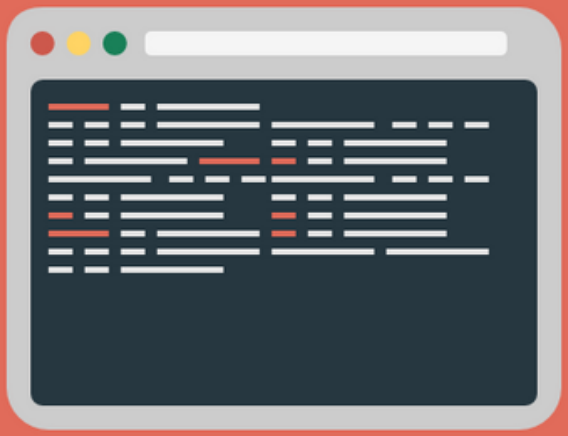
MODUL PRAKTIKUM

STRUKTUR DATA

DOSEN PENGAMPU:

- **DIKE BAYU M, S.T., M.MT**
- **TRI DEVI W, S.T., M. T.**



PROGRAM STUDI SISTEM INFORMASI

PRAKTIKUM 1 - STRUKTUR DATA

ARRAYS

Learning outcomes:

1. Mampu menjelaskan konsep dan implementasi array pada program
2. Mampu melakukan manipulasi data array: menambahkan item, melakukan pencarian, dan menghapus item pada array
3. Mampu mengimplementasikan *ordered array* pada program.
4. Mampu mengimplementasikan *binary search* pada *ordered array*.
5. Mampu menyimpan dan manipulasi objek pada array.

IDENTITAS PRAKTIKAN

NIM : _____

Nama Lengkap : _____

Kelas/ Hari/ Jam :_____/_____/_____

Nomor komputer : _____

Nama Asisten : _____

Tugas	**Praktikum**
Telah diperiksa pada tanggal _____ *(nilai dan paraf asisten)*	Telah diperiksa pada tanggal _____ *(nilai dan paraf asisten)*

A. PENDAHULUAN

1. Format penulisan code yang digunakan untuk mendeklarasikan sebuah array adalah:

```
TipeData namaVariable[ ] = new TipeData[panjang/ukuranArray];
```

Sedangkan format penulisan code untuk menambahkan item pada array yang telah dideklarasikan adalah:

```
namaVariable[index]= value;
```

Tulis dan jalankan listing program berikut:

```
public class classArray {

    public static void main(String[] args) {
        int[] array = new int[10];
        array[0] = 10;
        array[1] = 20;
        array[2] = 30;
        array[3] = 40;
        array[4] = 50;

        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println("");

    }
}
```

Pada listing program tersebut, ukuran array yang dideklarasikan adalah 10. *Insert* item dilakukan hingga index ke-4, artinya hanya terdapat 5 item.
Tuliskan output program tersebut dan jelaskan kenapa demikian!

jawaban

2. Tambahkan baris code berikut ini pada listing program nomer 1

```
            ...

            array = new int[20];

            for (int i = 0; i < array.length; i++) {
                System.out.print(array[i] + " ");
            }
            System.out.println("");

        } //akhir method main
}//akhir class
```

Jalankan program tersebut, apa output yang dihasilkan?

jawaban

Apakah item yang awal dimasukkan (pada listing no 1) masih tersimpan didalam array? Jelaskan kenapa demikian?

jawaban

Hingga tahap ini, yang dapat disimpulkan adalah:

Ukuran array bersifat *fixed* / *not fixed* *)

**)coret salah satu*

3. Lakukan *experiment* menggunakan listing nomer 1 untuk menjawab pertanyaan berikut (beri keterangan benar/salah untuk soal berupa statemen dan tulis jawaban untuk soal isian)

Insert item pada array hanya bisa dilakukan secara berurutan mulai index ke-0.	
Insert item pada array hanya bisa dilakukan hingga ukuran array – 1.	
Cell array untuk semua tipe data *primitive* yang belum diberi value secara *default* bernilai 0.	
Cell array untuk tipe data *String* yang belum diberi value secara *default* bernilai *null*.	
Keterangan yang muncul jika memasukkan item melebihi ukuran array adalah ……	

4. Lengkapi listing berikut:

```
run:
30 20 60 70 50 10
BUILD SUCCESSFUL (total time: 1 second)
```

```
public class ClassArray {

    public static void main(String[] args) {
        int[] array = new int[100];
        int nElemen = 0;
        array[0] = 30;
        array[1] = 20;
        array[2] = 60;
        array[3] = 70;
        array[4] = 50;
        array[5] = 10;
        nElemen = 6;

        for (int i = 0; i <        ; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println("");

    }
}
```

Jalankan dan tuliskan penjelasan dari listing yang telah Anda lengkapi!

jawaban

5. Berikut ini adalah listing program array yang dituliskan dalam bentuk *object oriented programming*. Class *HighArray* memiliki method untuk manipulasi array, yaitu *insert*, *find/search*, dan *delete* serta method *display* untuk menampilkan isi array. Method dalam class HighArray tersebut dipanggil dan dijalankan pada class HighArrayApp.
   Pahami listing berikut dengan menulis dan menjalankannya, kemudian tuliskan penjelasan tiap barisnya!

`class HighArray {`	*Awal sebuah kelas bernama HighArray*
`private int[] arr;`	*Deklarasi variable integer bertipe array bernama "arr" dengan akses private*
`private int nElemen;`	
`public HighArray(int max) {`	
`arr = new int[max];`	
`nElemen = 0;`	
`}`	
`public void insert(int value) {`	
`arr[nElemen] = value;`	
`nElemen++;`	
`}`	
`public boolean find(int key) {`	
`int i;`	
`for (i = 0; i < nElemen; i++) {`	
`if (arr[i] == key) {`	
`break;`	
`}`	
`}`	
`if (i == nElemen) {`	
`return false;`	
`} else {`	
`return true;`	
`}`	
`}`	
`public boolean delete(int value) {`	

```
        int i;
        for (i = 0; i < nElemen; i++) {
            if (value == arr[i]) {
                break;
            }
        }
        if (i == nElemen) {
            return false;
        } else {
          for (int j = i; j < nElemen; j++) {
                arr[j] = arr[j + 1];
            }
            nElemen--;
            return true;
        }
    }

    public void display() {
        for (int i = 0; i < nElemen; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println("");
    }
}

public class HighArrayApp {

    public static void main(String[] args) {
        int maxSize = 100;
        HighArray arr;
        arr = new HighArray(maxSize);

        arr.insert(70);
        arr.insert(80);
        arr.insert(75);
        arr.insert(55);
        arr.insert(85);
        arr.insert(25);
        arr.insert(30);
        arr.insert(00);
        arr.insert(90);
        arr.insert(40)

        arr.display();
```

```
        int key = 25;
        if (arr.find(key)) {
            System.out.println(key +
                    " ditemukan");
        } else {
            System.out.println(key +
                    " tidak ditemukan");
        }

        arr.delete(00);
        arr.delete(80);
        arr.delete(55);

        arr.display();
    }
}
```

Output program tersebut adalah….

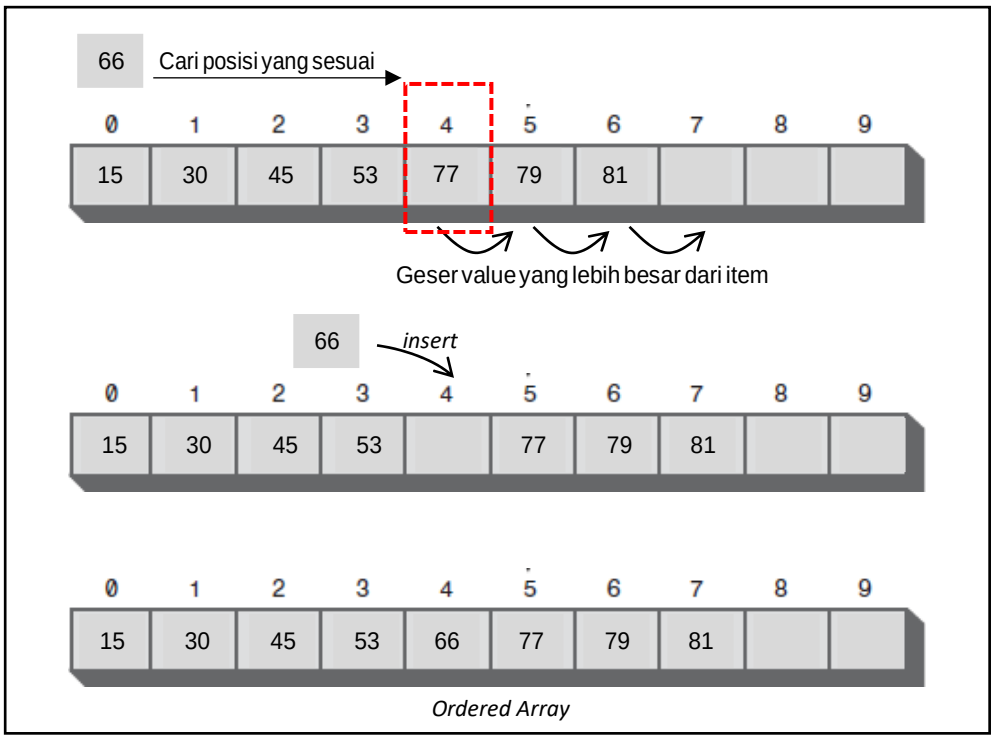
jawaban

6. Tambahkan sebuah method *size* pada class *HighArray* yang mempu mengembalikan nilai jumlah elemen array. Panggil method tersebut pada class *HighArrayApp* untuk menampilkan jumlah elemen.
   Tulis code dan penjelasannya!

jawaban

B. PRAKTIKUM

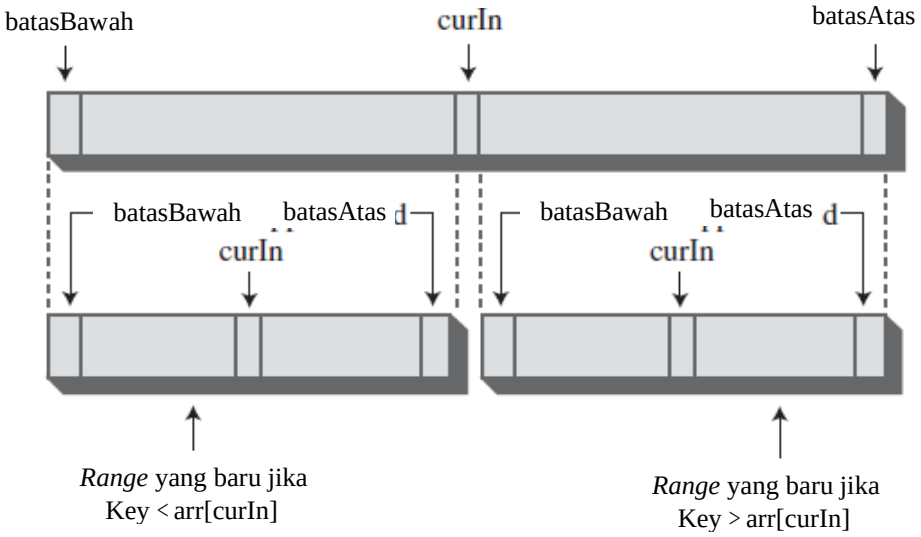
1. Pada listing nomer 5 (tugas pendahuluan), method *insert* digunakan untuk menambahkan item pada *cell* yang belum terisi tanpa memberhatikan *value* item yang ditambahkan sehingga elemen pada array disimpan secara tidak berurutan (*unordered*).
   Agar item dapat disimpan pada urutan yang sesuai dengan *value*-nya maka perlu dilakukan pencarian posisi *cell* yang tepat bagi item yang akan dimasukkan dengan cara membandingkan tiap item pada *cell* dengan item yang akan dimasukkan, yaitu pencarian secara *linier*. Setelah *cell* tepat ditemukan, langkah selanjutnya adalah menyiapkan *cell* tersebut untuk diisi jika sudah ada item yang tersimpan pada *cell* tersebut. Hal ini bisa dilakukan dengan cara menggeser item yang memiliki value lebih besar dari item yang akan dimasukkan, dengan demikian terdapat *cell* kosong untuk diisi dengan item baru. Langkah-langkah insert item pada ordered array ditunjukkan pada Gambar 1.1 berikut ini.



Gambar 1.1 Langkah *insert* item pada *ordered array*

Tuliskan listing untuk method *insert* untuk menyimpan elemen array secara berurutan (*ordered*)!

2. Pencarian pada method *find* listing nomer 5 (tugas pendahuluan) menggunakan *linier search*, artinya terhadap key yang dicari, program akan melakukan pencarian pada array secara berurutan mulai dari elemen pertama hingga elemen terakhir. Hal ini tidak efisien. Pada *ordered array*, dapat dilakukan pencarian menggunakan *binary serach* yang lebih efisien dibandingkan dengan *linier search*.
   Pada *binary search*, *range* elemen array dibagi dua secara berulang-ulang. Hal ini menjadikan *range* pencarian semakin kecil dan terpusat pada item yang memiliki *value* mendekati *key* pencarian. Pembagian *range* pencarian pada *binary search* ditunjukkan pada Gambar 1.2 berikut ini.



Gambar 1.2 Pembagian *range* pencarian pada *binary Search*

Tuliskan listing untuk method *find* yang mengimplementasikan *binary search*!

3. *Storing object*

Item data pada *real world* tidak direpresentasikan dalam bentuk data *primitive* tapi berupa *record* yang merupakan kombinasi dari beberapa *field*. Misalkan untuk *record* personal, kita dapat menyimpan nama, tempat tanggal lahir, nomer telpon, email, dsb. Untuk data mahasiswa, kita dapat menyimpan nim, nama, jurusan, asal, dsb. Dalam java, *record* data biasanya direpresentasikan dengan sebuah *class object*.

Berikut ini listing yang menunjukkan implementasi *storing object*. Terdapat tiga class, yaitu class "Mahasiswa", "DataArray", dan "DataArrayApp". *Record* yang disimpan adalah data mahasiswa yang terdiri dari *field* nim, nama, dan asal. *Record* mahasiswa ini direpresentasikan dalam sebuah *class object* dengan nama "Mahasiswa".

Tulis dan pahamilah listing program untuk menyimpan object berikut ini.

```
class Mahasiswa {

    private long nim;
    private String nama;
    private String asal;

    public Mahasiswa(long nim, String nama, String asal) {
        this.nim = nim;
        this.nama = nama;
        this.asal = asal;
    }

    public void displayMhs() {
        System.out.print("\tNIM: " + nim);
        System.out.print(", Nama: " + nama);
        System.out.println(", Asal: " + asal);
    }

    public long getNim() {
        return nim;
    }
}
```

```
public class DataArray {

    private Mahasiswa[] mhs;
    private int nElemen;

    public DataArray(int max) {
        mhs = new Mahasiswa[max];
        nElemen = 0;
    }

    public Mahasiswa find(long searchNim) {
        int i;
        for (i = 0; i < nElemen; i++) {
            if (mhs[i].getNim()==searchNim) {
                break;
            }
        }
        if (i == nElemen) {
            return null;
        } else {
            return mhs[i];
        }
    }

    public void insert(long nim, String nama, String asal) {
        mhs[nElemen] = new Mahasiswa(nim, nama, asal);
        nElemen++;
    }

    public boolean delete(long searchNim) {
        int i;
        for (i = 0; i < nElemen; i++) {
            if (mhs[i].getNim()==searchNim) {
                break;
            }
        }
        if (i == nElemen) {
            return false;
        } else {
            for (int j = i; j < nElemen; j++) {
                mhs[j] = mhs[j + 1];
            }
            nElemen--;
            return true;
        }
    }
    public void displayArray(){
        for (int i = 0; i < nElemen; i++) {
            mhs[i].displayMhs();
        }
    }
}
```

Objek mahasiswa disimpan dalam array. Class "DataArray" berisi method-method untuk manipulasi object mahasiswa, yaitu *insert*, *find*, dan *delete*, serta method untuk menampilkan array berisi objek mahasiswa, yaitu *displayArray*.
Class yang digunakan untuk menjalankan program adalah class "DataArrayApp". Class ini memiliki method main yang didalamnya terdapat listing untuk memanggil dan menjalankan fungsi-fungsi pada class DataArray yang telah dibuat.

```
public class DataArrayApp {

    public static void main(String[] args) {
        int maxSize = 100;
        DataArray arr;
        arr = new DataArray(maxSize);
        arr.insert(16650200, "Jundi", "Malang");
        arr.insert(16650210, "Ahmad", "Sidoarjo");
        arr.insert(16650220, "Ismail", "Banyuwangi");
        arr.insert(16650230, "Sofi", "Semarang");
        arr.insert(16650240, "Dinda", "Bandung");
        arr.insert(16650250, "Rais", "Ambon");
        arr.insert(16650260, "Helmi", "Madura");
        arr.insert(16650270, "Agung", "Madiun");
        arr.insert(16650280, "Arina", "Malang");

        arr.displayArray();

        long searchKey = 16650270;
        Mahasiswa mhs = arr.find(searchKey);
        if (mhs != null) {
            System.out.print("\nketemu");
            mhs.displayMhs();
        } else {
            System.out.println("ga ketemu " + searchKey);
        }

        searchKey=16650240;
        System.out.println("\nhapus nim: "+searchKey);
        arr.delete(searchKey);

        arr.displayArray();
    }
}
```

Output:

```
run:
        NIM: 16650200, Nama: Jundi, Asal: Malang
        NIM: 16650210, Nama: Ahmad, Asal: Sidoarjo
        NIM: 16650220, Nama: Ismail, Asal: Banyuwangi
        NIM: 16650230, Nama: Sofi, Asal: Semarang
        NIM: 16650240, Nama: Dinda, Asal: Bandung
        NIM: 16650250, Nama: Rais, Asal: Ambon
        NIM: 16650260, Nama: Helmi, Asal: Madura
        NIM: 16650270, Nama: Agung, Asal: Madiun
        NIM: 16650280, Nama: Arina, Asal: Malang

ketemu  NIM: 16650270, Nama: Agung, Asal: Madiun

hapus nim: 16650240
        NIM: 16650200, Nama: Jundi, Asal: Malang
        NIM: 16650210, Nama: Ahmad, Asal: Sidoarjo
        NIM: 16650220, Nama: Ismail, Asal: Banyuwangi
        NIM: 16650230, Nama: Sofi, Asal: Semarang
        NIM: 16650250, Nama: Rais, Asal: Ambon
        NIM: 16650260, Nama: Helmi, Asal: Madura
        NIM: 16650270, Nama: Agung, Asal: Madiun
        NIM: 16650280, Nama: Arina, Asal: Malang
BUILD SUCCESSFUL (total time: 1 second)
```

C. KESIMPULAN

Kesimpulan yang diperoleh dari pembahasan praktikum kali ini adalah:

1. Tentang *unordered arrays* dan *ordered arrays*

2. Tentang *linier search* dan *binary search*

3. Tentang menyimpan object (*storing object*)

PRAKTIKUM 2 - STRUKTUR DATA

SIMPLE SORTING

Learning outcomes:

1. Mampu menjelaskan langkah-langkah *sorting* algoritma *bubble sort, insertion sort,* dan *selection sort.*
2. Mampu mengimplementasikan algoritma *bubble sort, insertion sort,* dan *selection sort* pada program.
3. Mampu mengimplementasikan *sorting object* menggunakan algoritma *bubble sort, insertion sort,* dan *selection sort.*
4. Mampu menjelelaskan perbandingan *sorting* algoritma: *bubble sort, insertion sort,* dan *selection sort.*

IDENTITAS PRAKTIKAN

NIM : _____

Nama Lengkap : _____

Kelas/ Hari/ Jam :_____/_____/_____

Nomor komputer : _____

Nama Asisten : _____

Tugas	**Praktikum**
Telah diperiksa pada tanggal _____ *(nilai dan paraf asisten)*	Telah diperiksa pada tanggal _____ *(nilai dan paraf asisten)*

Tiga algoritma *sorting* sederhana yang dibahas pada modul ini adalah *bubble sort*, *selection sort*, dan *insertion sort*. Ketiga algoritma tersebut menggunakan dua langkah/tindakan yang terus dilakukan secara berulang hingga data berurutan. Dua tindakan tersebut adalah:

1. membandingkan dua item
2. menukar (*swap*) dua item, atau menyalin (*copy*) satu item

A. PENDAHULUAN

1. Algoritma *sorting* yang paling sederhana adalah ***bubble sort***. Langkah *sorting* secara *ascending* menggunakan *bubble sort* adalah sebagai berikut:
   a. Bandingkan dua item
   b. Jika item pertama (sebelah kiri) lebih besar daripada item kedua (sebelah kanan), maka tukar kedua item tersebut
   c. Pindah ke posisi kanan. Lakukan langkah a dan b hingga posisi sampai di batas akhir
   d. Ketika satu item telah berada sesuai urutan, ulangi langkah a-c hingga semua item sesuai urutan.

```
...
public void BubbleSort() {
int batas, i;
for (batas = nElemen-1; batas>0; batas--) {
    for (i = 0; i < batas; i++) {
        if (arr[i] > arr[i + 1]) {
            swap(i, i + 1);
        }
    }
}
}


public void swap(int one, int two) {
    int temp = arr[one];
    arr[one] = arr[two];
    arr[two] = temp;
}
...
```

Listing diatas adalah baris kode yang mengimplementasikan tiap langkah *bubble sort*. Implementasikan baris kode tersebut ke dalam sebuah program sehingga dapat digunakan untuk mengurutkan elemen array. Anda dapat menambahkan listing tersebut pada class *HighArray* (listing 5 praktikum 1) yang dipanggil pada class

HighArrayApp. Atau anda juga dapat menuliskan listing ini pada satu class secara terstruktur.
Insert 6 item pada array program tersebut. Tampilkan isi array sebelum dilakukan pengurutan. Kemudian urutkan dan tampilkan isi array setelah pengurutan.
Jalankan program yang telah Anda buat dan tuliskan output program tersebut.

jawaban

2. Untuk sorting array yang memiliki 6 item, berapa jumlah perbandingan item yang dilakukan hingga semua item sesuai urutan?

jawaban

3. Pada program nomer 1, tambahkan kode untuk menampilkan isi array setelah baris kode pertukaran item (`swap(i, i + 1);`). Jalankan kembali dan amati bagaimana proses pengurutan pada tiap iterasi. Tuliskan isi array pada 10 perulangan pertama. Jelaskan!

jawaban

4. Jika Anda ingin melakukan *sorting* menggunakan algoritma *bubble sort* secara *descending*, maka pada program nomer 1 bagian manakah yang harus diganti. Tuliskan code-nya dan jelaskan!

jawaban

5. Algoritma *sorting* yang lain adalah ***selection sort***. Langkah *selection sort* untuk pengurutan secara *ascending* yaitu:
   a. Cari item terkecil pada array
   b. Letakkan item terkecil sesuai urutannya dengan cara menukar item terkecil dengan item pada index awal
   c. Geser posisi awal pencarian ke kanan.
   d. Lakukan langkah a, b, dan c hingga semua item terurut.

   Berikut ini listing untuk algoritma *selection sort*. Lengkapi sebagaimana soal nomor 1, gunakan data array yang sama, jalankan dan jelaskan tiap baris code pengurutan berikut ini!

```
...
public void SelectionSort() {
   int awal, i, min;


   for (awal=0; awal< nElemen-1; awal++) {
      min = awal;
      for (i = awal + 1; i < nElemen; i++) {
           if (arr[i] < arr[min]) {
                min = i;
           }
      }
      swap(awal, min);
   }
}
...
```

6. Pada listing nomor 5, tambahkan kode untuk menampilkan isi elemen pada array setelah kode pertukaran (`swap(awal, min);`). Jalankan program, amati dan tulis outputnya, kemudian jelaskan!

jawaban

7. Sedikit berbeda dengan dua algoritma sebelumnya, pada ***insertion sort***, aksi yang dilakukan adalah membandingkan dan meng-*copy* item. Berikut ini langkah *insertion sort* untuk pengurutan secara *ascending*:
   a. Tandai sebuah item sebagai batas antara *partially sorted* dan *unsorted items*.
   b. Geser item pada *partially sorted* yang bernilai lebih besar dari pada item yang ditandai pada langkah a.
   c. Sisipkan item tersebut pada posisi yang sesuai di bagian *partially sorted*.
   d. Ulangi langkah a-c hingga semua *unsorted items* telah disisipkan (*insert*) ke *sorted group*.

   Berikut ini listing untuk algoritma *insertion sort*. Lengkapi sebagaimana soal nomor 1, gunakan data array yang sama, jalankan dan jelaskan tiap baris code pengurutan berikut ini!

`...`	
`public void InsertionSort() {`	
`  int i, curIn;`	
`  for (curIn= 1; curIn < nElemen; curIn++) {`	
`      int temp = arr[curIn];`	
`      i = curIn;`	

```
        while (i > 0 && arr[i - 1] >= temp) {
                arr[i] = arr[i - 1];
                i--;
        }
        arr[i] = temp;
    }
}
...
```

8. Pada listing nomor 7, tambahkan kode untuk menampilkan isi elemen pada array setelah kode pergeseran item (`arr[i] = arr[i - 1];`) dan setelah kode insert item (`arr[i] = temp;`). Jalankan program, amati dan tulis outputnya hingga 4 kali tahap penyisipan, kemudian jelaskan!

jawaban

B. PRAKTIKUM

1. *Sorting Object*

   Pada praktikum 1 (Arrays), Anda telah mengiimplementasikan *storing object* "mahasiswa". Implementasikan algoritma *sorting* untuk *sorting object* "Mahasiswa" berdasarkan NIM. Yaitu dengan cara menambahkan method *BubbleSort( ), SelectionSort( ),* dan *InsertionSort( )* sebagaimana yang dituliskan pada tugas pendahuluan modul ini (dengan sedikit penyesuaian) pada class *DataArray* (listing nomer 3 parktikum 1). Method tersebut dipanggil pada class *DataArrayApp*.

   Keterangan:
   - Praktikan dengan **NIM genap** mengerjakan *BubbleSort( )* dan *SelectionSort( )*
   - Praktikan dengan **NIM ganjil** mengerjakan *BubbleSort( )* dan *InsertionSort( )*

2. *Lexicographical Comparisons*

   NIM pada object mahasiswa adalah variable bertipe long. Anda dapat membandingkan item dengan tipe long dengan menggunakan operator *equality relational*. Pada *sorting object*, kita perlu mengetahui cara membandingkan item dengan tipe String (object). Misal, adakalanya pencarian *record* mahasiswa dapat menggunakan *field* "nama" sebagai *key*. Untuk kondisi ini, kita dapat menggunakan method **`compareTo()`** yang ada pada class String.

   Method `compareTo()` membandingkan dua string secara leksikograf (alfabetis). Perbandingan ini didasarkan pada nilai Unicode dari masing-masing karakter dalam String. Deretan karakter objek String ini dibandingkan secara leksikograf dengan deretan karakter string argumen.

   Value yang dikembalikan dalam pemanggilan method ini adalah nilai 0 jika objek string sama dengan string argumen; nilai kurang dari 0 jika objek string secara leksikografis kurang dari string argumen; dan nilai lebih besar dari 0 jika objek string secara leksikograf lebih besar dari string argumen. (lihat Tabel 2.1)

   Tabel 2.1 Operasi pada method `compareTo()`

`StringObjek.compareTo(StringArgumen)`	**Return value**
`StringObjek` < `StringArgumen`	< 0
`StringObjek` equal `StringArgumen`	0
`StringObjek` > `StringArgumen`	> 0

   Variasi lain dari method `compareTo()` adalah `compareToIgnoreCase()`. Method ini memiliki fungsi yang sama dengan method `compareTo()` namun dengan mengabaikan besar/kecilnya huruf.

   Gunakan method `compareTo()` atau `compareToIgnoreCase()` pada tahap perbandingan item untuk melakukan *sorting object* "mahasiswa" berdasarkan *field* "nama"!

   Keterangan:
   - Praktikan dengan **NIM genap** mengerjakan *InsertionSortbyName( )*
   - Praktikan dengan **NIM ganjil** mengerjakan *SelectionSortbyName( )*

```
run:
Data mahasiswa sebelum diurutkan
        NIM: 16650270, Nama: Agung, Asal: Madiun
        NIM: 16650230, Nama: Sofi, Asal: Semarang
        NIM: 16650280, Nama: Arina, Asal: Malang
        NIM: 16650260, Nama: Helmi, Asal: Madura
        NIM: 16650220, Nama: Ismail, Asal: Banyuwangi
        NIM: 16650240, Nama: Dinda, Asal: Bandung
        NIM: 16650250, Nama: Rais, Asal: Ambon
        NIM: 16650200, Nama: Jundi, Asal: Malang
        NIM: 16650210, Nama: Ahmad, Asal: Sidoarjo

1. Sorting Mahasiswa by NIM
        NIM: 16650200, Nama: Jundi, Asal: Malang
        NIM: 16650210, Nama: Ahmad, Asal: Sidoarjo
        NIM: 16650220, Nama: Ismail, Asal: Banyuwangi
        NIM: 16650230, Nama: Sofi, Asal: Semarang
        NIM: 16650240, Nama: Dinda, Asal: Bandung
        NIM: 16650250, Nama: Rais, Asal: Ambon
        NIM: 16650260, Nama: Helmi, Asal: Madura
        NIM: 16650270, Nama: Agung, Asal: Madiun
        NIM: 16650280, Nama: Arina, Asal: Malang

2. Sorting Mahasiswa by NAME
        NIM: 16650270, Nama: Agung, Asal: Madiun
        NIM: 16650210, Nama: Ahmad, Asal: Sidoarjo
        NIM: 16650280, Nama: Arina, Asal: Malang
        NIM: 16650240, Nama: Dinda, Asal: Bandung
        NIM: 16650260, Nama: Helmi, Asal: Madura
        NIM: 16650220, Nama: Ismail, Asal: Banyuwangi
        NIM: 16650200, Nama: Jundi, Asal: Malang
        NIM: 16650250, Nama: Rais, Asal: Ambon
        NIM: 16650230, Nama: Sofi, Asal: Semarang
BUILD SUCCESSFUL (total time: 1 second)
```

Gambar 2.1 Contoh output *sorting object* mahasiswa berdasarkan *field* NIM dan nama

C. KESIMPULAN

Kesimpulan yang diperoleh dari pembahasan praktikum kali ini adalah:

1. Tentang perbandingan algoritma sorting *bubble, selection,* dan *insertion*

2. Tentang *sorting object*

PRAKTIKUM 3 - STRUKTUR DATA

STACKS AND QUEUES

Learning outcomes:

1. Mampu menjelaskan konsep LIFO pada Stacks dan FIFO pada Queues
2. Mampu mengimplementasikan operasi dasar Stack dan Queue (*push*, *pop*, *peek*) dengan struktur data Array.
3. Mampu menjelaskan implementasi Stack pada *parsing arithmetic expressions.*
4. Mampu menjelaskan dan mengimplementasikan *circular* Queue.
5. Mampu mengimplementasikan struktur data Stacks dan Queues pada program.

IDENTITAS PRAKTIKAN

NIM : _____

Nama Lengkap : _____

Kelas/ Hari/ Jam : _____/_____/_____

Nomor komputer : _____

Nama Asisten : _____

Tugas	**Praktikum**
Telah diperiksa pada tanggal _____ *(nilai dan paraf asisten)*	Telah diperiksa pada tanggal _____ *(nilai dan paraf asisten)*

Struktur data yang dibahas pada praktikum kali ini adalah Stacks dan Queues. Selain itu, ada pula struktur data yang sejenis yaitu priority queue (tidak dibahas pada praktikum ini). Fungsi ketiga struktur data ini lebih sering digunakan sebagai *programmer's tool*. Yaitu, digunakan sebagai alat bantu konseptual penyimpanan data, bukan sebagai penyimpanan data itu sendiri.

Stack, queue, dan priority queue lebih abstrak dibandingkan dengan struktur penyimpanan data seperti arrays dan beberapa yang lainnya. Mekanisme mendasar untuk mengimplementasikan stack, queue, dan priority queue dapat berupa Arrays sebagaimana yang ditunjukkan pada pembahasan modul ini. Selain itu, dapat juga menggunakan "Linked list". Mekanisme mendasar untuk Priority queue dapat juga berupa salah satu jenis khusus dari struktur data *tree* yang disebut dengan *heap*.

Pada struktur data Arrays, semua item dapat diakses, sedangkan pada stack, queue, dan priority queue akses tersebut dibatasi, yaitu hanya satu item yang dapat diakses untuk dibaca atau dihapus.

A. PENDAHULUAN

1. Stacks (tumpukan) merupakan suatu susunan koleksi data dimana data yang dapat ditambahkan dan dihapus selalu dilakukan pada bagian akhir data, yang disebut dengan *top of stack*. Dengan kata lain, stack hanya mengijinkan akses pada item yang terakhir dimasukkan.
   Stacks bersifat LIFO (*Last In First Out*). Jelaskan sifat LIFO pada stacks dan gambarkan skema lengkap dari LIFO!

jawaban

2. Operasi utama pada Stacks yaitu *push* dan *pop*. Selain dua operasi tersebut, juga terdapat operasi *peek* pada Stacks. Jelaskan masing-masing dari tiga operasi tersebut!

jawaban

3. Tulislah listing program berikut ini dan jelaskan tiap barisnya!

```
class Stack {


    private int maxSize;
    private long[] stackArray;
    private int top;


    public Stack(int size) {
        maxSize = size;
        stackArray = new long[maxSize];
        top = -1;
    }


    public void push(long item) {
        stackArray[++top] = item;
    }


    public long pop() {
        return stackArray[top--];
    }

```

```
    public long peek() {
        return stackArray[top];
    }

    public boolean isEmpty() {
        return (top == -1);
    }

    public boolean isFull() {
        return (top == maxSize - 1);
    }
}

public class StackApp {

    public static void main(String[] args) {
        Stack theStack = new Stack(10);
        System.out.println(">> push some
                           items");
        theStack.push(20);
        theStack.push(40);
        theStack.push(60);
        theStack.push(80);

        System.out.println("\n>> pop items
                           in the stack");
        while (!theStack.isEmpty()) {
            long value = theStack.pop();
            System.out.print(value + " ");
        }
    }
}
```

Apa output dari listing program tersebut? Jelaskan!

jawaban

4. Setelah anda memahami tiap baris listing nomer 3. Tuliskan kesimpulan logika untuk setiap method pada class Stack: method `push()`, `pop()`, `peek()`, `isEmpty()`, dan `isFull()`!

jawaban

5. Salah satu aplikasi yang menggunakan struktur penyimpanan stack adalah parsing ekspresi aritmatika. Tuliskan langkah manual untuk merubah notasi *infix*:

   U * (I + N) / M ^ L – G

   menjadi notasi *postfix* menggunakan teknik stack!

jawaban

6. Queues (antrian) adalah struktur data yang hampir mirip dengan stack. Perbedaannya adalah pada queues, akses item bagi yang pertama dimasukkan. Queues bersifat FIFO (*First In First Out*). Jelaskan sifat FIFO pada queues dan gambarkan skema lengkap dari FIFO!

jawaban

7. Berikut ini listing Queue dengan Array. Tulis dan jelaskan!

```
class Queue {
    private int maxSize;
    private long[] queArray;
    private int front;
    private int rear;
    private int nItems;


    public Queue(int size) {
        this.maxSize = size;
        queArray = new long[maxSize];
        front = 0;
        rear = -1;
        nItems = 0;
    }


    public void insert(long value) {
        if (rear == maxSize - 1) {
            rear = -1;
        }
        queArray[++rear] = value;
        nItems++;
    }


    public long remove() {
        long temp = queArray[front++];
        if (front == maxSize) {
            front = 0;
        }
        nItems--;
        return temp;
    }
```

```
    public long peekFront() {
        return queArray[front];
    }

    public boolean isEmpty() {
        return (nItems == 0);
    }

    public boolean isFull() {
        return (nItems == maxSize);
    }

    public int size() {
        return nItems;
    }
}

public class QueueApp {
    public static void main(String[] args) {
        Queue theQueue = new Queue(5);
        theQueue.insert(10);
        theQueue.insert(20);
        theQueue.insert(30);
        theQueue.insert(40);
        theQueue.remove();
        theQueue.remove();
        theQueue.remove();
        theQueue.insert(50);
        theQueue.insert(60);
        theQueue.insert(70);
        theQueue.insert(80);

        while (!theQueue.isEmpty()){
```

```
            long n = theQueue.remove();
            System.out.print(n);
            System.out.print(" ");
        }
        System.out.println("");
    }
}
```

Tuliskan output dari listing tersebut dan jelaskan!

jawaban

8. Setelah anda memahami tiap baris listing nomer 7. Tuliskan kesimpulan logika untuk setiap method pada class Queue: method `insert()`, `remove()`, `peek()`, `isEmpty()`, dan `isFull()`!

jawaban

9. Apa jenis queue (*linier/circular*) yang diimplementasikan pada listing tersebut? Beri alasan/bukti dari jawaban anda!

jawaban

B. PRAKTIKUM

1. Implementasi Stack

   Salah satu contoh program sederhana yang mengimplementasikan stack adalah program pembalik kata. Stack digunakan untuk membalik huruf. Langkah pertama, tiap katakter pada String input diekstrak dan dimasukkan kedalam stack. Kemudian tiap karakter tersebut dikeluarkan dan ditampilkan sebagai output. Karena memiliki sifat LIFO, maka keluaran stack adalah karakter-karakter dengan urutan yang berkebalikan dengan input.

   Buatlah program pembalik kata tersebut dengan membuat 3 class, yaitu:

   a. class "stack". Class ini digunakan untuk menyimpan setiap karakter input pada stack array. Berisi constructor dan method-method operasi stack.
   b. class "pembalik". Class ini digunakan untuk membaca setiap karakter input, menyimpan karakter dengan memanggil method `push()` pada class "stack" (point a), dan membalik input dengan memanggil method `pop()` pada class "stack" (point a). Tiap karakter keluaran dari stack tersebut disimpan/ditambahkan (*append*) pada String output sebagai keluaran yang akan ditampilkan.
   c. class "AppPembalik". Class ini berisi method main. Digunakan untuk deklarasi dan inisialisasi input, memanggil class "pembalik" untuk membalik input dan mendapatkan output kata yang telah dibalik, serta menampilkan output pada console.

   Untuk membaca tiap karakter String, anda dapat mengunakan method **`charAt()`** yang terdapat pada class String, misal `StringInput.charAt(index)`.

   Contoh program pembalik kata ditunjukkan pada Gambar 3.1. Anda dapat mengerjakan sebagaimana Gambar 3.1 (a) dan point tambahan akan diberikan jika Anda mengerjakan sebagaimana Gambar 3.1 (b).

(a)

```
run:
>> katanya...
        kasur
>> dibalik jadi...
        rusak
BUILD SUCCESSFUL (total time: 2 seconds)
```

(b)

```
run:
Masukkan kata: bakso
kebalikan: oskab
Masukkan kata: soto
kebalikan: otos
Masukkan kata:
```

Gambar 3.1 Contoh output program pembalik kata. (a) String input diinisialisasi secara langsung pada listing program (*hardcode*). (b) String input didapat dari masukan dengan keyboard, program dapat membaca dan membalikkan input secara berulang-ulang

2. Implementasi Queue

   Implementasi queue banyak didunia nyata, sebagaimana antrian. Buatlah program simulasi antrian dengan mengimplementasikan Queue Stack. Simulasi antrian menunjukkan (lihat Gambar 3.2):

   - penambahan objek pada daftar antrian. Lakukan beberapa kali hingga antrian penuh. Ketika objek ditambahkan pada antrian yang penuh maka program akan menampilkan keterangan antrian penuh.
   - Menampilkan isi antrian
   - Satu persatu objek keluar antrian hingga antrian kosong

```
run:
>> beberapa mulai mengantri
Andi masuk antrian
Ahmad masuk antrian
Satrio masuk antrian
Afrizal masuk antrian
Maaf sukran, antrian masih penuh
Maaf Mahmud, antrian masih penuh

>> isi antrian
Andi,Ahmad,Satrio,Afrizal,

>> satu persatu keluar antrian
Andi Keluar antrian
Ahmad,Satrio,Afrizal,Kosong,

Ahmad Keluar antrian
Satrio,Afrizal,Kosong,Kosong,

Satrio Keluar antrian
Afrizal,Kosong,Kosong,Kosong,

Afrizal Keluar antrian
Kosong,Kosong,Kosong,Kosong,

antrian kosong
0 Person
Kosong,Kosong,Kosong,Kosong,
BUILD SUCCESSFUL (total time: 2 seconds)
```

Gambar 3.2 Output program simulasi antrian

C. KESIMPULAN

Kesimpulan yang diperoleh dari pembahasan praktikum kali ini adalah:

1. Tentang konsep dan implementasi Stack

2. Tentang konsep dan implementasi Queue

PRAKTIKUM 4 - STRUKTUR DATA

LINKED LIST

Learning outcomes:

1. Mampu menjelaskan empat macam linked list yaitu *linier singly-linked list, linier doubly-linked list, circular singly-linked list,* dan *circular doubly-linked list.*
2. Mampu mengimplementasikan singly-linked list
3. Mampu mengimplementasikan doubly-linked list
4. Mampu mengimplementasikan Stacks pada struktur data linked list
5. Mampu mengimplementasikan Queues pada struktur data linked list

IDENTITAS PRAKTIKAN

NIM : _____

Nama Lengkap : _____

Kelas/ Hari/ Jam : _____/_____/_____

Nomor komputer : _____

Nama Asisten : _____

Tugas	**Praktikum**
Telah diperiksa pada tanggal _____ *(nilai dan paraf asisten)*	Telah diperiksa pada tanggal _____ *(nilai dan paraf asisten)*

Linked List adalah salah satu bentuk struktur data, berisi kumpulan data (*node*) yang tersusun secara sekuensial, saling sambung-menyambung, dinamis dan tidak terbatas. Pada Linked List, setiap item berada di dalam sebuah `node` atau object `link`. Setiap objek `Link` memuat sebuah referensi (biasa disebut `next`) menuju link setelahnya pada list. Pada bahasa pemrograman C, referensi disini biasa disebut dengan pointer.

A. PENDAHULUAN

1. Dalam pengaplikasiannya, secara struktur link, linked list dibagi menjadi 2 macam, yaitu :
   a. Singly-Linked List, merupakan list dengan penghubung 1 referensi antar data yang menunjuk data selanjutnya (`next`).
   b. Doubly-Linked List, merupakan list dengan penghubung 2 referensi antar data yang menunjuk data sebelum (`prev`) dan setelahnya (`next`).

   Dan dari 2 jenis di atas, secara bentuk list yang terbentuk, masing-masing dapat diaplikasikan dengan 2 bentuk, yaitu :
   a. Linear: List yang berbentuk linear dengan *head/first* sebagai penunjuk awal list dan akan diteruskan dengan obyek-obyek yang saling berhubungan, diakhiri dengan *tail/ last* sebagai penunjuk akhir list, yang mana data terakhir ini akan memiliki referensi null sebagai indikator akhir list.
   b. Circular: List yang berbentuk circular dengan *head/first* sebagai penunjuk awal list dan akan diteruskan dengan obyek-obyek yang saling berhubungan, diakhiri dengan *tail/ last* sebagai penunjuk akhir list. Bedanya akhir list akan memiliki referensi yang akan menunjuk kembali ke awal list (*head/first*) dan atau *head/first* pun akan memiliki referensi yang menunjuk bahwa data sebelumnya adalah *tail/last* list.

   Sehingga, dapat dikatakan terdapat 4 macam struktur linked list, yaitu linier singly-linked list, linier doubly-linked list, circular singly-linked list, dan circular-doubly linked list. Dari ke-empat macam struktur linked list tersebut, gambarkan ilustrasi masing-masing struktur linked list tersebut sehingga tampak isi objek `link` dan hubungan tiap `link`!

jawaban

2. Berikut ini listing program yang menunjukkan implementasi linier singly-linked list.

```
class Link {


    public int Data;
    public Link next;


    public Link(int Data) {
        this.Data = Data;
    }


    public void displayLink() {
        System.out.print(Data +" ");
    }
}


class LinkedList {


    private Link first;
    public LinkedList() {
        first = null;
    }


    public boolean isEmpty() {
        return (first == null);
    }


    public void insertFirst(int Data) {
        Link newLink = new Link(Data);
        newLink.next = first;
        first = newLink;
    }
```

```
public Link deleteFirst() {
    Link temp = first;
    first = first.next;
    return temp;
}


public Link find(int key) {
    Link current = first;
    while (current.Data != key) {
        if (current.next == null) {
            return null;
        } else {
            current = current.next;
        }
    }
    return current;
}


public Link delete(int key) {
    Link current = first;
    Link previous = first;
    while (current.Data != key) {
        if (current.next == null) {
            return null;
        } else {
            previous = current;
            current = current.next;
        }
  }
    if (current == first) {
        first = first.next;
    } else {
```

```
            previous.next = current.next;
        }
        return current;
    }


    public void displayList() {
        System.out.println("List(first
                           -->last):");
        Link current = first;
        while (current != null) {
            current.displayLink();
            current = current.next;
        }
        System.out.println("");
    }
}


public class LinkedListApp {
    public static void main(String[] args) {
        LinkedList theList = new LinkedList();


        theList.insertFirst(22);
        theList.insertFirst(44);
        theList.insertFirst(66);
        theList.insertFirst(88);
        theList.displayList();


        while (!theList.isEmpty()) {
            Link aLink =
                   theList.deleteFirst();

            System.out.print("Deleted ");
            aLink.displayLink();
```

```
            System.out.println("");
        }
        theList.displayList();


        theList.insertFirst(33);
        theList.insertFirst(55);
        theList.insertFirst(77);
        theList.insertFirst(88);
        theList.displayList();

        Link f = theList.find(77);
        if (f != null) {
            System.out.println("ketemu..."
                               + f.Data);
        } else {
            System.out.println("link tidak
                                ditemukan");
        }

        Link d = theList.delete(88);
        if (d != null) {
            System.out.println("hapus link
                     dengan key " + d.Data);
        } else {
            System.out.println("link tidak
                                ditemukan");
        }
        theList.displayList();
    }
}
```

Tuliskan output dari listing program tersebut!

jawaban

3. Pada class `Link` (listing nomor 2) terdapat deklarasi variable "`next`" yang bertipe objek `Link` (sama dengan nama class tersebut). Variable tersebut tidak di-inisialisasi dengan value tertentu. Jelaskan maksud dan fungsi dari variable `next` pada baris `public Link next;`

jawaban

4. Jelaskan fungsi dan logika tiap method `insertFirst()`, `deleteFirst()`, `find()`, dan `delete()` yang terdapat pada listing nomer 2!

jawaban

5. Jelaskan langkah/logika untuk menambahkan operasi “insert Last” yang berfungsi untuk menambahkan data pada akhir list. Tuliskan listing program yang perlu ditambahkan pada program nomor 2!

jawaban

6. Jelaskan langkah/logika untuk menambahkan operasi “delete Last” yang berfungsi untuk menghapus data pada akhir list. Tuliskan listing program yang perlu ditambahkan pada program nomor 2!

jawaban

7. Berikut ini listing program yang menunjukkan implementasi doubly-linked list.

```
class Link {

    public int Data;

    public Link next;

    public Link previous;


    public Link(int Data) {

        this.Data = Data;

    }

    public void displayLink() {

        System.out.print(Data + " ");

    }

}
```

class DoublyLinkedList {
    private Link first;
    private Link last;
 	
    public DoublyLinkedList() {
        first = null;
        last = null;
    }	

public boolean isEmpty() {
        return first == null;
    }	

public void insertFirst(int Data) {
        Link newLink = new Link(Data);
        if (isEmpty()) {
            last = newLink;
        } else {
            first.previous = newLink;
        }
        newLink.next = first;
        first = newLink;
    }	

public void insertLast(int Data) {
        Link newLink = new Link(Data);
        if (isEmpty()) {
            first = newLink;
        } else {
            last.next = newLink;
            newLink.previous = last;
        }
        last = newLink;
    }	

```
    public Link deleteFirst() {
        Link temp = first;
        if (first.next == null) {
            last = null;
        } else {
            first.next.previous = null;
        }
        first = first.next;
        return temp;
    }
```

```
    public Link deleteLast() {
        Link temp = last;
        if (first.next == null) {
            first = null;
        } else {
            last.previous.next = null;
        }
        last = last.previous;
        return temp;
    }
```

```
    public boolean insertAfter(int key,
                               int Data) {
        Link current = first;
        while (current.Data != key) {
            current = current.next;
            if (current == null) {
                return false;
            }
        }
        Link newLink = new Link(Data);
        if (current == last) {
```

```
            newLink.next = null;
            last = newLink;
        } else {
            newLink.next = current.next;
            current.next.previous = newLink;
        }
        newLink.previous = current;
        current.next = newLink;
        return true;
    }
```

```
    public Link deleteKey(int key) {
        Link current = first;
        while (current.Data != key) {
            current = current.next;
            if (current == null) {
                return null;
            }
        }
        if (current == first) {
            first = current.next;
        } else {
            current.previous.next =
                                current.next;
        }
        if (current == last) {
            last = current.previous;
        } else {
            current.next.previous =
                            current.previous;
        }
        return current;
    }
```

```
    public void displayForward() {
        System.out.print("List ”
```

```
                        + "(first-->last): ");
        Link current = first;
        while (current != null) {
            current.displayLink();
            current = current.next;
        }
        System.out.println("");
    }
```

```
    public void displayBackward() {
        System.out.print("List "
                        + "(last-->first): ");
        Link current = last;
        while (current != null) {
            current.displayLink();
            current = current.previous;
        }
        System.out.println("");
    }
} // akhir class
```

```
public class DoublyLinkedListApp {
    public static void main(String[] args) {
        DoublyLinkedList theList =
                    new DoublyLinkedList();
        theList.insertFirst(22);
        theList.insertFirst(44);
        theList.insertFirst(66);
        theList.displayForward();
        theList.insertLast(11);
        theList.insertLast(33);
        theList.insertLast(55);
        theList.displayForward();
        theList.displayBackward();
        theList.deleteFirst();
```

Tuliskan output program ini:

```
        theList.displayForward();
        theList.deleteLast();
        theList.displayForward();
        theList.deleteKey(11);
        theList.displayForward();
        theList.insertAfter(22, 77);
        theList.insertAfter(33, 88);
        theList.displayForward();
    }
}
```

8. Dari penjelasan tiap bagian yang Anda tuliskan pada nomor 7, tuliskan kesimpulan logika yang digunakan pada tiap method: `insertFirst()`, `insertLast()`, `insertAfter()`, `deleteFirst()`, `deleteLast()`, `deleteKey()`!

jawaban

B. PRAKTIKUM

1. Implementasi Stack pada Linked List
   Pada modul 3, Stacks telah dibahas beserta implementasinya pada struktur data Array. Pada praktikum kali ini, buatlah program yang mengimplementasikan Stack menggunakan struktur data Linked list dengan ketentuan:
   - Data yang disimpan adalah record barang elektronik yang memiliki field id bertipe integer dan nama barang bertipe String.
   - Lakukan push, pop, dan display stack pada program

```
run:
Stack(top-- > bottom):
 {2, TV}
 {1, VCD}

Stack(top-- > bottom):
 {6, dispenser}
 {5, rice cooker}
 {4, PC}
 {3, kulkas}
 {2, TV}
 {1, VCD}

Stack(top-- > bottom):
 {4, PC}
 {3, kulkas}
 {2, TV}
 {1, VCD}

BUILD SUCCESSFUL (total time: 1 second)
```

Gambar 4.1 Contoh output program Stack pada Linked list

2. Implementasi Queue pada Linked List
   Sebagaimana soal praktikum nomer 1, buatlah program yang mengimplementasikan Queue menggunakan struktur data Linked list dengan ketentuan:
   - data yang disimpan adalah data mahasiswa (nim dan nama) yang melakukan antrian
   - Lakukan insert, remove, dan display queue pada program!

```
run:
Queue (front-->rear):
        1665100 Dee
        1665200 Deaja

Queue (front-->rear):
        1665100 Dee
        1665200 Deaja
        1665300 Ami
        1665400 Haya
        1665500 Yati

Queue (front-->rear):
        1665300 Ami
        1665400 Haya
        1665500 Yati

BUILD SUCCESSFUL (total time: 2 seconds)
```

Gambar 4.2 Contoh output program Queue pada Linked list

C. KESIMPULAN

Kesimpulan yang diperoleh dari pembahasan praktikum kali ini adalah:

1. Tentang perbandingan 4 macam struktur linked list, yaitu linier singly-linked list, linier doubly-linked list, circular singly-linked list, dan circular-doubly linked list.

2. Tentang implementasi Stack pada Linked list

3. Tentang implementasi Queue pada Linked list