Bab 7

Collection

Tujuan

1. Memahami Penggunaan *Collection*
2. Mengenal Struktur Data dasar dalam Komputer
3. Mampu memanipulasi *Collection*

Collection merupakan salah satu *framework* dalam Java yang berisi kumpulan *interface* yang dapat digunakan untuk menampung beberapa data dalam sebuah variabel, *Collection* mirip dengan *Array* akan tetapi terdapat beberapa kelebihan *collection* yang memudahkan *programmer* dalam memanipulasi struktur data seperti ukuran yang dinamis, terdapat *method* untuk *sorting*, *searching*, *insert* dan *delete*. *Collection* yang paling sering digunakan dalam java adalah **List**, **Set**, dan **Map**.

7.1 List

List merupakan *collection* yang memiliki struktur mirip dengan *array* dimana data disimpan secara teratur dan memiliki indeks berupa integer dari nol, salah satu implementasi dari *interface List* adalah *ArrayList*, untuk menggunakannya, program harus mengimport *package* ***java.util.ArrayLis*t**.

7.1.1 Deklarasi *ArrayList*

ArrayList dideklarasikan sebagai objek, berbeda dengan tipe data primitif ataupun *String*, *ArrayList* menggunakan *generic* untuk menentukan tipe data yang dapat ditampung, *generic* ditandai dengan <> (Kurung Sudut), di dalam <> dinyatakan tipe datanya.

```
import java.util.ArrayList;

class Main {
    public static void main(String[] args) {
         ArrayList<String> name = new ArrayList<>();
    }
}
```

Drawing 33: Deklarasi ArrayList dengan Tipe Data String

7.1.2 Menambahkan Data kedalam *ArrayList*

ArrayList memiliki *method* ***add(e)*** untuk menambahkan data ***e*** yang sesuai dengan tipe data pada saat deklarasi, perlu diperhatikan bahwa *ArrayList* memiliki ukuran yang dinamis sehingga tidak perlu dinyatakan ukurannya.

```
import java.util.ArrayList;

class Main {
    public static void main(String[] args) {
          ArrayList<String> name = new ArrayList<>();
          name.add(“Baco”);
          name.add(“Ken”);
          name.add(“Naim”)
    }
}
```

ArrayList di atas memiliki tiga elemen, ketika *method* ***add(e)*** dipanggil, maka secara otomatis ukuran *ArrayList* akan bertambah.

7.1.3 Mengakses Elemen *ArrayList*

Sama seperti *array*, elemen pada *ArrayList* dapat diakses menggunakan perulangan, untuk mendapatkan ukuran *ArrayList*, dapat digunakan *method* ***size()***, sedangkan untuk mengakses elemennya, terdapat *method* ***get(i)***. Java *Collection* juga menyediakan *Class Iterator* (*java.util.Iterator*) dan *method* ***forEach()*** yang dapat digunakan dalam melakukan perulangan *ArrayList*.

```
import java.util.ArrayList;
import java.util.Iterator;

class Main {
    public static void main(String[] args) {
          ArrayList<String> name = new ArrayList<>();
          name.add(“Baco”);
          name.add(“Ken”);
          name.add(“Naim”);

          for(int i = 0; i < name.size(); i++) {
               System.out.println(name.get(i));
          }
          Iterator nameIterator = name.iterator();


          while(nameIterator.hasNext()) {
               System.out.println(nameIterator.next());
          }
    }
}
```

7.1.4 Menghapus Elemen ArrayList

ArrayList menyediakan *method* ***remove(i)***, ***removeAll(c)***, dan ***clear()*** untuk menghapus elemen dari *ArrayList*. *Method* ***remove(i)*** digunakan untuk menghapus salah satu elemen berdasarkan indeks ***i*** atau nilai yang sama dengan ***i***, sedangkan *method* ***removeAll(c)*** digunakan untuk menghapus elemen yang sama dengan elemen di dalam ***ArrayList c***, *method* ***clear()*** digunakan untuk menghapus semua elemen dalam *ArrayList.*

```
import java.util.ArrayList;
import java.util.Arrays;

class Main {
    public static void main(String[] args) {
          ArrayList<String> name = new ArrayList<>();
          ArrayList<String> delete;
          name.add(“Baco”);
          name.add(“Ken”);
          name.add(“Naim”);
          name.add(“Suripto”);

          String[] arr = {“Baco”, “Naim”};
          delete = new ArrayList<>(Arrays.asList(arr));

          name.forEach(System.out::println);
          name.removeAll(delete);
          System.out.println(“\nAfter Deleted”);
          name.forEach(n -> System.out.println(n));

          name.remove(“Suripto”);
          System.out.println(“\nAfter Deleted”);
          name.forEach(System.out::println);

          name.clear();
          System.out.println(“\nAfter Cleaning”);
          name.forEach(System.out::println);
    }
}
```

7.

Drawing 34: Menghapus Elemen ArrayList menggunakan remove(), remove-All(), dan clear()

7.1.5 Mencari Elemen *ArrayList*

Sama seperti sebelumnya, untuk mencari elemen dalam *ArrayList*, telah ada *method* yang disediakan yaitu ***contains(s)*** dan ***containsAll(c)***, dimana method ini mengembalikan *boolean true* jika di dalam *ArrayList* terdapat elemen ***s*** atau ***collection c***.

```
import java.util.ArrayList;

class Main {
    public static void main(String[] args) {
         ArrayList<String> name = new ArrayList<>();
         name.add(“Baco”);
         name.add(“Ken”);
         name.add(“Naim”);
         name.add(“Suripto”);
         String s = “Ken”;
         boolean c = name.contains(s);
         System.out.printf(“%s is exist? %b\n”, s, c);

    }
}
```

Drawing 35: Mencari Elemen Yang Sama dengan String s

7.1.6 Mengurutkan *ArrayList*

Untuk mengurutkan *ArrayList*, dibutuhkan *package java.util.Collections* dimana di dalamnya terdapat *method* ***sort(c)*** untuk mengurutkan elemen dari ***ArrayList c***. Jika elemen *ArrayList* berupa *String*, maka akan diurutkan secara alfabetik, jika numerik, maka akan diurutkan dari yang terkecil ke yang terbesar, jika ingin mengurutkan elemennya secara terbalik, maka dapat digunakan *method* ***reverseOrder()*** atau ***reverse(c)*** yang juga berada dalam p*ackage java.util.Collections.*

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

class Main {
    public static void main(String[] args) {
         ArrayList<String> name = new ArrayList<>();
         name.add(“Ken”);
         name.add(“Naim”);
         name.add(“Baco”);
         name.forEach(System.out::println);
         Collections.sort(name);
         System.out.println(“\nSorted Name :”);
         name.forEach(System.out::println);
    }
}
```

Drawing 36: Mengurutkan ArrayList name Secara Alfabetik

7.2 Set

Set mirip dengan *List*, hanya saja elemen dalam *Set* tidak bisa diduplikasi, artinya jika elemen yang sama dimasukkan berkali-kali kedalam *set*, maka *set* hanya akan menyimpan elemen tersebut satu kali. Salah satu implementasi dari *interface Set* adalah *HashSet.*

```
import java.util.HashSet;

class Main {
    public static void main(String[] args) {
          HashSet<String> name = new HashSet<>();
          name.add(“Ken”);
          name.add(“Naim”);
          name.add(“Baco”);
          name.add(“Naim”);
          System.out.println(name);
    }
}
```

Drawing 37: HashSet Tidak Akan Menyimpan Elemen yang Sama Lebih dari Sekali

Method-method pada *ArrayList* seperti *add(e)* dan *remove(e)* juga dapat digunakan untuk *Set*, umumnya *Set* digunakan untuk menghindari duplikasi elemen *ArrayList*.

7.3 Map

Berbeda dengan *List* dan *Set*, *Map* merupakan struktur data yang berbentuk *Key-Value Pair*, atau dalam bahasa pemrograman PHP dikenal sebagai *Associative Array*. *Map* dapat dikatakan *Array* namun dapat menggunakan indeks yang bukan integer, *map* menggunakan indeks yang disebut sebagai *key* dan harus bersifat unik. Salah satu implementasi dari *Map* adalah *HashMap*.

7.3.1 Deklarasi dan Inisialisasi *HashMap*

HashMap dideklarasikan seperti *Collection* lainnya, yaitu dengan menentukan tipe data yang dapat ditampung di dalam <> (Kurung Sudut), pada *HashMap*, yang ditentukan bukan hanya tipe data elemennya, tapi juga tipe data *key*-nya

```
import java.util.HashMap;

class Main {
    public static void main(String[] args) {
          HashMap<String, String> name = new HashMap<>();
          name.put(“K”, “Ken”);
          name.put(“N”, “Naim”);
          name.put(“B”, “Baco”);
    }
}
```

Drawing 38: HashMap name dengan Key String dan Value String

Cara memasukkan nilai kedalam *HashMap* tidak menggunakan method ***add(e)*** tetapi ***put(key, value)***.

7.3.2 Mengakses Elemen *HashMap*

Elemen *HashMap* dapat diakses menggunakan perulangan, method *forEach* ataupun menggunakan *iterator*. Terdapat beberapa *method* penting yang dapat dimanfaatkan untuk mengakses elemen *HashMap* seperti:

Method	Fungsi
get(key)	Mendapatkan nilai elemen berdasarkan key
keySet()	Mendapatkan daftar key dari HashMap
value()	Mendapatkan nilai elemen HashMap
size()	Mendapatkan Ukuran HashMap

Table 10: Beberapa Method yang Dapat Digunakan Untuk Mengakses Elemen

```
import java.util.HashMap;

class Main {
    public static void main(String[] args) {
          HashMap<String, String> name = new HashMap<>();
          name.put(“K”, “Ken”);
          name.put(“N”, “Naim”);
          name.put(“B”, “Baco”);

          name.forEach((k, v) -> {
                System.out.println(k + “ -> “ + v);
          });

          for (String key : name.keySet()) {
                System.out.println(name.get(key));
          }
    }
}
```

Drawing 39: Mengakses Elemen HashMap dengan method forEach() dan perulangan for each

7.3.3 Menghapus Elemen *HashMap*

Sama seperti *ArrayList*, *HashMap* juga memiliki *method* ***remove(k)*** dan *clear()* untuk menghapus elemen, bedanya pada *HashMap*, *method* ***remove(k)*** membutuhkan parameter ***k*** sebagai *key* untuk menghapus elemen.

7.3.4 Merubah Nilai Elemen *HashMap*

Untuk merubah nilai elemen pada *HashMap* dapat menggunakan *method* ***put(k, v)*** atau ***replace(k, v)***, keduanya berfungsi untuk merubah nilai elemen *HashMap*, namun pada *method* ***put(k, v)***, jika ***k*** dari ***v*** yang ingin diubah belum ada, maka ***k*** dan ***v*** yang dimasukkan di *method* tersebut akan ditambahkan kedalam *HashMap* sedangkan ***replace(k, v)*** hanya akan mengganti ***v*** yang sudah ada.

```
import java.util.HashMap;

class Main {
    public static void main(String[] args) {
         HashMap<String, String> name = new HashMap<>();
         name.put(“K”, “Ken”);
         name.put(“N”, “Naim”);
         name.put(“B”, “Baco”);

         name.forEach((k, v) -> {
              System.out.println(k + “ -> “ + v);
         });

         name.put(“K”, “Kennedy”);
         name.replace(“B”, “Burhan”);

         for (String key : name.keySet()) {
              System.out.println(name.get(key));
         }
    }
}
```

Drawing 40: Mengubah Nilai Menggunalan Method put(k, v) dan replace(k, v)