BAB I
PEMROGRAMAN GENERIC

1.1. Pemrograman Generic

Pemrograman *Generic* merupakan suatu teknik yang digunakan untuk mengatasi permasalahan konsistensi tipe pada *collection*. Pada semester sebelumnya (Mata Kuliah Pemrograman Komputer 1), telah dijelaskan tentang penggunaan *collection*, salah satunya adalah penggunaan `List`. Seperti yang telah kita ketahui, bahwa penggunaan objek `List` adalah sebagai berikut:

```
List listV = new ArrayList();
listV.add("Kota Tegal");
listV.add(85);
listV.add(86.7);
```

Pada penggalan kode di atas, variabel `listV` dapat diisi dengan berbagai tipe data (*objek*). Hal ini akan berdampak buruk ketika kita hendak melakukan manipulasi data, dikarenakan kepastian tipe data yang dimasukkan dalam *collection* `List` bersifat tidak konsisten. Untuk mengatasi permasalahan tersebut dapat menggunakan *generic* sebagai solusinya. Pada Java, kita dapat mendeklarasikan *generic* dengan simbol sebagai berikut:

- `<E>` : untuk elemen dari sebuah *collection*
- `<T>` : untuk tipe
- `<K, V>` : untuk key dan value
- `<N>` : untuk number
- `S,U,V` : digunakan untuk parameter ke `2`, `3`, `4`

1.2. Kelas *Generic*

Salah satu cara realisasi *generic* dapat dilakukan pada kelas (dikenal dengan nama kelas *generic* atau *generic class*). Umumnya, kelas generic digunakan untuk *collection* dengan tanpa menentukan tipe data yang disimpannya. Contoh realisasi kelas generic dapat dilihat pada kode program di bawah ini:

```
public class MyGeneric<T> {
    // realisasi kelas
}
```

Untuk lebih jelasnya, akan diilustrasikan contoh dari pembuatan kelas *generic* dan cara penggunaan kelas *generic* sebagai berikut:

Ilustrasi: Membuat kelas *generic* dengan nama "`MyGeneric`", kemudian untuk melakukan pengujian dibuat kelas baru dengan nama "`MyGenericTest`". Pada kelas "`MyGenericTest`" dilakukan pengujian dengan 2 *generic* tipe data, yaitu "`Integer`" dan "`String`".

```
public class MyGeneric <T> {
      T ob;
      //deklarasi sebuah objek dari tipe T

      MyGeneric (T o){
            ob = o;
      }

      T getob() {
            return ob;
      }

      void showType() {
            System.out.println("Tipe T adalah"+
                ob.getClass().getName());
      }
}
```

```
public class MyGenericTest {
    public static void main(String[] args) {
        MyGeneric<Integer> iOb = new MyGeneric<>(88);
        iOb.showType();

        int v = iOb.getob();
        System.out.println("value : "+v);
        System.out.println();

        MyGeneric<String> strOb = new MyGeneric<>("Generic Test");
        String v2 = strOb.getob();
        strOb.showType();
        System.out.println("value: "+ v2);
        System.out.println();

        //Integer i = (Integer) strOb.getob();
        //Error: cannot cast from String to Integer
    }
}
```

1.3. Method Generic

Selain pada *class*, kita juga dapat membuat generic pada sebuah *method*. Pada dasarnya, *class* dibuat seperti *class* pada umumnya, tetapi pada *method* yang memanipulasi tipe *generic* harus menerapkan *method generic*. Contoh *method generic* adalah sebagai berikut:

```
public class MyGenericMethod {

    public static <E> void showData(E[] data){
        for (E e : data) {
            System.out.println(e);
        }
    }

    public static void main(String[] args) {
        Object[] objects = {1,2,3,"Empat",5};
        MyGenericMethod.showData(objects);
        Integer[] numbers = {1,2,3,4,5};

        MyGenericMethod.showData(numbers);
        MyGenericMethod.showData(objects);
    }
}
```

Pada kode di atas, dideklarasikan elemen *generic* `<E>` atau juga bisa menggunakan generic `<T>` pada *method* `showData()`. Berikutnya, dideklarasikan sebuah *array* `objects` dan `numbers` kemudian diisi dengan data sampel. Setelah itu, dilakukan pemanggilan method `showData()` untuk menampilkan data pada *array* `objects` dan `numbers`.

1.4. *Diamond Interface*

Dalam pemrograman *generic*, terdapat sebuah fitur yang dapat digunakan untuk menyederhanakan dalam penulisan kode program, yaitu dengan menghilangkan tipe atau elemen pada saat instansiasi objeknya. Fitur tersebut tersedia pada Java 7 ke atas.

Contoh:

```
List<Object>  o = new ArrayList<Object>();
List<Integer> i = new ArrayList<Integer>();
```

Dapat dirubah menjadi:

```
List<Object>  o = new ArrayList<>();
List<Integer> i = new ArrayList<>();
```

1.5. Wildcard

Pada kondisi tertentu, kita menginginkan tipe data yang dimasukkan ke dalam objek atau method *generic* dibatasi pada tipe tertentu untuk menjaga konsisten data yang akan dimanipulasi. Teknik tersebut biasa disebut dengan *Wildcard Generic*. *Wildcard generic* memiliki 3 (tiga) bentuk, yaitu: *Unbounded*, *Upper bounded*, dan *Lower bounded*. Sebagai ilustrasi, dibuat method *generic* untuk mengurutkan data dengan metode *bubble sort*. Pada *generic method* di elemen yang masuk dibatasi harus *inheritance* dari `Comparable<E>`.

Contoh:

```
public class GeneicType {
    public static <E extends Comparable<E>> void bubbleSort(E[] list){
       boolean flag = true;
       E temp;
       while (flag) {
           flag = false;
           for (int j = 0; j < list.length-1; j++) {
               if(list[j].compareTo(list[j+1]) > 0){
                   temp = list[j];
                   list[j] = list[j+1];
                   list[j+1] = temp;
                   flag = true;
               }
           }
       }
    }
    public static <E> void showData(E[] list){
        for (E e : list) {
            System.out.println(e);
        }
    }
    public static void main(String[] args) {
        Integer[] numbers = {3,6,1,12,8,9,16,2,7};
        GeneicType.bubbleSort(numbers);
        showData(numbers);
    }
}
```

2.5.1. *Unbounded Wildcard*

Terkadang kita memiliki situasi dimana kita menginginkan metode generik kita untuk bekerja dengan semua tipe data, dalam hal ini wildcard tak terbatas (*unbounded wildcard*) dapat digunakan. Sama seperti menggunakan `<? extends Object>`.

Contoh:

```
public class UnboundedWildcrad {

    public static void printData(List<?> list){
        for(Object obj : list){
            System.out.println(obj);
        }
    }

    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        numbers.add(7);
        numbers.add(2);
        numbers.add(10);
        printData(numbers);

        List<String> fruits = new ArrayList<>();
        fruits.add("Apel");
        fruits.add("Alpukat");
        fruits.add("Wortel");
        printData(fruits);
    }
}
```

Pada penggalan kode di atas, kita dapat menggunakan `List<String>` atau `List <Integer>` atau tipe lainnya dari argumen objek `list` ke metode `printData()`. Hal ini dilakukan untuk memastikan bahwa data yang dimasukkan ke dalam `list` dapat dijaga konsistensinya.

2.5.2. Upper Bounded Wildcard

Upper Bounded Wildcard digunakan untuk membatasi tipe yang tidak dikenal menjadi tipe tertentu atau sub tipenya dengan *keyword* `extends`. Misalkan kita ingin menulis sebuah metode yang akan mengembalikan jumlah angka dalam sebuah list, maka implementasinya dapat kita lakukan dengan cara berikut:

```
public class UpperBoundedWildcard {

    static double sumOf(List<? extends Number> list){
       double sum = 0;
       for(Number n : list){
           sum += n.doubleValue();
       }
       return sum;
    }
```

```
    public static void main(String[] args) {
        List numbers = new ArrayList<>();
        numbers.add(80);
        numbers.add(90.65);
        numbers.add(70.87);
        numbers.add(0.89);
        double sum = sumOf(numbers);
        System.out.println(sum);
    }
}
```

2.5.3. Lower Bounded Wildcard

Lower Bounded Wildcard digunakan untuk membatasi tipe yang tidak dikenal menjadi tipe tertentu atau super tipenya dengan *keyword* `super`. Sintaks yang digunakan adalah `<? super T>`. Misalkan kita ingin menulis sebuah metode yang akan mencetak semua item yang ada dalam sebuah `list` dan dapat berfungsi dalam `List<Integer>`, `List<Number>` dan `List<Object>` atau semua tipe yang mengandung nilai tipe `Integer`. Normalnya kita akan menggunakan kode seperti berikut:

```
public static void printItems(List<Integer> list){
    for (Integer num : list){
        System.out.println(num);
    }
}
```

Tetapi, metode di atas tidak dapat berfungsi untuk `List<Number>` dan `List<Object>`. Untuk mengatasi hal tersebut, kita dapat menggunakan *lower bounded wildcard* dengan menggunakan `List<? super Integer>` dan juga sesuai dengan semua super tipe dari Integer. Contoh penggunaan metode yang yang benar adalah sebagai berikut:

Contoh:

```
public static void print(List<? super Integer> list){
    for (Object num : list){
        System.out.println(num);
    }
}
```

Contoh Kode Lengkap:

```
public class LowerBoundedExample {

    static void printItems(List<? super Integer> list){
        for (Object num : list){
            System.out.println(num);
        }
    }
```

```
    public static void main(String[] args) {
        List<Integer> list1 = new ArrayList<>();
        list1.add(1);
        list1.add(2);
        list1.add(3);
        List<Number> list2 = new ArrayList<>();
        list2.add(1.4);
        list2.add(2.3);
        list2.add(3.4);
        printItems(list1);
        printItems(list2);
    }
}
```

1.6. Praktik

1.6.1. Membuat Projek Baru

Untuk membuat project baru di Netbeans ikuti langkah-langkah berikut:

1. Klik menu **File → New Project**, selanjutnya akan terbuka dialog **New Project**
2. Pada dialog **New Project**, untuk *Categories* pilih **Java**, dan untuk jenis **Projects** pilih **Java Application →** klik **Next** untuk melanjutkan ke tahapan berikutnya.
3. Pada dialog **New Java Application**, isi field dan pilihan yang ada sebagai berikut:
   A. Project Name : **`P01-Generic`**
   B. Project Location : Lokasi opsional, sesuai dengan keinginan anda
   C. Project Folder : Otomatis oleh Netbeans IDE
   D. Create Main Class : **Hilangkan centang pada checkbox**
4. Klik **Finish**

1.6.2. Membuat Paket Baru

Untuk membuat paket baru dalam sebuah project, ikuti langkah-langkah berikut:

1. Klik kanan pada Node **Source Package → New → Java Package**. Selanjutnya Netbeans akan membuka dialog **New Java Package**.
2. Pada dialog **New Java Package**, isikan "***generic***" pada kotak isian nama paket (*package name*)
3. Klik **Finish**.

1.6.3. Membuat Form Baru

Untuk membuat form baru dalam sebuah project, ikuti langkah-langkah berikut:

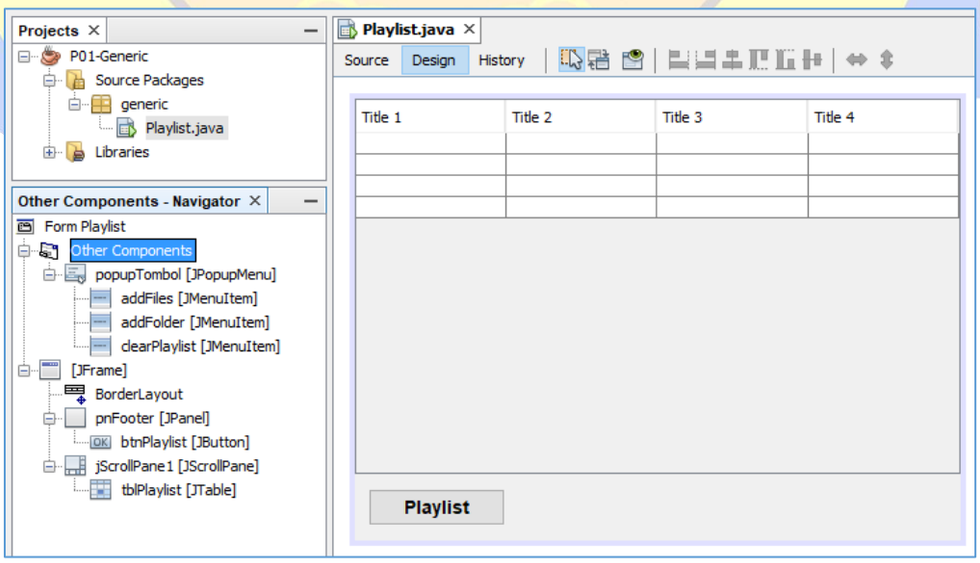
1. Klik kanan pada *package* ***generic*** → klik **New** → klik **JFrame Form**. Selanjutnya Netbeans akan menampilkan dialog untuk membuat form baru (dialog **New JFrame Form**).
2. Pada field **Class Name**, isikan "**Playlist**".
3. Klik **Finish**.

1.6.4. Mendesain Form

Setelah *form* selesai dibuat, berikutnya desainlah *form* tersebut dengan mengikuti langkah-langkah berikut:

1. Atur layout *form* menjadi "`Border Layout`".
2. Tambahkan komponen/palette "**`JPanel`**" pada posisi "BOTTOM". Selanjutnya, *rename* nama komponen dengan "`pnFooter`", dan ubah tinggi panel menjadi `50` piksel.
3. Tambahkan komponen/palette "**`JTable`**" pada posisi "CENTER". Selanjutnya, *rename* nama komponen dengan "`tblPlaylist`".
4. Tambahkan komponen/palette "**`JButton`**" pada "`pnFooter`" (letakkan pada posisi kiri). Kemudian ubah teks-Nya menjadi "**`Playlist`**", dan ubah namanya menjadi "`btnPlaylist`".
5. Tambahkan komponen/palette "**`Popup Menu`**" pada "Other Components". Kemudian *rename* nama komponen menjadi "`popupTombol`".
6. Selanjutnya tambahkan 3 (tiga) komponen/palette "**`Menu Item`**" pada "`popupTombol`". Kemudian ubah nama ketiga komponen tersebut seperti berikut:
    a. **Teks:** Add File(s), **Nama:** addFiles
    b. **Teks:** Add Folder, **Nama:** addFolder
    c. **Teks:** Clear Playlist, **Nama:** clearPlaylist

Setelah langkah-langkah di atas selesai, maka desain *form*, struktur *project*, dan struktur komponen pada navigator akan terlihat seperti Gambar 1 di bawah ini:



Gambar 1. Desain *form* playlist (Pemrograman Generic)

1.6.5. Membuat Kelas Musik

Objek ini digunakan untuk menyimpan informasi detail dari music, seperti nama file, lokasi file, ukuran file, dan ekstensi file. Untuk membuat objek ini ikuti langkah-langkah berikuut:

1. Klik kanan pada *package* ***generic*** →pilih **New** → pilih **Java Class**. Selanjutnya Netbeans akan menampilkan dialog untuk membuat kelas baru (dialog **New Java Class**).
2. Pada field **Class Name**, isikan **Musik**.
3. Klik **Finish**.

Selanjutnya, edit kode program sehinggan menjadi seperti di bawah ini:

```
public class Musik {
    String path, fileName, fileSize, extention;

    public Musik(String p, String fn, String fs, String e) {
        this.path = p;
        this.fileName = fn;
        this.fileSize = fs;
        this.extention = e;
    }

    public String getPath() {
        return path;
    }

    public String getFileName() {
        return fileName;
    }

    public String getFileSize() {
        return fileSize;
    }

    public String getExtention() {
        return extention;
    }
}
```

1.6.6. Membuat Kelas KoleksiMusik

Objek ini digunakan untuk menyimpan kumpulan objek `Music` yang digunakan sebagai *generic* dalam praktik pemrograman *generic* ini. Untuk membuat objek ini ikuti langkah-langkah berikuut:

1. Klik kanan pada *package* **generic** → **New** → **Java Class**. Selanjutnya Netbeans akan menampilkan dialog untuk membuat kelas baru (dialog **New Java Class**).
2. Pada field **Class Name**, isikan **KoleksiMusik**.
3. Klik **Finish**.

Selanjutnya, edit kode program sehingga menjadi seperti di bawah ini:

```
import java.util.ArrayList;
import java.util.List;
import javax.swing.table.AbstractTableModel;

public class KoleksiMusik extends AbstractTableModel {
    List<Musik> list = new ArrayList<>();

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

    @Override
    public int getColumnCount() {
        return 2;
    }

    @Override
   synchronized
    public Object getValueAt(int rowIndex, int columnIndex) {
        switch(columnIndex){
            case 0 : return list.get(rowIndex).getFileName();
            case 1 : return list.get(rowIndex).getFileSize();
            default: return "";
        }
    }
    @Override
    public String getColumnName(int column) {
        switch(column){
            case 0 : return "JUDUL LAGU";
            case 1 : return "UKURAN";
            default: return "";
        }
    }

    public void add(Musik m){
       list.add(m);
       fireTableRowsInserted(getRowCount(),getColumnCount());
    }

    public void set(int i , Musik m){
        list.set(i, m);
        fireTableDataChanged();
    }

    public void clear(){
        list.clear();
        fireTableDataChanged();
    }

    public void remove(int row){
        list.remove(row);
        fireTableRowsDeleted(row, row);
    }

    public Musik get(int row){
        return (Musik) list.get(row);
    }

}
```

1.6.7. Membuat Method dan Memberikan Event pada Komponen

Kelas `Musik` dan `KoleksiMusik` telah selesai dibuat, untuk tahapan selanjutnya adalah melakukan pengkodingan dan menambahkan *event* pada setiap komponen serta melakukan pengaturan-pengaturan lainnya. Untuk melakukannya buka `Playlist.java`, kemudian ikuti langkah-langkah berikut:

1. Pastikan kelas-kelas yang dibutuhkan dalam kelas ini telah di-import semua.

```
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.io.File;
import java.text.DecimalFormat;
import java.util.List;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.SwingWorker;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.TableColumn;
import javax.swing.table.TableColumnModel;
```

*) Lewati langkah di atas, jika anda adalah programmer Java.

2. Lakukan instansiasi kelas `KoleksiMusik` (baris `20`)

```
15  /**
16   *
17   * @author nishom
18   */
19  public class Playlist extends javax.swing.JFrame {
20      KoleksiMusik koleksi = new KoleksiMusik();
```

3. Buat beberapa *method* yang diperlukan dalam aplikasi. Diantaranya:

Method untuk pengaturan lebar dan tinggi kolom pada tabel

```
private void resizeListener(){
    addComponentListener(new ComponentAdapter() {
        @Override
        public void componentResized(ComponentEvent e) {
            resizeColumns();
        }
    });
}

private void resizeColumns() {
    float[] columnWidthPercentage = {90.0f,10.0f};
    int tW = tblPlaylist.getWidth();
    TableColumn column;
    TableColumnModel jTableColumnModel = tblPlaylist.getColumnModel();
    int cantCols = jTableColumnModel.getColumnCount();
    for (int i = 0; i < cantCols; i++) {
        column = jTableColumnModel.getColumn(i);
        int pWidth = Math.round(columnWidthPercentage[i] * tW);
        column.setPreferredWidth(pWidth);
        tblPlaylist.setRowHeight(27);
    }
}
```

Method untuk membaca ukuran file

```
 private String fileSizeOf(File file){
        DecimalFormat format = new DecimalFormat("#.##");
        long MB = 1024 * 1024;
        long KB = 1024;
        final double length = file.length();
        if (length > MB) {
            return format.format(length / MB) + " MB";
        }
        if (length > KB) {
            return format.format(length / KB) + " KB";
        }
        return format.format(length) + " B";
   }
```

Method untuk mendapatkan ekstensi sebuah file

```
private String extensionOf(File file) {
   String fileExtension="";
   String fileName=file.getName();
   if(fileName.contains(".") && fileName.lastIndexOf(".")!= 0){
         fileExtension =
         fileName.substring(fileName.lastIndexOf(".")+1);
   }
   return fileExtension;
}
```

Method untuk mengkoleksi semua file mp3 yang dipilih

```
private void addFiles(File[] files){
   for (File file : files) {
         String path = file.getAbsolutePath();
         String fn = file.getName();
         String fileName = fn.substring(0, fn.length()-4);
         String fileSize = fileSizeOf(file);
         String extension = "";
         int i = path.lastIndexOf('.');
         if (i > 0) {
             extension = extensionOf(file);
         }
         Musik m = new Musik(path,fileName,fileSize,extension);
         koleksi.add(m);
   }
}
```

Method untuk membaca semua file mp3 di dalam folder dan sub-folder

```
private void addFolder(File dir){
   File[] listOfFiles = dir.listFiles();
   for (File listOfFile : listOfFiles) {
         if (listOfFile.isFile()) {
               String path = listOfFile.getAbsolutePath();
               String fn = listOfFile.getName();
               String fileName = fn.substring(0, fn.length()-4);
               String fileSize = fileSizeOf(listOfFile);
               String extension;
               int i = path.lastIndexOf('.');
               if (i > 0) {
                     extension = extensionOf(listOfFile);
                     if("mp3".equalsIgnoreCase(extension)){
                           Musik m = new
                           Musik(path,fileName,fileSize,extension);
                           koleksi.add(m);
                     }
               }
         } else if (listOfFile.isDirectory()) {
               addFolder(listOfFile);
         }
   }
}
```

4. Selanjutnya, lakukan pemanggilan beberapa method pada konstruktor.

```
22      public Playlist() {
23          initComponents();
24          tblPlaylist.setModel(koleksi);
25          resizeColumns();
26          resizeListener();
27      }
```

Keterangan:

Baris 24: mengatur model tabel

Baris 25: mengubah ukuran kolom pada tabel

Baris 26: melakukan *resize* otomatis ketika jendela aplikasi di-resize.

5. Selanjutnya tambahkan event pada komponen-komponen berikut:

Event pada tombol "Playlist"

```
private void btnPlaylistActionPerformed(java.awt.event.ActionEvent evt) {
    popupTombol.show(btnPlaylist,
            btnPlaylist.getWidth(),
            btnPlaylist.getHeight()/2);
}
```

Event pada menu "Add File(s)"

```
private void addFilesActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fc = new JFileChooser();
    fc.setDialogType(JFileChooser.FILES_ONLY);
    fc.setMultiSelectionEnabled(true);
    fc.setDialogTitle("Add Files");
    fc.setAcceptAllFileFilterUsed(false);
    fc.setFileFilter(new FileNameExtensionFilter("MP3 File (*.mp3)","mp3"));
    fc.setApproveButtonText("Add Files");
    int show = fc.showOpenDialog(this);
    if(show == JFileChooser.APPROVE_OPTION){
        File[] files = fc.getSelectedFiles();
        addFiles(files);
    }
}
```

Event pada menu "Add Folder"

```
private void addFolderActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fc = new JFileChooser();
    fc.setDialogType(JFileChooser.DIRECTORIES_ONLY);
    fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    fc.setDialogTitle("Add Folder");
    fc.setApproveButtonText("Add Folder");
    int show = fc.showOpenDialog(this);
    if(show == JFileChooser.APPROVE_OPTION){
        File file = fc.getSelectedFile();
        addFolder(file);
    }
}
```

Event pada menu "Clear Playlist"

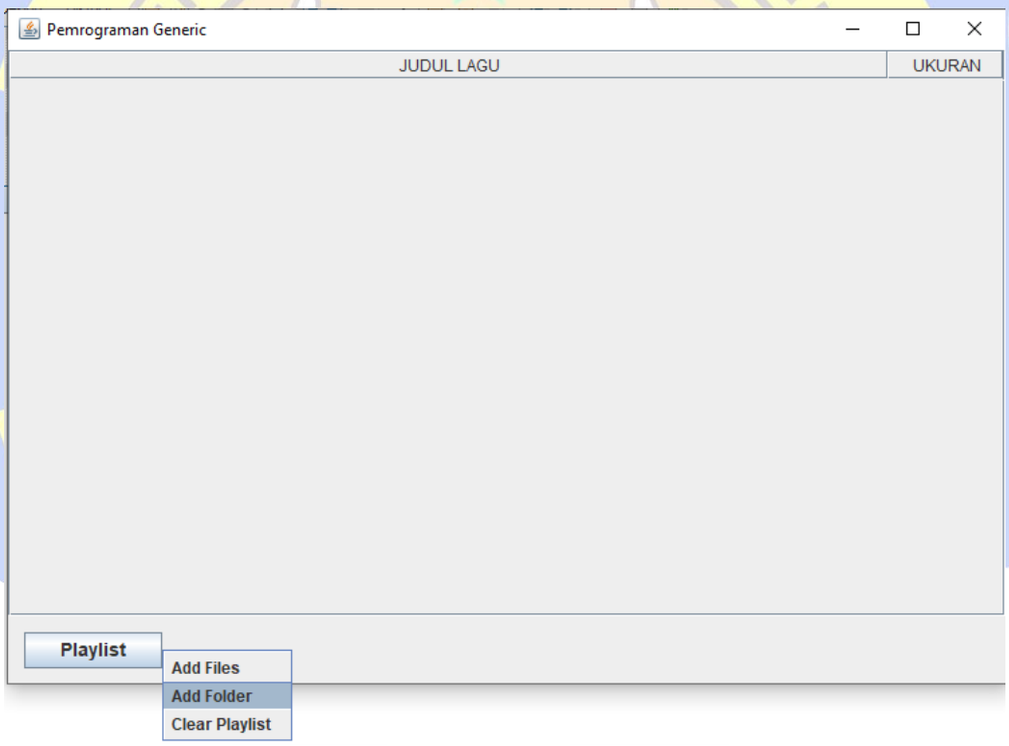
```
private void clearPlaylistActionPerformed(java.awt.event.ActionEvent evt) {
    koleksi.clear();
}
```

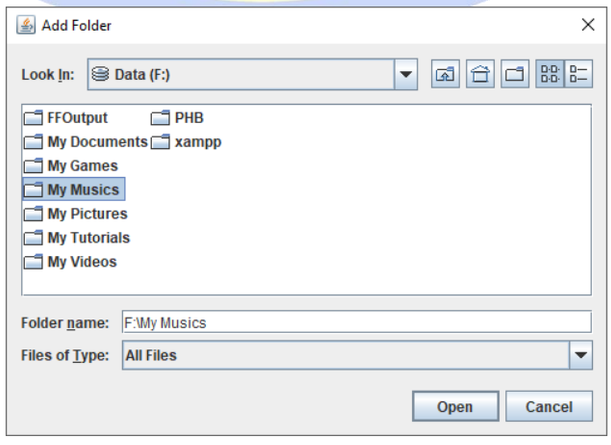
Event pada tabel ketika diklik 2 kali (*double click)*

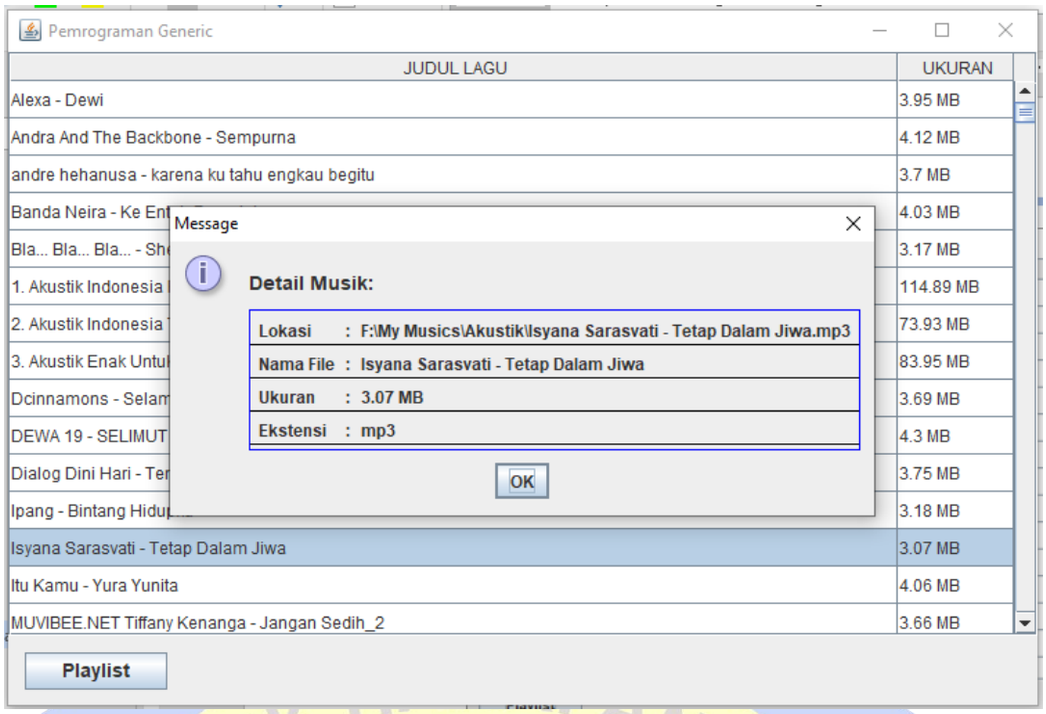
```
private void tblPlaylistMouseClicked(java.awt.event.MouseEvent evt) {
    int i = tblPlaylist.getSelectedRow();
    if (evt.getClickCount() == 2 && i != -1) {
        Musik m = koleksi.get(i);
        JOptionPane.showMessageDialog(this,
        "<html>"
        + "<head>"
        + "<style>"
        + "table { border-collapse: collapse; border: lpx solid blue; }"
        + "tr { border-bottom:lpt solid black; }"
        + "</style>"
        + "</head>"
        + "<body>"
        + "<h3>Detail Musik:</h3>"
        + "<table>"
        + "<tr><td>Lokasi</td><td>:</td><td> "+ m.getPath()+ "</td></tr>"
        + "<tr><td>Nama File</td><td>:</td><td> "+ m.getFileName()+ "</td></tr>"
        + "<tr><td>Ukuran</td><td>:</td><td> "+ m.getFileSize()+ "</td></tr>"
        + "<tr><td>Ekstensi</td><td>:</td><td> "+ m.getExtention()+ "</td></tr>"
        + "</table>"
        + "</body>"
        + "</html>");
    }
}
```

Jalankan Program: Klik tool atau tekan **F6**

Hasil Praktik:







1.7. Tugas

Buatlah sebuah aplikasi berbasis GUI dengan mengimplementasikan beberapa poin berikut:

a. *Kelas Generic*
b. *Method Generic*
c. *Diamond Interface*
d. *Wildcard (Unbounded, Upper/Lower Bounded)*
e. Format nama project adalah KELAS_NIM_NAMA

Pengumpulan tugas:

- ✓ Tugas kolektif ke komting
- ✓ Tugas dikirim ke email: m.nishom.dosen@gmail.com paling lambat 1 (satu) minggu dari pelaksanaan perkuliahan, pukul 23:59:59.