

Laporan Tugas

Functional Programming



Dipersiapkan oleh :

Muhammad Ariq Faridzki

2242004

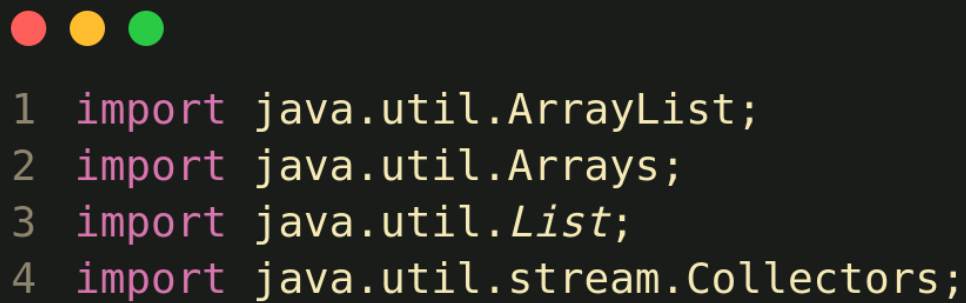
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER
"AMIK BANDUNG"
2023

Penjelasan Function

Project yang saya buat adalah untuk menemukan data email dengan kriteria yang ingin dicari.

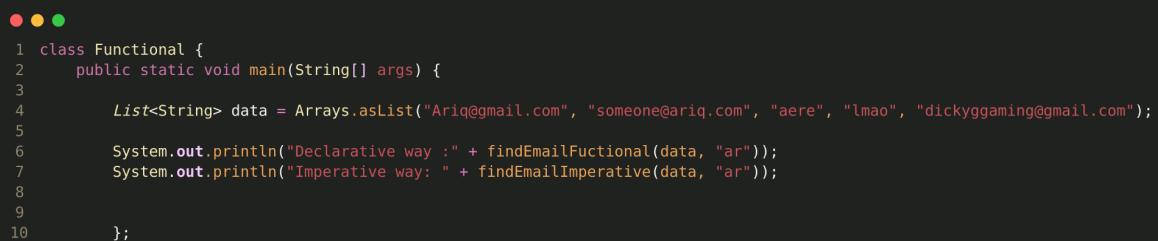
Persiapan

Agar method berjalan saya menggunakan class bawaan java yaitu :



```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.List;
4 import java.util.stream.Collectors;
```

Dan berikut pemanggilan method :



```
1 class Functional {
2     public static void main(String[] args) {
3
4         List<String> data = Arrays.asList("Ariq@gmail.com", "someone@ariq.com", "aere", "lmao", "dickygaming@gmail.com");
5
6         System.out.println("Declarative way : " + findEmailFuctional(data, "ar"));
7         System.out.println("Imperative way: " + findEmailImperative(data, "ar"));
8
9     }
10 }
```

Kode Declarative

```
1  /**
2   * Mencari email dengan string biasa - Functional Way
3   *
4   * @param array data emailnya
5   * @param searchArgs kriteria data yang ingin dicari
6   * @return array List tipe String
7   */
8
9  static List<String> findEmailFuctional (final List<String> array, final String searchArgs){
10
11  return array.stream()
12  .map(string -> string.toLowerCase())
13  .filter(string -> string.contains("@") && string.contains(searchArgs.toLowerCase()))
14  .collect(Collectors.toList());
15
16 }
```

Kode tersebut declarative (Pure Function) dikarenakan tidak mengubah data argumen yang telah diberikan.

Berikut penjelasannya :

- Method **.map** itu hanya membuat stream array baru dan menerapkan function yang telah diberikan(Chaudhary).
- setelah itu method **.filter** membuat stream array baru dengan data yang memenuhi kriteria. (Marimuthu) jadi seperti method **.map**
- Dan yang terakhir method **.collect** mengumpulkan stream array data yang telah diberikan oleh method **.filter** dan mengubahnya menjadi tipe data **List<Integer>**
- Terlihat kode lebih mudah dimengerti dan jelas

Kode Imperative

```
1  /**
2   * Mencari email dengan string biasa - Imperative Way
3   *
4   * @param array data emailnya
5   * @param searchArgs kriteria data yang ingin dicari
6   * @return array List tipe String
7   */
8  static List<String> findEmailImperative (List<String> array, String searchArgs){
9
10     List<String> result = new ArrayList<>();
11
12     for (int i = 0; i < array.size(); i++) {
13         String email = array.get(i).toLowerCase();
14         if (email.contains("@") && email.contains(searchArgs.toLowerCase())) {
15
16             result.add(array.get(i).toLowerCase());
17
18         }
19     }
20 }
21
22 return result;
23 }
```

Kode tersebut imperative (Non - Pure Function) dikarenakan adanya perubahan data argumen yang telah diberikan.


Berikut penjelasannya :

- Saat melakukan pengulangan di baris 17 ada pengubahan struktur data (variable result) yaitu dengan menambah data di dalamnya
- Kode tersebut terlihat seperti prosedur yang berurutan, seperti membuat variabel baru, setelah itu pengulangan data yaitu memfilter data menggunakan **if** dan menambahkan data ke variable result.
- Tidak seperti kode declarative, kalau urutan tidak terlalu mempengaruhi proses.

Hasil Output

Berikut hasil yang diberikan dengan argumen :

```
["Ariq@gmail.com", "someone@ariq.com", "aere", "lmao",  
"dickyggaming@gmail.com"]
```



```
1 // Hasil  
2  
3 Declarative way :[ariq@gmail.com, someone@ariq.com]  
4 Imperative way: [ariq@gmail.com, someone@ariq.com]
```

Kesimpulan

Setelah mempelajari metode declarative, saya merasa ada perubahan dalam cara menggunakan method bawaan java walaupun ada kendala seperti penggunaan method yang belum terbiasa tapi efek semua itu terlihat dari kode yang rapi dan jelas yang akan memudahkan untuk mencari kesalahan.

Saran

- Masih belum terbiasa dengan method bawaan java 8
- Masih terbiasa untuk menggunakan metode imperative
- Bingung untuk membuat control flow (if) dalam metode declarative

Referensi

Chaudhary, Namita. "Java Stream map()." *Scaler*, Namita Chaudhary, 20 12 2022, <https://www.scaler.com/topics/java-stream-map/>. Accessed 1 October 2023.

Marimuthu, Ganesh Kumar. "Functional Programming in Java With Examples." *Scaler*, 16 06 2023, <https://www.scaler.com/topics/java/functional-programming-in-java/>. Accessed 1 October 2023.

Marimuthu, Ganesh Kumar. "Streams in Java- Complete Tutorial with Examples." *Scaler*, Scaler, 4 5 2023, <https://www.scaler.com/topics/java/streams-in-java/>. Accessed 1 October 2023.

Oracle. "Stream (Java Platform SE 8)." *Oracle Help Center*, Oracle, <https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html>. Accessed 1 October 2023.