

Pengenalan doubly linked list

- Apa itu doubly linked list?

Salah satu bentuk struktur data yang dikenal sebagai senarai berantai ganda (Doubly linked list) terdiri dari sejumlah simpul berantai (linked list) yang saling terhubung satu sama lain dan mengizinkan penjelajahan di kedua arah dari satu simpul (node). Setiap simpul dari senarai berantai ganda berisi bidang data, penunjuk sebelumnya, dan penunjuk berikutnya¹².

Implementasi Java dari senarai berantai ganda membutuhkan pengembangan kelas simpul dan penambahan simpul ke dalam senarai. Menjelang akhir daftar, node baru sering ditambahkan³.

- Keuntungan dan kerugian dari senarai berantai ganda (doubly linked list)

Manfaat Senarai Berantai Ganda

- Senarai berantai ganda memungkinkan penjelajahan dua arah yang efisien, yang membuatnya bermanfaat dalam beberapa aplikasi karena dapat bergerak maju dan mundur.
- Memberikan kebebasan tambahan pada algoritma dan struktur data tertentu.
- Ini membuat penghapusan simpul lebih cepat.
- Dapat digunakan untuk mengimplementasikan banyak struktur data yang berbeda.
- Dapat dengan mudah mengalokasikan ulang atau mengalokasikan memori saat sedang berjalan.
- Dalam beberapa aplikasi pemrograman, ini lebih refleksif dan lebih mudah digunakan.
- Konstruksi cache MRU/LRU (Most/Least Recently Used) menggunakannya.
- Ini dapat digunakan untuk menavigasi maju dan mundur melalui halaman online.
- Berfungsi sebagai setumpuk kartu dalam permainan.

Kekurangan Senarai Berantai Ganda:

- Membutuhkan lebih banyak memori untuk menyimpan pointer ekstra, yang dapat menyebabkan peningkatan penggunaan memori.
- Implementasi yang lebih kompleks daripada daftar berantai tunggal (Single linked list).
- Mereka membutuhkan operasi ekstra untuk menjaga integritas daftar (list).
- Penjelajahannya (Traversal) lebih lambat daripada larik untuk akses acak.
- Karena elemen dalam memori disimpan secara acak, maka elemen-elemen tersebut diakses secara berurutan, dan tidak ada akses langsung yang diperbolehkan.
- Pembaruan pada daftar, seperti menyisipkan atau menghapus elemen, dapat lebih memakan waktu dibandingkan dengan array atau struktur data lainnya.
- Ini menggunakan memori ekstra bila dibandingkan dengan larik dan senarai berantai tunggal.
- Manajemen penunjuk dapat menjadi rumit.

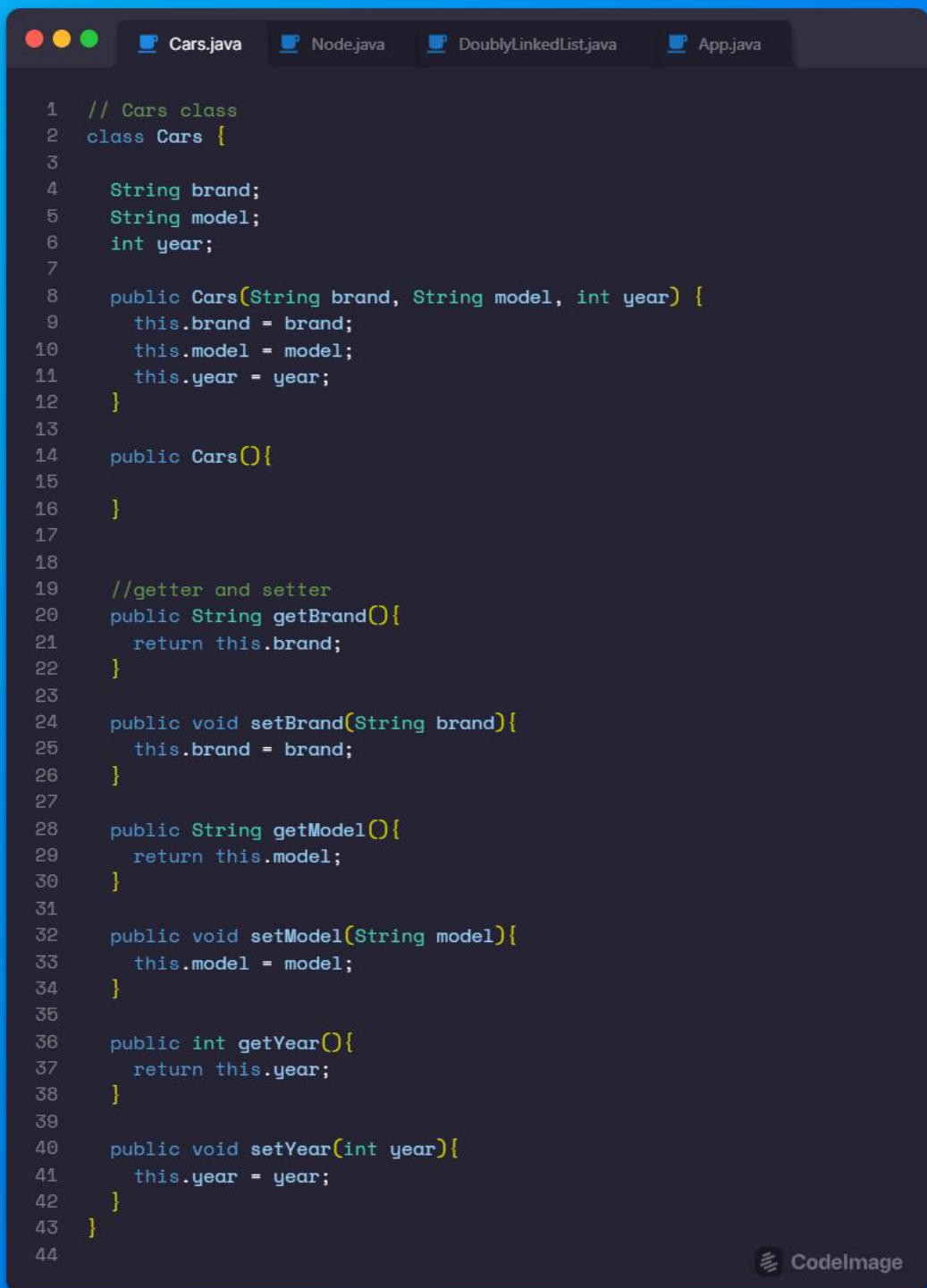
Membuat dan memanipulasi double linked list

- Bagaimana cara membuat doubly linked list?


¹ <https://www.scaler.com/topics/doubly-linked-list-in-java/>

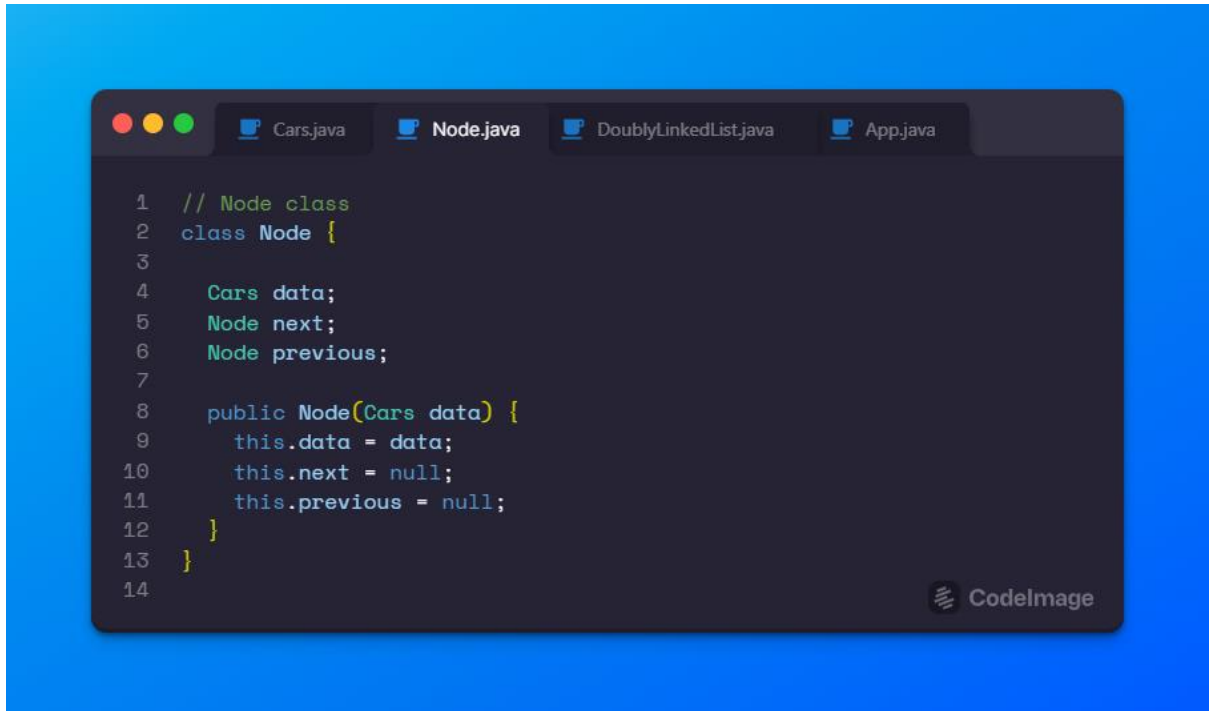
² <https://www.javatpoint.com/java-program-to-create-and-display-a-doubly-linked-list>

³ <https://www.softwaretestinghelp.com/doubly-linked-list-in-java/>



```
1 // Cars class
2 class Cars {
3
4     String brand;
5     String model;
6     int year;
7
8     public Cars(String brand, String model, int year) {
9         this.brand = brand;
10        this.model = model;
11        this.year = year;
12    }
13
14    public Cars(){
15
16    }
17
18    //getter and setter
19    public String getBrand(){
20        return this.brand;
21    }
22
23    public void setBrand(String brand){
24        this.brand = brand;
25    }
26
27    public String getModel(){
28        return this.model;
29    }
30
31    public void setModel(String model){
32        this.model = model;
33    }
34
35    public int getYear(){
36        return this.year;
37    }
38
39    public void setYear(int year){
40        this.year = year;
41    }
42 }
43
44
```

 CodeImage



```
1 // Node class
2 class Node {
3
4     Cars data;
5     Node next;
6     Node previous;
7
8     public Node(Cars data) {
9         this.data = data;
10        this.next = null;
11        this.previous = null;
12    }
13 }
14
```

The image shows a code editor window with four tabs: Cars.java, Node.java, DoublyLinkedList.java, and App.java. The Node.java tab is active, displaying the code for the Node class. The code defines a Node class with attributes data (type Cars), next (type Node), and previous (type Node). It includes a constructor that initializes these attributes. The background of the editor is dark, and the text is light-colored. The CodeImage logo is visible in the bottom right corner of the editor window.

- Bagaimana cara menambahkan dan menghapus di doubly linked list?

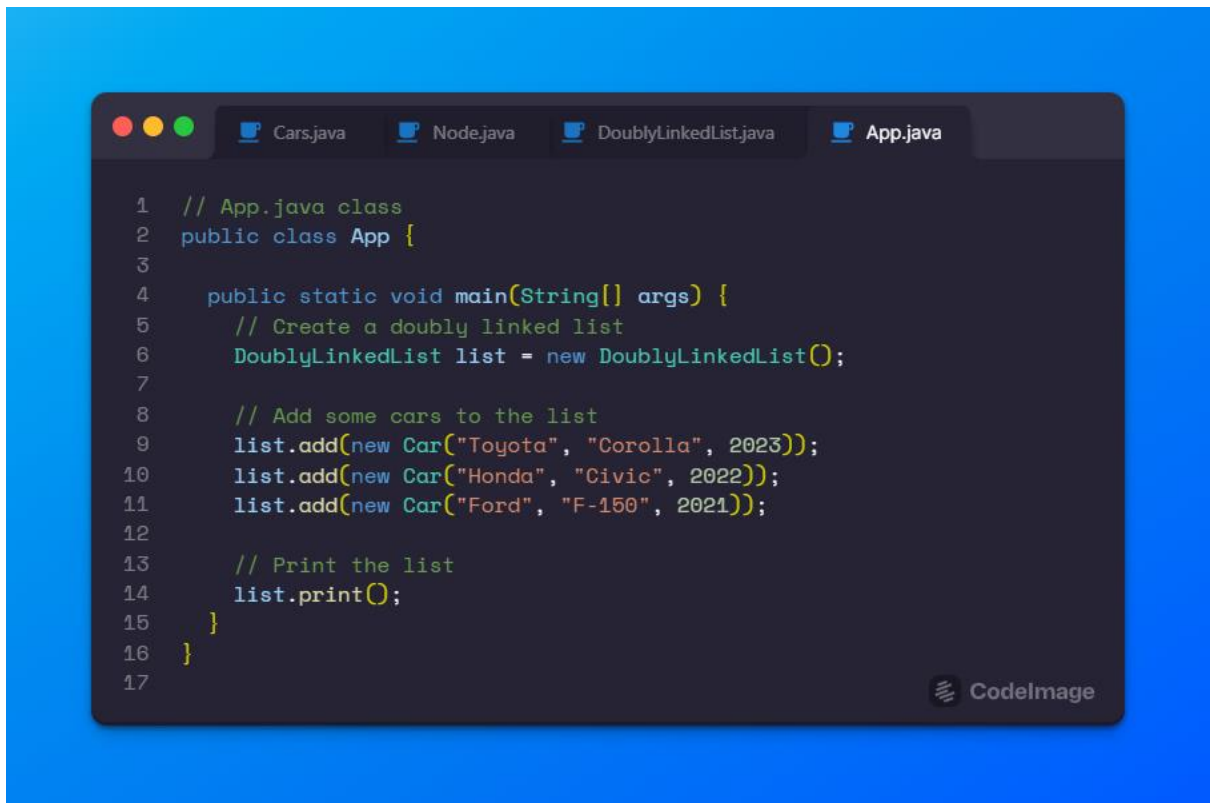
Cars.java Node.java DoublyLinkedList.java App.java

```
1 // DoublyLinkedList class
2 class DoublyLinkedList {
3
4     Node head;
5     Node tail;
6
7     public DoublyLinkedList() {
8         head = null;
9         tail = null;
10    }
11
12    public void add(Cars car) {
13        Node node = new Node(car);
14
15        if (head == null) {
16            head = node;
17            tail = node;
18        } else {
19            tail.next = node;
20            node.previous = tail;
21            tail = node;
22        }
23    }
24
25    public void remove(String model) {
26        Node current = head;
27
28        while (current != null) {
29            if (current.data.equals(model)) {
30                if (current == head) {
31                    head = current.next;
32                } else {
33                    current.previous.next = current.next;
34                }
35
36                if (current == tail) {
37                    tail = current.previous;
38                } else {
39                    current.next.previous = current.previous;
40                }
41            }
42            current = current.next;
43        }
44    }
45
46    public void print() {
47        Node current = head;
48
49        while (current != null) {
50            System.out.println(current.data);
51            current = current.next;
52        }
53    }
54 }
55
56
```

- Bagaimana cara melakukan perulangan dengan doubly linked list?

Cars.java Node.java DoublyLinkedList.java App.java

```
1 // DoublyLinkedList class
2 class DoublyLinkedList {
3
4     Node head;
5     Node tail;
6
7     public DoublyLinkedList() {
8         head = null;
9         tail = null;
10    }
11
12    public void add(Cars car) {
13        Node node = new Node(car);
14
15        if (head == null) {
16            head = node;
17            tail = node;
18        } else {
19            tail.next = node;
20            node.previous = tail;
21            tail = node;
22        }
23    }
24
25    public void remove(String model) {
26        Node current = head;
27
28        while (current != null) {
29            if (current.data.equals(model)) {
30                if (current == head) {
31                    head = current.next;
32                } else {
33                    current.previous.next = current.next;
34                }
35
36                if (current == tail) {
37                    tail = current.previous;
38                } else {
39                    current.next.previous = current.previous;
40                }
41            }
42            current = current.next;
43        }
44    }
45
46    public void print() {
47        Node current = head;
48
49        while (current != null) {
50            System.out.println(current.data);
51            current = current.next;
52        }
53    }
54 }
55
56
```

A screenshot of a code editor window with a dark theme. The window has four tabs at the top: 'Cars.java', 'Node.java', 'DoublyLinkedList.java', and 'App.java'. The 'App.java' tab is active. The code is written in Java and is as follows:

```
1 // App.java class
2 public class App {
3
4     public static void main(String[] args) {
5         // Create a doubly linked list
6         DoublyLinkedList list = new DoublyLinkedList();
7
8         // Add some cars to the list
9         list.add(new Car("Toyota", "Corolla", 2023));
10        list.add(new Car("Honda", "Civic", 2022));
11        list.add(new Car("Ford", "F-150", 2021));
12
13        // Print the list
14        list.print();
15    }
16 }
17
```

The code is numbered from 1 to 17. The 'CodeImage' logo is visible in the bottom right corner of the editor window.

Latihan

Write a program in Java to implement the following operations on a doubly linked list:

1. Create a doubly linked list.
2. Display the doubly linked list.
3. Insert a node at the beginning of the doubly linked list.
4. Insert a node at the end of the doubly linked list.
5. Insert a node at a given position in the doubly linked list.
6. Delete a node from the beginning of the doubly linked list.
7. Delete a node from the end of the doubly linked list.
8. Delete a node from a given position in the doubly linked list.
9. Reverse the doubly linked list.
10. Check if the doubly linked list is empty.