



Array  
Array List  
Iterator  
Vector  
Stack  
Queue



# Tentang Array

- Array adalah sekumpulan variabel yang memiliki tipe data yang sama dan dinyatakan dengan nama yang sama
- Array menggunakan indeks integer untuk menentukan urutan elemen-elemennya, dimana elemen pertamanya dimulai dari indeks 0, elemen kedua memiliki indeks 1, dan seterusnya.

# Mendeklarasikan Variabel Array

- Mendeklarasikan variabel array dengan tipe data yang diinginkan dengan cara yang hampir sama dengan variabel biasa
- perbedaan utama pendeklarasian variabel array dengan variabel biasa adalah adanya tanda kurung [ ] di akhir tipe data atau di akhir nama variabel array
- Contoh  
**int[ ] bilangan; atau int bilangan[ ];**

# Mendefinisikan Array

- menentukan besar array yang diinginkan.

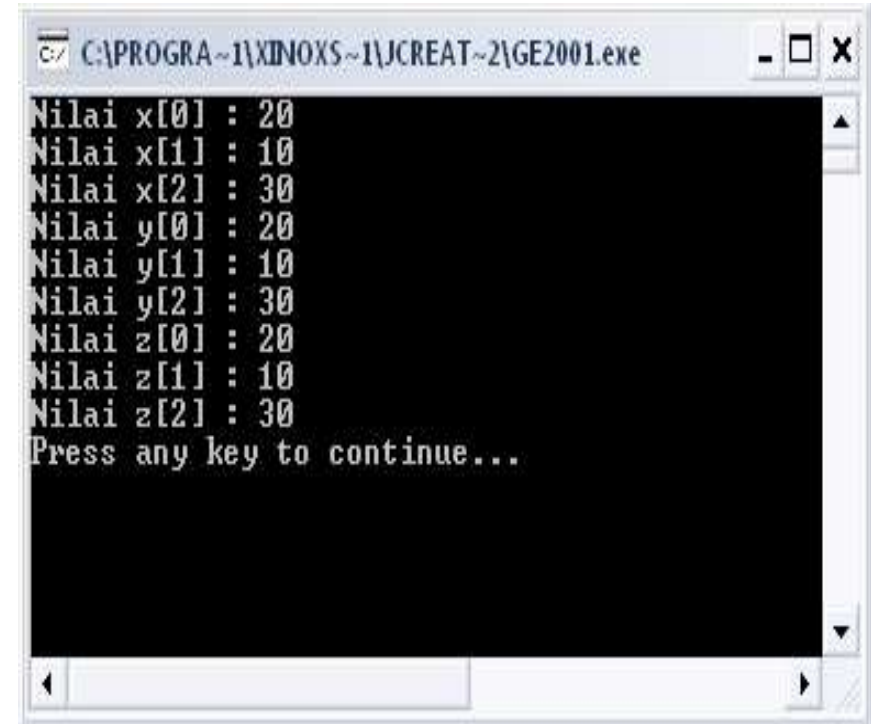
- Contoh

**Bilangan = new int[5];**

- variabel dengan nama bilangan dapat menyimpan 5 nilai integer yang dapat diakses melalui indeks 0 sampai indeks 4.

# Contoh

```
class SingleArray {  
    public static void main(String[] args) {  
        int [] x; // Cara 1  
        x = new int[3];  
        x[0] = 20 ;  
        x[1] = 10 ;  
        x[2] = 30;  
        System.out.println("Nilai x[0] :" + x[0]);  
        System.out.println("Nilai x[1] :" + x[1]);  
        System.out.println("Nilai x[2] :" + x[2]);  
        int [] y = new int[3]; // Cara 2  
        y[0] = 20 ;  
        y[1] = 10 ;  
        y[2] = 30;  
        System.out.println("Nilai y[0] :" + y[0]);  
        System.out.println("Nilai y[1] :" + y[1]);  
        System.out.println("Nilai y[2] :" + y[2]);  
        int[] z = {20,10,30}; // Cara 3 tdk menggunakan  
                               new  
        System.out.println("Nilai z[0] :" + z[0]);  
        System.out.println("Nilai z[1] :" + z[1]);  
        System.out.println("Nilai z[2] :" + z[2]);  
    }  
}
```



```
Nilai x[0] : 20  
Nilai x[1] : 10  
Nilai x[2] : 30  
Nilai y[0] : 20  
Nilai y[1] : 10  
Nilai y[2] : 30  
Nilai z[0] : 20  
Nilai z[1] : 10  
Nilai z[2] : 30  
Press any key to continue...
```

# Array Dua Dimensi

- Array dua dimensi sebenarnya adalah array yang berisi array
- Jumlah index array kolom \* baris
- Contoh :

```
int[][] arrx;
```

```
arrx = new int[3][3];
```

ada  $3 \times 3 = 9$  elemen, mulai dari

```
arrx[0][0]..arrx[2][2]
```





# Latihan

- Buka latihan menghitung nilai PBO, modifikasi agar dapat menampung data lebih dari satu.
- Buat method daftarNilai yang digunakan untuk menampilkan data yang sudah dimasukkan

# Contoh Array 2 Dimensi

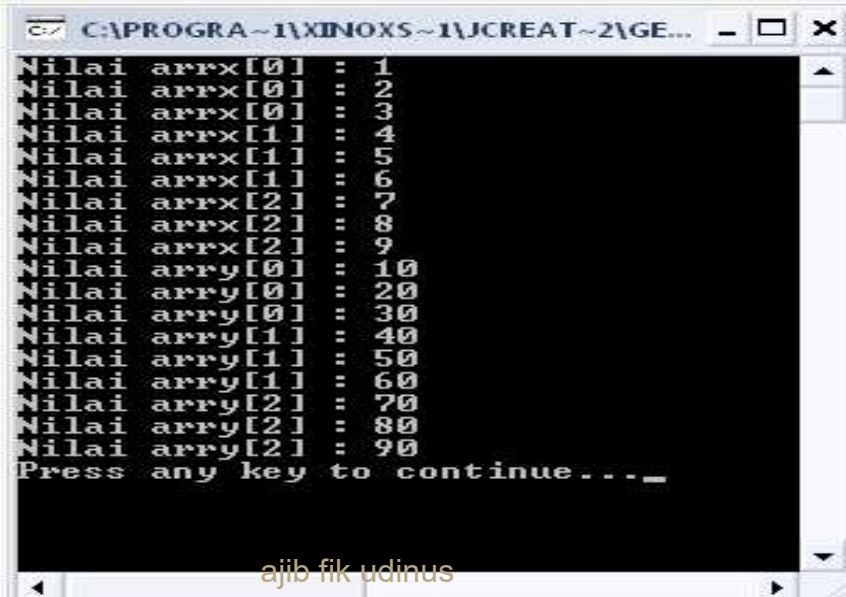
```
class Array2D {  
    public static void main(String[] args) {  
        int[][] arrx; // Cara 1 Array 2 Dimensi  
        arrx = new int[3][3];  
        arrx[0][0] = 1;  
        arrx[0][1] = 2;  
        arrx[0][2] = 3;  
        arrx[1][0] = 4;  
        arrx[1][1] = 5;  
        arrx[1][2] = 6;  
        arrx[2][0] = 7;  
        arrx[2][1] = 8;  
        arrx[2][2] = 9;  
        System.out.println("Nilai arrx[0] : " + arrx[0][0]);  
        System.out.println("Nilai arrx[0] : " + arrx[0][1]);  
        System.out.println("Nilai arrx[0] : " + arrx[0][2]);  
        System.out.println("Nilai arrx[1] : " + arrx[1][0]);  
        System.out.println("Nilai arrx[1] : " + arrx[1][1]);  
        System.out.println("Nilai arrx[1] : " + arrx[1][2]);  
        System.out.println("Nilai arrx[2] : " + arrx[2][0]);  
        System.out.println("Nilai arrx[2] : " + arrx[2][1]);  
        System.out.println("Nilai arrx[2] : " + arrx[2][2]);  
    }  
}
```

```
int[][] array = {{10,20,30},{40,50,60},{70,80,90}} ; //
```

Cara 2 Array 2

Dimensi dgn ukuran 3 \* 3 = 9

```
System.out.println("Nilai array[0] : " + array[0][0]);  
System.out.println("Nilai array[0] : " + array[0][1]);  
System.out.println("Nilai array[0] : " + array[0][2]);  
System.out.println("Nilai array[1] : " + array[1][0]);  
System.out.println("Nilai array[1] : " + array[1][1]);  
System.out.println("Nilai array[1] : " + array[1][2]);  
System.out.println("Nilai array[2] : " + array[2][0]);  
System.out.println("Nilai array[2] : " + array[2][1]);  
System.out.println("Nilai array[2] : " + array[2][2]);  
}
```



```
C:\PROGRA~1\XINOX~1\JCREAT~2\GE... - □ ×  
Nilai arrx[0] : 1  
Nilai arrx[0] : 2  
Nilai arrx[0] : 3  
Nilai arrx[1] : 4  
Nilai arrx[1] : 5  
Nilai arrx[1] : 6  
Nilai arrx[2] : 7  
Nilai arrx[2] : 8  
Nilai arrx[2] : 9  
Nilai array[0] : 10  
Nilai array[0] : 20  
Nilai array[0] : 30  
Nilai array[1] : 40  
Nilai array[1] : 50  
Nilai array[1] : 60  
Nilai array[2] : 70  
Nilai array[2] : 80  
Nilai array[2] : 90  
Press any key to continue..._
```

ajib fik udinus



# Array Multidimensi

- Array multidimensi merupakan array yang terdiri dari array lebih dari dua dimensi.

- Contoh :

```
int[][][]array dimensi = new  
int[5][10][5];
```

- dapat menentukan ukuran array yang berbeda pada tiap array

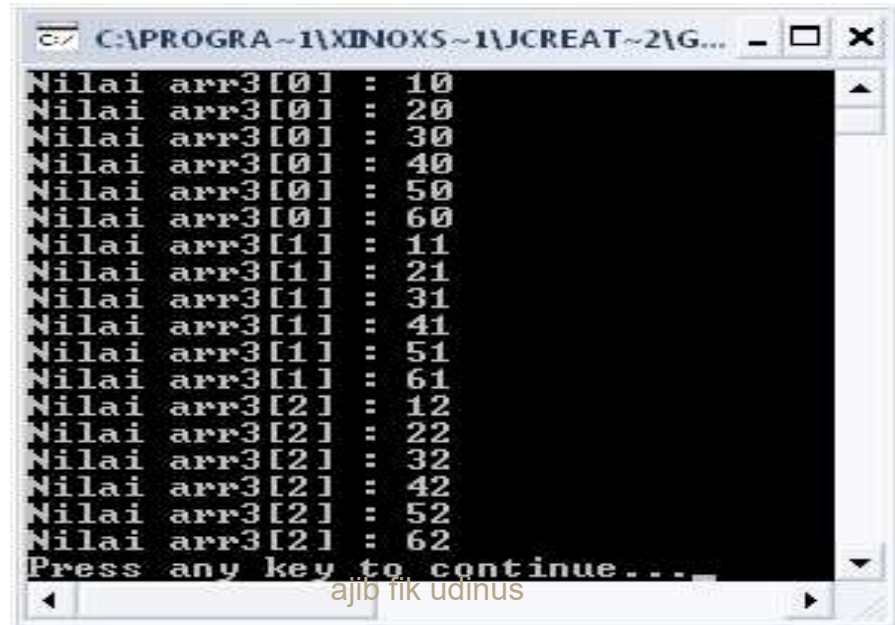
- Misal :

```
int[][][] mdimensi = new int[5][][];
```

# Contoh Multidimensi

```
class ArrayMultiD {  
    public static void main(String[] args) {  
        int[][][] arr3 = {{{10,20,30},{40,50,60}},  
        {{11,21,31},{41,51,61}},  
        {{12,22,32},{42,52,62}}}; //3 * 6 = 18  
        System.out.println("Nilai arr3[0] : " +  
            arr3[0][0][0]);  
        System.out.println("Nilai arr3[0] : " +  
            arr3[0][0][1]);  
        System.out.println("Nilai arr3[0] : " +  
            arr3[0][0][2]);  
        System.out.println("Nilai arr3[0] : " +  
            arr3[0][1][0]);  
        System.out.println("Nilai arr3[0] : " +  
            arr3[0][1][1]);  
        System.out.println("Nilai arr3[0] : " +  
            arr3[0][1][2]);  
        System.out.println("Nilai arr3[1] : " +  
            arr3[1][0][0]);  
        System.out.println("Nilai arr3[1] : " +  
            arr3[1][0][1]);
```

```
        System.out.println("Nilai arr3[1] : " + arr3[1][0][2]);  
        System.out.println("Nilai arr3[1] : " + arr3[1][1][0]);  
        System.out.println("Nilai arr3[1] : " + arr3[1][1][1]);  
        System.out.println("Nilai arr3[1] : " + arr3[1][1][2]);  
        System.out.println("Nilai arr3[2] : " + arr3[2][0][0]);  
        System.out.println("Nilai arr3[2] : " + arr3[2][0][1]);  
        System.out.println("Nilai arr3[2] : " + arr3[2][0][2]);  
        System.out.println("Nilai arr3[2] : " + arr3[2][1][0]);  
        System.out.println("Nilai arr3[2] : " + arr3[2][1][1]);  
        System.out.println("Nilai arr3[2] : " + arr3[2][1][2]);  
    }  
}
```



```
Nilai arr3[0] : 10  
Nilai arr3[0] : 20  
Nilai arr3[0] : 30  
Nilai arr3[0] : 40  
Nilai arr3[0] : 50  
Nilai arr3[0] : 60  
Nilai arr3[1] : 11  
Nilai arr3[1] : 21  
Nilai arr3[1] : 31  
Nilai arr3[1] : 41  
Nilai arr3[1] : 51  
Nilai arr3[1] : 61  
Nilai arr3[2] : 12  
Nilai arr3[2] : 22  
Nilai arr3[2] : 32  
Nilai arr3[2] : 42  
Nilai arr3[2] : 52  
Nilai arr3[2] : 62  
Press any key to continue...
```

# Array Object

- Contoh:

**Siswa[] s=new Siswa[5];**

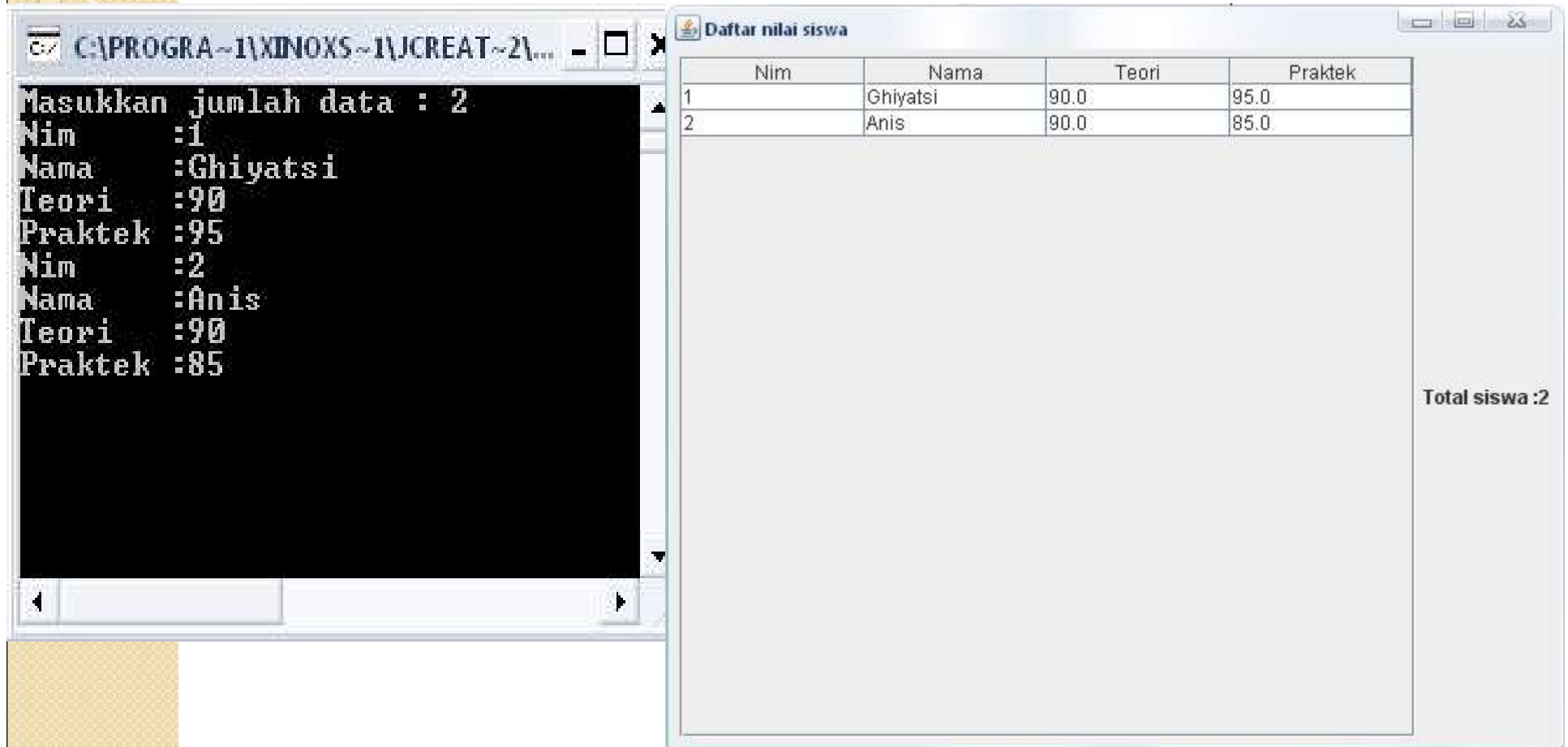
- Menciptakan variabel s yang berupa referensi ke objek null, untuk membuat objek siswa sesungguhnya, perlu dibuat instance dr masing-masing elemen.

```
for (i=0;i<5;i++)  
{s[i]=new Siswa();}
```

# Latihan

- Buat class Siswa dengan atribut nim, nama, nilai teori dan nilai praktek
- Class Siswa memiliki method setNim, setNama, setTeori, setPraktek, getNim, getNama, getTeori, getPraktek, getRata
- Buat class nilaiSiswa yang digunakan untuk memasukkan sejumlah data disimpan dalam array dan sekaligus dapat menampilkan data yang sudah di masukkan.

# Hasil yang diharapkan



The screenshot shows a C++ program running in a Windows environment. The program is titled "Daftar nilai siswa" and displays a table of student data. The output window shows the user input for two students: Ghiyatsi and Anis, with their respective scores in Theory and Practice.

**Daftar nilai siswa**

Nim	Nama	Teori	Praktek
1	Ghiyatsi	90.0	95.0
2	Anis	90.0	85.0

Total siswa :2

**Output Window:**

```
Masukkan jumlah data : 2
Nim      :1
Nama     :Ghiyatsi
Teori    :90
Praktek  :95
Nim      :2
Nama     :Anis
Teori    :90
Praktek  :85
```

# ArrayList

- Kelas yang memungkinkan pembuatan list objek array yang ukurannya dapat berubah secara dinamis atau ukuran ArrayList dapat berubah sesuai dengan jumlah data yang dimasukkan.
- Hampir mirip seperti fungsi array, arraylist digunakan untuk menyimpan data objek. Namun perbedaan dengan array biasa terletak pada tipe data dari objek yang akan disimpan ke dalam arraylist.
- ArrayList terletak pada kelas `java.util`, menggunakan ArrayList harus meng-importkan `java.util.ArrayList`



## ArrayList #2

```
ArrayList<Kelas> nama_variable = new  
ArrayList<Kelas>();
```

```
ArrayList<String> list = new ArrayList<String>();
```

Kita hanya dapat menambahkan **String** dalam objek list, Contoh :

```
list.add("Red");
```

Jika memberikan nilai bukan **String**, maka akan terjadi *error* ketika di kompilasi, contoh :

```
list.add(new Integer(1));
```

# ArrayList #3

- Dalam Array biasa, harus menunjukkan kemana arah data akan disimpan,

Misalnya :

nilaiI[1] = 3;

nilaiI[2] = 9;

nilaiI[3] = 4;

- Tetapi jika di ArrayList, hanya menambahkan saja tanpa menunjuk kemana posisi data yang ingin ditambah, secara otomatis data disimpan berdasarkan urutan array.

nilaiI.add(3);

nilaiI.add(9);

nilaiI.add(4);

# ArrayList #4

```
import java.util.ArrayList;
public class ArrayListManual{
public static void main(String args[]){
    ArrayList<String> data = new ArrayList<String>();
    //memasukkan data kedalam array
    data.add("Data 1");
    data.add("Data 2");
    data.add("Data 3");
    data.add("Data 4");
    data.add("Data 5");
    //menampilkan data
    System.out.println(data.get(0)); //data 1
    System.out.println(data.get(1)); //data 2
    System.out.println(data.get(2)); //data 3
    System.out.println(data.get(3)); //data 4
    System.out.println(data.get(4)); //data 5
}}
```

# ArrayList #5

```
import java.util.ArrayList;

public class ArrayListDenganFor {

    public static void main(String args[]) {
        ArrayList<String> data = new ArrayList<String>();
        //memasukkan data kedalam array
        for(int i=0; i<5; i++){
            data.add("Data " + (i+1));
        }
        //menampilkan data
        for(int i=0; i<5; i++){
            System.out.println(data.get(i));
        }
    }
}
```

# ArrayList #6

```
import java.util.ArrayList;

public class ArrayListPenjumlahan {

    public static void main(String args[]){
        ArrayList<Integer> nilai1 = new ArrayList<Integer>();
        ArrayList<Integer> nilai2 = new ArrayList<Integer>();
        ArrayList<Integer> jumlah = new ArrayList<Integer>();
        System.out.println("Nilai 1");
        for(int i=0; i<5; i++){
            nilai1.add(i); System.out.println("Index ke " + i + " = " +
            nilai1.get(i)); }
        System.out.println("Nilai 2");
        for(int i=0; i<5; i++){
            nilai2.add(i); System.out.println("Index ke " + i + " = " +
            nilai2.get(i)); }
        for(int i=0; i<5; i++){
            jumlah.add(nilai1.get(i) + nilai2.get(i)); }
            System.out.println("Hasil Jumlah nilai1 + nilai2");
        for(int i=0; i<5; i++){
            System.out.println("Jumlah index ke " + i + " = " +jumlah.get(i));
        }
    }
}
```

# Iterator

- Salah satu interface yang tersedia di Library Java.
- Terdapat di *java.util package*.
- Salah satu interface yang tersedia di Library Java.
- Iterator digunakan untuk membuat element-element seperti *collection*.
- ListIterator adalah *extend* dari *class* Iterator, bisa memudahkan untuk mengambil element-element yang ada di *collection* dengan cara maju atau mundur.



# Iterator

## The Methods Declared by Iterator

Sr.No.	Method & Description
1	<b>boolean hasNext( )</b> Returns true if there are more elements. Otherwise, returns false.
2	<b>Object next( )</b> Returns the next element. Throws NoSuchElementException if there is not a next element.
3	<b>void remove( )</b> Removes the current element. Throws IllegalStateException if an attempt is made to call remove( ) that is not preceded by a call to next( ).

# Iterator

## The Methods Declared by ListIterator

Sr.No.	Method & Description
1	<b>void add(Object obj)</b> Inserts obj into the list in front of the element that will be returned by the next call to next( ).
2	<b>boolean hasNext( )</b> Returns true if there is a next element. Otherwise, returns false.
3	<b>boolean hasPrevious( )</b> Returns true if there is a previous element. Otherwise, returns false.
4	<b>Object next( )</b> Returns the next element. A NoSuchElementException is thrown if there is not a next element.
5	<b>int nextIndex( )</b> Returns the index of the next element. If there is not a next element, returns the size of the list.

# Iterator

6	<b>Object previous( )</b> Returns the previous element. A <code>NoSuchElementException</code> is thrown if there is not a previous element.
7	<b>int previousIndex( )</b> Returns the index of the previous element. If there is not a previous element, returns -1.
8	<b>void remove( )</b> Removes the current element from the list. An <code>IllegalStateException</code> is thrown if <code>remove( )</code> is called before <code>next( )</code> or <code>previous( )</code> is invoked.
9	<b>void set(Object obj)</b> Assigns <code>obj</code> to the current element. This is the element last returned by a call to either <code>next( )</code> or <code>previous( )</code> .



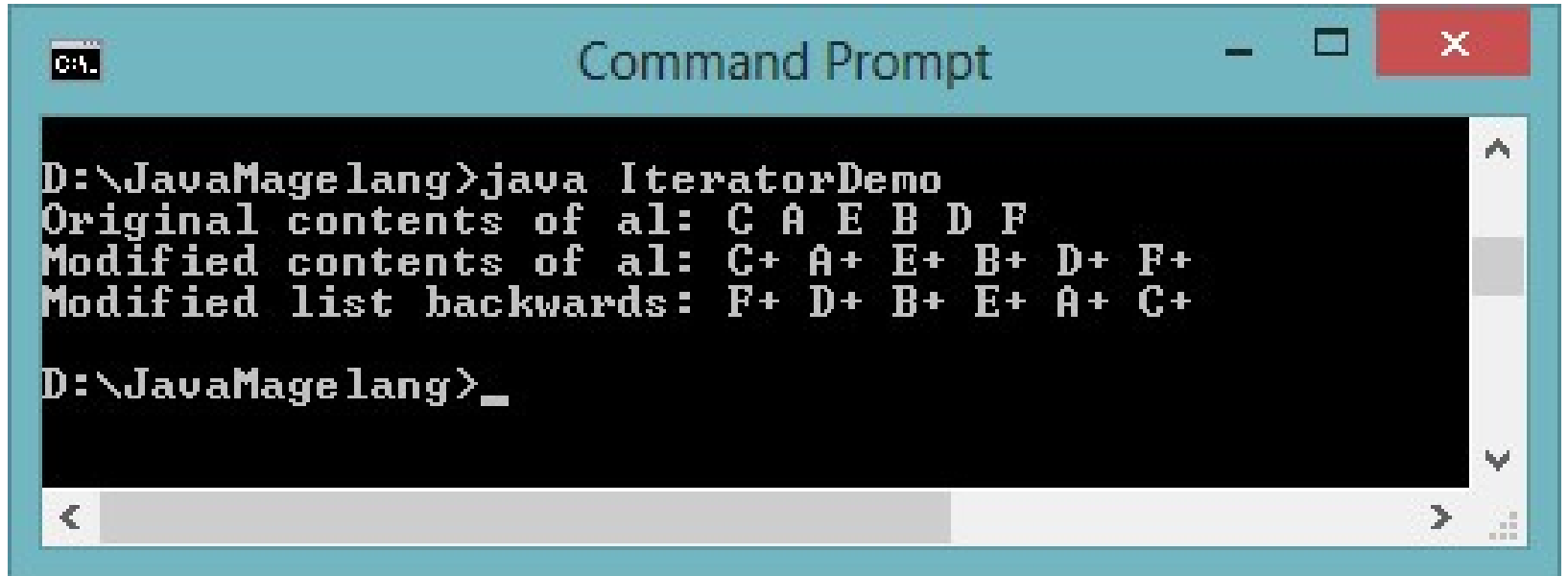
# Contoh Iterator

```
1  import java.util.*;
2  public class IteratorDemo {
3      public static void main(String args[]) {
4          // Create an array list
5          ArrayList al = new ArrayList();
6          // add elements to the array list
7          al.add("C");
8          al.add("A");
9          al.add("E");
10         al.add("B");
11         al.add("D");
12         al.add("F");
13         // Use iterator to display contents of al
14         System.out.print("Original contents of al: ");
15         Iterator itr = al.iterator();
16         while(itr.hasNext()) {
17             Object element = itr.next();
18             System.out.print(element + " ");
19         }
20         System.out.println();
```

# Contoh Iterator

```
22      // Modify objects being iterated
23      ListIterator itr = al.listIterator();
24      while(itr.hasNext()) {
25          Object element = itr.next();
26          itr.set(element + "+");
27      }
28      System.out.print("Modified contents of al: ");
29      itr = al.iterator();
30      while(itr.hasNext()) {
31          Object element = itr.next();
32          System.out.print(element + " ");
33      }
34      System.out.println();
35      // Now, display the list backwards
36      System.out.print("Modified list backwards: ");
37      while(itr.hasPrevious()) {
38          Object element = itr.previous();
39          System.out.print(element + " ");
40      }
41      System.out.println();
42  }
43 }
```

# Contoh Iterator



```
Command Prompt

D:\JavaMagelang>java IteratorDemo
Original contents of al: C A E B D F
Modified contents of al: C+ A+ E+ B+ D+ F+
Modified list backwards: F+ D+ B+ E+ A+ C+

D:\JavaMagelang>_
```

The screenshot shows a Windows Command Prompt window with a light blue title bar and standard window controls. The command prompt displays the execution of a Java program named 'IteratorDemo'. The output shows the original contents of a list 'al' as 'C A E B D F', the modified contents as 'C+ A+ E+ B+ D+ F+', and the modified list backwards as 'F+ D+ B+ E+ A+ C+'. The prompt ends with a cursor on a new line.



# Vector

- Hampir serupa dengan *ArrayList*.
- Memiliki sejumlah *method* yang sama dengan *ArrayList*, digunakan untuk memanipulasi suatu *vector*.
  - **size()**, untuk mencari panjang *ArrayList*
  - **add()**, untuk menambah elemen baru
  - **get()**, untuk mengambil elemen pada indeks tertentu
  - **isEmpty()**, untuk memeriksa apakah *ArrayList* kosong atau tidak
  - **indexOf()**, untuk mengetahui indeks dari suatu nilai
  - **contains()**, untuk memeriksa apakah suatu nilai ada dalam *ArrayList*
  - **set()**, untuk menimpa nilai pada indeks tertentu
  - **remove()**, untuk menghapus nilai pada indeks tertentu

# Vector

- Operasi di Vector
  - **firstElement()**, mengambil nilai elemen pertama dari *vector*
  - **lastElement()**, mengambil nilai elemen terakhir dari *vector*
  - **insertElementAt()**, menyisipkan nilai elemen pada indeks tertentu di dalam *vector*

# Contoh Vector

```
1 import java.util.Vector;
2 class DemoVector {
3     public static void main (String[] args){
4         Vector<Integer> deret = new Vector<Integer>();
5
6         System.out.println("Panjang deret awal: "+deret.size());
7
8         for (int i = 0; i < 10; i++){
9             deret.add(i * 10);
10        }
11
12        System.out.println("\nPanjang deret setelah ditambah elemen: "+deret.size());
13
14        for (int i = 0; i < deret.size(); i++){
15            System.out.println("elemen ke - " + i + " : " + deret.get(i));
16        }
17
18        System.out.println("\nMemeriksa apakah vector kosong: ");
19        System.out.println(deret.isEmpty());
20
21        System.out.println("\nMencari indeks dari suatu nilai di dalam vector: ");
22        System.out.println(deret.indexOf(60));
```

# Contoh Vector

```
23 System.out.println(deret.indexOf(70));
24 System.out.println(deret.indexOf(80));
25
26 System.out.println("\nMemeriksa keberadaan suatu nilai di dalam vector: ");
27 System.out.println(deret.contains(60));
28 System.out.println(deret.contains(70));
29 System.out.println(deret.contains(80));
30
31 System.out.println("\nMencari elemen pertama dan terakhir di dalam vector: ");
32 System.out.println(deret.firstElement());
33 System.out.println(deret.lastElement());
34
35 System.out.println("\nUpdate suatu nilai di dalam vector: ");
36 deret.set(6, 600);
37 deret.set(7, 700);
38 deret.set(8, 800);
39
40 System.out.println(deret.get(6));
41 System.out.println(deret.get(7));
42 System.out.println(deret.get(8));
43
44 System.out.println("\nMenghapus suatu nilai di dalam vector... ");
```

# Contoh Vector

```
45 deret.remove(6);
46 deret.remove(6);
47 deret.remove(6);
48
49 System.out.println("\nPanjang deret setelah menghapus beberapa elemen: "+
50 deret.size());
51
52 for (int i = 0; i < deret.size(); i++){
53     System.out.println("elemen ke - " + i + " : " + deret.get(i));
54 }
55
56 System.out.println("\nMenyisipkan suatu nilai di dalam vector... ");
57 deret.insertElementAt( 35005, 3 );
58
59 System.out.println("\nPanjang deret setelah menyisipkan elemen: "+deret.size());
60
61 for (int i = 0; i < deret.size(); i++){
62     System.out.println("elemen ke - " + i + " : " + deret.get(i));
63 }
64 }
65 }
```



# Stack

- Stack adalah subkelas dari Vector yang mengimplementasikan LIFO.
- Stack hanya mendefinisikan konstruktor default, yang menciptakan tumpukan kosong.
- Stack mencakup semua metode yang didefinisikan oleh Vector, dan menambahkan beberapa fiturnya sendiri.
- Terlepas dari metode yang diwarisi dari kelas induknya Vector, Stack mendefinisikan metode berikut :



# Stack

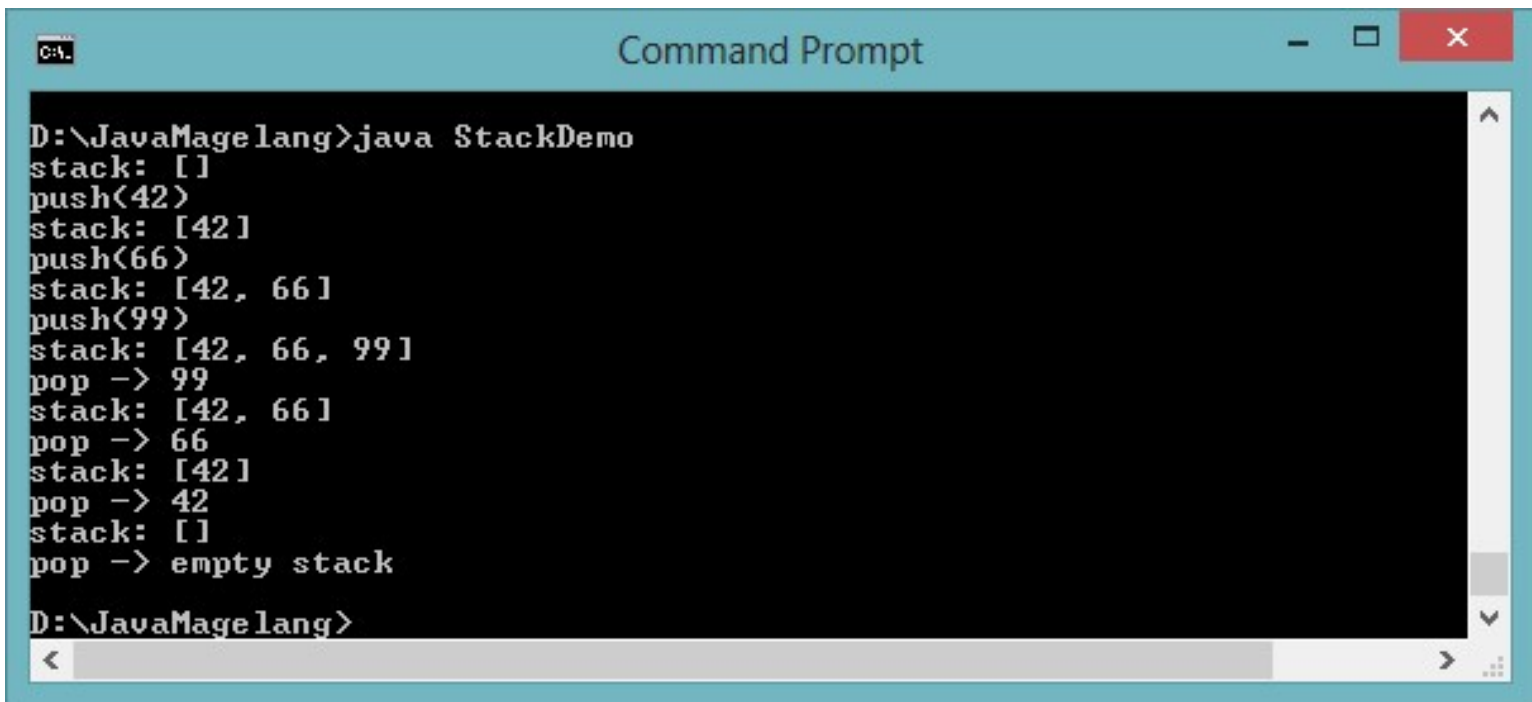
Sr.No.	Method & Description
1	<b>boolean empty()</b> Tests if this stack is empty. Returns true if the stack is empty, and returns false if the stack contains elements.
2	<b>Object peek( )</b> Returns the element on the top of the stack, but does not remove it.
3	<b>Object pop( )</b> Returns the element on the top of the stack, removing it in the process.
4	<b>Object push(Object element)</b> Pushes the element onto the stack. Element is also returned.
5	<b>int search(Object element)</b> Searches for element in the stack. If found, its offset from the top of the stack is returned. Otherwise, -1 is returned.

# Contoh Stack

```
1  import java.util.*;
2  public class StackDemo {
3      static void showpush(Stack st, int a) {
4          st.push(new Integer(a));
5          System.out.println("push(" + a + ")");
6          System.out.println("stack: " + st);
7      }
8      static void showpop(Stack st) {
9          System.out.print("pop -> ");
10         Integer a = (Integer) st.pop();
11         System.out.println(a);
12         System.out.println("stack: " + st);
13     }
14     public static void main(String args[]) {
15         Stack st = new Stack();
16         System.out.println("stack: " + st);
17         showpush(st, 42);
18         showpush(st, 66);
19         showpush(st, 99);
20         showpop(st);
21         showpop(st);
22         showpop(st);
```

# Contoh Stack

```
23     try {  
24         showpop(st);  
25     } catch (EmptyStackException e) {  
26         System.out.println("empty stack");  
27     }  
28 }  
29 }
```



```
C:\>  
D:\JavaMagelang>java StackDemo  
stack: []  
push(42)  
stack: [42]  
push(66)  
stack: [42, 66]  
push(99)  
stack: [42, 66, 99]  
pop -> 99  
stack: [42, 66]  
pop -> 66  
stack: [42]  
pop -> 42  
stack: []  
pop -> empty stack  
D:\JavaMagelang>
```

# Queue

- `java.util.Queue` adalah subtype antarmuka `java.util.Collection`.
- Mengikuti prinsip FIFO.
- Ada banyak cara untuk menginisialisasi objek Antrian, yang paling umum :
  - Sebagai Antrian Prioritas
  - Sebagai `LinkedList`
- Perlu diketahui bahwa kedua implementasinya tidak aman.
- `PriorityBlockingQueue` adalah salah satu implementasi alternatif jika memerlukan implementasi yang aman.



# Queue

## Operations on Queue :

- **Add()**-Adds an element at the tail of queue. More specifically, at the last of linkedlist if it is used, or according to the priority in case of priority queue implementation.
- **peek()**-To view the head of queue without removing it. Returns null if queue is empty.
- **element()**-Similar to peek(). Throws NoSuchElementException if queue is empty.
- **remove()**-Removes and returns the head of the queue. Throws NoSuchElementException when queue is empty.
- **poll()**-Removes and returns the head of the queue. Returns null if queue is empty.

Since it is a subtype of Collections class, it inherits all the methods of it namely **size()**, **isEmpty()**, **contains()** etc.



# Contoh Queue

```
// Java program to demonstrate working of Queue
// interface in Java
import java.util.LinkedList;
import java.util.Queue;

public class QueueExample
{
    public static void main(String[] args)
    {
        Queue<Integer> q = new LinkedList<>();

        // Adds elements {0, 1, 2, 3, 4} to queue
        for (int i=0; i<5; i++)
            q.add(i);

        // Display contents of the queue.
        System.out.println("Elements of queue-"+q);

        // To remove the head of queue.
        int removedele = q.remove();
        System.out.println("removed element-" + removedele);

        System.out.println(q);

        // To view the head of queue
        int head = q.peek();
        System.out.println("head of queue-" + head);

        // Rest all methods of collection interface,
        // Like size and contains can be used with this
        // implementation.
        int size = q.size();
        System.out.println("Size of queue-" + size);
    }
}
```

Output:

```
Elements of queue-[0, 1, 2, 3, 4]
removed element-0
[1, 2, 3, 4]
head of queue-1
Size of queue-4
```



# Latihan ArrayList

- Tambahkan method pengurangan, perkalian dan pembagian



# Ada pertanyaan





# Rehat Sejenak

- Berkeringat
- Big Package

# Referensi

- <https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html>
- [https://www.tutorialspoint.com/java/java\\_using\\_iterator.htm](https://www.tutorialspoint.com/java/java_using_iterator.htm)
- [https://www.tutorialspoint.com/java/java\\_vector\\_class.htm](https://www.tutorialspoint.com/java/java_vector_class.htm)
- [https://www.tutorialspoint.com/java/java\\_stack\\_class.htm](https://www.tutorialspoint.com/java/java_stack_class.htm)
- <https://www.codepolitan.com/menggunakan-vector-di-java>
- [https://www.tutorialspoint.com/java/util/java\\_util\\_stack.htm](https://www.tutorialspoint.com/java/util/java_util_stack.htm)
- <https://www.sanfoundry.com/java-program-implement-queue/>
- <https://www.geeksforgeeks.org/queue-interface-java/>
- <http://homepage.divms.uiowa.edu/~sriram/21/spring07/code/queue.java>