LAPORAN PRAKTIKUM
Task 1

Disusun Oleh:
Dzakir Tsabit Asy Syafiq (241511071)
Jurusan Teknik Komputer dan Informatika

Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
27/10/2025

Pembahasan

1. Penjelasan Kode Student.java

Implementasi Generics

Kode public class Student<T> mendefinisikan sebuah **kelas generik**. Penggunaan <T> (tipe parameter) memungkinkan kelas Student menjadi fleksibel.

Ini terlihat langsung pada atribut:

```
private T department;
```

Mirip dengan contoh CellGenericCollection<T> dalam materi, penggunaan T di sini berarti bahwa saat objek Student dibuat, tipe data untuk department dapat ditentukan (misalnya, String atau Integer). Ini adalah inti dari "create a collection using generics", di mana Student adalah objek yang akan dimasukkan ke dalam koleksi tersebut.

Objek untuk Koleksi

Kelas Student ini berfungsi sebagai cetakan (blueprint) untuk objek-objek yang akan disimpan di dalam sebuah **koleksi (Collection)**. Objek-objek Student yang dibuat dari kelas ini dapat ditambahkan ke implementasi List seperti ArrayList atau ke implementasi Set seperti HashSet.

Mekanisme Pengurutan (Sorting)

Materi pelajaran mencantumkan aktivitas untuk mengurutkan ArrayList . Metode Collections.sort() digunakan dalam materi tersebut untuk mengurutkan daftar String.

Agar Collections.sort() dapat mengurutkan objek kustom (seperti Student), objek tersebut harus menyediakan logika perbandingan. Kode ini menyediakannya melalui:

```
implements Comparable<Student<T>>
```

Dan implementasi metodenya:

```
@Override
public int compareTo(Student<T> other) {
    return this.name.compareTo(other.getName());
}
```

Metode compareTo ini adalah instruksi yang memberi tahu Collections.sort() untuk mengurutkan daftar objek Student berdasarkan **Nama** (Name) mereka secara alfabetis (leksikografis).

2. File: StudentDemo.java

Ini adalah file yang *menggunakan* kelas Student. File ini berisi logika utama program (metode main) untuk membuat objek dan memanipulasi koleksi.

1. Implementasi ArrayList

Materi pelajaran membahas ArrayList sebagai implementasi dari List. ArrayList adalah koleksi yang ukurannya dinamis, tidak seperti array biasa.

Implementasi ini terlihat di Bagian 1 kode:

```
List<Student> studentList = new ArrayList<>();
```

- Sebuah ArrayList baru dibuat untuk menyimpan objek Student. Ini sesuai dengan materi tentang "Implement an ArrayList".

Selanjutnya, metode .add() digunakan untuk menambah 5 objek Student baru:

```
studentList.add(new Student<String>("241511071", ...));
studentList.add(new Student<Integer>("241511082", ...));
// ... dan seterusnya
```

- Ini sesuai dengan metode add(E e) yang dijelaskan dalam materi (Hal. 15), yang berfungsi untuk "menambahkan elemen ke akhir list".

2. Penggunaan Generics

Materi pelajaran secara khusus membahas penggunaan **Generics** untuk membuat koleksi yang *type-safe* (aman secara tipe data).

Kode tersebut menggunakan Generics secara ekstensif pada kelas Student itu sendiri:

```
new Student<String>(..., "Teknik Informatika")
new Student<Integer>(..., 101)
```

- Generics berhasil digunakan untuk mengizinkan atribut department di kelas Student agar bisa menyimpan String atau Integer. Ini adalah penerapan praktis dari konsep yang dibahas di materi.

3. Mengurutkan List

Materi pelajaran memiliki aktivitas khusus untuk mengurutkan ArrayList.

Tugas ini dijalankan oleh kode:

```
Collections.sort(studentList);
```

- Metode Collections.sort() dipanggil pada studentList.
- Seperti yang dijelaskan di materi, Collections.sort menerima sebuah List sebagai parameternya , dan ArrayList adalah sebuah List. Ini berhasil mengurutkan daftar mahasiswa berdasarkan nama (seperti yang didefinisikan dalam metode compareTo di kelas Student).

4. Perbedaan List vs. Set

Materi pelajaran menekankan perbedaan utama antara List dan Set:

- **List**: Adalah koleksi *terurut* dan **mengizinkan duplikat**.
- **Set**: Adalah koleksi yang **tidak mengizinkan duplikat**.

Bagian 2 kode (menggunakan Vector, yang juga merupakan List) secara tidak langsung membuktikan penggunaan List:

```
studentVector.add(new Student<String>("...","Farah",..., "Teknik Informatika"));

studentVector.add(new Student<String>("...","Hana", ..., "Teknik Informatika"));

studentVector.add(new Student<String>("...","Joko", ..., "Teknik Informatika"));
```

- Saat proses *filter* (penyaringan) dilakukan, ketiga mahasiswa (Farah, Hana, dan Joko) berhasil ditampilkan.
- Ini membuktikan bahwa koleksi tersebut adalah List (karena mengizinkan elemen yang "mirip" atau duplikat secara departemen). Jika Set digunakan, perilaku penyimpanannya akan berbeda

3. Konsep Enkapsulasi (Encapsulation)

Enkapsulasi adalah salah satu pilar utama dalam Pemrograman Berbasis Objek (OOP).

Konsep intinya adalah **pembungkusan**. Enkapsulasi membungkus data (atribut) dan metode (fungsi) yang mengoperasikan data tersebut ke dalam satu unit tunggal, yang disebut **Kelas** (Class).

Tujuan utamanya adalah **menyembunyikan detail internal** (Data Hiding). Data di dalam objek tidak boleh diakses atau diubah secara langsung dari luar. Interaksi harus dilakukan melalui "perantara" yang sudah disediakan oleh kelas tersebut.

Cara Kerja Enkapsulasi

Implementasi Enkapsulasi dalam kode (seperti pada kelas Student yang sudah dibuat) mengikuti dua aturan sederhana:

1. **Jadikan Atribut private:** Semua atribut (variabel) di dalam kelas dideklarasikan sebagai private. Ini mengunci data tersebut sehingga hanya bisa diakses dari *dalam* kelas itu sendiri.

```
public class Student<T> {
    private String studentID;
    private String name;
    ...
}
```

2. **Sediakan Metode public (Getters & Setters):** Kelas menyediakan metode public (terbuka untuk umum) untuk berinteraksi dengan data private tersebut.
    - **Getter (Pengambil):** Metode untuk *membaca* nilai atribut. (Contoh: getName()).
    - **Setter (Pengatur):** Metode untuk *mengubah* nilai atribut. (Kelas Student kita tidak memiliki *setter*, tapi jika ada, contohnya adalah public void setName(String newName)).

```
  public String getName() {
        return name; // Memberi akses baca ke 'name' yang private
    }

    public T getDepartment() {
```

```
        return department; // Memberi akses baca ke 'department' yang
private
    }
```

Mengapa Enkapsulasi Penting?

- **Kontrol Penuh:** Kelas memiliki kendali penuh atas datanya. Jika ada *setter* (misal setStudentID), kelas bisa menambahkan logika validasi. Contoh: "ID tidak boleh kosong" atau "ID harus diawali huruf S".
- **Keamanan Data:** Melindungi data dari modifikasi yang tidak disengaja atau tidak sah. Kode di luar kelas tidak bisa "mengacaukan" nilai atribut secara sembarangan.
- **Fleksibilitas (Mudah Dipelihara):** Implementasi internal kelas bisa diubah tanpa merusak kode lain yang menggunakannya. Misalnya, kita bisa mengubah private String name menjadi private String firstName dan private String lastName. Selama metode public String getName() tetap ada (dan diubah agar mengembalikan firstName + " " + lastName), kelas StudentDemo tidak akan error dan tidak perlu tahu ada perubahan internal.

4. Output

```
Note: StudentDemo.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
 Bagian 1: ArrayList dan Sorting

[SEBELUM DIURUTKAN BERDASARKAN NAMA]
Student [ID=241511071, Name=Dzakir, Address=Jl. Merdeka, Department=Teknik Informatika]
Student [ID=241511082, Name=Ibnu, Address=Jl. Setiabudi, Department=101]
Student [ID=241511091, Name=Asep, Address=Jl. Dago, Department=Teknik Informatika]
Student [ID=241511088, Name=Ican, Address=Jl. Cihampelas, Department=Teknik Informatika]
Student [ID=241511099, Name=Umem, Address=Jl. Riau, Department=102]

[SETELAH DIURUTKAN BERDASARKAN NAMA]
Student [ID=241511091, Name=Asep, Address=Jl. Dago, Department=Teknik Informatika]
Student [ID=241511071, Name=Dzakir, Address=Jl. Merdeka, Department=Teknik Informatika]
Student [ID=241511082, Name=Ibnu, Address=Jl. Setiabudi, Department=101]
Student [ID=241511088, Name=Ican, Address=Jl. Cihampelas, Department=Teknik Informatika]
Student [ID=241511099, Name=Umem, Address=Jl. Riau, Department=102]

 Bagian 2: Vector dan Filtering

[MAHASISWA DI DEPARTEMEN 'Teknik Informatika']
Student [ID=24151123, Name=Farah, Address=Jl. Sudirman, Department=Teknik Informatika]
Student [ID=24159921, Name=Hana, Address=Jl. Pasteur, Department=Teknik Informatika]
Student [ID=23141991, Name=Joko, Address=Jl. Buahbatu, Department=Teknik Informatika]
PS D:\SEMESTER 3 TEKNIK INFORMATIKA\PEMROGRAMAN BERORIENTASI OBJECT\GIT\Object-Oriented-SM
```