LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

BAB	:	COLLECTION
NAMA	:	NAMA PRAKTIKAN
NIM	:	NIM PRAKTIKAN
ASISTEN	:	Tengku Muhammad Rafi Rahardiansyah Muhammad Bin Djafar Almasyhur
TGL PRAKTIKUM	:	17 Mei 2023

BAB 9
COLLECTION

Tujuan

1. Memberikan pemahaman kepada mahasiswa tentang Collection
2. Dapat menggunakan dengan benar jenis Collection yaitu Set dan Map dalam program java

Ringkasan Materi

1. Collection

Koleksi adalah struktur data—sebenarnya, sebuah objek—yang dapat menyimpan referensi ke objek lain. Biasanya, koleksi berisi referensi ke objek yang semuanya bertipe sama. Collection-framework interface mendeklarasikan operasi-operasi yang akan dipanggil secara generic pada berbagai jenis collection. Collection-framework memiliki beberapa interface seperti yang ada pada tabel berikut:

Interface	Deskripsi
Collection	Root interface dalam hierarki collection dimana interface Set, Queue, dan List diturunkan
Set	Collection yang tidak mengandung duplikat
List	Collection terurut yang dapat berisi elemen duplikat
Set	Collection yang mengaitkan key ke nilai dan tidak dapat berisi key yang sama.
Queue	Biasanya berupa collection first-in, first-out yang memodelkan antrean.

2. Set

Set adalah Kumpulan elemen unik yang tidak berurutan (tidak ada elemen yang sama). Framework collection berisi beberapa implementasi Set, termasuk *HashSet* dan *TreeSet*. *HashSet* menyimpan elemennya dalam tabel hash, dan *TreeSet* menyimpan elemennya dalam tree.

Program *SetTest.java* menggunakan HashSet untuk menghapus string duplikat dari List. Ingat bahwa List dan Collection adalah tipe generic, jadi baris 14 membuat List yang berisi objek String, dan baris 17 melewatkan Collection String ke metode *printNonDuplicates*.

Metode *printNonDuplicates* (baris 20-28) mengambil argumen Collection. Baris 23 membuat *HashSet<String>* dari argumen *Collection<String>*. Menurut definisi, Set tidak mengandung duplikat, jadi ketika *HashSet* dibangun, ia menghapus duplikat apapun dalam Collection. Baris 25–26 elemen keluaran pada Set.

SetTest.java

```
1  // HashSet used to remove duplicate values from array of strings.
2  import java.util.List; import
3  import java.util.Arrays;
4  import java.util.HashSet;
5  import java.util.Set;
6  import java.util.Collection;
7  public class SetTest
8  {
9          public static void main( String[] args )
10         {
11                 // create and display a List< String >
12                 String[] colors = { "red", "white", "blue", "green", "gray",
13                 "orange", "tan", "white", "cyan", "peach", "gray", "orange" };
14                 List< String > list = Arrays.asList( colors );
15                 System.out.printf( "List: %s\n", list );
16                 // eliminate duplicates then print the unique values
17                 printNonDuplicates( list );
18         } // end main
19         // create a Set from a Collection to eliminate duplicates
20         private static void printNonDuplicates( Collection< String > values )
21         {
22                 // create a HashSet
23                 Set< String > set = new HashSet< String >( values );
24                 System.out.print( "\nNonduplicates are: " );
25                 for ( String value : set )
26                         System.out.printf( "%s ", value );
27                 System.out.println();
28         } // end method printNonDuplicates
29 } // end class SetTest
```

```
List: [red, white, blue, green, gray, orange, tan, white, cyan, peach, gray,
orange]

Nonduplicates are: orange green white peach gray cyan red blue tan
```

Set Terurut

Framework collection juga mencakup interface *SortedSet* (yang meng-extends Set) untuk set yang mempertahankan elemennya dalam kondisi terurut—baik urutan alami elemen (misalnya, angka dalam urutan menaik/ascending) atau urutan yang ditentukan oleh *Comparator*. Class *TreeSet* mengimplementasikan *SortedSet*. Program *SortedSetTest.java* menempatkan string ke dalam *TreeSet*. String diurutkan saat ditambahkan ke *TreeSet*. Contoh ini juga mendemonstrasikan metode range-view, yang memungkinkan program untuk melihat sebagian dari collection.

Baris 12–13 membuat *TreeSet<String>* yang berisi elemen array *colors*, lalu meng-assignkan *TreeSet<String>* baru ke variabel *tree* yang bertipe *SortedSet<String>*. Baris 16 menampilkan string set awal menggunakan metode *printSet* (baris 31–36). Baris 20 memanggil metode pada *TreeSet* yaitu *headSet* untuk mendapatkan subset dari *TreeSet* yaitu setiap elemen yang kurang dari "orange". Hasil yang dikembalikan dari *headSet* kemudian ditampilkan dengan *printSet*. Jika ada perubahan yang dibuat pada subset, perubahan tersebut juga akan dilakukan pada *TreeSet* awal, karena subset yang dikembalikan oleh *headSet* adalah tampilan *TreeSet*.

Baris 24 memanggil metode lainnya pada *TreeSet* yaitu *tailSet* untuk mendapatkan subset di mana setiap elemen yang nilainya lebih besar dari atau sama dengan "orange", lalu menampilkan hasilnya. Setiap perubahan yang dilakukan melalui *tailSet* akan mengubah *TreeSet* awal. Baris 26–27 memanggil metode *SortedSet* yaitu *first* dan *last* untuk mendapatkan elemen terkecil dan terbesar dari masing-masing set.

Metode *printSet* (baris 31–36) menerima *SortedSet* sebagai argumen dan mencetaknya. Baris 33–34 mencetak setiap elemen dari *SortedSet* menggunakan enhanced *for* statement.

SortedSetTest.java

```
1  // Using SortedSets and TreeSets.
2  import java.util.Arrays;
3  import java.util.SortedSet;
4  import java.util.TreeSet;
5  public class SortedSetTest
6  {
7      public static void main( String[] args )
8      {
9          // create TreeSet from array colors
10         String[] colors = { "yellow", "green", "black", "tan", "grey",
11         "white", "orange", "red", "green" };
12         SortedSet< String > tree =
13             new TreeSet< String >( Arrays.asList( colors ) );
```

```
14
15                      System.out.print( "sorted set: " );
16                      printSet( tree ); // output contents of tree
17
18                      // get headSet based on "orange"
19                      System.out.print( "headSet (\"orange\"): " );
20                      printSet( tree.headSet( "orange" ) );
21
22                      // get tailSet based upon "orange"
23                      System.out.print( "tailSet (\"orange\"): " );
24                      printSet( tree.tailSet( "orange" ) );
25                      // get first and last elements
26                      System.out.printf( "first: %s\n", );
27                      System.out.printf( "last : %s\n", );
28              } // end main
29
30              // output SortedSet using enhanced for statement
31              private static void printSet( SortedSet< String > set )
32              {
33                      for ( String s : set )
34                              System.out.printf( "%s ", s );
35                      System.out.println();
36              } // end method printSet
37  } // end class SortedSetTest
```

```
sorted set: black green grey orange red tan white yellow
headSet ("orange"):  black green grey
tailSet ("orange"):  orange red tan white yellow
first: black
last : yellow
```

3. Map

Map mengaitkan *key* dengan nilai. *Key* dalam Map harus unik, tetapi nilai yang terkait dengannya tidak perlu unik. Jika Map berisi *key* unik dan nilai unik, dikatakan menerapkan pemetaan satu-ke-satu. Jika hanya *key* yang unik, Map dikatakan mengimplementasikan pemetaan banyak-ke-satu—banyak *key* dapat dipetakan ke satu nilai.

Map berbeda dari Set karena Map berisi *key* dan nilai, sedangkan Set hanya berisi nilai. Tiga dari beberapa class yang mengimplementasikan interface Map adalah *Hashtable*, *HashMap* dan *TreeMap*. *Hashtable* dan *HashMap* menyimpan elemen dalam hash table, dan *TreeMaps* menyimpan elemen di tree. Bagian ini membahas hash table dan memberikan contoh yang menggunakan *HashMap* untuk menyimpan pasangan *key*/nilai. Interface *SortedMap* memperluas Map dan mempertahankan key-nya dalam urutan yang diurutkan—baik urutan alami elemen atau urutan yang ditentukan oleh Pembanding. Class *TreeMap* mengimplementasikan *SortedMap*.

Implementasi Map menggunakan hash table

Ketika sebuah program membuat objek dari tipe baru atau yang sudah ada, terkadang perlu untuk menyimpan dan me-retreivenya secara efisien. Menyimpan dan mengambil informasi dengan array akan efisien jika beberapa aspek data secara langsung cocok dengan nilai *key* numerik nya serta jika key-nya unik dan dikemas dengan mampat. Jika kita memiliki 100 karyawan dengan sembilan digit nomor identitas (Social Security Number/SSN) dan kita ingin menyimpan dan mengambil data karyawan dengan menggunakan nomor identitas sebagai *key*, maka dibutuhkan array dengan lebih dari 700 juta elemen, karena sembilan digit nomor identitas harus dimulai dengan 001–733 sesuai dengan apa yang ada pada situs web Social Security Administration. Ini tidak praktis untuk hampir semua aplikasi yang menggunakan nomor SSN sebagai *key*.

Apa yang dibutuhkan adalah skema kecepatan tinggi untuk mengubah *key* seperti nomor SSN, *inventory part number* dan sejenisnya menjadi indeks array yang unik. Kemudian, ketika aplikasi perlu menyimpan sesuatu, skema dapat mengubah *key* aplikasi dengan cepat menjadi indeks, dan data dapat disimpan di slot tersebut dalam array. Pengambilan dilakukan dengan cara yang sama: Setelah aplikasi memiliki *key* yang diinginkan untuk mengambil rekaman data, aplikasi hanya menerapkan konversi ke *key*—hal tersebut menghasilkan indeks array tempat dimana data disimpan dan diambil. Skema tersebut disebut *hashing*. Class *Hashtable* dan *HashMap* memungkinkan kita menggunakan *hashing* tanpa harus mengimplementasikan mekanisme hash table. Program *WordTypeCount.java* menggunakan *HashMap* untuk menghitung jumlah kemunculan setiap kata dalam sebuah string.

WordTypeCount.java	
1	// Program counts the number of occurrences of each word in a String.
2	import java.util.Map;
3	import java.util.HashMap;
4	import java.util.Set;

```
5  import java.util.TreeSet;
6  import java.util.Scanner;
7  public class WordTypeCount
8  {
9          public static void main( String[] args )
10         {
11                 // create HashMap to store String keys and Integer values
12                 Map< String, Integer > myMap = new HashMap< String, Integer >();
13                 createMap( myMap ); // create map based on user input
14                 displayMap( myMap ); // display map content
15         } // end main
16         // create map from user input
17         private static void createMap( Map< String, Integer > map )
18         {
19                 Scanner scanner = new Scanner( System.in ); // create scanner
20                 System.out.println( "Enter a string:" ); // prompt for user input
21                 String input = scanner.nextLine();
22                 // tokenize the input
23                 String[] tokens = input.split( " " );
24                 // processing input text
25                 for ( String token : tokens )
26                 {
27                         String word = token.toLowerCase(); // get lowercase word
28                         // if the map contains the word
29                         if ( map.containsKey( word ) ) // is word in map
30                         {
31                                 int count = map.get( word ); // get current count
32                                 map.put( word, count + 1 ); // increment count
33                         }//endif
```

```
34                                  else
35                                      map.put( word, 1 ); // add new word with a count of 1 to map
36                          }//endfor
37              } // end method createMap
38              // display map content
39              private static void displayMap( Map< String, Integer > map )
40              {
41                          Set< String > keys = map.keySet(); // get keys
42                          // sort keys
43                          TreeSet< String > sortedKeys = new TreeSet< String >( keys );
44                          System.out.println( "\nMap contains:\nKey\t\tValue" );
45                          // generate output for each key in map
46                          for ( String key : sortedKeys )
47                                      System.out.printf( "%-10s%10s\n", key, map.get( key ) );
48                          System.out.printf("\nsize: %d\nisEmpty: %b\n", map.size(), map.isEmpty() );
49              } // end method displayMap
50  } // end class WordTypeCount
```

```
Enter a string:
this is a sample sentence with several words this is another sample
sentence with several different words

Map contains:
Key             Value
a                   1
another             1
different           1
is                  2
sample              2
sentence            2
several             2
this                2
with                2
words               2

size: 10
isEmpty: false
```

Baris 12 membuat *HashMap* kosong dengan kapasitas awal default (16 elemen) dan faktor beban default (0,75)—default ini dibangun ke dalam implementasi *HashMap*.

Ketika jumlah slot yang ditempati di HashMap menjadi lebih besar dari kapasitas dikalikan dengan faktor beban, kapasitas digandakan secara otomatis. HashMap adalah class generik yang mengambil dua jenis argumen—jenis key (yaitu, String) dan jenis nilai (yaitu, Integer). Ingat bahwa tipe argumen yang dilewatkan ke class generik harus tipe referensi, maka argumen tipe kedua adalah Integer, bukan int.
Baris 13 memanggil metode createMap (baris 17–37), yang menggunakan Map untuk menyimpan jumlah kemunculan setiap kata dalam kalimat. Baris 21 memperoleh input pengguna, dan baris 23 melakukan tokenisasi (me-split kalimat per kata). Perulangan pada baris 25–36 mengubah token berikutnya menjadi huruf kecil (baris 27), kemudian memanggil metode pada Map yaitu *containsKey* (baris 29) untuk menentukan apakah kata tersebut ada di dalam Map. Jika Map tidak berisi pemetaan untuk kata tersebut, baris 35 menggunakan metode Map yaitu *put* untuk membuat entri baru di Map, dengan kata sebagai key dan objek Integer yang berisi 1 sebagai nilainya. Jika kata tersebut memang ada di Map, baris 31 menggunakan metode Map yaitu *get* untuk mendapatkan nilai terkait key di Map. Baris 32 menambah nilai itu dan menggunakan *put* untuk menggantikan nilai key terkait di Map. Metode *put* mengembalikan nilai key terkait sebelumnya, atau nol jika key tidak ada di Map.
Metode *displayMap* (baris 39–49) menampilkan semua entri dalam Map. Disana menggunakan metode dalam *HashMap* yaitu *keySet* (baris 41) untuk mendapatkan satu set key. Key memiliki tipe String di Map, jadi metode *keySet* mengembalikan tipe generik Set dengan parameter tipe String. Baris 43 membuat *TreeSet* dari key, di mana key diurutkan. Loop pada baris 46–47 mengakses setiap key dan nilainya di Map. Baris 47 menampilkan setiap key dan nilainya. Key ditampilkan dalam urutan menaik. Baris 48 memanggil ukuran metode Map untuk mendapatkan jumlah pasangan key/nilai di Map. Baris 48 juga memanggil metode pada Map yaitu *isEmpty*, yang mengembalikan boolean yang menunjukkan apakah Map kosong atau tidak.

Pelaksanaan Percobaan

Set

Ketikkan kode program dibawah ini dan analisis output dari program tersebut!

SetExample.java

```
1   // Java Program Demonstrating Operations on the Set
2   // such as Union, Intersection and Difference operations
3
4   // Importing all utility classes
5   import java.util.*;
6
7   // Main class
8   public class SetExample {
9
10      // Main driver method
11      public static void main(String args[])
12      {
13          // Creating an object of Set class
14          // Declaring object of Integer type
15          Set<Integer> a = new HashSet<Integer>();
16
```

```
17                  // Adding all elements to List
18                  a.addAll(Arrays.asList(
19                          new Integer[] { 1, 3, 2, 4, 8, 9, 0 }));
20
21          // Again declaring object of Set class
22          // with reference to HashSet
23                  Set<Integer> b = new HashSet<Integer>();
24
25          b.addAll(Arrays.asList(
26                          new Integer[] { 1, 3, 7, 5, 4, 0, 7, 5 }));
27
28
29                  // To find union
30                  Set<Integer> union = new HashSet<Integer>(a);
31                  union.addAll(b);
32                  System.out.print("Union of the two Set");
33                  System.out.println(union);
34
35                  // To find intersection
36                  Set<Integer> intersection = new HashSet<Integer>(a);
37                  intersection.retainAll(b);
38                  System.out.print("Intersection of the two Set");
39                  System.out.println(intersection);
40
41                  // To find the symmetric difference
42                  Set<Integer> difference = new HashSet<Integer>(a);
43                  difference.removeAll(b);
44                  System.out.print("Difference of the two Set");
45                  System.out.println(difference);
46          }
47      }
```

Source code

```
1   Tulis source code di sini pake courier new 12
```

Output

Penjelasan

Map

Ketikkan kode program dibawah ini dan analisis output dari program tersebut!

MapExample.java

```
1   // Java program to demonstrate
2   // the working of Map interface
3
4   import java.util.*;
5   class MapExample {
6       public static void main(String args[])
7       {
8
9           // Initialization of a Map
10          // using Generics
11          Map<Integer, String> hm1
12                  = new HashMap<Integer, String>();
13
14          // Inserting the Elements
15          hm1.put(new Integer(1), "Geeks");
16          hm1.put(new Integer(2), "Geeks");
17          hm1.put(new Integer(3), "Geeks");
18
19          System.out.println("Initial Map " + hm1);
20
21          hm1.put(new Integer(2), "For");
22
23          System.out.println("Updated Map " + hm1);
24
25          hm1.remove(new Integer(3));
26
27          // Final Map
28          System.out.println(("Final Map " + hm1);
29      }
30  }
```

Source code

```
1   Tulis source code di sini pake courier new 12
```

Output

Penjelasan

Tugas Praktikum

Apabila diketahui data anggota tim futsal sebagai berikut:

No	Tim A		Tim B	
	Tinggi Badan (cm)	Berat Badan (kg)	Tinggi Badan (cm)	Berat Badan (kg)
1	168	50	170	66
2	170	60	167	60
3	165	56	165	59
4	168	55	166	58
5	172	60	168	58
6	170	70	175	71
7	169	66	172	68
8	165	56	171	68
9	171	72	168	65
10	166	56	169	60

1. Dengan program java, carilah data pemain diantara kedua tim tersebut:
   a. Yang mempunyai tinggi badan sama

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

   b. Yang mempunyai berat badan sama

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

   c. Rentang nilai dari tinggi badan kedua tim

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

   d. Rentang nilai dari berat badan kedua tim

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

e. Tinggi badan pada tim A yang tidak ada pada tim B

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

f. Berat badan pada tim B yang tidak ada pada tim A

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

2. Buatlah implementasi Map dalam program java berdasarkan kondisi berikut:
   a. Implementasikan Map untuk menyimpan data tim A dan tim B dalam bentuk Map terpisah.

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

b. Dari data tim B, ternyata ada kesalahan pencatatan berat badan yaitu untuk pemain yang memiliki tinggi badan 168, berat badannya adalah 66. Update data Map untuk tim B.

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

c. Implementasikan Map untuk menyimpan data tinggi badan dan berat badan dari tim A yang tinggi badannya sama dengan tim B.

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan

d. Dari data tim A yang mempunyai tinggi badan sama dengan tim B tadi, pelatih memutuskan untuk hanya mengambil pemain yang memiliki tinggi badan 168 keatas, sehingga pemain dengan tinggi badan kurang dari 168 dinyatakan dieleminasi dari tim. Update data Map tersebut sesuai dengan kondisi terakhir.

Source code

1	Tulis source code di sini pake courier new 12

Output

Penjelasan