

Nama: Arya Raihan Hanif

Kelas: Praktikum Pemrograman C

NPM: 233040101

Link Repo: https://github.com/aryahanif/PP12025_C_233040101

List (Operasi Add Head & Tail)

Latihan 1

Latihan ini akan memberikan implementasi pembuatan elemen list. Elemen list direpresentasikan dengan Node. Sebuah node terdiri dari atribut data/nilai dan atribut next. Atribut next akan menunjuk ke node yang lain.

Pseudocode	Bahasa Pemrograman
<pre>class Node { data: tipe data (T) next: Node }</pre>	<pre>public class Node { private int data; private Node next; /** Inisialisasi atribut node */ public Node(int data) { this.data = data; } /** Setter & Getter */ }</pre>

Jawab:

```
package Pertemuan3;
```

```
public class Node {  
    private int data;  
    private Node next;  
  
    //inisialisasi Node  
    public Node(int data) {  
        this.data = data;  
        this.next = null;  
    }  
  
    //setter and getter  
    public void setData(int data) {  
        this.data = data;  
    }  
  
    public int getData() {  
        return data;  
    }  
  
    //setter untuk next  
    public void setNext(Node next) {
```

```

        this.next = next;
    }

    //getter untuk next
    public Node getNext() {
        return next;
    }
}

```

Latihan 2

Latihan ini akan memberikan implementasi operasi penambahan elemen list di akhir/tail dengan notasi algoritma. Operasi ini direpresentasikan dengan fungsi addTail dengan parameter data yaitu node yang akan ditambahkan ke List.

- Buatlah kelas StrukturList kemudian tambahkan atribut HEAD dengan tipe data Node
- Tambahkan fungsi dibawah ini di kelas StrukturList. Fungsi addTail di bawah dikonversi ke dalam bahasa pemrograman
- function isEmpty() harus diimplementasikan dengan melakukan apakah list kosong atau tidak (HEAD != null)!

Algoritma addTail	Program addTail
<pre> procedure addTail(data: integer) deklarasi posNode, curNode: Node {current node} deskripsi newNode ← new Node(data) IF (isEmpty()) THEN HEAD ← newNode ELSE curNode ← HEAD WHILE (curNode <> null) DO posNode ← curNode curNode ← curNode.next ENDWHILE posNode.next ← newNode ENDIF </pre>	<pre> public void addTail(int data) { Node posNode=null, curNode=null; Node newNode = new Node(data); if (isEmpty()) { HEAD = newNode; } else { curNode = HEAD; while (curNode != null) { posNode = curNode; curNode = curNode.getNext(); } posNode.setNext(newNode); } } </pre>

Latihan 3

Latihan ini akan memberikan implementasi untuk menampilkan elemen list. Elemen list yang ditampilkan ke layar diawali dari nilai HEAD.

Algoritma	Bahasa Pemrograman
<pre>procedure displayElement() deklarasi curNode: Node deskripsi curNode ← HEAD; WHILE (curNode <> null) DO print(curNode.data) curNode ← curNode.next ENDWHILE</pre>	<pre>public void displayElement() { Node curNode = HEAD; while (curNode != null) { System.out.print(curNode.getData() + " "); curNode = curNode.getNext(); } }</pre>

Jawab:

```
package Pertemuan3;
```

```
public class StrukturList {
    private Node HEAD;
```

```
    public boolean isEmpty() {
        return HEAD == null;
    }
```

```
    public void addHead(int data) {
        Node newNode = new Node(data);
        if (isEmpty()) {
            HEAD = newNode;
        }

        else {
            newNode.setNext(HEAD);
            HEAD = newNode;
        }
    }
```

```
    public void addTail(int data) {
        Node posNode = null, curNode = null;

        Node newNode = new Node(data);
        if (isEmpty()) {
            HEAD = newNode;
        }

        else {
            curNode = HEAD;
            while (curNode != null) {
                posNode = curNode;
            }
        }
    }
```

```

        curNode = curNode.getNext();
    }
    posNode.setNext(newNode);
}

}

public void displayElement() {
    Node curNode = HEAD;
    while (curNode != null) {
        System.out.print(curNode.getData()+ " ");
        curNode = curNode.getNext();
    }
}

}

```

Latihan 4

Latihan ini akan memberikan penggunaan operasi penambahan elemen di akhir list dan kemudian menampilkan setiap elemen yang terdapat di list. Buatlah kelas ListTest berikut fungsi main untuk mengeksekusi program. Konversikan urutan instruksi berikut di bawah ini ke fungsi tersebut!

Urutan Instruksi	Program
<ol style="list-style-type: none"> 1. Create list dengan keyword new 2. Tambah elemen 3 di akhir list 3. Tambah elemen 4 di akhir list 4. Tambah elemen 5 di akhir list 5. Tampilkan elemen list 	<pre> public class ListMain { public static void main(String[] args) { StrukturList list = new StrukturList(); list.addTail(3); list.addTail(4); list.addTail(5); System.out.println("Elemen: "); list.display(); } } </pre>

Outputnya adalah: Elemen: 3 4 5

Tes-1

Lakukan seperti diatas dengan output elemen list seperti berikut:

A. 3 2 1

B. 1 4 5 7

Jawab:

```
package Pertemuan3;
```

```
public class ListMainTail {
```

```

    public static void main(String[] args) {
        // 1. Create list dengan keyword new
        StrukturList list = new StrukturList();

```

```

        // 2. Tambah elemen 3 di akhir list

```

```

        list.addTail(3);

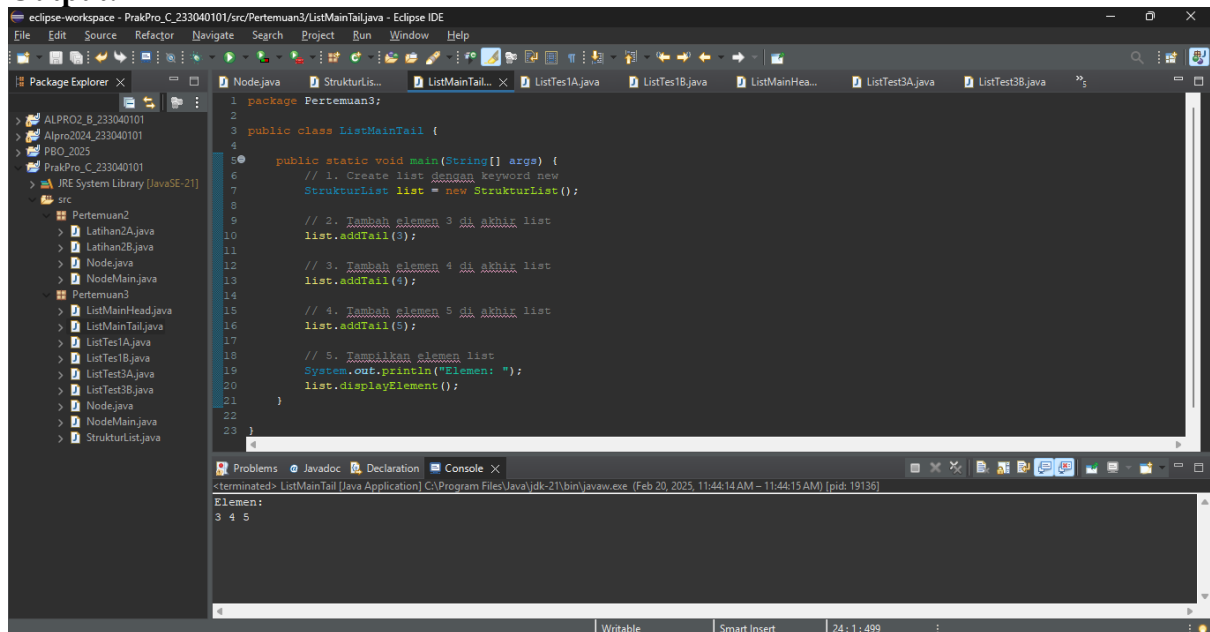
// 3. Tambah elemen 4 di akhir list
        list.addTail(4);

// 4. Tambah elemen 5 di akhir list
        list.addTail(5);

// 5. Tampilkan elemen list
        System.out.println("Elemen: ");
        list.displayElement();
    }
}

```

Output:



Tes-1A

package Pertemuan3;

public class ListTes1A {

```

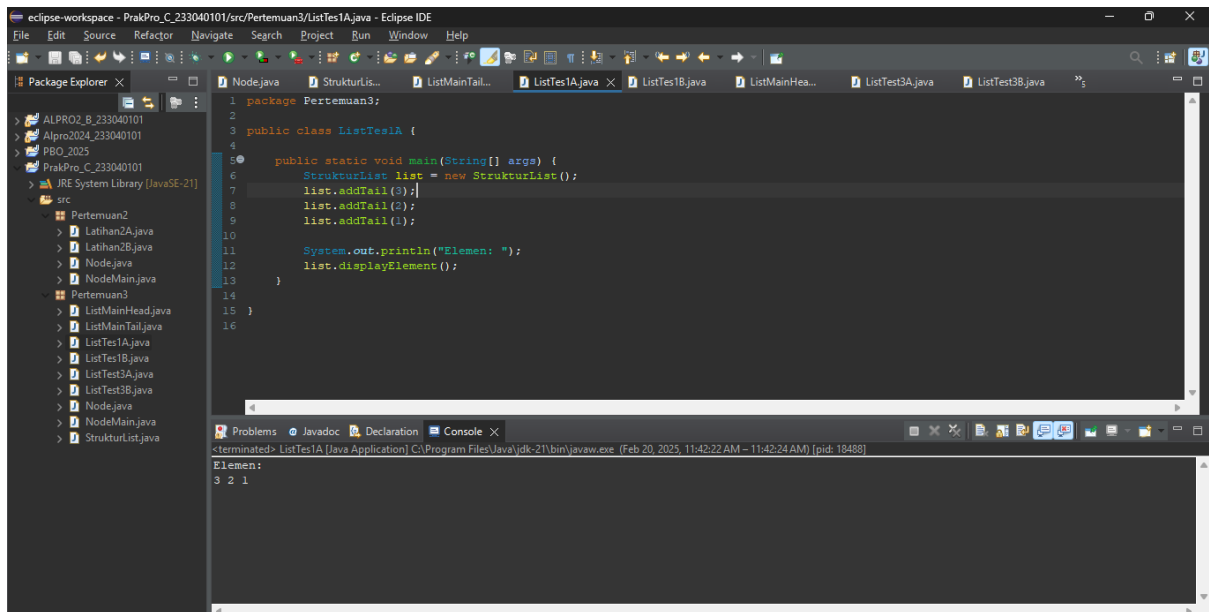
    public static void main(String[] args) {
        StrukturList list = new StrukturList();
        list.addTail(3);
        list.addTail(2);
        list.addTail(1);

        System.out.println("Elemen: ");
        list.displayElement();
    }
}

```

}

Output:



Tes-1B

package Pertemuan3;

public class ListTes1B {

```
    public static void main(String[] args) {  
        StrukturList list = new StrukturList();  
        list.addTail(1);  
        list.addTail(4);  
        list.addTail(5);  
        list.addTail(7);  
  
        System.out.println("Elemen: ");  
        list.displayElement();  
    }
```

}

Output:

```
1 package Pertemuan3;
2
3 public class ListTest1B {
4
5     public static void main(String[] args) {
6         StrukturList list = new StrukturList();
7         list.addTail(1);
8         list.addTail(4);
9         list.addTail(5);
10        list.addTail(7);
11
12        System.out.println("Elemen: ");
13        list.displayElement();
14    }
15 }
16
17
```

Elemen:
1 4 5 7

Latihan 5

Latihan ini akan memberikan implementasi operasi penambahan elemen list di awal/head. Operasi ini direpresentasikan dengan procedure addHead dengan parameter data yang akan ditambahkan. Tambahkan atribut HEAD dengan tipe data Node kelas StrukturList.

Algoritma addHead	Bahasa Pemrograman
<pre>procedure addHead(data: integer) deskripsi newNode ← new Node(data); IF (HEAD = null) THEN HEAD ← newNode ELSE newNode.next ← HEAD HEAD ← newNode ENDIF</pre>	<pre>public void addHead(int data) { Node newNode = new Node(data); if (isEmpty()) { HEAD = newNode; } else { newNode.setNext(HEAD); HEAD = newNode; } }</pre>

Tes-2

Latihan ini akan memberikan penggunaan operasi penambahan elemen di awal list dan kemudian menampilkan setiap elemen yang terdapat di list. Konversikan urutan instruksi berikut di bawah ini ke dalam Bahasa pemrograman!

Urutan Instruksi	Output
<ol style="list-style-type: none">1. Create list dengan keyword new2. Tambah elemen 5 di awal list3. Tambah elemen 4 di awal list4. Tambah elemen 3 di awal list5. Tampilkan elemen list	3 4 5

Tes-3

Lakukan seperti diatas dengan output elemen list seperti berikut:

A. 3 2 1

B. 1 4 5 7

Jawab:

package Pertemuan3;

```
public class StrukturList {
    private Node HEAD; // Pointer ke node pertama dalam list (head)

    // Method untuk mengecek apakah list kosong
    public boolean isEmpty() {
        return HEAD == null; // Jika HEAD null, berarti list kosong
    }

    // Method untuk menambahkan elemen di awal (head) list
    public void addHead(int data) {
        Node newNode = new Node(data); // Buat node baru dengan data yang
        diberikan
        if (isEmpty()) { // Jika list kosong
            HEAD = newNode; // Jadikan node baru sebagai head
        } else { // Jika list tidak kosong
            newNode.setNext(HEAD); // Hubungkan node baru ke head lama
            HEAD = newNode; // Jadikan node baru sebagai head
        }
    }

    // Method untuk menambahkan elemen di akhir (tail) list
    public void addTail(int data) {
        Node posNode = null, curNode = null;

        Node newNode = new Node(data); // Buat node baru dengan data yang
        diberikan
        if (isEmpty()) { // Jika list kosong
            HEAD = newNode; // Jadikan node baru sebagai head
        } else {
            curNode = HEAD; // Mulai dari head
            while (curNode != null) { // Perulangan untuk mencapai node terakhir
                posNode = curNode; // Simpan node sebelumnya
                curNode = curNode.getNext(); // Pindah ke node berikutnya
            }
            posNode.setNext(newNode); // Hubungkan node terakhir ke node baru
        }
    }

    // Method untuk menampilkan elemen-elemen dalam list
    public void displayElement() {
        Node curNode = HEAD; // Mulai dari head
```



```

        while (curNode != null) { // Perulangan selama masih ada node
            System.out.print(curNode.getData() + " "); // Cetak data node saat ini
            curNode = curNode.getNext(); // Pindah ke node berikutnya
        }
        System.out.println(); // Tambahkan baris baru setelah mencetak semua elemen
    }
}

```

Tes-2

package Pertemuan3;

```

public class ListMainHead {

    public static void main(String[] args) {
        // 1. Create list dengan keyword new
        StrukturList list = new StrukturList();

        // 2. Tambah elemen 5 di awal list
        list.addHead(5);

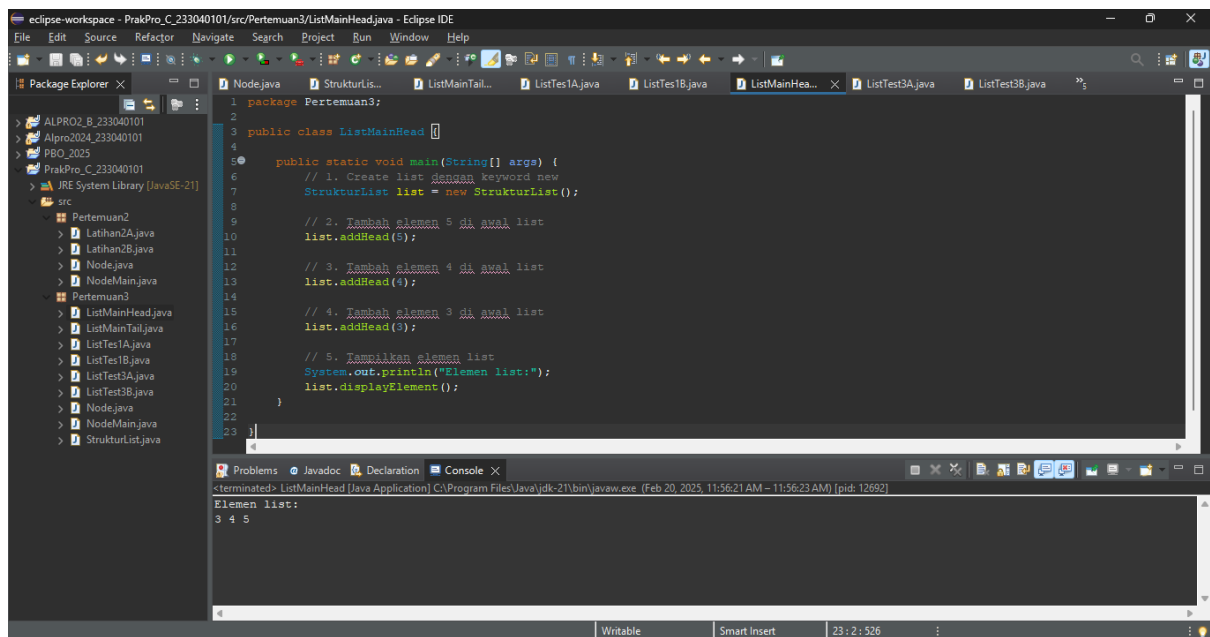
        // 3. Tambah elemen 4 di awal list
        list.addHead(4);

        // 4. Tambah elemen 3 di awal list
        list.addHead(3);

        // 5. Tampilkan elemen list
        System.out.println("Elemen list:");
        list.displayElement();
    }
}

```

Output:



Tes-3A

package Pertemuan3;

public class ListTest3A {

public static void main(String[] args) {
 // 1. Create list dengan keyword new
 StrukturList list = new StrukturList();

// 2. Tambah elemen 1 di awal list
 list.addHead(1);

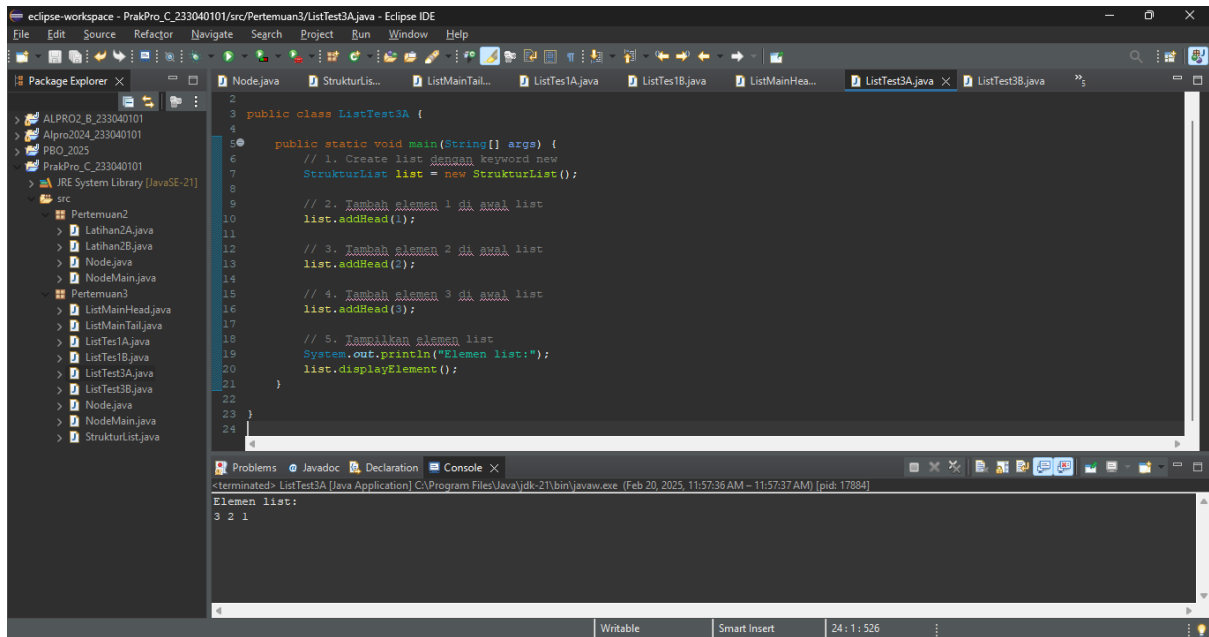
// 3. Tambah elemen 2 di awal list
 list.addHead(2);

// 4. Tambah elemen 3 di awal list
 list.addHead(3);

// 5. Tampilkan elemen list
 System.out.println("Elemen list:");
 list.displayElement();
 }

}

Output:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure with a package named 'Pertemuan3' containing several Java files, including 'ListTest3A.java'. The main editor window shows the source code of 'ListTest3A.java'. The code defines a 'StrukturList' class and a 'ListTest3A' class. The 'ListTest3A' class has a 'main' method that creates a 'StrukturList' object, adds elements 1, 2, and 3 to the list, and then prints the list. The console at the bottom shows the output of the program: 'Elemen list: 3 2 1'.

```
2 public class ListTest3A {
3
4
5     public static void main(String[] args) {
6         // 1. Create list dengan keyword new
7         StrukturList list = new StrukturList();
8
9         // 2. Tambah elemen 1 di awal list
10        list.addHead(1);
11
12        // 3. Tambah elemen 2 di awal list
13        list.addHead(2);
14
15        // 4. Tambah elemen 3 di awal list
16        list.addHead(3);
17
18        // 5. Tampilkan elemen list
19        System.out.println("Elemen list:");
20        list.displayElement();
21    }
22 }
23
24
```

Elemen list:
3 2 1

Tes-3B

package Pertemuan3;

```
public class ListTest3B {

    public static void main(String[] args) {
        // 1. Create list dengan keyword new
        StrukturList list = new StrukturList();

        // 2. Tambah elemen 7 di awal list
        list.addHead(7);

        // 3. Tambah elemen 5 di awal list
        list.addHead(5);

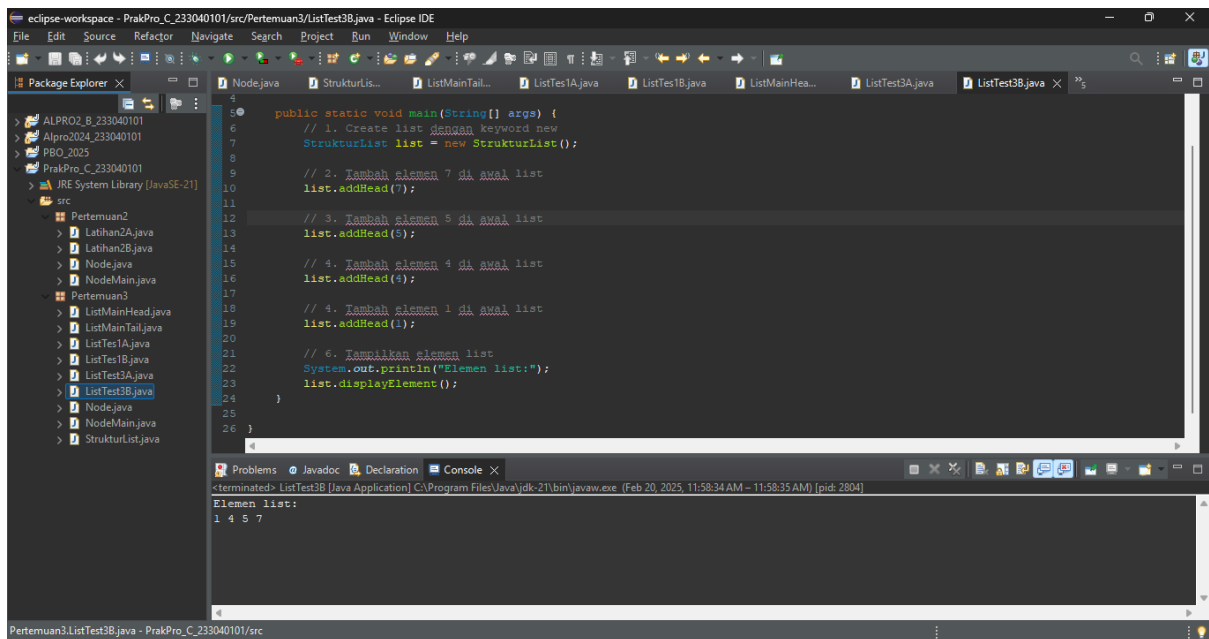
        // 4. Tambah elemen 4 di awal list
        list.addHead(4);

        // 4. Tambah elemen 1 di awal list
        list.addHead(1);

        // 6. Tampilkan elemen list
        System.out.println("Elemen list:");
        list.displayElement();
    }

}
```

Output:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project structure with a package named 'Pertemuan3' containing several Java files, including 'ListTest3B.java'. The main editor window shows the source code of 'ListTest3B.java', which implements a linked list structure. The code includes a 'main' method that creates a 'StrukturList' object, adds elements 7, 5, 4, and 1 to the list, and then prints the list elements. The Console window at the bottom shows the output of the program, which is 'Elemen list: 1 4 5 7'.

```
public static void main(String[] args) {  
    // 1. Create list dengan keyword new  
    StrukturList list = new StrukturList();  
  
    // 2. Tambah elemen 7 di awal list  
    list.addHead(7);  
  
    // 3. Tambah elemen 5 di awal list  
    list.addHead(5);  
  
    // 4. Tambah elemen 4 di awal list  
    list.addHead(4);  
  
    // 4. Tambah elemen 1 di awal list  
    list.addHead(1);  
  
    // 6. Tampilkan elemen list  
    System.out.println("Elemen list:");  
    list.displayElement();  
}
```

Elemen list:
1 4 5 7