Bab 2 Single Link List

Identitas

Kajian

Single Link List

Topik

1. Single Link List

Referensi

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms Third Edition. Cambridge: The MIT Press.
2. Sedgewick, R. (2002). Algorithm in Java: parts 1-4 Third Edition. Boston: Pearson Education, Inc.
3. P. Deitel and H. Deitel, Java How To Program 9th Edition, Prentice Hall, 2011.
4. Poo, Danny, Derek Kiong, and Swarnalatha Ashok. Object-Oriented Programming and Java, 2nd Edition. Springer, 2010.
5. Wu, C. Thomas. An Introduction to Object-Oriented Programming with Java. McGraw-Hill, 2009.

Kompetensi Utama

1. Mahasiswa mampu menggunakan bahasa java untuk membangun sebuah aplikasi sederhana yang digunakan untuk mengelola data dalam struktur Single Link list

Lama Kegiatan Kajian

1. Pertemuan Terbimbing : 2 x 500 menit
2. Kegiatan Mandiri : 1 x 70 menit

Parameter Penilaian

1. Jurnal 60%
2. Tugas Akhir 40%

Pengumpulan

Kumpulkan jawaban dari Jurnal tembimbing dan jurnal mandiri pada LMS yang telah disediakan

Jurnal terbimbing

Buatlah sebuah aplikasi yang dapat digunakan untuk melakukan pendataan deret angka dengan memanfaatkan struktur Single Link List.

1. Deret terinput secara bebas
2. Deret terinput terurut secara ascending.

Data disimpan dalam sebuah kelas bernama Node, dimana dalam Node tersimpan informasi mengenai data itu sendiri dan informasi Node selanjutnya. Dalam kelas SingleLinkList, data pertama akan disimpan dalam sebuah Node utama yaitu Head. Berikut kode tersebut.

```
import java.io.*;
class Node {
    int data;
    Node next;

    Node(int d) {
        data = d;
        next = null;
    }
}

public class SingleLinkedList {
    Node head; // head of list

    /*Insert last tidak terurut*/
    public int insert(int data) {
           Node nn = new Node(data);
           Node current = head;

           if (current == null) {
                 nn.next=null;
                 head = nn;
           } else {
                 while (current.next != null) {
                 current = current.next;
           }
           current.next = nn;
       }
       System.out.println("Node baru "+data+" ditambahkan");
       return 0;
   }
```

```
/* Insert sorted */
public void insertSorted(int data) {
      Node nn = new Node(data);   //node baru
      Node current = head;
      // Jika list kosong atau data lebih kecil dari head
      if (head == null || head.data >= nn.data) {
            nn.next = head;
            head = nn;
      } else {
      // Cari posisi yang tepat untuk menyisipkan node
            while (current.next != null && (current.next.data < nn.data) ) {
                  current = current.next;
            }
            nn.next = current.next;
            current.next = nn;
      }
      System.out.println("Node baru "+data+" ditambahkan");
} //eoinsertSorted
```

```
/*Delete*/
    public int delete(int data) {
        // Jika list kosong
        if (head == null) {
            System.out.println("List kosong");
            return 0;
        } else {
            // Cek apakah data yang dicar ada di head
            if (head.data == data){
                head = head.next;
                System.out.println("Node "+data+" telah dihapus");
                return 0;
            } else{
                //jika data yang dicari bukan di head
                Node current = head;
                Node prec = head;
                boolean found=false;
                while (current != null) {
                    if (current.data == data){
                        prec.next = current.next;
                        current.next = null;
                        System.out.println("Node "+data+" telah dihapus");
                        return 0;
                    }
                    prec = current;
                    current = current.next;
                }
            }
        }
        System.out.println("Node "+data+" tidak ditemukan");
        return 0;
    } //eodelete
```

```
    public void printList() {
        Node current = head;
        System.out.print("Single LinkedList: ");

        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println("");
    } //eoprintlist

    public static void main(String[] args) {
        SingleLinkedList sll = new SingleLinkedList();
        sll.runThis();
    }
    void runThis(){
           //Buat proses pengisian data
    } //eoRunThis
} //eoSingleLinkList
```

Jurnal Terbimbing

1. Buatlah kode program lengkap dari contoh diatas
2. Buatlah isi dari method runThis untuk melakukan proses penambahan data untuk deret 5 1 3 2 7 6 9 8
    a. Deret tidak terurut, dengan hasil sebagai berikut
    Node baru 5 ditambahkan
    Node baru 1 ditambahkan
    Node baru 3 ditambahkan
    Node baru 2 ditambahkan
    Node baru 7 ditambahkan
    Node baru 6 ditambahkan
    Node baru 9 ditambahkan
    Node baru 8 ditambahkan
    Single LinkedList: 5 1 3 2 7 6 9 8

    b. Deret terurut, dengan hasil sebagai berikut
    Node baru 5 ditambahkan
    Node baru 1 ditambahkan
    Node baru 3 ditambahkan
    Node baru 2 ditambahkan
    Node baru 7 ditambahkan
    Node baru 6 ditambahkan
    Node baru 9 ditambahkan
    Node baru 8 ditambahkan
    Single LinkedList: 1 2 3 5 6 7 8 9
3. Lakukan penghapusan data masing-masing untuk nilai 5, 1, 9 dan 99 dan tampilkan hasilnya.

Jurnal Mandiri.

Maksimal pengumpulan adalah 1 jam setelah praktikum berakhir.

Tambahkan sebuah method, yang dapat digunakan untuk melakukan proses pencarian data yang ada didalam list.

a. Jika data ditemukan : Data [yang dicari] ditemukan pada urutan ke-N
b. Jika data tidak ditemukan : Data [yang dicari] tidak ditemukan