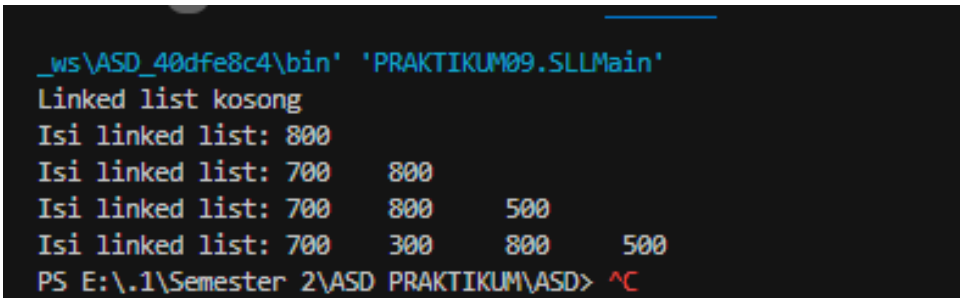
LAPORAN PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA

PRAKTIKUM 9
Linked List

Oleh:
JESSICA AMELIA
2341760185
SIB 1A/16

PROGRAM STUDI SISTEM INFORMASI BISNIS
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2023/2024

Praktikum 1



1. Mengapa class LinkedList tidak memerlukan method isFull() seperti halnya Stack dan Queue?
   Jawab : Karena LinkedList adalah struktur data yang menggunakan alokasi memori secara dinamis saat elemen baru ditambahkan, tidak ada batasan pada jumlah elemen yang dapat disimpan di dalamnya. Dengan kata lain, LinkedList dapat terus tumbuh seiring penambahan elemen baru tanpa perlu khawatir tentang ketersediaan ruang memori. Oleh karena itu, tidak perlu untuk memiliki method isFull() dalam LinkedList, karena konsep "penuh" tidak berlaku dalam konteks alokasi memori dinamis yang digunakan oleh LinkedList. Sementara Stack dan Queue memiliki batasan kapasitas maksimum karena struktur datanya yang berbeda. Stack memiliki aturan "Last In, First Out" (LIFO) di mana elemen terakhir yang ditambahkan adalah yang pertama kali dihapus. Queue memiliki aturan "First In, First Out" (FIFO) di mana elemen pertama yang ditambahkan adalah yang pertama kali dihapus. Karena keterbatasan ini, Stack dan Queue perlu mengecek apakah daftar tersebut penuh sebelum menambahkan elemen baru.
2. Mengapa class LinkedList hanya memiliki atribut head yang menyimpan informasi node pertama? Bagaimana informasi node kedua dan lainnya diakses?
   Jawab : Class LinkedList hanya memiliki atribut head yang menyimpan informasi node pertama karena struktur data LinkedList didesain untuk menyimpan elemen secara berurutan, di mana setiap node terhubung dengan node berikutnya melalui pointer. Head menyimpan pointer ke node kedua. Node kedua memiliki pointer ke node ketiga, dst.
3. Pada langkah, jelaskan kegunaan kode berikut

```
if (currentNode.data == key) {
    newNode.next = currentNode.next;
    currentNode.next = newNode;
    break;
}
```

   Jawab : Kode tersebut digunakan untuk menambahkan node baru pada posisi setelah node yang berisi data tertentu (key)
   if (currentNode.data == key): Ini adalah kondisi yang memeriksa apakah data pada currentNode sama dengan key. Ini menandakan bahwa kita telah menemukan tempat di mana kita ingin menyisipkan newNode.
   Jika kondisi tersebut terpenuhi (currentNode.data sama dengan key), maka langkah-langkah berikut dilakukan:

a. newNode.next = currentNode.next;: Ini mengatur next dari newNode menjadi next dari currentNode. Ini berarti newNode akan menunjuk ke node yang saat ini ditunjuk oleh currentNode.
b. currentNode.next = newNode;: Ini menyisipkan newNode setelah currentNode, sehingga newNode sekarang menjadi node yang berikutnya dari currentNode.
c. break;: Ini mengakhiri loop karena operasi pencarian dan penyisipan telah selesai.

4. Implementasikan method insertAt(int index, int key) dari tugas mata kuliah ASD (Teori)
Jawab :

```java
// Fungsi untuk menambahkan data baru pada indeks tertentu
public void insertAt(int index, int key) {
    Node newNode = new Node(key, next:null);

    if (index == 0) {
        newNode.next = head;
        head = newNode;
    } else {
        Node currentNode = head;
        int currentIndex = 0;

        // Iterasi hingga mencapai indeks sebelumnya
        while (currentNode != null && currentIndex < index - 1) {
            currentNode = currentNode.next;
            currentIndex++;
        }

        // Jika indeks lebih besar dari panjang linked list,
        // atau jika indeks adalah indeks terakhir
        if (currentNode == null || currentNode.next == null) {
            System.out.println(x:"Index out of bounds.");
            return;
        }

        // Tautkan node baru di antara node pada indeks sebelumnya dan node pada indeks
        newNode.next = currentNode.next;
        currentNode.next = newNode;
    }
}
```

```java
public class SLLMain {
    Run | Debug
    public static void main(String[] args) {
        LinkedList myLinkedList = new LinkedList();
        myLinkedList.print();
        myLinkedList.addFirst(input:800);
        myLinkedList.print();
        myLinkedList.addFirst(input:700);
        myLinkedList.print();
        myLinkedList.addLast(input:500);
        myLinkedList.print();
        myLinkedList.insertAfter(key:700, input:300);
        myLinkedList.print();
        myLinkedList.insertAt(index:2, key:400);
        myLinkedList.print();
    }
}
```

```
_ws\ASD_40dfe8c4\bin' 'PRAKTIKUM09.SLLMain'
Linked list kosong
Isi linked list: 800
Isi linked list: 700    800
Isi linked list: 700    800     500
Isi linked list: 700    300     800     500
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASD> ^C
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASD>
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASD>  e:; cd 'e:\.1\Semester
& 'C:\Program Files\Java\jdk1.8.0_121\bin\java.exe' '-cp' 'C:\U
a\Roaming\Code\User\workspaceStorage\ca7ac3c3e86cab05f2ab5cd3b9
_ws\ASD_40dfe8c4\bin' 'PRAKTIKUM09.SLLMain'
Linked list kosong
Isi linked list: 800
Isi linked list: 700    800
Isi linked list: 700    800     500
Isi linked list: 700    300     800     500
Isi linked list: 700    300     400     800     500
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASD>
```

Praktikum 2

```
0dfe8c4\bin' 'PRAKTIKUM09.SLLMain'
Linked list kosong
Isi linked list: 800
Isi linked list: 700    800
Isi linked list: 700    800     500
Isi linked list: 700    300     800     500
Data pada index ke-1: 300
Data 300 berada pada index ke: 1
Isi linked list: 700    800     500
Isi linked list: 800    500
Isi linked list: 800
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASD>
```

1. Jelaskan maksud potongan kode di bawah pada method remove()

```
if (currentNode.next.data == key) {
    currentNode.next = currentNode.next.next;
    break;
}
```

Jawab : Potongan kode tersebut merupakan bagian dari metode remove() yang bertujuan untuk menghapus node yang memiliki nilai data yang sama dengan key yang diberikan.

a. currentNode.next.data == key: Kode tersebut memeriksa apakah nilai data dari node berikutnya (currentNode.next) sama dengan nilai key. Ini menandakan bahwa node berikutnya adalah node yang ingin dihapus.
b. Jika kondisi tersebut terpenuhi, berarti kita telah menemukan node yang ingin dihapus, maka langkah-langkah berikut dilakukan:a. currentNode.next = currentNode.next.next;: Kode ini menautkan node saat ini (currentNode) langsung ke node setelah node yang akan dihapus (currentNode.next.next). Dengan melakukan ini, kita melewati atau "menghilangkan" node yang ingin dihapus dari linked list.b. break;: Ini mengakhiri loop karena operasi penghapusan telah selesai.

2. Jelaskan maksud if-else block pada method indexOf() berikut

```
if (currentNode == null) {
    return -1;
} else {
    return index;
}
```

Jawab :

a. if(currentNode == null): Kode ini memeriksa apakah currentNode saat ini adalah null. Ini terjadi ketika iterasi melalui linked list telah mencapai akhir tanpa menemukan node dengan nilai data yang sama dengan key.
   Jika kondisi ini terpenuhi, maka metode indexOf() akan mengembalikan -1. Ini menunjukkan bahwa nilai key tidak ditemukan dalam linked list, karena tidak ada node yang memiliki nilai data yang sama.
b. else: Bagian else menangani kasus ketika currentNode bukan null, yang berarti masih ada node yang perlu diperiksa atau bahwa node dengan nilai data yang sama dengan key telah ditemukan.
   Pada bagian else, jika node dengan nilai data yang sama dengan key ditemukan, maka metode indexOf() akan mengembalikan nilai index, yang merupakan indeks dari node tersebut dalam linked list.

3. Error apa yang muncul jika argumen method getData() lebih besar dari jumlah node pada linked list? Modifikasi kode program untuk menghandle hal tersebut.

Jawab :

Jika argumen index pada metode getData() lebih besar dari jumlah node pada linked list, maka akan terjadi error NullPointerException. Hal ini terjadi ketika iterasi mencoba untuk mengakses next dari currentNode, tetapi currentNode menjadi null karena telah mencapai akhir linked list.

Untuk mengatasi hal tersebut, kita perlu memeriksa apakah currentNode tidak null sebelum mengakses data. Jika currentNode adalah null, itu berarti kita telah mencapai akhir linked list

dan nilai index yang diminta melebihi jumlah node. Dalam hal ini, kita dapat mengembalikan nilai yang menandakan bahwa indeks tersebut tidak valid. Berikut adalah modifikasi kode untuk mengatasi masalah ini:

```
public int getData(int index){
    Node currentNode = head;
    //modifikasi
    for (int i=0; i< index; i++){
        if(currentNode == null){
            return -1;
        }
        currentNode = currentNode.next;
    }

    return currentNode.data;
}
```

```
C:\Program Files\Java\jdk1.8.0_121\bin\java.ex
ing\Code\User\workspaceStorage\ca7ac3c3e86cab6
0dfe8c4\bin' 'PRAKTIKUM09.SLLMain'
Linked list kosong
Isi linked list: 800
Isi linked list: 700    800
Isi linked list: 700    800     500
Isi linked list: 700    300     800     500
Data pada index ke-7: -1
Data 300 berada pada index ke: 1
Isi linked list: 700    800     500
Isi linked list: 800    500
Isi linked list: 800
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASD>
```

4. Apa fungsi keyword break pada method remove()? Bagaimana efeknya jika baris tersebut dihapus?
   Jawab : Keyword break pada metode remove() memiliki fungsi untuk mengakhiri loop while saat kondisi tertentu terpenuhi. Ketika kita menemukan node yang memiliki nilai data yang sama dengan key yang ingin dihapus, kita tidak perlu melanjutkan iterasi melalui linked list lagi, karena kita sudah menemukan node yang ingin dihapus dan sudah menghapusnya. Oleh karena itu, kita menggunakan break untuk keluar dari loop while setelah menghapus node yang diinginkan dan menghindari pengulangan yang tidak perlu dan meningkatkan efisiensi.

Tugas

1. Implementasikan method-method berikut pada class LinkedList:
   a. insertBefore() untuk menambahkan node sebelum keyword yang diinginkan

```java
//TUGAS

// Metode untuk menambahkan node sebelum keyword yang diinginkan
public void insertBefore(int key, int newData) {
    Node newNode = new Node(newData, next:null);

    // Jika linked list kosong atau keyword adalah head
    if (isEmpty() || head.data == key) {
        newNode.next = head;
        head = newNode;
        return;
    }

    Node current = head;

    // Iterasi hingga menemukan node sebelum keyword
    while (current.next != null && current.next.data != key) {
        current = current.next;
    }

    // Jika keyword tidak ditemukan
    if (current.next == null) {
        System.out.println(x:"Keyword not found.");
        return;
    }

    // Tautkan node baru sebelum node dengan keyword
    newNode.next = current.next;
    current.next = newNode;
}
```

b. insertAt(int index, int key) untuk menambahkan node pada index tertentu

```java
JM09 > LinkedList.java > LinkedList > insertBefore(int, int)
public class LinkedList {
    public void insertBefore(int key, int newData) {
        // Tautkan node baru sebelum node dengan keyword
        newNode.next = current.next;
        current.next = newNode;
    }

    // Method untuk menambahkan data baru pada indeks tertentu
    public void insertAt(int index, int key) {
        Node newNode = new Node(key, next:null);

        if (index == 0) {
            newNode.next = head;
            head = newNode;
        } else {
            Node currentNode = head;
            int currentIndex = 0;

            // Iterasi hingga mencapai indeks sebelumnya
            while (currentNode != null && currentIndex < index - 1) {
                currentNode = currentNode.next;
                currentIndex++;
            }

            // Jika indeks lebih besar dari panjang linked list,
            // atau jika indeks adalah indeks terakhir
            if (currentNode == null || currentNode.next == null) {
                System.out.println(x:"Index out of bounds.");
                return;
            }

            // Tautkan node baru di antara node pada indeks sebelumnya dan node pada indeks
            newNode.next = currentNode.next;
            currentNode.next = newNode;
        }
    }
```

c. removeAt(int index) untuk menghapus node pada index tertentu

```java
// Metode untuk menghapus node pada index tertentu
public void removeAt(int index) {
    // Jika linked list kosong
    if (isEmpty()) {
        System.out.println(x:"Linked list is empty.");
        return;
    }

    // Jika index adalah 0
    if (index == 0) {
        head = head.next;
        return;
    }

    Node current = head;
    int currentIndex = 0;

    // Iterasi hingga mencapai indeks sebelumnya
    while (current != null && currentIndex < index - 1) {
        current = current.next;
        currentIndex++;
    }

    // Jika indeks lebih besar dari panjang linked list
    if (current == null || current.next == null) {
        System.out.println(x:"Index out of bounds.");
        return;
    }

    // Hapus node pada indeks tertentu
    current.next = current.next.next;
}
```

Main dan Output

```java
public class SLLMain {
    Run | Debug
    public static void main(String[] args) {
        LinkedList myLinkedList = new LinkedList();
        myLinkedList.print();
        myLinkedList.addFirst(input:800);
        myLinkedList.print();
        myLinkedList.addFirst(input:700);
        myLinkedList.print();
        myLinkedList.addLast(input:500);
        myLinkedList.print();
        myLinkedList.insertAfter(key:700, input:300);
        myLinkedList.print();
        myLinkedList.insertAt(index:2, key:400);
        myLinkedList.print();

        System.out.println("Data pada index ke-7: " + myLinkedList.getData(index:7));
        System.out.println("Data 300 berada pada index ke: " + myLinkedList.indexOf(key:300));

        myLinkedList.remove(key:300);
        myLinkedList.print();
        myLinkedList.removeFirst();
        myLinkedList.print();
        myLinkedList.removeLast();
        myLinkedList.print();

        //Tugas
        myLinkedList.insertBefore(key:400, newData:300);
        myLinkedList.print();
        myLinkedList.insertAt(index:1, key:350);
        myLinkedList.print();
        myLinkedList.removeAt(index:2);
        myLinkedList.print();
    }
}
```

```
Storage\ca7ac3c3e86cab05f2ab5cd3b92ef8ec\redhat.java\jdt_
Linked list kosong
Isi linked list: 800
Isi linked list: 700    800
Isi linked list: 700    800     500
Isi linked list: 700    300     800     500
Isi linked list: 700    300     400     800     500
Data pada index ke-7: -1
Data 300 berada pada index ke: 1
Isi linked list: 700    400     800     500
Isi linked list: 400    800     500
Isi linked list: 400    800
Isi linked list: 300    400     800
Isi linked list: 300    350     400     800
Isi linked list: 300    350     800
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASD>
```

2. Dalam suatu game scavenger hunt, terdapat beberapa point yang harus dilalui peserta untuk menemukan harta karun. Setiap point memiliki soal yang harus dijawab, kunci jawaban, dan pointer ke point selanjutnya. Buatlah implementasi game tersebut dengan linked list

```
KTIKUM09 > Tugas > Point.java > Point
package PRAKTIKUM09.Tugas;

class Point {
    String question;
    String answer;
    Point next;

    Point(String question, String answer)
        this.question = question;
        this.answer = answer;
        this.next = null;
    }
}
```

```
IKUM09 > Tugas > ScavengerHunt.java > ...
package PRAKTIKUM09.Tugas;
import java.util.Scanner;
class ScavengerHunt {
    Point head;
    Point currentPoint;

    // Metode untuk menambahkan titik baru dengan pertanyaan dan jawaban
    public void addPoint(String question, String answer) {
        Point newPoint = new Point(question, answer);
        if (head == null) {
            head = newPoint;
            currentPoint = head;
        } else {
            Point temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newPoint;
        }
    }

    // Metode untuk memainkan permainan scavenger hunt
    public void play() {
        Scanner scanner = new Scanner(System.in);

        System.out.println(x:"Selamat datang di Scavenger Hunt!");

        while (currentPoint != null) {
            System.out.println("Pertanyaan: " + currentPoint.question);
            System.out.print(s:"Jawaban Anda: ");
            String userAnswer = scanner.nextLine();

            if (userAnswer.equalsIgnoreCase(currentPoint.answer)) {
                System.out.println(x:"Jawaban Anda benar! Anda dapat melanjutkan ke titik berikutnya.");
                currentPoint = currentPoint.next;
            } else {
                System.out.println(x:"Jawaban Anda salah! Coba lagi.");
            }
        }

        scanner.close();

        System.out.println(x:"Selamat! Anda telah menyelesaikan Scavenger Hunt dan menemukan harta karun!");
    }
}
```

```java
package PRAKTIKUM09.Tugas;

public class MainScavager {
    Run | Debug
    public static void main(String[] args) {
        // Membuat objek untuk permainan scavenger hunt
        ScavengerHunt game = new ScavengerHunt();

        // Menambahkan titik-titik dan pertanyaan serta jawaban
        game.addPoint(question:"Siswa newSiswa = new Siswa(); , Tunjukkan yang disebut tipe data?", answer:"Siswa");
        game.addPoint(question:"Apa tipe data untuk kumpulan karakter?", answer:"String");
        game.addPoint(question:"Construktor juga dimaksud dengan method apa ?", answer:"Istimewa");
        game.addPoint(question:"Class digunakan untuk membentuk apa?", answer:"Objek");
        game.addPoint(question:"Kata kunci new digunakan untuk ?", answer:"Instansiasi");

        // Memulai permainan
        game.play();
    }
}
```

```
Storage\ca7ac3c3e86cab05f2ab5cd3b92ef8ec\redhat.java\jdt_ws\ASD_40dfe8c4\bin' 'PR
Selamat datang di Scavenger Hunt!
Pertanyaan: Siswa newSiswa = new Siswa(); , Tunjukkan yang disebut tipe data?
Jawaban Anda: Siswa
Jawaban Anda benar! Anda dapat melanjutkan ke titik berikutnya.
Pertanyaan: Apa tipe data untuk kumpulan karakter?
Jawaban Anda: String
Jawaban Anda benar! Anda dapat melanjutkan ke titik berikutnya.
Pertanyaan: Construktor juga dimaksud dengan method apa ?
Jawaban Anda: Istimewa
Jawaban Anda benar! Anda dapat melanjutkan ke titik berikutnya.
Pertanyaan: Class digunakan untuk membentuk apa?
Jawaban Anda: atribut
Jawaban Anda salah! Coba lagi.
Pertanyaan: Class digunakan untuk membentuk apa?
Jawaban Anda: objek
Jawaban Anda benar! Anda dapat melanjutkan ke titik berikutnya.
Pertanyaan: Kata kunci new digunakan untuk ?
Jawaban Anda: instansiasi
Jawaban Anda benar! Anda dapat melanjutkan ke titik berikutnya.
Selamat! Anda telah menyelesaikan Scavenger Hunt dan menemukan harta karun!
PS E:\.1\Semester 2\ASD PRAKTIKUM\ASD>
```