Laporan

Email Student GUI Linked List

Disusun untuk Memenuhi Laporan Praktikum Algoritma dan Struktur Data

Oleh:

Nama	: Bilal Abdul Qowiy
NIM	: 1910130007
Nama	: Muhammad Akbar Riandi
NIM	: 1910130011
Nama	: Muhammad Zulfikri Maulana
NIM	: 1810130008

SEKOLAH TINGGI ILMU MANAJEMEN DAN ILMU KOMPUTER ESQ

JAKARTA

2020

BAB I PENDAHULUAN

1.1. **DASAR TEORI** **Pengertian**

Linked List

a. Nodes

Self-referential objects (object yang mereferensikan dirinya sendiri) yang disebut *nodes*, yang dihubungkan dengan links, membentuk kata “linked” list.

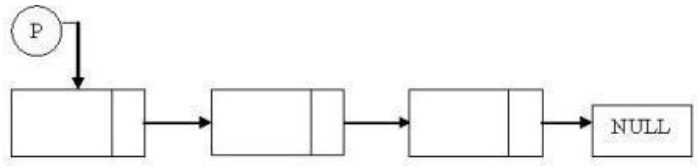
b. Linked List (LL)

Adalah koleksi data item yang tersusun dalam sebuah barisan secara linear, dengan penyisipan dan pemindahan dapat dilakukan dalam semua tempat di LL tersebut. ***c.***

Single Linked List

Adalah sebuah LL yang menggunakan sebuah variabel pointer saja untuk menyimpan banyak data dengan metode LL, suatu daftar isi yang saling berhubungan.

Ilustrasi single LL:



Pada gambar di atas, data terletak pada sebuah lokasi dalam sebuah memory, tempat yang disediakan memory untuk menyimpan data disebut node ? simpul, setiap node memiliki pointer (penunjuk) yang menunjuk ke node berikutnya sehingga terbentuk suatu untaian yang disebut single LL.

Bila dalam single LL pointer hanya dapat bergerak ke satu arah saja, maju / mundur, kanan / kiri, sehingga pencarian datanya juga hanya satu arah saja.

Definisi GUI

Pengertian GUI (Graphical User Interface) adalah bentuk antarmuka pengguna untuk memungkinkan user dapat berinteraksi dengan perangkat elektronik.

GUI memiliki beberapa elemen, mulai dari elemen windows, menu, icon, widget dan juga tab. Untuk menggunakan elemen ini biasanya GUI akan mendapatkan inputan dari perangkat masukan, baik secara manual maupun dengan teknologi touchscreen.

Dalam perkembangannya, GUI akan terus dikembangkan untuk semakin memudahkan penggunaannya seperti misalnya pengembangan teknologi gesture dan juga teknologi remote jarak jauh sebagai salah satu cara inputan di masa depan.

GUI merupakan salah satu jenis user interface yang digunakan untuk melakukan komunikasi antara manusia dengan perangkat seperti laptop, komputer, ponsel dan tablet. Hal ini menjadikan komponen GUI selalu berhubungan dengan representasi visual dari sebuah sistem operasi ataupun software.

GUI sangat berperan penting serta sangat memudahkan user untuk mengoperasikan sebuah sistem atau software aplikasi dengan navigasi yang ada. Untuk itu penting bagi Anda mengetahui lebih detail tentang apa itu pengertian, fungsi, jenis, komponen, kelebihan dan kekurangan dari graphical user interface.

Email

Pada tahun 1971, seorang insinyur bernama Ray Tomlinson ditugaskan dalam proyek yang disebut SNDMSG. Program ini bukan merupakan program baru, karena sebenarnya program tersebut sudah ada selama sekian tahun. Dengan standar masa kini, program tersebut bisa dikatakan lebih dari primitif. Apa yang dilakukan program tersebut hanyalah memungkinkan pengguna pada mesin yang sama dapat saling mengirim pesan satu sama lain. Pengguna dapat membuat dokumen teks yang kemudian akan dikirimkan ke dalam kotak surat pada mesin yang sama.

Awalnya Ray bereksperimen dengan sebuah program yang bernama SNDMSG yang bisa digunakan untuk meninggalkan pesan pada sebuah komputer, sehingga orang lain yang memakai komputer itu dapat membaca pesan yang ditinggalkan. Lalu ia melanjutkan eksperimennya dengan menggunakan file protocol yang bernama CYPNET sehingga program SNDMSG tadi bisa mengirim pesan ke komputer lain yang berada di dalam jaringan ARPAnet. Itulah awal terciptanya sebuah ‘e-mail’. Pesan e-mail yang pertama kali dikirim Ray, dan merupakan e-mail yang pertama di dunia adalah “QWERTYUIOP”

BAB II PEMBAHASAN

2.1. SOURCE CODE

StudentEmailExeption.java

```
public class StudentEmailException extends Exception
{

  public StudentEmailException ()
  {
    super ();
  }

  public StudentEmailException (String message)
  {
    super (message);
  }

  public StudentEmailException (Throwable cause)
  {
    super (cause);
  }

  public StudentEmailException (String message, Throwable cause)
  {
    super (message, cause);
  }
}
```

StudentEmailGUI.java

```
import javax.swing.*;
import java.awt.*;
import java.util.LinkedList;

public class StudentEmailGUI extends JFrame
{
  // Constants:

  // GUI Componet:
  JTextArea  studentTextArea   = new JTextArea ();

  JLabel     idLabel          = new JLabel     ("ID: ");
  JTextField idTextField      = new JTextField (10);
  JLabel     nameLabel        = new JLabel     ("Name: ");
  JTextField nameTextField    = new JTextField (10);

  JButton    testDataButton    = new JButton ("Test Data");
  JButton    addButton         = new JButton ("Add");
  JButton    deleteButton      = new JButton ("Delete");
  JButton    editButton        = new JButton ("Edit");
  JButton    editSaveButton    = new JButton ("Save");
  JButton    displayAllButton  = new JButton ("Display All");
  JButton    exitButton        = new JButton ("Exit");


  // Class Instance Data:
  private LinkedList<StudentEmail> studentLinkedList = new
LinkedList<StudentEmail> ();
  private int editIndex;


  public StudentEmailGUI ()
  {
    JPanel flow1Panel = new JPanel (new FlowLayout (FlowLayout.CENTER));
    JPanel flow2Panel = new JPanel (new FlowLayout (FlowLayout.CENTER));
    JPanel gridPanel  = new JPanel (new GridLayout (2, 1));

    studentTextArea.setEditable (false);

    flow1Panel.add (idLabel);
    flow1Panel.add (idTextField);
    flow1Panel.add (nameLabel);
    flow1Panel.add (nameTextField);

    flow2Panel.add (testDataButton);
    flow2Panel.add (addButton);
    flow2Panel.add (editButton);
    flow2Panel.add (editSaveButton);
```

```
  flow2Panel.add (deleteButton);
  flow2Panel.add (displayAllButton);
  flow2Panel.add (exitButton);

  gridPanel.add (flow1Panel);
  gridPanel.add (flow2Panel);

  editSaveButton.setEnabled (false);

  add (studentTextArea, BorderLayout.CENTER);
  add (gridPanel,       BorderLayout.SOUTH);


  addButton.addActionListener       (event -> addStudent ());
  displayAllButton.addActionListener (event -> displayAll ());
  editButton.addActionListener      (event -> editStudent ());
  editSaveButton.addActionListener   (event -> editSaveStudent ());
  exitButton.addActionListener      (event -> exitApplication ());
  deleteButton.addActionListener     (event -> deleteStudent ());
  testDataButton.addActionListener   (event -> addTestData ());

  setTitle ("ESQBS EMAIL STUDENT v0.1");
}

private boolean isStudentIdInLinkedList (String idStr)
{
  boolean inList = false;

  for (StudentEmail stud : studentLinkedList)
  {
    if (stud.getId ().compareToIgnoreCase (idStr) == 0)
    {
      inList = true;
    }
  }

  return inList;
}

private void addStudent ()
{
  if (isStudentIdInLinkedList (idTextField.getText()) == true)
  {
    JOptionPane.showMessageDialog (StudentEmailGUI.this,
                   "Error: student id sudah ada di database");
  }
  else
  {
    try
    {
```

```
      StudentEmail stud = new StudentEmail (nameTextField.getText(),
                                  idTextField.getText());

      studentLinkedList.add (stud);                          // tambah student

      displayAll ();

      nameTextField.setText("");
      idTextField.setText("");


    }
    catch (StudentEmailException error)
    {
      JOptionPane.showMessageDialog (StudentEmailGUI.this, error.toString ());
      // myLabel.setText (error.toString ());


    }

  }
}

private void deleteStudent ()
{
  if (studentLinkedList.size() == 0)
  {
    JOptionPane.showMessageDialog (StudentEmailGUI.this,
                          "Error: database kosong.");
  }
  else if (isStudentIdInLinkedList (idTextField.getText()) == false)
  {
    JOptionPane.showMessageDialog (StudentEmailGUI.this,
                         "Error: id student tidak ada di database.");
  }
  else
  {
    for (int s = 0; s < studentLinkedList.size(); s++)
    {
      String currId = studentLinkedList.get (s).getId ();

      if (currId.compareToIgnoreCase (idTextField.getText()) == 0)
      {
        studentLinkedList.remove (s);                        // menghapus
      }
    }

    displayAll ();

    nameTextField.setText("");
```

```
    idTextField.setText("");
  }
}

private void editStudent ()
{
  if (studentLinkedList.size() == 0)
  {
    JOptionPane.showMessageDialog (StudentEmailGUI.this,
                                   "Error: database kosong.");
  }
  else if (isStudentIdInLinkedList (idTextField.getText()) == false)
  {
    JOptionPane.showMessageDialog (StudentEmailGUI.this,
                                   "Error: id student tidak ada.");
  }
  else
  {
    editIndex = -1;

    for (int s = 0; s < studentLinkedList.size(); s++)
    {
      String currId = studentLinkedList.get (s).getId ();

      if (currId.compareToIgnoreCase (idTextField.getText()) == 0)
      {
        editIndex = s;
        s = studentLinkedList.size(); // Exit Loop
      }
    }

    // index cannot be less than 0, because we checked if the Stud Id was in
    // the linked list before we looped above.
    if (editIndex >= 0)
    {
      editSaveButton.setEnabled   (true);

      editButton.setEnabled       (false);
      testDataButton.setEnabled   (false);
      addButton.setEnabled        (false);
      deleteButton.setEnabled     (false);
      displayAllButton.setEnabled (false);
      exitButton.setEnabled       (false);

      nameTextField.setText (studentLinkedList.get (editIndex).getName () );
      //idTextField.setText   (studentLinkedList.get (editIndex).getId () );
    }


  }
```

```
}

private void editSaveStudent ()
{
  // This code will preserve the changes the user made to the student
  // they were editing - and save them back into the Linked List.

  studentLinkedList.get (editIndex).setName (nameTextField.getText() );
  studentLinkedList.get (editIndex).setId   (idTextField.getText() );

  displayAll ();

  nameTextField.setText ("");
  idTextField.setText   ("");

  editSaveButton.setEnabled   (false);

  editButton.setEnabled      (true);
  testDataButton.setEnabled   (true);
  addButton.setEnabled       (true);
  deleteButton.setEnabled     (true);
  displayAllButton.setEnabled (true);
  exitButton.setEnabled      (true);
}

private void addTestData ()
{
  nameTextField.setText ("Faiz");
  idTextField.setText   ("ESQ21");
  addStudent ();

  nameTextField.setText ("Muhsin");
  idTextField.setText   ("ESQ31");
  addStudent ();

  nameTextField.setText ("Bahara");
  idTextField.setText   ("ESQ41");
  addStudent ();
}

private void displayAll ()
{
  studentTextArea.setText ("");

  for (StudentEmail stud : studentLinkedList)
  {
    studentTextArea.append (stud + "\n");
  }
}
```

```
  private void exitApplication ()
  {
    System.exit (0); // All is OK.
  }

  public static void main (String[] args)
  {
    StudentEmailGUI app = new StudentEmailGUI ();
    app.setVisible  (true);
    app.setSize     (500, 310);
    app.setLocation (200, 100);
    app.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
  }
} // public class StudentEmailGUI
```

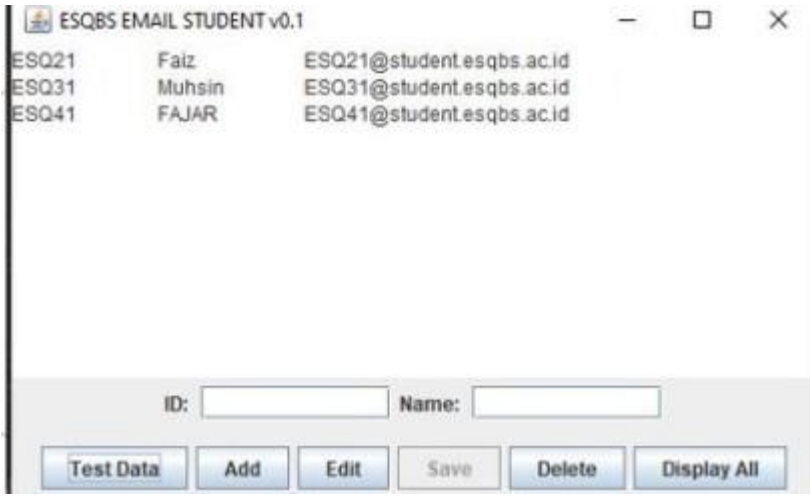
StudentEmail.java

```
public class StudentEmail
{
  public static final String EMAIL_STUB = "@student.esqbs.ac.id";

  private String name;
  private String id;

  public StudentEmail ()
  {
    name = "";
    id   = "";
  }

  public StudentEmail (String name, String id)
                  throws StudentEmailException
  {
    // Remove elading and trailing spaces, tabs.
    name = name.trim ();
    id   = id.trim ();

    if (name.length () == 0)
    {
      //JOptionPane.showMessageDialog (null, "Error: name cannot be blank.");
      throw new StudentEmailException ("Error: nama harus terisi.");
    }

    else if (id.length () == 0)
    {
      //JOptionPane.showMessageDialog (null, "Error: Id cannot be blank.");
      throw new StudentEmailException ("Error: id harus terisi.");
    }
```

```
    else
    {
      // All is OK, set class data to the values passed in.
      this.name = name;
      this.id   = id;
    }
  }

  public String getName ()
  {
    return name;
  }

  public String getId ()
  {
    return id;
  }

  public void setName (String name)
  {
    this.name = name;
  }

  public void setId (String id)
  {
    this.id = id;
  }

  @Override
  public String toString ()
  {
    return id + "\t" + name + "\t" + id + EMAIL_STUB;
  }

} // public class StudentEmail
```

2.2. **OUTPUT**

Screenshot run



2.3. ANALISA

Pembahasan program praktikum

Pada program diatas methot SisipUrut akan menginialisasi tipe data dan akan mengurutkan nilai yang sebelumnya di set.

Kesimpulan dari program diatas bahwa dalam penggunaan Single Linked List itu proses pengaksesannya sama dengan stack yakni LIFO. Namun demikian, Linked List tidak sekaku stack. Pada Linked List dapat menyisipkan data sesuai keinginan penggunanya.