LAPORAN PRAKTIKUM PEMROGRAMAN
GENERIC JAVA

Oleh:

Ihsan Fauzi 241524048

PROGRAM STUDI D4-TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG
2025

DAFTAR ISI

DAFTAR ISI i
BAB I PENDAHULUAN 1
BAB II ANALISIS GENERIC PROGRAM 2
BAB III KESIMPULAN 6
BAB IV LINK GITHUB 7

BAB I
PENDAHULUAN

Generic Programming di Java adalah fitur penting yang diperkenalkan pada Java 5.0. Konsep ini memungkinkan pemrograman lebih fleksibel dengan membuat kode yang dapat bekerja dengan berbagai tipe data yang berbeda namun tetap menjaga tipe safety. Dalam laporan ini, akan dilakukan analisis terhadap implementasi berbagai konsep generic programming dalam sebuah sistem machine learning sederhana. Sistem ini mencakup berbagai model yang bekerja dengan tipe data yang berbeda seperti gambar, teks, dan kombinasi keduanya, menunjukkan bagaimana generic programming dapat meningkatkan desain perangkat lunak dengan mengurangi duplikasi kode dan meningkatkan fleksibilitas.

BAB II
ANALISIS GENERIC PROGRAM

1. Generic class adalah kelas yang menerima satu atau lebih tipe parameter yang dapat digunakan di seluruh kelas tersebut. Pembuatan generic class memungkinkan kita untuk menulis satu kelas yang dapat bekerja dengan berbagai tipe data, tanpa perlu menulis ulang kode untuk setiap tipe data yang berbeda. Dengan menggunakan parameter tipe seperti itu, kita dapat menentukan secara spesifik tipe data yang akan digunakan saat membuat instance dari kelas tersebut, sehingga compiler dapat memeriksa tipe data dan mencegah kesalahan tipe saat kompilasi.

```java
// Dataset.java
import java.util.List;

// Ini adalah contoh dari Generic Class
// Dataset<T> dapat menyimpan data dengan tipe apapun, selama konsisten (Image, Text, dll)
public class Dataset<T> {
    private List<T> data;

    public Dataset(List<T> data) {
        this.data = data;
    }

    public List<T> getData() {
        return data;
    }
}
```

Pada kode di atas, Dataset<T> adalah sebuah generic class dengan parameter tipe T. Parameter T digunakan untuk mendefinisikan tipe data yang akan disimpan dalam list data. Saat membuat instance dari Dataset, pengguna dapat menentukan tipe data spesifik yang akan digunakan, seperti Dataset<Image> untuk koleksi gambar atau Dataset<Text> untuk koleksi teks. Generic parameter T digunakan pada empat tempat dalam kelas ini:

a. Deklarasi kelas: public class Dataset<T>
b. Deklarasi atribut: private List<T> data
c. Parameter konstruktor: public Dataset(List<T> data)
d. Tipe return method: public List<T> getData()

Penggunaan generic class ini memungkinkan pembuatan dataset untuk berbagai tipe data (Image, Text, Video, ImageText) tanpa perlu menulis ulang kode yang sama untuk setiap tipe.

2. Generic sebagai Parameter / Method / Wildcard

Generic dapat digunakan sebagai parameter, dalam method, atau menggunakan wildcard. Konsep ini memperluas fleksibilitas generic programming dengan memungkinkan method untuk menerima atau mengembalikan tipe generic, serta menggunakan wildcard untuk menangani tipe yang tidak diketahui secara spesifik. Wildcard (?) digunakan ketika kita ingin membuat method yang lebih fleksibel dalam menerima beragam tipe generic, baik dengan batasan tertentu (bounded wildcard) maupun tanpa batasan.

```java
// Trainer.java
import java.util.List;

// Ini adalah contoh penggunaan Wildcard (? extends Model<?>)
// Trainer dapat menerima list model dengan tipe tidak pasti selama mereka turunan dari Model<?>
```

```java
public class Trainer {
    public static void trainAll(List<? extends TrainableModel<?>>
models, Dataset<?> dataset) {
        for (TrainableModel<?> model : models) {
            System.out.println("Training model...");
            model.safeTrain(dataset);
        }
    }
}
```

Pada kode di atas, method trainAll menggunakan dua wildcard generic:

a. List<? extends TrainableModel<?>> - menunjukkan bahwa method menerima List yang berisi objek dari tipe TrainableModel atau turunannya
b. Dataset<?> - menunjukkan dataset dengan tipe data yang tidak ditentukan
c. TrainableModel<?> - menunjukkan model yang bekerja dengan tipe data yang tidak ditentukan
d. Penggunaan wildcard ? untuk memungkinkan method menerima berbagai tipe model dan dataset

Method safeTrain dalam TrainableModel interface juga merupakan contoh penggunaan generic parameter:

```java
// TrainableModel.java
public interface TrainableModel<T> extends Model<T> {
    void safeTrain(Dataset<?> dataset);
}
```

Penggunaan wildcard membuat method trainAll sangat fleksibel karena dapat menerima berbagai implementasi TrainableModel dengan berbagai tipe data tanpa harus menulis ulang method yang sama untuk setiap tipe.

3. Generic Inheritance

Generic inheritance terjadi ketika kelas turunan (subclass) mewarisi parameter generic dari kelas induk (superclass), atau ketika interface mewarisi parameter generic dari interface lain. Ini memungkinkan subclass untuk menggunakan parameter tipe yang sama dengan superclass-nya, atau membatasi tipe data yang dapat digunakan oleh subclass.

```java
// ImageDataset.java
import java.util.List;

// Ini contoh dari Generic Inheritance
// ImageDataset adalah subclass dari Dataset<Image>
public class ImageDataset extends Dataset<Image> {
    public ImageDataset(List<Image> data) {
        super(data);
    }
}
```

Pada kode di atas, ImageDataset adalah subclass dari Dataset<Image>. Ini menunjukkan generic inheritance dimana:

a. ImageDataset mewarisi semua method dan field dari Dataset<Image>
b. Parameter generic T dari Dataset<T> dibatasi menjadi tipe Image pada ImageDataset
c. Konstruktor ImageDataset memanggil konstruktor super(data) dari kelas induk
d. ImageDataset mengkhususkan Dataset hanya untuk tipe data Image

Contoh lain adalah interface TrainableModel<T> yang merupakan turunan dari Model<T>:

```java
// TrainableModel.java
public interface TrainableModel<T> extends Model<T> {
    void safeTrain(Dataset<?> dataset);
}
```

Generic inheritance memungkinkan penggunaan hirarki tipe yang lebih spesifik, seperti menentukan bahwa ImageDataset hanya dapat berisi objek dengan tipe Image, sementara menyimpan semua fungsionalitas dari kelas Dataset generic.

4. Generic Multiple (Parameter <T> <M>)
   Generic multiple parameter adalah penggunaan lebih dari satu parameter tipe dalam deklarasi kelas atau interface. Hal ini memungkinkan pembuatan komponen yang dapat bekerja dengan berbagai kombinasi tipe data yang berbeda, meningkatkan fleksibilitas dan kemampuan penggunaan kembali kode.

```java
// Experiment.java
// Ini contoh dari Generic Multiple Type Parameter
// M = Model<D>, D = Tipe data
public class Experiment<M extends Model<D>, D> {
    private M model;
    private Dataset<D> dataset;

    public Experiment(M model, Dataset<D> dataset) {
        this.model = model;
        this.dataset = dataset;
    }

    public void run() {
        model.train(dataset);
        System.out.println("Experiment done.");
    }
}
```

Pada kode Experiment di atas, terdapat multiple generic parameter yang digunakan:

a. Parameter M dengan batasan extends Model<D> - menunjukkan bahwa M harus merupakan implementasi dari interface Model
b. Parameter D yang menunjukkan tipe data yang digunakan oleh model dan dataset
c. Field model dengan tipe M yang merupakan parameter generic
d. Field dataset dengan tipe Dataset<D> yang menggunakan parameter generic kedua

Multiple generic parameter ini memungkinkan Experiment untuk bekerja dengan berbagai pasangan model dan dataset yang bersesuaian, seperti Experiment<KNNImageModel, Image> untuk percobaan dengan model KNN pada data gambar, atau Experiment<SVMTextModel, Text> untuk percobaan dengan model SVM pada data teks.

5. Generic Interface
   Generic interface adalah interface yang menggunakan satu atau lebih parameter tipe, memungkinkan implementasi interface tersebut untuk bekerja dengan berbagai tipe data yang berbeda. Dengan generic interface, kita dapat mendefinisikan kontrak yang harus dipenuhi oleh implementasi, sambil tetap memberikan fleksibilitas terkait tipe data yang digunakan.

```java
// Model.java
// Ini adalah contoh Generic Interface
// Interface Model<T> mendefinisikan bahwa setiap model bekerja
dengan jenis data tertentu (T)
public interface Model<T> {
    void train(Dataset<T> dataset);
    double predict(T input);
}
```

Pada kode di atas, Model<T> adalah sebuah generic interface yang menyediakan kontrak untuk kelas-kelas model machine learning:

a. Parameter tipe T digunakan untuk menunjukkan jenis data yang digunakan oleh model
b. Method train() menerima Dataset<T> yang sesuai dengan tipe model
c. Method predict() menerima input dengan tipe T dan mengembalikan hasil prediksi

d. Interface ini diimplementasikan oleh berbagai model seperti KNNImageModel, SVMTextModel, dll

   Generic interface memungkinkan kita untuk menerapkan kontrak yang sama (seperti kemampuan untuk melakukan training dan prediksi) ke berbagai jenis model, sambil mempertahankan keamanan tipe untuk setiap implementasi spesifik.

BAB III
KESIMPULAN

Berdasarkan analisis implementasi generic programming dalam sistem machine learning sederhana di atas, dapat disimpulkan bahwa:

1. **Multiple Generic Parameter** adalah penggunaan lebih dari satu parameter tipe dalam deklarasi kelas atau interface. Hal ini terlihat pada class Experiment<M extends Model<D>, D> yang menggunakan dua parameter tipe berbeda. Parameter ini diimplementasikan pada Main.java saat membuat instance seperti Experiment<KNNImageModel, Image>, Experiment<SVMTextModel, Text>, dan lainnya, memungkinkan eksperimen dijalankan dengan berbagai kombinasi model dan tipe data.
2. **Generic Class** memungkinkan pembuatan struktur data yang fleksibel seperti Dataset<T> yang dapat digunakan untuk berbagai tipe data tanpa menulis ulang kode.
3. **Generic Parameter dan Wildcard** meningkatkan fleksibilitas dalam pembuatan method yang dapat bekerja dengan berbagai tipe data, seperti method trainAll yang dapat melatih berbagai jenis model dengan berbagai jenis dataset.
4. **Generic Inheritance** memungkinkan pembuatan hirarki kelas yang lebih spesifik namun tetap mempertahankan fleksibilitas generic, seperti pada ImageDataset yang mewarisi dari Dataset yang adalah kelas generik. Sehingga ImageDataset extend Dataset<T> Ini memungkinkan ImageDataset untuk bekerja dengan berbagai tipe data, tidak hanya terbatas pada tipe Image.
5. **Generic Interface** mendefinisikan kontrak umum yang dapat diimplementasikan oleh berbagai kelas dengan tipe data spesifik, seperti Model<T> yang diimplementasikan oleh model untuk gambar, teks, dan tipe data lainnya.

BAB IV
LINK GITHUB

https://github.com/isanzzi/Tekpro-sem-2/tree/main/W8-Generic%20Programming-machine%20learning