadi dalam satu class, pada percobaan diatas!
4. Tunjukkan contoh Overloading yang terjadi antara superclass dan subclass pada percobaan diatas!
5. Tunjukkan contoh Virtual Method Invocation yang terjadi pada percobaan diatas!
6. Tunjukkan contoh Polimorfism pada percobaan diatas!
7. Tunjukkan contoh inheritance pada percobaan diatas!

F. LAPORAN RESMI

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