LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

KE - 14 (Polymorphism 2)

Nama : Azzahra Rachmadia Mumtaz

NRP : 3124600095

Kelas: D4 IT D

Dosen Pengajar : Grezio Arifiyan Primajaya S.Kom, M.Kom

PROGRAM STUDI D4 TEKNIK INFORMATIKA POLITEKNIK ELEKTRONIKA NEGERI SURABAYA (PENS) TAHUN 2025

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. Percobaan

1. Percobaan 1 :

a. Listing Program

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

class Fans {
   private String name;
   public Fans(){
      this("noname");

   }

   public Fans(String name){
      this.name = name;

   }

   public void showName(){
      System.out.print(name);

   }

   public void watchingPerformance(){
      this.showName();
      System.out.println(": melihat idolanya dari youtube ");
   }

   public void watchingPerformance(Musician musician){
      this.showName();
      System.out.print(": melihat idolanya");
```

```
        if (musician instanceof Singer){
            ((Singer) musician).perform();
        } else if (musician instanceof Kpop){
            ((Kpop) musician).perform();
        } else if (musician instanceof Biduan){
            ((Biduan) musician).perform();
        } else {
            musician.perform();
        }
        System.out.println();
    }
}

class KpopFans extends Fans{
    public KpopFans(){
        super();
    }

    public KpopFans(String name){
        super(name);
    }

    public void watchingPerformance(Musician musician, String expression){
        super.showName();
        System.out.print(": " + expression + " melihat idolanya ");
        musician.perform();
        System.out.println();
    }
}

class Musician {
    public void perform(){
        System.out.print(" Beraksi diatas panggung");
    }
}

class Singer extends Musician{
    public void perform(){
        super.perform();
        System.out.print(", bernyanyi dengan merdu");
    }
}

class Biduan extends Singer{
    public void perform(){
        super.perform();
        System.out.print(", dengan cengkok melayu");
    }
```

```
}

class Kpop extends Singer{
   public void perform(){
      super.perform();
      System.out.print(", dan ngedance");
   }
}
```

b. Output

```
PS D:\azza\pbo> java .\Test.java
noname: melihat idolanya dari youtube
Mona: melihat idolanya Beraksi diatas panggung
Mona: melihat idolanya Beraksi diatas panggung, bernyanyi dengan merdu
Tomi: melihat idolanya Beraksi diatas panggung, bernyanyi dengan merdu, dengan cengkok melayu
Febi: teriak histeris melihat idolanya  Beraksi diatas panggung, bernyanyi dengan merdu, dan ngedance
```

C. Latihan

1. Tunjukkan contoh Overloading dalam percobaan diatas!

a. Class Fans

Overloading Constructor

```
public Fans(String name){
      this.name = name;

   }

   public void showName(){
      System.out.print(name);

   }
```

1. Overloading Method

```
public void watchingPerformance(){
      this.showName();
      System.out.println(": melihat idolanya dari youtube ");
   }

   public void watchingPerformance(Musician musician){
      this.showName();
      System.out.print(": melihat idolanya ");
      if (musician instanceof Singer){
         ((Singer) musician).perform();
      } else if (musician instanceof Kpop){
         ((Kpop) musician).perform();
      } else if (musician instanceof Biduan){
         ((Biduan) musician).perform();
      } else {
         musician.perform();
      }
      System.out.println();
```

```
    }
}
```

b. **Class KpopFans**

Overloading Constructor

```
public KpopFans(){
    super();
}

public KpopFans(String name){
    super(name);
}
```

2. **Tunjukkan contoh Overriding method dan overriden method pada percobaan diatas!**
   a. **Overridden di kelas Musician**

```
public void perform(){
    System.out.print(" Beraksi diatas panggung");
}
```

   b. **Overriding di kelas Singer**

```
public void perform(){
    super.perform();
    System.out.print(", bernyanyi dengan merdu");
}
```

3. **Tunjukkan contoh Overloading yang terjadi dalam satu class, pada percobaan diatas!**
   a. **Class Fans**

Tanpa parameter

```
public void watchingPerformance(){
    System.out.print(name + ": melihat idolanya di atas panggung ");
}
```

Dengan parameter Musician

```
public void watchingPerformance(Musician musician){
    …
}
```

   b. **Class KpopFans**

Tanpa Parameter

```
public KpopFans(){
    super();
}
```

Dengan parameter String

```
public KpopFans(String name){
    super(name);
}
```

4. **Tunjukkan contoh Overloading yang terjadi antara superclass dan subclass pada percobaan diatas!**

a. **Kelas Fans (Superclass)**

```
public void watchingPerformance(Musician musician){
…
}
```

b. **Kelas KpopFans (Subclass)**

```
public void watchingPerformance(Musician musician, String expression){
…
}
```

5. **Tunjukkan contoh Virtual Method Invocation yang terjadi pada percobaan diatas!**

```
if (musician instanceof Singer){
        ((Singer) musician).perform();
    } else if (musician instanceof Kpop){
        ((Kpop) musician).perform();
    } else if (musician instanceof Biduan){
        ((Biduan) musician).perform();
    } else {
        musician.perform();
    }
```

6. **Tunjukkan contoh Polimorfism pada percobaan diatas!**

```
public void watchingPerformance(Musician musician){
    this.showName();
    System.out.print(": melihat idolanya ");
    if (musician instanceof Singer){
        ((Singer) musician).perform();
    } else if (musician instanceof Kpop){
        ((Kpop) musician).perform();
    } else if (musician instanceof Biduan){
        ((Biduan) musician).perform();
    } else {
        musician.perform();
    }
    System.out.println();
  }
}
```

7. **Tunjukkan contoh inheritance pada percobaan diatas!**

a. **Biduan mewarisi dari Singer**

```
public class Biduan extends Singer{ ... }
```

b. **Kpop mewarisi dari Singer**

```
public class Kpop extends Singer{ ... }
```

c. **KpopFans mewarisi dari Fans**

```
public class KpopFans extends Fans{ ... }
```

d. **Singer mewarisi dari Musician**

```
public class Singer extends Musician{ ... }
```