LAPORAN PRAKTIKUM
MATA KULIAH PEMROGRAMAN BERBASIS OBJEK

Dosen Pengampu : Vit Zuraida, S.Kom, M.Kom

JOBSHEET - 12

Nama : **M. Zidna Billah Faza**

NIM : **2341760030**

Prodi : **D-IV Sistem Informasi Bisnis**

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

POLIMORFISME

A. Percobaan 1 (Heterogenous Collection)

1) Buatlah folder baru dengan nama Praktikum10. Di dalamnya, buat class Pegawai

```
public class Pegawai {
    public String nip;
    public String nama;

    public Pegawai() {

    }

    public Pegawai(String nip, String nama) {
        this.nip = nip;
        this.nama = nama;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void displayInfo() {
        System.out.println("NIP : " + this.nip);
        System.out.println("Nama : " + this.nama);
    }
}
```

2) Tambahkan class Dosen

```
public class Dosen extends Pegawai{
    public String nidn;

    public Dosen() {

    }

    public Dosen(String nip, String nama, String nidn) {
        super(nip, nama);
        this.nidn = nidn;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void displayInfo() {
        super.displayInfo();
        System.out.println("NIDN : " + this.nidn);
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void mengajar() {
        System.out.println(x:"Membuat rencana pembelajaran");
        System.out.println(x:"Menyusun materi");
        System.out.println(x:"Melakukan evaluasi");
        System.out.println(x:"Melaksanakan PBM");
    }
}
```

3) Tambahkan class TenagaKependidikan

```
public class TenagaKependidikan extends Pegawai {
    public String kategori;

    public TenagaKependidikan() {

    }

    public TenagaKependidikan(String nip, String nama, String kategori) {
        super(nip, nama);
        this.kategori = kategori;
    }

    Codeium: Refactor | Explain | Generate Javadoc | X
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Kategori : " + kategori);
    }
}
```

4) Buatlah class Demo beserta fungsi main(). Di dalam fungsi main(), instansiasi beberapa object dosen dan tenaga kependidikan sebagai berikut

```
Dosen dosen1 = new Dosen(nip:"19940201", nama:"Widia, S.Kom. M.Kom", nidn:"199402");
Dosen dosen2 = new Dosen(nip:"19700105", nama:"Muhammad, S.T, M.T", nidn:"197001");

TenagaKependidikan tendik1 = new TenagaKependidikan(nip:"19750301", nama:"Aida, A.Md.", kategori:"Tenaga Administrasi");
TenagaKependidikan tendik2 = new TenagaKependidikan(nip:"19650304", nama:"Rika, S.T.", kategori:"Tenaga Laboratorium");
```

5) Buatlah arrayList daftarPegawai bertipe ArrayList of Pegawai.

```
ArrayList<Pegawai> daftarPegawai = new ArrayList<Pegawai>();
```

6) Konsep polimorfisme mengizinkan object dosen1, dosen2, tendik1, dan tendik2 untuk ditambahkan ke Array List daftarPegawai meskipun tidak secara eksplisit bertipe Pegawai.

```
daftarPegawai.add(dosen1);
daftarPegawai.add(dosen2);
daftarPegawai.add(tendik1);
daftarPegawai.add(tendik2);
```

7) Compile dan run program untuk memastikan bahwa heterogenous collection dapat dibuat.

B. Percobaan 2 (Object Casting)

1) Pada langkah sebelumnya, Anda telah membuat object dosen1 yang diinstansiasi dari class Dosen. Object dosen1 bertipe Dosen. Dengan kata lain, object tersebut akan dikenali/diperlakukan sebagai object bertipe Dosen. Oleh karena itu, dosen1 memiliki atribut NIDN dan dapat memanggil method mengajar().

2) Modifikasi fungsi main() sebagai berikut

```
Dosen dosen1 = new Dosen(nip:"19940201", nama:"Widia, S.Kom. M.Kom", nidn:"199402");

System.out.println(dosen1.nip);
System.out.println(dosen1.nama);
System.out.println(dosen1.nidn);
dosen1.mengajar();
```

3) Run dan compile kode program. Amati hasilnya.

```
19940201
Widia, S.Kom. M.Kom
199402
Membuat rencana pembelajaran
Menyusun materi
Melakukan evaluasi
Melaksanakan PBM
```

4) Lakukan upcasting object dosen1 menjadi object dari parent class nya, yaitu Pegawai. Object pegawai1 merupakan hasil instansiasi dari class Dosen, tetapi proses upcasting ini membuat pegawai1 dikenali dan diperlakukan sebagai object bertipe Pegawai.

```
Pegawai pegawai1 = dosen1;
```

5) Modifikasi kode program sebagai berikut

```
Dosen dosen1 = new Dosen(nip:"19940201", nama:"Widia, S.Kom. M.Kom", nidn:"199402");

Pegawai pegawai1 = dosen1;

System.out.println(pegawai1.nip);
System.out.println(pegawai1.nama);
System.out.println(pegawai1.nidn);
pegawai1.mengajar();
```

6) Tidak ada compile error pada baris kode upcasting. Error muncul saat mengakses atribut NIDN dan memanggil method mengajar() karena object1 dikenali sebagai object bertipe Pegawai, sementara class Pegawai tidak memiliki atribut NIDN dan method mengajar().

7) Modifikasi fungsi main sebagai berikut

```
Dosen dosen1 = new Dosen(nip:"19940201", nama:"Widia, S.Kom. M.Kom", nidn:"199402");

Pegawai pegawai1 = dosen1;

System.out.println(pegawai1.nip);
System.out.println(pegawai1.nama);
pegawai1.displayInfo();
```

8) Run dan compile kode program, amati hasilnya

```
19940201
Widia, S.Kom. M.Kom
NIP : 19940201
Nama : Widia, S.Kom. M.Kom
NIDN : 199402
```

9) Perhatikan bahwa method displayInfo() dapat dipanggil oleh object pegawai1 karena terdapat method displayInfo() pada class Pegawai sehingga tidak muncul compile error. Tetapi saat program di-run, yang dieksekusi adalah method displayInfo() pada class Dosen, karena adanya overriding

10) Cobalah lakukan downcasting object pegawai1 ke class TenagaKependididikan. Perhatikan bahwa downcasting harus dilakukan secara eksplisit dengan menyebutkan nama subclass nya.

```
System.out.println(pegawai1.nip);
System.out.println(pegawai1.nama);
pegawai1.displayInfo();

TenagaKependidikan test = (TenagaKependidikan) pegawai1;
```

11) Tidak terdapat warning pada kode program karena tidak ada compile error sebab TenagaKependidikan merupakan subclass dari class Pegawai.

12) Run program dan amati bahawa terdapat runtime error java.lang.ClassCastException kerena object tersebut bukan instance dari class TenagaKependidikan.

```
Exception in thread "main" java.lang.ClassCastException: Jobsheet12.src.Dosen cannot be cast to Jobsheet12.src.TenagaKependidikan
        at Jobsheet12.src.Demo.main(Demo.java:16)
```

13) Cobalah lakukan downcasting object pegawai1 kembali ke class Dosen.

```
Dosen dosen1 = new Dosen(nip:"19940201", nama:"Widia, S.Kom. M.Kom", nidn:"199402");

Pegawai pegawai1 = dosen1;

System.out.println(pegawai1.nip);
System.out.println(pegawai1.nama);
pegawai1.displayInfo();

Dosen newDosen = (Dosen) pegawai1;

System.out.println(newDosen.nama);
System.out.println(newDosen.nidn);
newDosen.mengajar();
```

Object newDosen sekarang sudah dikenali kembali sebagai object bertipe Dosen. Oleh karena itu, atribut NIDN dapat diakses dan method mengajar() juga dapat dipanggil

14) Run dan compile kode program kemudian amati hasilnya

```
19940201
Widia, S.Kom. M.Kom
NIP : 19940201
Nama : Widia, S.Kom. M.Kom
NIDN : 199402
Widia, S.Kom. M.Kom
199402
Membuat rencana pembelajaran
Menyusun materi
Melakukan evaluasi
Melaksanakan PBM
```

C. Percobaan 3 (Polymorphic Arguments & instanceOf)

1) Misalnya pada class Demo terdapat method train() yang bertujuan untuk memberikan pelatihan bagi pegawai baru.

```
public static void train(Pegawai pegawai) {
    System.out.println(x:"Memberi pelatihan untuk pegawai");
```

2) Dengan konsep polimorfisme, method train() tidak hanya dapat dipanggil dengan argument bertipe Pegawai, tetapi juga subclass Pegawai, yaitu Dosen dan TenagaKependidikan.

3) Modifikasi kode program sebagai berikut

```
    Dosen dosen1 = new Dosen(nip:"19940201", nama:"Widia, S.Kom. M.Kom", nidn:"199402");
    TenagaKependidikan tendik1 = new TenagaKependidikan(nip:"19750301", nama:"Aida, A.Md.", kategori:"Tenaga Administrasi");

    train(dosen1);
    train(tendik1);
}


public static void train(Pegawai pegawai) {
    System.out.println(x:"Memberi pelatihan untuk pegawai");
```

4) Perhatikan bahwa terdapat proses upcasting (secara implisit) dalam polymorphic argument, artinya di dalam method train() object pegawai akan dikenali sebagai object bertipe Pegawai, sehingga atribut NIDN dan kategori tidak dapat diakses. Di samping itu, method mengajar() juga tidak dapat dipanggil.

```
    Dosen dosen1 = new Dosen(nip:"19940201", nama:"Widia, S.Kom. M.Kom", nidn:"199402");
    TenagaKependidikan tendik1 = new TenagaKependidikan(nip:"19750301", nama:"Aida, A.Md.", kategori:"Tenaga Administrasi");

    train(dosen1);
    train(tendik1);
}

public static void train(Pegawai pegawai) {
    System.out.println(x:"Memberi pelatihan untuk pegawai");
    pegawai.displayInfo();

    // test
    System.out.println(pegawai.nidn);
    System.out.println(pegawai.kategori);
    pegawai.mengajar();
}
```

5) Jika object perlu dikenali sebagai class asalnya, lakukan proses downcasting seperti percobaan sebelumnya.

6) Misalnya method train() memiliki proses yang sedikit berbeda untuk dosen dan tenaga kependidikan. Keyword instanceOf dapat digunakan untuk mengetahui dari class mana suatu object diinstansiasi untuk memastikan tidak ada runtime error saat proses downcasting.

```
    Dosen dosen1 = new Dosen(nip:"19940201", nama:"Widia, S.Kom. M.Kom", nidn:"199402");
    TenagaKependidikan tendik1 = new TenagaKependidikan(nip:"19750301", nama:"Aida, A.Md.", kategori:"Tenaga Administrasi");

    train(dosen1);
    train(tendik1);
}

public static void train(Pegawai pegawai) {
    pegawai.displayInfo();
    System.out.println(x:"Mengenalkan lingkungan kampus");
    System.out.println(x:"Menginfokan SOP/Juknis");

    if (pegawai instanceof Dosen) {
        System.out.println(x:"Memberi pelatihan pedagogik");
    }
}
```

7) Run program kemudian amati hasilnya.

```
NIP : 19940201
Nama : Widia, S.Kom. M.Kom
NIDN : 199402
Mengenalkan lingkungan kampus
Menginfokan SOP/Juknis
Memberi pelatihan pedagogik
NIP : 19750301
Nama : Aida, A.Md.
Kategori : Tenaga Administrasi
Mengenalkan lingkungan kampus
Menginfokan SOP/Juknis
```

D. Pertanyaan

1) Apakah upcasting dan downcasting dapat dilakukan dari suatu class terhadap class lain yang tidak memiliki relasi inheritance?

   Upcasting dan Downcasting hanya dapat dilakukan anatar class yang berelasi inheritance.

2) Dari 2 baris kode program berikut, manakah proses upcasting yang tepat? Jelaskan

   Pegawai pegawai1 = new Dosen();

   Pegawai pegawai1 = (Pegawai) new Dosen();

   Kedua cara tersebut akan melakukan upcasting, namun cara pertama lebih sederhana karena Dosen adalah subclass dari pegawai maka secara otomatis objek Dosen akan dianggap sebagai Pegawai.

3) Apa fungsi dari keyword instanceOf?

   instanceOf berfungsi sebagai pengecekan apakah sebuah objek merupakan instance dari class lain atau tidak. Jika objek merupakan instance dari sebuah class maka instanceOf akan mengembalikan nilai true.

4) Apa yang dimaksud heterogenous collection?

   Heterogenous Collection adalah sebuah collection yang dapat menyimpan objek-objek seperti ArrayList, LinkedList, Stack, dan lain-lain.

5) Sebuah object diinstansiasi dari class Pegawai. Kemudian dilakukan downcasting menjadi object bertipe Dosen. Apakah hal ini dapat dilakukan? Lakukan percobaan untuk membuktikannya.

   Tidak dapat dilakukan karena downcasting hanya dapat dilakukan ketika objek yang diinstansiasi dari parent class sebenarnya adalah objek dari child class.

E. Percobaan 4 (Heterogenous Collection & Polymorphic Arguments dengan Interface)

1) Buatlah interface Printable

```
public interface Printable {
    void print();
}
```

2) Buatlah class yang meng-implements interface Printable yaitu class Document

```
public class Document implements Printable{
    public String fileName;
    public String content;

    Codeium: Refactor | Explain | Generate Javadoc | X
    @Override
    public void print() {
        System.out.println("Mencetak dokumen: " + content);

    }
}
```

3) Buatlah class lain yang meng-implements interface Printable yaitu Webpage

```
public class Webpage implements Printable {
    public String url;
    public String title;
    public String content;

    Codeium: Refactor | Explain | Generate Javadoc | X
    @Override
    public void print() {
        System.out.println("URL : " + url);
        System.out.println("Title : " + title);

        String plainText = content.replaceAll(regex:"<[^>]*>", replacement:"");
        System.out.println("Content : " + plainText);

    }

}
```

4) Tambahkan class NewDemo.java beserta fungsi main() lalu instansiasi object dari class Document dan Webpage. Selanjutnya instasiasi ArrayList of Printables. Konsep polimorfime mengizinkan ArrayList ini menampung object-object dari class yang berbeda asal class tersebut mengimplements interface Printables.

```
public class NewDemo {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X
    public static void main(String[] args) {
        Webpage webpage1 = new Webpage();
        webpage1.title = "Jurusan Teknologi Informasi Polinema";
        webpage1.url = "https://www.jti.polinema.ac.id/";
        webpage1.content = "<h1>Ini contoh heading</h1><br/>";

        Document doc1 = new Document();
        doc1.fileName = "PBO 2024.docx";
        doc1.content = "Ini adalah contoh konten PBO";

        ArrayList<Printable> printables = new ArrayList<>();
        printables.add(webpage1);
        printables.add(doc1);

        Printer printer1 = new Printer();
        printer1.cetak(webpage1);
        printer1.cetak(doc1);
    }

}
```

5) Buatlah class Printer dengan method cetak() yang memiliki parameter bertipe Printable

```
public class Printer {
    Codeium: Refactor | Explain | Generate Javadoc | X
    public void cetak(Printable printable) {
        printable.print();
    }
}
```

6) Pada NewDemo, instansiasi object dari class Printer kemudian panggil method cetak()dengan argumen webpage1 bertipe Webpage dan doc1 bertipe Document.

```
Printer printer1 = new Printer();
printer1.cetak(webpage1);
printer1.cetak(doc1);
```

Konsep polymoriphic arguments juga diperbolehkan tidak hanya untuk class-class dalam hirarki inheritance, tetapi juga class-class yang meng-implements interface yang sama. Pada contoh di atas, method cetak() dapat menerima argumen bertipe class apapun asal class tersebut meng-implement interface Printable.

F. Polymorphic Arguments dengan Built-In Interface

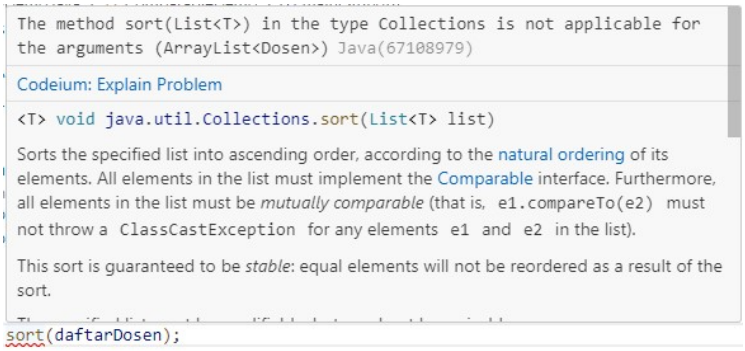
1) Buatlah class baru bernama ComparableDemo.java beserta fungsi main() nya
2) Pada fungsi main() buatlah ArrayList of Dosen bernama daftarDosen kemudian tambahkan beberapa object bertipe Dosen ke dalam ArrayList daftarDosen.

```
public class ComparableDemo {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        ArrayList<Dosen> daftarDosen = new ArrayList<>();
        daftarDosen.add(new Dosen(nip:"321", nama:"Bella", nidn:"0091"));
        daftarDosen.add(new Dosen(nip:"123", nama:"Adila", nidn:"0081"));
        daftarDosen.add(new Dosen(nip:"231", nama:"Petrus", nidn:"0071"));
    }
}
```

3) Java telah menyediakan static method sort() pada class Collections yang dapat digunakan untuk melakukan sorting elemen-elemen di dalam list.

```
public class ComparableDemo {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        ArrayList<Dosen> daftarDosen = new ArrayList<>();
        daftarDosen.add(new Dosen(nip:"321", nama:"Bella", nidn:"0091"));
        daftarDosen.add(new Dosen(nip:"123", nama:"Adila", nidn:"0081"));
        daftarDosen.add(new Dosen(nip:"231", nama:"Petrus", nidn:"0071"));
        Collections.sort(daftarDosen);
    }
}
```

4) Method sort() dapat menerima polymorphic arguments, artinya method ini dapat menerima argumen list of objects dengan berbagai tipe data yang berbeda. Tetapi dapat dilihat bahwa pada langkah sebelumnya, masih terdapat error sebab meskipun bersifat polymorphic, class dari elemen di dalam list harus meng-implement interface Comparable.



5) Oleh karena itu, class Dosen harus mengimplements interface Comparable

```
public class Dosen extends Pegawai implements Comparable<Dosen> {
```

6) Interface Comparable akan "memaksa" class Dosen untuk mengimplements method compareTo(). Gunakan kode program berikut jika Anda ingin mengurutkan object Dosen berdasarkan atribut nama.

```
@Override
public int compareTo(Dosen otherDosen) {
    return this.nama.compareTo(otherDosen.nama);
}
```

Catatan: *implementasi method compareTo() sesuai tipe data atribut yang dipilih. Anda juga dapat menggunakan static method Double.compare() atau Integer.compare().*

7) Untuk mengecek hasil sorting, tambahkan kode program untuk menampilkan daftarDosen

```
public class ComparableDemo {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        ArrayList<Dosen> daftarDosen = new ArrayList<>();
        daftarDosen.add(new Dosen(nip:"321", nama:"Bella", nidn:"0091"));
        daftarDosen.add(new Dosen(nip:"123", nama:"Adila", nidn:"0081"));
        daftarDosen.add(new Dosen(nip:"231", nama:"Petrus", nidn:"0071"));

        Collections.sort(daftarDosen);

        for (Dosen dosen : daftarDosen) {
            dosen.displayInfo();
        }
    }
}
```

8) Note: jika tidak diimport oleh VS Code secara otomatis, lakukan import

```
import java.util.ArrayList;
import java.util.Collections;
```