BAB 11
COLLECTION SORTING DAN SEARCHING

Tujuan

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

Ringkasan Materi

1. Method Collection

Kelas Collection menyediakan beberapa algoritme kinerja tinggi untuk memanipulasi elemen Collection. Algoritma dalam tabel dibawah diimplementasikan sebagai method statis. Method sort, binarySearch, reverse, shuffle, fill dan copy beroperasi pada Lists. Method min, max, addAll, frekuensi dan disjoint beroperasi pada Collections.

Method	Deskripsi
Sort	Mengurutkan elemen List.
binarySearch	Menempatkan sebuah objek dalam sebuah List.
reverse	Membalikkan urutan elemen List.
shuffle	Mengurutkan elemen List secara acak.
fill	Mengatur setiap elemen List untuk merujuk ke objek tertentu.
copy	Menyalin referensi dari satu List ke List lainnya.
min	Mengembalikan elemen terkecil dalam Collection.
max	Mengembalikan elemen terbesar dalam Collection.
addAll	Menambahkan semua elemen dalam array ke Collection.
frequency	Menghitung berapa banyak elemen Collection yang sama dengan elemen tertentu.
disjoint	Menentukan apakah dua Colection tidak memiliki elemen yang sama.

2. Method Sort

Metode sort mengurutkan elemen List, dimana harus mengimplementasikan interface Comparable. Urutan ditentukan oleh urutan alami dari tipe elemen seperti yang diterapkan oleh method compareTo. Method compareTo dideklarasikan dalam interface Comparable dan kadang disebut sebagai method perbandingan alami.

Sorting dalam Urutan Ascending

Program *Sort1.java* dibawah menggunakan metode Collection sort untuk mengurutkan elemen List dalam urutan menaik (baris 17). Ingat bahwa List adalah tipe generik dan menerima satu argumen tipe yang menentukan tipe elemen List—baris 14 membuat List sebagai List of String. Baris 15 dan 20 masing-masing menggunakan panggilan implisit ke method toString dalam List untuk menampilkan konten List dalam format yang ditunjukkan dalam output.

Sort1.java

```
1  // Program Sort1.java
2  // Collections method sort.
3  import java.util.List;
4  import java.util.Arrays;
5  import java.util.Collections;
```

```
6
7   public class Sort1
8   {
9               public static void main( String[] args )
10              {
11                      String[] suits = { "Hearts", "Diamonds", "Clubs", "Spades" };
12
13                      // Create and display a list containing the suits array elements
14                      List< String > list = Arrays.asList( suits ); // create List
15                      System.out.printf( "Unsorted array elements: %s\n", list );
16
17                      Collections.sort( list ); // sort ArrayList
18
19                      // output list
20                      System.out.printf( "Sorted array elements: %s\n", list );
21              } // end main
22  } // end class Sort1
```

```
Unsorted array elements: [Hearts, Diamonds, Clubs, Spades]
Sorted array elements: [Clubs, Diamonds, Hearts, Spades]
```

Mengurutkan dalam Urutan Descending

Program *Sort2.java* dibawah mengurutkan daftar string yang sama yang digunakan pada program *Sort1.java* dalam urutan menurun. Dalam contoh program dibawah memperkenalkan interface Comparator, yang digunakan untuk mengurutkan elemen Collection dalam urutan yang berbeda. Baris 18 memanggil method Collection sort untuk mengurutkan List dalam urutan menurun. Method Collection statis reverseOrder mengembalikan objek Comparator yang mengurutkan elemen Collection dalam urutan terbalik.

Sort2.java

```
1   // Program Sort2.java
2   // Using a Comparator object with method sort.
3   import java.util.List;
4   import java.util.Arrays;
5   import java.util.Collections;
6
7   public class Sort2
8   {
9               public static void main( String[] args )
10              {
11                      String[] suits = { "Hearts", "Diamonds", "Clubs", "Spades" };
12
13                      // Create and display a list containing the suits array elements
14                      List< String > list = Arrays.asList( suits ); // create List
15                      System.out.printf( "Unsorted array elements: %s\n", list );
16
17                      // sort in descending order using a comparator
18                      Collections.sort( list, Collections.reverseOrder() );
19
20                      // output List elements
21                      System.out.printf( "Sorted list elements: %s\n", list );
22              } // end main
23  } // end class Sort2
```

```
Unsorted array elements: [Hearts, Diamonds, Clubs, Spades]
Sorted list elements: [Spades, Hearts, Diamonds, Clubs]
```

Method reverse, fill, copy, max dan min

Kelas Collection menyediakan method untuk membalikkan, mengisi, dan menyalin List. Method reverse membalikkan urutan elemen dalam List, dan metode fill meng-overwrite elemen dalam List dengan nilai tertentu. Operasi fill berguna untuk menginisialisasi ulang List. Methode copy membutuhkan dua argumen—List tujuan dan List sumber. Setiap elemen List sumber disalin ke List tujuan. List tujuan harus setidaknya sepanjang List sumber; jika tidak, akan terjadi IndexOutOfBoundsException. Jika List tujuan lebih panjang, elemen yang tidak di-overwrite tidak akan berubah.

Setiap method yang ada sejauh ini beroperasi pada List. Method min dan max masing-masing beroperasi pada seluruh jenis Collection. Method min mengembalikan elemen terkecil dalam Collection, dan method max mengembalikan elemen terbesar dalam Collection. Kedua method ini dapat dipanggil dengan objek Comparator sebagai argumen kedua untuk melakukan perbandingan kustom objek. Program *Algorithms1.java* dibawah menunjukkan method reverse, fill, copy, max dan min.

Algorithms1.java

```
1  // Program Algorithms1.java
2  // Collections methods reverse, fill, copy, max and min.
3  import java.util.List;
4  import java.util.Arrays;
5  import java.util.Collections;
6
7  public class Algorithms1
8  {
9          public static void main( String[] args )
10         {
11                 // create and display a List< Character >
12                 Character[] letters = { 'P', 'C', 'M' };
13                 List< Character > list = Arrays.asList( letters ); // get List
14                 System.out.println( "list contains: " );
15                 output( list );
16
17                 // reverse and display the List< Character >
18                 Collections.reverse( list ); // reverse order the elements
19                 System.out.println( "\nAfter calling reverse, list contains: " );
20                 output( list );
21
22                 // create copyList from an array of 3 Characters
23                 Character[] lettersCopy = new Character[ 3 ];
24                 List< Character > copyList = Arrays.asList( lettersCopy );
25
26                 // copy the contents of list into copyList
27                 Collections.copy( copyList, list );
28                 System.out.println( "\nAfter copying, copyList contains: " );
29                 output( copyList );
30
31                 // fill list with Rs
32                 Collections.fill( list, 'R' );
33                 System.out.println( "\nAfter calling fill, list contains: " );
34                 output( list );
35         } // end main
36
37         // output List information
38         private static void output( List< Character > listRef )
39         {
40                 System.out.print( "The list is: " );
41
42                 for ( Character element : listRef )
43                         System.out.printf( "%s ", element );
44
45                 System.out.printf( "\nMax: %s", );
46                 System.out.printf( " Min: %s\n", );
47         } // end method output
48 } // end class Algorithms1
```

```
list contains:
The list is: P C M
Max: P  Min: C

After calling reverse, list contains:
The list is: M C P
Max: P  Min: C

After copying, copyList contains:
The list is: M C P
Max: P  Min: C

After calling fill, list contains:
The list is: R R R
Max: R  Min: R
```

Baris 13 membuat List variabel List<Character> dan menginisialisasinya dengan tampilan List dari huruf array Character. Baris 14–15 menampilkan konten List saat ini. Baris 18 memanggil method Collection reverse untuk membalik urutan List. Method reverse membutuhkan satu argumen List. Karena List adalah tampilan List dari huruf array, elemen array sekarang dalam urutan terbalik. Isi yang telah dibalik ditampilkan pada baris 19–20. Baris 27 menggunakan method Collections copy untuk menyalin elemen list ke dalam copyList. Perubahan pada copyList tidak mengubah huruf, karena copyList adalah List terpisah yang bukan merupakan tampilan List dari huruf array. Method copy memerlukan dua argumen List—List tujuan dan List sumber. Baris 32 memanggil method Collections fill untuk menempatkan karakter 'R' di setiap elemen List. Karena List adalah tampilan List dari huruf array, operasi ini mengubah setiap elemen dalam huruf menjadi 'R'. Method fill memerlukan List untuk argumen pertama dan Objek untuk argumen kedua—dalam hal ini, Objek adalah versi boxed dari karakter 'R'. Baris 45–46 memanggil method Collections max dan min untuk menemukan elemen terbesar dan terkecil dari Collection masing-masing. Ingat bahwa interface List mengextends interface Collection, jadi List merupakan Collection.

3. Method Binary Search

Algoritme ini dibangun ke dalam Java Collections Framework sebagai method Collection statis binarySearch, yang menempatkan objek dalam List (mis., LinkedList atau ArrayList). Jika objek ditemukan, indeksnya dikembalikan. Jika objek tidak ditemukan, binarySearch mengembalikan nilai negatif. Method binarySearch menentukan nilai negatif ini dengan terlebih dahulu menghitung titik penyisipan dan membuat tandanya negatif. Kemudian, binarySearch mengurangi 1 dari titik penyisipan untuk mendapatkan nilai kembalian, yang menjamin bahwa method binarySearch mengembalikan bilangan positif (>= 0) jika dan hanya jika objek ditemukan. Jika lebih dari satu elemen dalam daftar cocok dengan kunci pencarian, tidak ada jaminan mana yang akan ditemukan lebih dulu. Program *BinarySearchTest.java* dibawah menggunakan metode binarySearch untuk mencari rangkaian string dalam ArrayList.

BinarySearchTest.java

```
1  // Program BinarySearchTest.java
2  // Collections method binarySearch.
3  import java.util.List;
4  import java.util.Arrays;
```

```
5  import java.util.Collections;
6  import java.util.ArrayList;
7
8  public class BinarySearchTest
9  {
10         public static void main( String[] args )
11         {
12                 // create an ArrayList< String > from the contents of colors array
13                 String[] colors = { "red", "white", "blue", "black", "yellow",
14                         "purple", "tan", "pink" };
15                 List< String > list =
16                         new ArrayList< String >( Arrays.asList( colors ) );
17
18                 Collections.sort( list ); // sort the ArrayList
19                 System.out.printf( "Sorted ArrayList: %s\n", list );
20
21                 // search list for various values
22                 printSearchResults( list, colors[ 3 ] ); // first item
23                 printSearchResults( list, colors[ 0 ] ); // middle item
24                 printSearchResults( list, colors[ 7 ] ); // last item
25                 printSearchResults( list, "aqua" ); // below lowest
26                 printSearchResults( list, "gray" ); // does not exist
27                 printSearchResults( list, "teal" ); // does not exist
28         } // end main
29
30         // perform search and display result
31         private static void printSearchResults(
32                 List< String > list, String key )
33         {
34                 int result = 0;
35
36                 System.out.printf( "\nSearching for: %s\n", key );
37                 result = Collections.binarySearch( list, key );
38
39                 if(result>=0)
40                         System.out.printf( "Found at index %d\n", result );
41                 else
42                         System.out.printf( "Not Found (%d)\n",result );
43         } // end method printSearchResults
44 } // end class BinarySearchTest
```

```
Sorted ArrayList: [black, blue, pink, purple, red, tan, white, yellow]

Searching for: black
Found at index 0

Searching for: red
Found at index 4

Searching for: pink
Found at index 2

Searching for: aqua
Not Found (-1)

Searching for: gray
Not Found (-3)

Searching for: teal
Not Found (-7)
```

Baris 15–16 menginisialisasi List dengan ArrayList yang berisi salinan elemen dalam array warna. Method collection binarySearch mengharapkan elemen argumen List nya diurutkan dalam urutan menaik, jadi baris 18 menggunakan pengurutan method collection untuk mengurutkan list. Jika elemen argumen List tidak diurutkan, hasil dari penggunaan binarySearch tidak dapat ditentukan. Baris 19 menampilkan list yang diurutkan. Baris 22–27 memanggil method printSearchResults (baris 31–43) untuk

melakukan pencarian dan menampilkan hasilnya. Baris 37 memanggil method Collections binarySearch untuk melakukan pencarian dalam list terhadap kunci yang ditentukan. Metode binarySearch mengambil List sebagai argumen pertama dan Objek sebagai argumen kedua. Baris 39–42 menampilkan hasil pencarian. Method binarySearch yang dioverload menggunakan objek Comparator sebagai argumen ketiganya, yang menentukan bagaimana binarySearch harus membandingkan kunci pencarian dengan elemen List.

Method addAll, frequency dan disjoint

Class Collections juga menyediakan method addAll, frequency dan disjoint. Method Collection addAll membutuhkan dua argumen—Collection untuk menyisipkan elemen baru dan array yang menyediakan elemen untuk disisipkan. Method Collection frequency membutuhkan dua argumen—Collection yang akan dicari dan Objek yang akan dicari dalam collection. Method frequency mengembalikan berapa kali argumen kedua muncul dalam koleksi. Method Collection disjoint mengambil dua Collection dan mengembalikan true jika mereka tidak memiliki elemen yang sama. Program *Algorithms2.java* dibawah menunjukkan penggunaan method addAll, frequency dan disjoint.

Algorithms2.java

```
1  // Fig. 20.13: Algorithms2.java
2  // Collections methods addAll, frequency and disjoint.
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.Arrays;
6  import java.util.Collections;
7
8  public class Algorithms2
9  {
10         public static void main( String[] args )
11         {
12                 // initialize list1 and list2
13                 String[] colors = { "red", "white", "yellow", "blue" };
14                 List< String > list1 = Arrays.asList( colors );
15                 ArrayList< String > list2 = new ArrayList< String >();
16
17                 list2.add( "black" ); // add "black" to the end of list2
18                 list2.add( "red" ); // add "red" to the end of list2
19                 list2.add( "green" ); // add "green" to the end of list2
20
21                 System.out.print( "Before addAll, list2 contains: " );
22
23                 // display elements in list2
24                 for(Strings:list2)
25                         System.out.printf( "%s ", s );
26
27                 Collections.addAll( list2, colors ); // add colors Strings to list2
28
29                 System.out.print( "\nAfter addAll, list2 contains: " );
30
31                 // display elements in list2
32                 for(Strings:list2)
33                         System.out.printf( "%s ", s );
34
35                 // get frequency of "red"
36                 int frequency = Collections.frequency( list2, "red" );
37                 System.out.printf(
38                         "\nFrequency of red in list2: %d\n", frequency );
39
40                 // check whether list1 and list2 have elements in common
41                 boolean disjoint = Collections.disjoint( list1, list2 );
42
43                 System.out.printf( "list1 and list2 %s elements in common\n",
```

```
44                                  ( disjoint ? "do not have" : "have" ) );
45              } // end main
46  } // end class Algorithms2
```

```
Before addAll, list2 contains: black red green
After addAll, list2 contains: black red green red white yellow blue
Frequency of red in list2: 2
list1 and list2 have elements in common
```

Baris 14 menginisialisasi list1 dengan elemen dalam array warna, dan baris 17–19 menambahkan String "hitam", "merah" dan "hijau" ke list2. Baris 27 memanggil method addAll untuk menambahkan elemen dalam array warna ke list2. Baris 36 mendapatkan frekuensi kemunculan String "merah" di list2 menggunakan method frequency. Baris 41 memanggil method disjoint untuk menguji apakah Collection list1 dan list2 memiliki elemen yang sama.

Pelaksanaan Percobaan

A. Sort

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

SetExample.java

```
1     // Java program to demonstrate working of Comparator
2     // interface and Collections.sort() to sort according
3     // to user defined criteria.
4     import java.util.*;
5     import java.lang.*;
6     import java.io.*;
7
8     // A class to represent a student.
9     class Student
10    {
11       int rollno;
12       String name, address;
13
14       // Constructor
15       public Student(int rollno, String name,
16                              String address)
17       {
18          this.rollno = rollno;
19          this.name = name;
20          this.address = address;
21       }
22
23       // Used to print student details in main()
24       public String toString()
25       {
26          return this.rollno + " " + this.name +
27                          " " + this.address;
```

```
28          }
29      }
30
31      class Sortbyroll implements Comparator<Student>
32      {
33          // Used for sorting in ascending order of
34          // roll number
35          public int compare(Student a, Student b)
36          {
37              return a.rollno - b.rollno;
38          }
39      }
40
41      // Driver class
42      class Main
43      {
44          public static void main (String[] args)
45          {
46              ArrayList<Student> ar = new ArrayList<Student>();
47              ar.add(new Student(111, "bbbb", "london"));
48              ar.add(new Student(131, "aaaa", "nyc"));
49              ar.add(new Student(121, "cccc", "jaipur"));
50
51              System.out.println("Unsorted");
52              for (int i=0; i<ar.size(); i++)
53                  System.out.println(ar.get(i));
54
55              Collections.sort(ar, new Sortbyroll());
56
57              System.out.println("\nSorted by rollno");
58              for (int i=0; i<ar.size(); i++)
59                  System.out.println(ar.get(i));
60          }
61  }
```

B. Searching

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

MapExample.java

```
1       // Java Program to Demonstrate Working of binarySearch()
2       // method of Collections class
3
4       // Importing required classes
5       import java.util.ArrayList;
6       import java.util.Collections;
7       import java.util.List;
8
```

```
9     // Main class
10    public class GFG {
11        // Main driver method
12        public static void main(String[] args)
13        {
14                // Creating an empty ArrayList of integer type
15                List<Integer> al = new ArrayList<Integer>();
16
17                // Populating the Arraylist
18                al.add(1);
19                al.add(2);
20                al.add(3);
21                al.add(10);
22                al.add(20);
23
24                // 10 is present at index 3
25                int key = 10;
26                int res = Collections.binarySearch(al, key);
27
28                if (res >= 0)
29                        System.out.println(
30                                key + " found at index = " + res);
31                else
32                        System.out.println(key + " Not found");
33
34                key = 15;
35                res = Collections.binarySearch(al, key);
36
37                if (res >= 0)
38                        System.out.println(
39                                key + " found at index = " + res);
40                else
41                        System.out.println(key + " Not found");
42        }
43    }
```

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, urutkan data pemain diantara kedua tim tersebut:
   a. Berdasarkan Tinggi Badannya secara Ascending/menaik dan Descending/menurun
   b. Berdasarkan Berat Badannya secara Ascending/menaik dan Descending/menurun
   c. Cari nilai maksimum dan minimum Tinggi Badan dan Berat Badan untuk pemain dari masing-masing tim.
   d. Copy seluruh anggota Tim B ke Tim C yang baru dibentuk

2. Buatlah implementasi Binary Search dalam program java berdasarkan kondisi berikut:
   a) Implementasikan ArrayList untuk menyimpan data tim A dan tim B dalam bentuk ArrayList terpisah.
   b) Dari data tim B, dicari jumlah pemain yang mempunyai tinggi badan 168 cm dan 160 cm.
   c) Dari data tim A, dicari jumlah pemain yang mempunyai berat badan 56 kg dan 53 kg.
   d) Ingin diketahui apakah pemain di Tim A ada yang mempunyai tinggi badan atau berat badan yang sama dengan pemain di Tim B?