LAPORAN MODUL (8)
Pemrograman Berorientasi Objek Lanjut

“ArrayList”

Dosen Pengampu: Eduardus Hardika Sandy Atmaja, S.Kom., M.Cs.

DIBUAT OLEH:

Nama : Yohanis Celvin D. P. U. Pati

Nim : 245314033

KELAS : BP

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2025

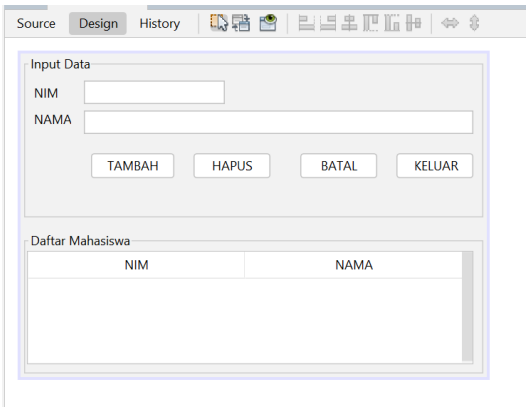
A. TUJUAN

1. Tujuan praktik materi ArrayList ini untuk memahami cara menyimpan, menambah, menghapus, dan menampilkan data secara dinamis menggunakan struktur data ArrayList dalam program Java.

B. TUGAS

Latihan 1

- **Source code (di ss)**
  - **JFrame**



  - **Menu Tambah**

```
206
207     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
208         Mahasiswa baru = new Mahasiswa();
209         baru.setNim(nim.getText());
210         baru.setNama(nama.getText());
211
212         daftarMahasiswa.add(baru);
213         //clearfield();
214         refreshTable();
215         JOptionPane.showMessageDialog(this, "Data Berhasil Ditambahkan");
216
217     }
```

Penjelasan:

- Mahasiswa baru = new Mahasiswa(); untuk membuat objek baru dari class Mahasiswa.
- baru.setNim(nim.getText()); untuk mengambil teks dari TextField nim dan menyimpannya ke atribut nim milik objek Mahasiswa.
- baru.setNama(nama.getText()); mengambil teks dari TextField nama dan menyimpannya ke atribut nama.daftarMahasiswa.add(baru); dan Menambahkan objek Mahasiswa baru ke dalam ArrayList daftarMahasiswa (penyimpanan data sementara di memori).
- refreshTable(); untuk memperbarui tampilan tabel agar data yang baru ditambahkan langsung muncul di daftar mahasiswa.

- JOptionPane.showMessageDialog(...) untuk menampilkan pesan bahwa data berhasil ditambahkan.

- **Menu Hapus**

```
218
219     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
220
221         int selectedRow = tabel.getSelectedRow();
222         if (selectedRow >= 0) {
223             daftarMahasiswa.remove(selectedRow);
224             refreshTable();
225             JOptionPane.showMessageDialog(this, "Data berhasil dihapus");
226         } else {
227             JOptionPane.showMessageDialog(this, "Pilih data yang ingin dihapus dulu");
228         }
```

Penjelasan:

- int selectedRow = tabel.getSelectedRow(); untuk mengecek baris mana yang sedang dipilih di tabel.
- if (selectedRow >= 0) untuk mengecek apakah ada baris yang dipilih (kalau tidak, nilainya -1).
- daftarMahasiswa.remove(selectedRow); untuk Menghapus objek mahasiswa pada posisi (index) sesuai baris yang dipilih.
- refreshTable(); untuk memperbarui tampilan tabel supaya data yang dihapus langsung hilang dari daftar.
- JOptionPane.showMessageDialog(...) untuk memberi pesan bahwa data sudah dihapus.

- **Menu Batal**

```
230
231     private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
232         clearField();
233     }
234
```

Penjelasan:

private void clearField() {

nim.setText("");

nama.setText("");

}

Ini method bantu yang dipanggil

- clearField()untuk mengosongkan field input NIM dan NAMA.
- Jadi ketika kamu klik tombol *Batal*, semua isian yang sudah kamu ketik akan hilang (kosong kembali).

- **Menu Keluar**

```
234
235     private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
236         System.exit(0);
237     }
238
```

Penjelasan:

- System.exit(0); perintah bawaan Java untuk menghentikan program sepenuhnya.

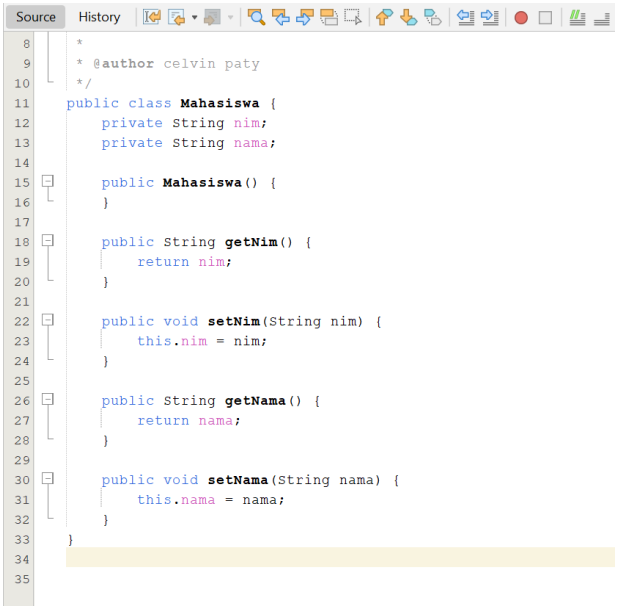
- **Method publick Test()**

```
20  public class Test extends javax.swing.JFrame {
21      private List<Mahasiswa> daftarMahasiswa = new ArrayList();
22      /**
23       * Creates new form Test
24       */
25      public Test() {
26          initComponents();
27
28          Mahasiswa m1 = new Mahasiswa(); m1.setNim("245314001"); m1.setNama("Andi");
29          Mahasiswa m2 = new Mahasiswa(); m2.setNim("245314002"); m2.setNama("Budi");
30          Mahasiswa m3 = new Mahasiswa(); m3.setNim("245314003"); m3.setNama("Citra");
31
32          daftarMahasiswa.add(m1);
33          daftarMahasiswa.add(m2);
34          daftarMahasiswa.add(m3);
35
36          refreshTable();
37      }
38
```

Penjelasan:

- initComponents(); untuk memanggil method otomatis dari NetBeans untuk membuat tampilan GUI (label, tombol, tabel, dll).
- Tiga baris Mahasiswa m1, m2, m3 untuk membuat 3 objek Mahasiswa sebagai data contoh.
- daftarMahasiswa.add(...) untuk menambahkan ketiga objek tadi ke ArrayList daftarMahasiswa**.**
- refreshTable(); memperbarui tampilan tabel agar data Andi, Budi, dan Citra langsung muncul ketika program dijalankan.

- **Mahasiswa**



Penjelasan:

- public class Mahasiswa { deklarasikan sebuah class publik bernama Mahasiswa.

  private String nim;

  private String nama;
- nim = untuk menyimpan nomor Induk Mahasiswa**.**
- nama = untuk menyimpan nama Mahasiswa**.**
- Keduanya bersifat private, artinya tidak bisa diakses langsung dari luar class **(harus melalui getter & setter).**

  public Mahasiswa() {

  }
- Konstruktor method khusus yang dijalankan setiap kali objek dibuat**.**

  public String getNim() {

  return nim;

  }
- Getter untuk NIM

  public void setNim(String nim) {

  this.nim = nim;

  }
- Setter untuk NIM

public String getNama() {

return nama;

}

- Getter untuk nama

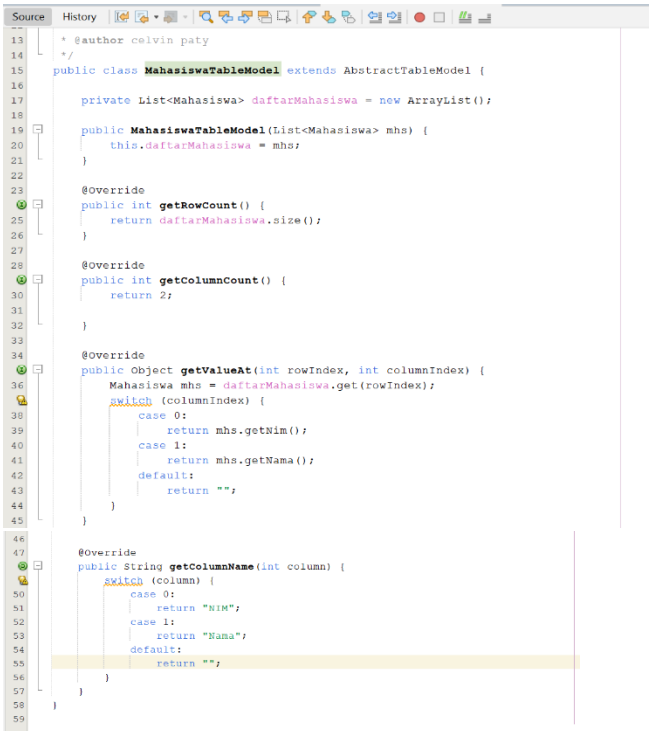
public void setNama(String nama) {

this.nama = nama;

}

- Setter untuk nama

- **MahasiswaTableModel**



```
* @author celvin paty
*/
public class MahasiswaTableModel extends AbstractTableModel {

    private List<Mahasiswa> daftarMahasiswa = new ArrayList();

    public MahasiswaTableModel(List<Mahasiswa> mhs) {
        this.daftarMahasiswa = mhs;
    }

    @Override
    public int getRowCount() {
        return daftarMahasiswa.size();
    }

    @Override
    public int getColumnCount() {
        return 2;

    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Mahasiswa mhs = daftarMahasiswa.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return mhs.getNim();
            case 1:
                return mhs.getNama();
            default:
                return "";
        }
    }

    @Override
    public String getColumnName(int column) {
        switch (column) {
            case 0:
                return "NIM";
            case 1:
                return "Nama";
            default:
                return "";
        }
    }
}
```

Penjelasan:

private List<Mahasiswa> daftarMahasiswa = new ArrayList();

- Untuk menyimpan semua data mahasiswa yang akan ditampilkan di tabel.

public MahasiswaTableModel(List<Mahasiswa> mhs) { ... }

- Konstruktor untuk menerima data mahasiswa dari Test.java.

getRowCount()

- Mengembalikan jumlah baris tabel = jumlah data mahasiswa.

  getColumnCount()
- Menentukan jumlah kolom tabel (di sini 2: NIM dan Nama).

  getValueAt(int rowIndex, int columnIndex)
- Mengambil isi sel tabel berdasarkan baris dan kolom (kolom 0 = NIM, kolom 1 = Nama).

  getColumnName(int column)
- Memberi nama header kolom tabel (“NIM” dan “Nama”).

- **DataMahasiswa**

```
 */
package com.mycompany.test1;
import java.util.ArrayList;
import java.util.Scanner;
/**
 *
 * @author celvin paty
 */
public class DataMahasiswa {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        ArrayList<String> mahasiswa = new ArrayList<String>();

        System.out.println("Masukkan nama 10 mahasiswa:");
        for (int i = 0; i < 10; i++) {
            System.out.print("Nama mahasiswa ke-" + (i + 1) + ": ");
            String nama = input.nextLine();
            mahasiswa.add(nama);
        }

        System.out.println("\n====================");
        System.out.println("Daftar Mahasiswa:");
        System.out.println("====================");
        for (int i = 0; i < mahasiswa.size(); i++) {
            System.out.println((i + 1) + ". " + mahasiswa.get(i));
        }
    }
}
```

Penjelasan:

Scanner input = new Scanner(System.in);

ArrayList<String> mahasiswa = new ArrayList<String>();

- Membuat objek Scanner untuk membaca input dari pengguna, dan ArrayList untuk menyimpan daftar nama mahasiswa.

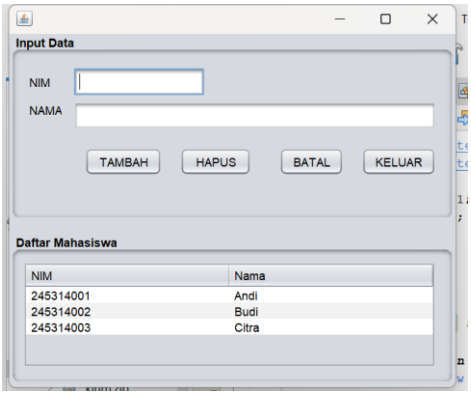
```
System.out.println("Masukkan nama 10 mahasiswa:");
for (int i = 0; i < 10; i++) {
   System.out.print("Nama mahasiswa ke-" + (i + 1) + ": ");
   String nama = input.nextLine();
   mahasiswa.add(nama);
}
```

- Melakukan perulangan sebanyak 10 kali untuk meminta pengguna mengetikkan nama mahasiswa satu per satu, lalu setiap nama disimpan ke dalam ArrayList mahasiswa.

```
System.out.println("\n====================");
System.out.println("Daftar Mahasiswa:");
System.out.println("====================");
for (int i = 0; i < mahasiswa.size(); i++) {
   System.out.println((i + 1) + ". " + mahasiswa.get(i));
}
```

- Setelah semua data dimasukkan, program menampilkan seluruh nama mahasiswa dalam format daftar bernomor urut.

- **Output**
  - **Ss hasil1**



Penjelasan:

Daftar data mahasiswa langsung muncul saat program dijalankan karena di dalam konstruktor public Test() sudah ditambahkan data awal secara manual seperti ini:

Mahasiswa m1 = new Mahasiswa(); m1.setNim("245314001"); m1.setNama("Andi");

Mahasiswa m2 = new Mahasiswa(); m2.setNim("245314002"); m2.setNama("Budi");

Mahasiswa m3 = new Mahasiswa(); m3.setNim("245314003"); m3.setNama("Citra");

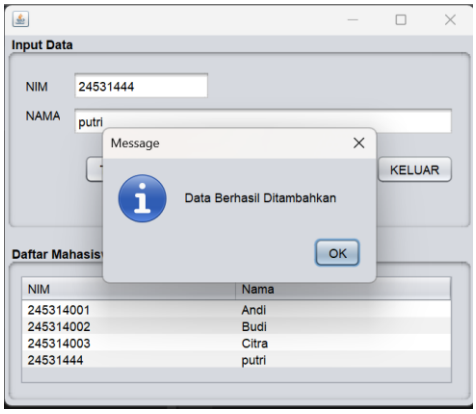
daftarMahasiswa.add(m1);

daftarMahasiswa.add(m2);

daftarMahasiswa.add(m3);

refreshTable();

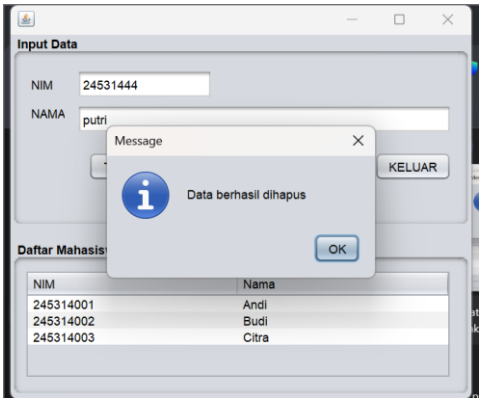
- **Ss hasil2**



Penjelasan:

- Saat tombol TAMBAH ditekan, program mengambil nilai dari field NIM dan NAMA**,** membuat objek baru Mahasiswa, lalu menambahkannya ke ArrayList daftarMahasiswa.
- Setelah itu, method refreshTable() memperbarui tabel, dan muncul pesan "Data Berhasil Ditambahkan" dari JOptionPane.showMessageDialog() sebagai konfirmasi bahwa proses penambahan data sukses.

- **Ss hasil3**

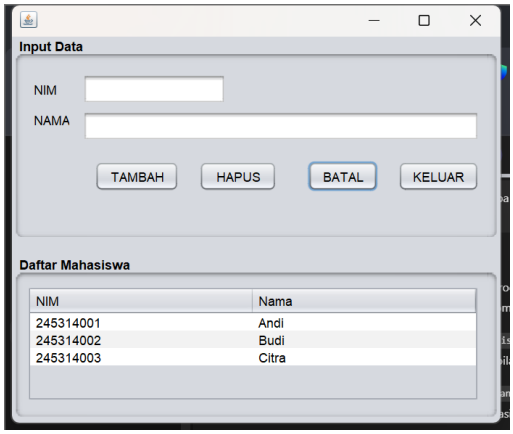


Penjelasan:

- Saat tombol HAPUS ditekan, program mencari data mahasiswa dengan NIM yang sama seperti yang kamu isi di kolom input (24531444).
- Begitu ditemukan, data tersebut dihapus dari ArrayList daftarMahasiswa**,** lalu refreshTable() dijalankan untuk memperbarui tampilan tabel.

- Pesan “Data berhasil dihapus” muncul dari JOptionPane.showMessageDialog() sebagai tanda bahwa proses penghapusan data berhasil.

- **Ss hasil4**



Penjelasan:

- Saat klik tombol BATAL, program menjalankan method clearField(), yang berisi:

  nim.setText("");

  nama.setText("");
- Kode ini mengosongkan (menghapus isi) kedua text field NIM dan NAMA**,** sehingga input yang sebelumnya kamu ketik otomatis terhapus dan kembali kosong.