MODUL 11
REKURSIF : definisi, cara kerja

CAPAIAN PEMBELAJARAN

Mahasiswa dapat memahami dan menyelesaikan kasus dengan rekursif.

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad
2. JDK

DASAR TEORI

Program-program yang telah kita bahas sejauh ini umumnya disusun sebagai method yang memanggil satu sama lain secara hierarkis. Untuk beberapa masalah, ada baiknya memanggil method itu sendiri. Method yang dikenal sebagai method rekursif. Method rekursif dapat memanggil dirinya baik secara langsung atau tidak langsung melalui method lain.

Konsep rekursif.

Pendekatan pemecahan masalah rekursif memiliki sejumlah elemen yang sama. Ketika method rekursif dipanggil untuk memecahkan masalah, sebenarnya hanya mampu menyelesaikan kasus yang paling sederhana, atau kasus dasar. Jika method ini disebut dengan kasus dasar, method itu mengembalikan hasil. Jika method ini disebut dengan masalah yang lebih kompleks, method tersebut biasanya membagi masalah menjadi dua bagian konseptual — bagian yang diketahui cara melakukannya dan bagian yang tidak diketahui bagaimana melakukannya. Untuk membuat rekursi menjadi layak, bagian yang terakhir

harus menyerupai masalah aslinya, tetapi versi yang sedikit lebih sederhana atau lebih kecil. Karena masalah baru ini terlihat seperti masalah asli, method ini memanggil salinan baru dari dirinya untuk bekerja pada masalah yang lebih kecil — ini disebut sebagai panggilan rekursif dan juga disebut langkah rekursi. Langkah rekursi biasanya mencakup pernyataan pengembalian, karena hasilnya akan digabungkan dengan bagian dari masalah yang diketahui method untuk menyelesaikan untuk membentuk hasil yang akan diteruskan kembali ke penelepon asli. Langkah rekursi dijalankan sementara pemanggilan method asli masih aktif (mis., Belum selesai dieksekusi). Ini dapat menghasilkan lebih banyak panggilan rekursif karena method ini membagi setiap sub-masalah baru menjadi dua bagian konseptual. Agar rekursi akhirnya berakhir, setiap kali method memanggil dirinya sendiri dengan versi yang lebih sederhana dari masalah aslinya, urutan masalah yang lebih kecil dan lebih kecil harus menyatu pada kasus dasar. Ketika method mengenali kasus dasar, itu mengembalikan hasil ke salinan method sebelumnya. Urutan pengembalian terjadi sampai pemanggilan method asli mengembalikan hasil akhir ke pemanggil.

Method rekursif dapat memanggil method lain, yang pada gilirannya membuat panggilan kembali ke method rekursif. Ini dikenal sebagai panggilan rekursif tidak langsung atau rekursi tidak langsung. Misalnya, method A memanggil method B, yang membuat panggilan kembali ke method A. Ini masih rekursi, karena panggilan kedua ke method A dilakukan saat panggilan pertama ke method A aktif — yaitu panggilan pertama ke method A belum selesai dieksekusi (karena sedang menunggu method B untuk mengembalikan hasilnya) dan belum kembali ke pemanggil asli method A. Untuk lebih memahami konsep rekursi, mari kita lihat contoh yang cukup akrab bagi pengguna komputer — definisi rekursif direktori pada komputer. Komputer biasanya menyimpan file terkait dalam direktori. Direktori dapat kosong, dapat berisi file dan / atau dapat berisi direktori lain (biasanya disebut sebagai subdirektori). Masing-masing subdirektori ini, pada gilirannya, dapat juga berisi file dan direktori. Jika kita ingin membuat daftar setiap file dalam direktori (termasuk semua file dalam subdirektori direktori), kita perlu membuat method yang pertama-tama mendaftar file direktori awal, kemudian membuat panggilan rekursif untuk membuat daftar file di masing-masing subdirektori direktori itu. Kasus dasar terjadi ketika direktori tercapai yang tidak mengandung subdirektori. Pada titik ini, semua file dalam direktori asli telah terdaftar dan tidak perlu rekursi lebih lanjut. (Sumber : Deitel, Deitel edisi 9)

Dalam pemrograman, rekursif dapat diimplementasikan pada method. Syarat supaya method rekursi dapat terjadi adalah harus ada kondisi dimana pemanggilan terhadap method itu sendiri berakhir.

Contoh rekursif : Faktorial.

Kita sudah pernah membuat sebuah algoritma dan program untuk menghitung faktorial dengan iterasi. Sekarang, kita akan membuatnya dengan rekursif.

Kita ingat lagi :

Faktorial N dengan N=5 artinya

N! = 5 * 4 * 3 * 2 * 1

Dan faktorial 1 = 1.

Dihitung dengan iterasi dan pernyataan for menjadi sebagai berikut.

```
faktorial = 1;

for (int counter = N; counter>=1; counter--)

        faktorial = faktorial * counter;
```

Sekarang kita akan menghitung faktorial secara rekursif. Rumusan umum method faktorial secara rekursif adalah sebagai berikut.

N! = N * (N-1)!

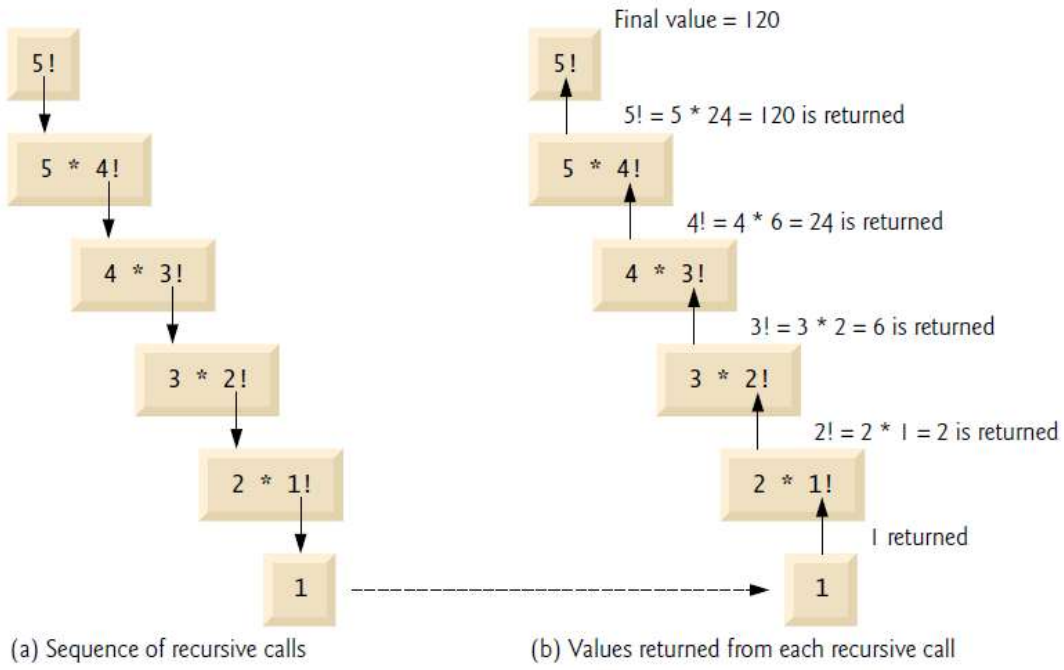
Sebagai contoh, untuk menghitung faktorial dengan N=5.

5! = 5 * 4 * 3 * 2 * 1

5! = 5 * (4 * 3 * 2 * 1)

5! = 5 * 4!

Evaluasi untuk 5! Ditunjukkan pada Gambar 1. Gambar 1 menunjukkan bagaimana urutan pemanggilan rekursif sampai kepada pemanggilan faktorial 1 yang merupakan nilai akhir dengan hasil 1.



(a) Sequence of recursive calls (b) Values returned from each recursive call

Gambar 1. Evaluasi Faktorial secara Rekursif dengan N=5.

(sumber Deitel, Deitel edisi 9 Gambar 18.2)

Algoritma tersebut dapat kita implementasikan ke dalam program menjadis ebagai berikut.

```
public class FaktorialRekursif{
    //method rekursif
    public static long faktorial( long N ){
        if ( N <= 1 ) // kondisi terpenuhi untuk akhir rekursif
            return 1; // nilai akhir: 0! = 1 dan 1! = 1
        else // step rekursif, ada pemanggilan kembali method 'faktorial'
            return N * faktorial( N - 1 );
    }

    public static void main( String[] args ){
        // pemanggilan awal method faktorial
        System.out.println( "Faktorial 5 = "+ faktorial(5));
    }
 }
```

PRAKTIK

Praktik 1. Program faktorial dengan rekursif

```
public class FaktorialRekursif{
   //method rekursif
   public static long faktorial( long N ){
         if ( N <= 1 ) // kondisi terpenuhi untuk akhir rekursif
               return 1; // nilai akhir: 0! = 1 dan 1! = 1
         else // step rekursif, ada pemanggilan kembali method 'faktorial'
               return N * faktorial( N - 1 );
   }

   public static void main( String[] args ){
         // pemanggilan awal method faktorial
         System.out.println( "Faktorial 5 = "+ faktorial(5));
   }
 }
```

Praktik 2. Program fibonacci dengan rekursif

```
public class FibonacciRekursif{

   //method rekursif

   public static long fibonacci( long N ){

          if (( N == 0 ) || ( N == 1 )) // kondisi terpenuhi untuk akhir rekursif

                 return N; // nilai akhir : N=0 atau N=1

          else // step rekursif, ada pemanggilan kembali method 'fibonacci'

                 return fibonacci(N-1) + fibonacci(N-2);

   }


   public static void main( String[] args ){

          System.out.println("Deret fibonacci dengan N = 3");

          for (int i=0; i<=3;i++)

                 // pemanggilan awal method fibonacci

                 if (i<3)

                        System.out.print(fibonacci(i)+", ");

                 else

                        System.out.println(fibonacci(i));

   }

 }
```

Praktik 3. Modifikasi method faktorial diatas menjadi menampilkan semua nilai hasil faktorial dari bilangan-bilangan sebelumnya. Misalnya jika dari method main dipanggil untuk menghasilkan faktorial dari 5 maka akan ditampilkan faktorial 1, faktorial 2, faktorial 3, faktorial 4 dan terakhir faktorial 5.

LATIHAN

1. Buat program dengan method rekursif untuk menghitung nilai

   A^B dimana A dan B bilangan bulat yang lebih besar atau sama dengan 1.

   Deklarasi methodnya ada sebagai berikut

   <*acces modifier, tipe, etc yg diperlukan*> pangkat(A,B)

2. Greatest Common Divisor (gcd) dari bilangan bulat x dan y adalah bilangan bulat terbesar yang terbagi rata menjadi x dan y. Buat algoritma dan program dengan method rekursif gcd yang mengembalikan reatest Common Divisor x dan y. Gcd x dan y didefinisikan secara rekursif sebagai berikut: Jika y sama dengan 0, maka gcd (x, y) adalah x; jika tidak, gcd (x, y) adalah gcd (y, x% y), di mana% adalah operator sisanya.

TUGAS

1. Palindrome adalah string yang dieja dengan cara yang sama maju dan mundur. Beberapa contoh palindrom adalah "radar," "makam". Tulis method rekursif testPalindrome yang mengembalikan nilai boolean true jika string yang disimpan dalam array adalah palindrom dan false jika tidak. Metode ini harus mengabaikan spasi dan tanda baca dalam string.
2. Soal diberikan oleh dosen.

REFERENSI

1. Deitel H, Deitel P, 2012, Java How to Program, 9th edition, Prentice Hall.

MODUL 12
REKURSIF vs ITERASI

CAPAIAN PEMBELAJARAN

Mahasiswa dapat membedakan rekursif dan iterasi dan mengubah rekursif menjadi iterasi.

KEBUTUHAN ALAT/BAHAN/SOFTWARE

3. TextPad
4. JDK

DASAR TEORI

Rekursif dan Iterasi.

Kita telah mempelajari metode faktorial dan Fibonacci, yang dapat dengan mudah diimplementasikan baik secara rekursif atau iteratif. Sekarang kita akan membandingkan dua pendekatan dan mendiskusikan mengapa dapat memilih satu pendekatan dari yang lain dalam situasi tertentu. Baik iterasi dan rekursi didasarkan pada pernyataan kontrol: Iterasi menggunakan pernyataan pengulangan (mis., for, while atau do ... while), sedangkan rekursi menggunakan pernyataan seleksi (mis., if, if ... else). Baik iterasi dan rekursi melibatkan pengulangan: Iterasi secara eksplisit menggunakan pernyataan pengulangan, sedangkan rekursi mencapai pengulangan melalui pemanggilan method berulang. Iterasi dan rekursi masing-masing melibatkan tes terminasi: Iterasi berakhir ketika kondisi loop bernilai false, sedangkan rekursi berakhir ketika case dasar tercapai. Iterasi dengan pengulangan dan rekursi yang

dikontrol secara bertahap mendekati penghentian: Iterasi terus memodifikasi penghitung hingga penghitung mengasumsikan nilai yang membuat kondisi loop bernilai false, sedangkan rekursi terus menghasilkan versi yang lebih kecil dari masalah awal hingga case dasar tercapai. Baik iterasi dan rekursi dapat terjadi hingga tak terhingga: Loop tak terbatas terjadi dengan iterasi jika uji kelanjutan loop tidak pernah menjadi salah, sedangkan rekursi tak terbatas terjadi jika langkah rekursi tidak mengurangi masalah setiap kali dengan cara yang menyatu pada case dasar, atau jika case dasar tidak diuji.

Untuk mengilustrasikan perbedaan antara iterasi dan rekursi, mari kita periksa solusi berulang untuk masalah faktorial. Pernyataan pengulangan digunakan (baris 2-3) dibandingkan dengan pernyataan pemilihan solusi rekursif (baris 1-6). Kedua solusi menggunakan tes terminasi. Dalam solusi rekursif, baris 3 menguji untuk kasus dasar. Dalam solusi berulang, baris 2 menguji kondisi kelanjutan loop — jika tes gagal, loop berakhir. Akhirnya, alih-alih menghasilkan versi yang lebih kecil dari masalah aslinya, solusi iteratif menggunakan penghitung yang dimodifikasi sampai kondisi perulangan menjadi salah.

No.	Iterasi	Rekursif
1	faktorial = 1;	public static long faktorial(long N){
2	for(int faktor=2; faktor<=counter; faktor++)	if (N <= 1)
3	faktorial *= faktor;	return 1;
4	System.out.println("Nilai " + counter + "!" +	else
5	" adalah " + faktorial);	return N * faktorial(N - 1);
6		}

Rekursi memiliki banyak hal negatif. Berulang kali memanggil mekanisme, dan akibatnya overhead, panggilan metode. Pengulangan ini bisa mahal dalam hal waktu prosesor dan ruang memori. Setiap panggilan rekursif menyebabkan salinan metode lain (sebenarnya, hanya variabel metode, yang disimpan dalam catatan aktivasi) yang akan dibuat — set salinan ini dapat menggunakan ruang memori yang cukup besar. Karena iterasi terjadi dalam suatu metode, panggilan metode berulang dan penugasan memori tambahan dihindari.

Setiap masalah yang dapat diselesaikan secara rekursif juga dapat diselesaikan secara iteratif (non-rekursif). Pendekatan rekursif biasanya lebih disukai daripada pendekatan iteratif ketika pendekatan rekursif lebih mencerminkan masalah secara alami dan menghasilkan program yang lebih mudah dipahami dan didebug. Pendekatan rekursif seringkali dapat diimplementasikan dengan lebih sedikit baris kode. Alasan lain untuk memilih pendekatan rekursif adalah bahwa pendekatan iteratif mungkin tidak memungkinkan.

PRAKTIK

Praktik 1. Program faktorial dengan rekursif dan iterasi. Buat program faktorial dengan rekursif (ihat modul 11) dan kemudian ubah dengan iterasi (lihat pada bagian teori).

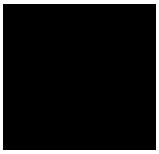
```
public class FaktorialIterasi {
public static void main(String[] args) {
   int batas = 5;
   int counter = 0;
   int faktorial = 1;
   for(counter=1; counter<=batas; counter++)
      faktorial *= counter;
     System.out.println("Nilai " + batas + "!" + " adalah " + faktorial);
   }
}
```

Praktik 2. Program fibonacci dengan rekursif dan iterasi. . Buat program faktorial dengan rekursif (ihat modul 11) dan kemudian ubah dengan iterasi.

```
public class FibonacciIterasi{

        public static void main( String[] args ){
                System.out.println("Deret fibonacci dengan N = 5");
                int fibn;
                int fibn1 = 1;
                int fibn2 = 0;
                for (int i=0; i<=5;i++)
                        if (i==0 || i==1)
                                System.out.print(i+ " ");

                        else {
                                fibn = fibn1+fibn2;
                                fibn2 = fibn1;
                                fibn1 = fibn;
                                System.out.print(fibn+" ");
                        }
        }
}
```

Praktik 3. Lihat kembali praktik 3 dari modul 11. Ubah ke dalam bentuk iterasi.

Jelaskan ketiga program diatas dan bandingkan antara rekursif dan iterasi.



LATIHAN

1. Buat algoritma dan program yang digunakan untuk menampilkan semua elemen dalam array, tiap elemen bertipe integer.
    a. Dengan method rekursif printArray
    b. Dengan iterasi
2. Buat program yang mengambil array karakter yang berisi string sebagai argumen dan mencetak string dari belakang. Gunakan metode String toCharArray, yang tidak memerlukan argumen, untuk mendapatkan array char yang berisi karakter dalam String. Masukan adalah sebuah string.
    a. dengan metode recursive stringReverse
    b. dengan iterasi
3. Buat program untuk menentukan elemen terkecil dalam array bilangan bulat.
    a. dengan metode recursive recursiveMinimum. Methode harus kembali ketika menerima array dari satu elemen.
    b. dengan iterasi

TUGAS

Kerjakan soal berikut, jika memungkinkan menggunakan iterasi, jika tidak, jelaskan alasannya.

3. Buat program dengan iterasi untuk menghitung nilai

    A^B dimana A dan B bilangan bulat yang lebih besar atau sama dengan 1.

    Deklarasi methodnya ada sebagai berikut

    <*acces modifier, tipe, etc yg diperlukan*> pangkat(A,B)

4. Palindrome adalah string yang dieja dengan cara yang sama maju dan mundur. Beberapa contoh palindrom adalah "radar," "makam". Tulis program dengan iterasi yang akan menghasilkan keluaran true jika string yang disimpan dalam array adalah

palindrom dan false jika tidak. Program ini harus mengabaikan spasi dan tanda baca dalam string.

5. Soal diberikan oleh dosen.

REFERENSI

2. Deitel H, Deitel P, 2012, Java How to Program, 9th edition, Prentice Hall.

MODUL 13
SORTING

CAPAIAN PEMBELAJARAN

Mahasiswa dapat mengurutkan data dengan metode bubble sort, selection sort dan mengimplementasikannya dalam program.

KEBUTUHAN ALAT/BAHAN/SOFTWARE

5. TextPad
6. JDK

DASAR TEORI

SORTING (PENGURUTAN)

Dalam banyak aplikasi, pengurutan menjadi algoritma yang sering banyak digunakan. Kalau kita berhubungan dengan data yang jumlahnya besar, maka data tersebut akan mudah kita kelola kalau dalam keadaan terurut dengan suatu kunci pengurutan tertentu. Dengan data yang sudah terurut, kita akan dengan mudah mencari data, mengelompokkan data, dan lain-lain. Ada banyak metode pengurutan. Beberapa diantaranya adalah sebagai berikut.

a. BUBBLE SORT

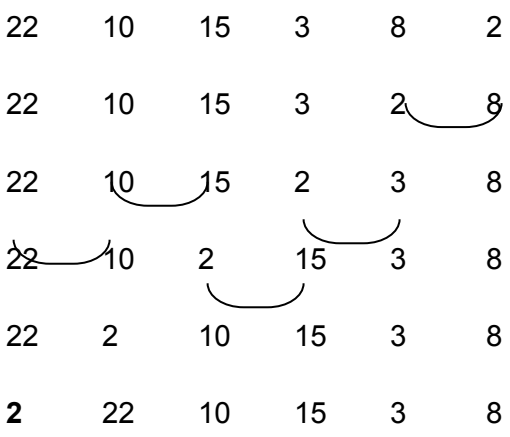
Metode Sorting Bubble sort

adalah pengurutan dengan membandingkan elemen berikutnya jika elemen sekarang lebih besar dari elemen berikutnya maka elemen tersebut ditukar.

Contoh :

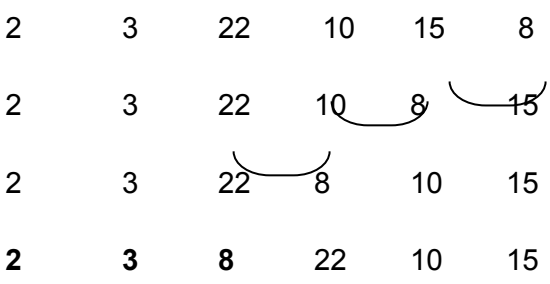
Data : 22, 10, 15, 3, 8, 2

Langkah 1:

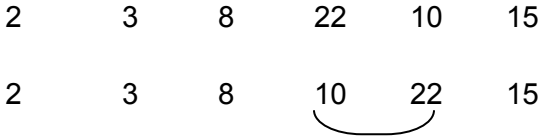


Langkah 2:

Langkah 3:



Langkah 4:



2 **3** **8** **10** 22 15

Langkah 5:

2 3 8 10 ~~22~~ ~~15~~

2 **3** **8** **10** **15** 22

Hasil pengurutannya adalah : 2, 3, 8, 10, 15, 22

b. SELECTION SORT

Metode pengurutan Selection Sort adalah

Pengurutan dengan mencari dari elemen yang berikutanya sampai dengan elemen terakhir jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka elemen yang bersangkutan akan ditukar. Oleh karena itu setiap langkah selalu memilih satu baris dari baris berikutnya.

Contoh :

Data : 22, 10, 15, 3, 8, 2

Penyelesaian :

Langkah 1 :		**Langkah 2:**	
22, 10, 15, 3, 8, 2		2,10,15,3,8,22	
Pembanding ||	Posisi data terkecil	Pembanding ||	Posisi data terkecil
22 > 10	2	10 < 15	2
10 < 15	2	15 > 3	4

15 > 3	4	3 < 8	4
3 < 8	4	8 < 22	5
8 > 2	6		
Posisi ke1=22, ditukar posisi ke-6		Posisi ke2=10, ditukar posisi ke-4	
Hasil : **2**, 10, 15, 3, 8, 22		Hasil : **2, 3**, 15, 10, 8, 22	

Langkah 3:

2, 3, 15, 10, 8, 22

Pembanding || posisi data terkecil		keterangan :
15 > 10	4	posisi ke-3 =15, tukar posisi ke-4
10 > 8	5	hasil : **2, 3, 8**, 10, 15, 22
8 < 22	5	

Langkah 4:

2, 3, 8, 10, 15, 22

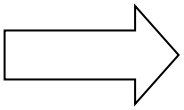
Pembanding || posisi data terkecil		keterangan :
10 < 15	5	tidak ada pertukaran data
10 < 22	5	hasil : **2, 3, 8**, **10**, 15, 22

Langkah 5:

2, 3, 8, 10, 15, 22

Pembanding || posisi data terkecil

15 < 22 5



keterangan :

tidak ada pertukaran data

hasil : **2, 3, 8**, **10**, **15**, 22

Hasil pengurutannya : 2, 3, 8, 10, 15, 22

PRAKTIK

Praktik 1. Program bubble sort.

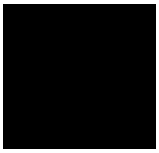
```
import java.util.Scanner;

public class BubbleSort{

public void bubbleSort(float larik2[])
{
      for (int i=0;i<larik2.length;i++)
      {
            for (int elemen=0;elemen<larik2.length-1;elemen++)
            {
                  if (larik2[elemen]>larik2[elemen+1])
                        tukar(larik2, elemen,elemen+1);
            }
      }
}

public void tukar(float larik3[], int satu, int dua)
{
      float temp;
      temp = larik3[satu];
      larik3[satu] = larik3[dua];
      larik3[dua] = temp;
}
```

```
public static void main(String args[]){
      Scanner masuk = new Scanner(System.in);
      BubbleSort lrk = new BubbleSort();
      float nilai[]= new float[5];
      System.out.println("Masukan 5 buat data nilai");
      for (int i = 0; i < 5; i++)
      {
            System.out.print( (i + 1 )+" : ");
            nilai[i]=masuk.nextFloat();
      }
      System.out.println("Data nilai yang dimasukan");
      for (int i = 0; i < 5; i++)
            System.out.println(nilai[i]);
      System.out.println("Data hasil pengurutan    ");
      lrk.bubbleSort(nilai);
      for (int i = 0; i < 5; i++)
            System.out.println(nilai[i]);

}
}
```

Praktik 2. Program Selection sort.

```
import java.util.Scanner;

public class SelectionSort{

public void selectionSort(float larik2[])
{
      for (int i=0;i<larik2.length;i++)
      {
            int min =i;
            for (int elemen=i+1;elemen<larik2.length;elemen++)
            {
                  if (larik2[min]>larik2[elemen])
                        min = elemen;
            }
            tukar(larik2, min,i);
      }
}

public void tukar(float larik3[], int satu, int dua)
{
      float temp;
      temp = larik3[satu];
      larik3[satu] = larik3[dua];
      larik3[dua] = temp;
}
```

```
public static void main(String args[]){
      Scanner masuk = new Scanner(System.in);
      SelectionSort lrk = new SelectionSort();
      float nilai[]= new float[5];
      System.out.println("Masukan 5 buat data nilai");
      for (int i = 0; i < 5; i++)
      {
            System.out.print( (i + 1 )+" : ");
            nilai[i]=masuk.nextFloat();
      }
      System.out.println("Data nilai yang dimasukan");
      for (int i = 0; i < 5; i++)
            System.out.println(nilai[i]);
      System.out.println("Data hasil pengurutan    ");
      lrk.selectionSort(nilai);
      for (int i = 0; i < 5; i++)
            System.out.println(nilai[i]);

}
}
```



LATIHAN

Kita bisa melakukan sorting dengan menggunakan kelas Collection. Jalankan program berikut, dan jelaskan.

4. Sorting ascending

```
// Collections method sort.
import java.util.List;
import java.util.Arrays;
import java.util.Collections;

public class Sort1
{
  public static void main( String[] args )
  {
    String[] suits = { "Hearts", "Diamonds", "Clubs", "Spades" };

    // Create and display a list containing the suits array elements
    List< String > list = Arrays.asList( suits ); // create List
    System.out.printf( "Unsorted array elements: %s\n", list );
```

```
    Collections.sort( list ); // sort ArrayList

    // output list
    System.out.printf( "Sorted array elements: %s\n", list );
  } // end main
} // end class Sort1
```

5. Sorting descending

```
// Using a Comparator object with method sort.
import java.util.List;
import java.util.Arrays;
import java.util.Collections;

public class Sort2
{
  public static void main( String[] args )
  {
    String[] suits = { "Hearts", "Diamonds", "Clubs", "Spades" };

    // Create and display a list containing the suits array elements
    List< String > list = Arrays.asList( suits ); // create List
    System.out.printf( "Unsorted array elements: %s\n", list );

    // sort in descending order using a comparator
    Collections.sort( list, Collections.reverseOrder() );

    // output List elements
    System.out.printf( "Sorted list elements: %s\n", list );
  } // end main
} // end class Sort2
```

TUGAS

6. Jelaskan tentang metode sorting Insertion sort, Merge sort dan Quick sort.
7. Soal diberikan oleh dosen.

REFERENSI

3. Deitel H, Deitel P, 2012, Java How to Program, 9th edition, Prentice Hall.
4. Harnaningrum, 2009, Algoritma dan Pemrograman dengan Java, Ghara Ilmu

MODUL 14
SEARCHING

CAPAIAN PEMBELAJARAN

Mahasiswa dapat melakukan pencarian data dengan metode linear search dan mengimplementasikannya dalam program.

KEBUTUHAN ALAT/BAHAN/SOFTWARE

7. TextPad
8. JDK

DASAR TEORI

SEARCHING (PENCARIAN)

Searching (pencarian) adalah aplikasi komputer yang sangat penting. Dalam pencarian, hal yang paling penting adalah adanya kunci pencarian. Searching adalah algoritma untuk mencari data yang berada pada suatu kumpulan data tertentu. Pencarian terhadap data, bisa merupakan data yang sudah terurut maupun yang belum terurut.

Pencarian Linear

Pencarian yang membandingkan data kunci dengan seluruh data, mulai dari data pertama kasus terburuk adalah data yang dicari ditemukan pada deret terakhir.

Contoh :

Larik ke	:	0	1	2	3	4	5
Data	:	10	4	14	7	5	23

Akan dicari data 14. Langkah yang ditempuh adalah mulai dari data ke-0 dibandingkan dengan data yang dicari

Larik ke-0 : 10 tidak sama dengan 14, pencarian bergeser ke larik ke-1

Larik ke-1 : 4 tidak sama dengan 14, bergeser lagi ke larik ke-2

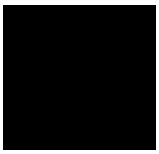
Larik ke-2 : 14 sama dengan 14, pencarian berakhir karena data ditemukan.

PRAKTIK

Praktik 1. Program pencarian linear.

```
import java.util.Scanner;
public class Larik9
{
      public static void main(String args[])
      {
            Scanner input = new Scanner(System.in);
            int i, x;
            boolean ketemu;
            int data[] = {12, 20, 14, 9, 34};
            System.out.print("Masukan Data yang dicari = ");
            x = input.nextInt();
```

```
            ketemu = false;
            for(i = 0; i < 5; i++)
            {
            if (data[i] ==x)
              {
               ketemu = ! ketemu;
               break;
               }
            }
            if (ketemu)
              System.out.println("Data tersebut ada pada posisi ke-"
              +(i+1));
            else
              System.out.println("Data tidak ketemu !");
      }
}
```



LATIHAN

Ubah program pencarian Linear dengan menggunakan metode rekursif linearSearch untuk melakukan pencarian linear array. Method harus menerima kunci pencarian dan indeks awal sebagai argumen. Jika kunci pencarian ditemukan, kembalikan indeksnya dalam array; jika tidak, kembalikan –1. Setiap panggilan ke method rekursif harus memeriksa satu indeks dalam array.

```
import java.util.Scanner;
public class LinearSearchR
{
     static int data[]={12, 20, 14, 9, 34};

     public static int linearSearch (int kunci, int indeksAwal)
     {
        if(kunci == data[indeksAwal])
          return indeksAwal;
        if(indeksAwal + 1 < data.length)
          return linearSearch(kunci, indeksAwal + 1);
        return  -1;
     }


     public static void main(String args[])
     {
               Scanner input = new Scanner(System.in);
               int i, x;
               System.out.print("Masukan Data yang dicari = ");
```

```
            x = input.nextInt();
            i = linearSearch(x,0);
            if (i!=-1)
              System.out.println("Data tersebut ada pada posisi ke-"+i);
            else
              System.out.println("Data tidak ketemu !");
    }
}
```

TUGAS

8. Jelaskan tentang metode pencarian biner, kemudian coba jalankan program berikut

```
// Collections method binarySearch.
import java.util.List;
import java.util.Arrays;
import java.util.Collections;
import java.util.ArrayList;

public class BinarySearchTest
{
  public static void main( String[] args )
  {
    // create an ArrayList< String > from the contents of colors array
    String[] colors = { "red", "white", "blue", "black", "yellow",
      "purple", "tan", "pink" };
    List< String > list =
      new ArrayList< String >( Arrays.asList( colors ) );

    Collections.sort( list ); // sort the ArrayList
    System.out.printf( "Sorted ArrayList: %s\n", list );

    // search list for various values
    printSearchResults( list, colors[ 3 ] ); // first item
    printSearchResults( list, colors[ 0 ] ); // middle item
    printSearchResults( list, colors[ 7 ] ); // last item
    printSearchResults( list, "aqua" ); // below lowest
    printSearchResults( list, "gray" ); // does not exist
    printSearchResults( list, "teal" ); // does not exist
```

```
  } // end main

  // perform search and display result
  private static void printSearchResults(
    List< String > list, String key )
  {
    int result = 0;

    System.out.printf( "\nSearching for: %s\n", key );
    result = Collections.binarySearch( list, key );

    if ( result >= 0 )
      System.out.printf( "Found at index %d\n", result );
    else
      System.out.printf( "Not Found (%d)\n",result );
  } // end method printSearchResults
} // end class BinarySearchTest
```

9. Soal diberikan oleh dosen.

REFERENSI

5. Deitel H, Deitel P, 2012, Java How to Program, 9th edition, Prentice Hall.
6. Harnaningrum, 2009, Algoritma dan Pemrograman dengan Java, Ghara Ilmu