Mengolah Row Jtable Berbasis Vector pada Pemrograman Java

(Working with Rows in Vector Based Jtable in Java Programming Language)

Rinta Kridalukmana, S.Kom., M.T.
Program Studi Sistem Komputer, Universitas Diponegoro

rintakrida@undip.ac.id

Abstract

Jtable is a component in Java Programming Language that is used to display and edit two-dimensional tables of cells. The first dimension will be a column and the other is row. Basically, Jtable in can be based on vector or object. In this paper, working with rows in Vector-based Jtable will be ligthed on inserting multiple rows, deleting multiple rows and refresh rows. By using getMinSelectionIndex() and getMaxSelectionIndex(), we can get the minimum or maximum index selected rows in Jtable. After that, the addNotify() function will execute inserting, deleting or refrehing rows in Jtable. Keystroke support will be enabled also to trigger the execution of function that have been developed.

Keywords : Jtable, rows, Java

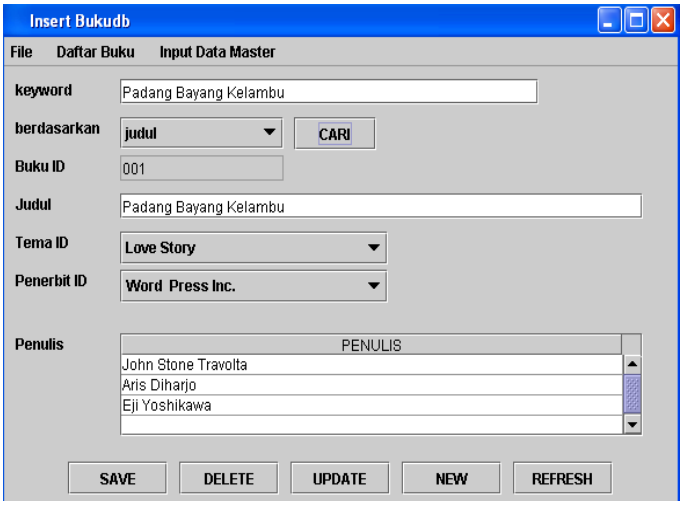
Pendahuluan

Jtable dalam lingkungan bahasa pemrograman Java biasa digunakan untuk menampilkan data dalam bentuk tabel, di mana data tersebut dapat juga berasal dari database. Bila untuk keperluan menampilkan data saja, maka pada Jtable tidak perlu dilakukan operasi-operasi perubahan nilai pada sel tabel dan bahkan tabel dapat diset agar bersifat *non-editable.*

Operasi pada Jtable yang meliputi penambahan baris, penghapusan baris, atau refresh tabel berguna apabila Jtable akan digunakan sebagai fasilitas untuk melakukan input data, terutama pada database. Sebagai contoh pada suatu database terdapat suatu tabel yang berisi data daftar penulis suatu buku, di mana tiap buku dapat terdiri dari satu atau lebih penulis. Karena jumlah yang tidak pasti dari penulis buku, maka komponen input data pada suatu form, seperti misalnya text box, akan sulit untuk mengakomodasi kebutuhan tersebut. Hal ini disebabkan text box hanya dapat berisi satu nilai data saja.

Berbeda bila menggunakan komponen input berbentuk tabel, yang dapat memasukkan beberapa data sekaligus. Untuk mendapatkan gambaran yang lebih jelas perbedaan

penggunaan komponen tabel dengan komponen lainnya dapat dilihat pada **Gambar 1** di bawah ini.



Gambar 1. Contoh Input Form

Pada Gambar 1, nama-nama penulis akan diinput dengan menggunakan Jtable karena dapat terdiri lebih dari satu nilai. Karena jumlah penulis yang tidak pasti, maka tabel harus dapat bersifat dinamis, tidak statis. Yang dimaksud statis di sini adalah bahwa tabel diset hanya untuk menampung 6 baris data saja. Sehingga sejak awal tampilan tabel hanya terdapar 6 baris data. Sedangkan yang dimaksud dinamis adalah bahwa sejak awal tabel mungkin hanya ditampilkan 2 baris data, namun untuk selanjutnya dapat ditambah sesuai kebutuhan, atau dapat dikurangi apabila ada penghapusan data.

Berbeda dengan menggunakan pemrograman visual, pada pemrograman Java operasi pada baris, seperti misalnya penambahan, penyisipan, atau penghapusan baris, *statement* harus dideklarasikan secara manual. Sebaliknya, pada pemrograman visual operasi tersebut dapat secara otomatis ter-*generate*. Oleh karena harus dideklarasikan sendiri oleh *programmer*, maka tiap *programmer* dapat menggunakan teknik yang berbeda-beda dalam mendeklarasikan *statement* tersebut.

Perumusan Masalah

Dari penjelasan di atas, maka permasalahan yang akan dibahas adalah perumusan *statement* dalam pemrograman Java untuk melakukan operasi baris pada JTable yang meliputi penambahan baris, penghapusan baris dan me-*refresh* tabel.

Batasan Masalah

Pembahasan akan difokuskan pada *statement* untuk mendefinisikan JTable dan operasi pada JTable, sedangkan komponen Java lainnya yang mendukung dalam pembahasan seperti JFrame dan JScrollPane tidak akan dijabarkan lebih lanjut.

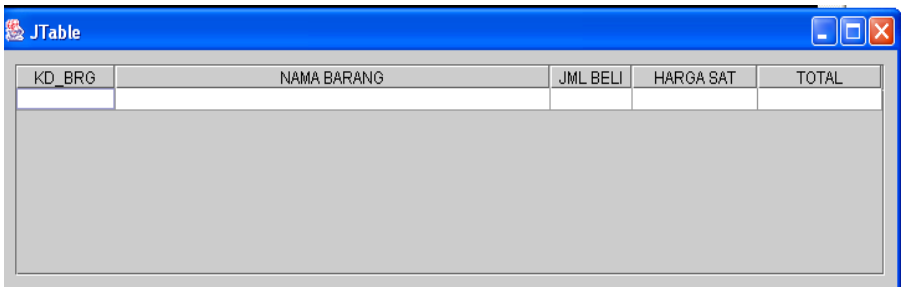
Alur Pembahasan Masalah

Pada awal pembahasan akan ditentukan terlebih dahulu asumsi kondisi awal yang meliputi bentuk form yang akan dibuat dan bagaimana cara form bekerja, deskripsi tiap operasi yang akan dilakukan, dan deskripsi fitur pendukung dalam membantu operasi tabel, yaitu *keystroke*.

Setelah membuat asumsi kondisi awal, akan dilanjutkan dengan pendefinisian *statement* operasi pada Jtabel, pendefinisian JTable, dan fitur pendukung. Dan pada akhir pembahasan akan dilakukan penggabungan seluruh statement tersebut dengan komponen pendukung lainnya sehingga membentuk kesatuan pemrograman java yang akan diberi nama tabel.class.

Asumsi Kondisi Awal

Untuk lebih dapat memberikan gambaran yang jelas dalam operasi pada JTable, dapat dilihat dari bagaimana JTable bekerja dalam sebuah form. Dengan menggunakan form yang memuat JTable seperti yang terlihat pada Gambar 2, penulis akan mencoba memberikan gambaran bagaimana operasi baris bekerja.



Gambar 2. Form JTable

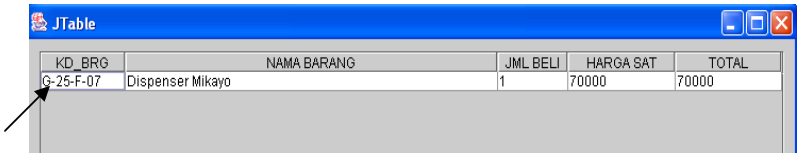
Pada awal pemanggilan form, terdapat 1 baris JTable. Operasi baris yang dapat dilakukan pada form ini adalah sebagai berikut :

1. Penambahan baris (1 baris atau beberapa baris)

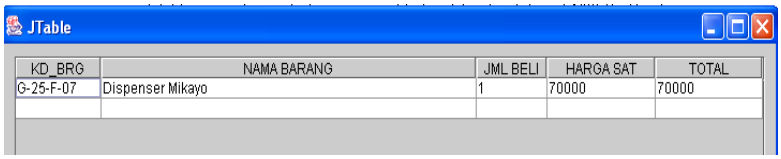
   Untuk melakukan penambahan baris pada Form JTable, dilakukan dengan menekan tombol ENTER. Berikut ini beberapa variasi operasi penambahan baris pada JTable :

   a. Penambahan 1 baris di bawah baris yang dipilih

      Untuk melakukan penambahan 1 baris di bawah baris yang dipilih, letakkan pointer pada baris yang dikehendaki dan tekan ENTER. Hasil akan terlihat seperti berikut ini :



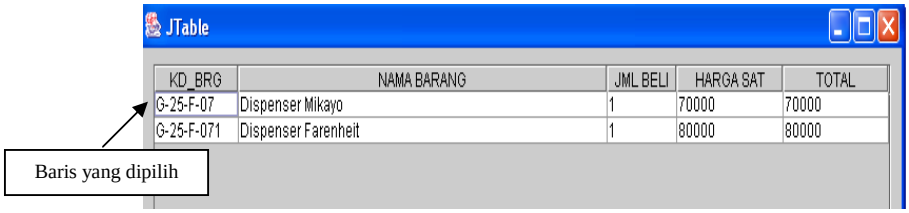
Kondisi Awal



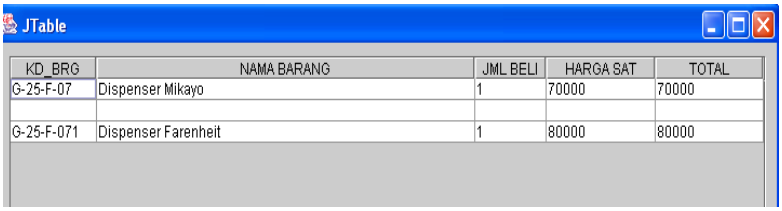
Setelah ditekan tombol ENTER

   b. Penyisipan 1 baris di bawah baris yang dipilih

      Untuk menyisipkan satu baris di bawah baris yang dipilih letakkan pointer pada baris yang dikehendaki dan tekan ENTER. Hasil akan terlihat seperti berikut ini :



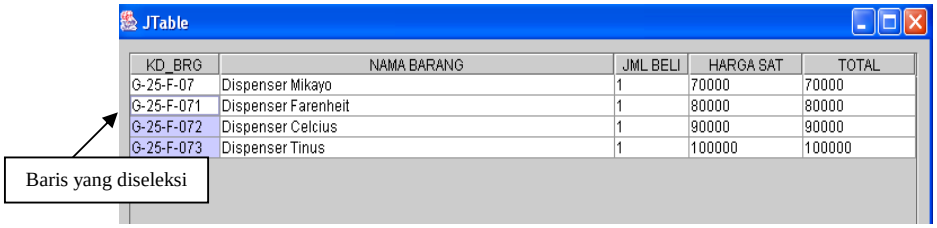
Kondisi Awal



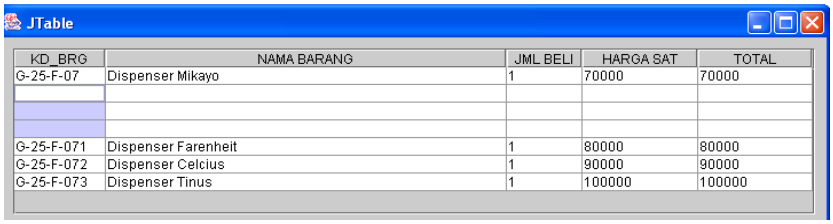
Setelah ditekan tombol ENTER

c. Penyisipan lebih dari satu baris

Untuk penyisipan atau penambahan lebih dari satu baris, seleksi beberapa baris yang diinginkan, kemudian tekan F1. Hasil akan terlihat seperti berikut ini :



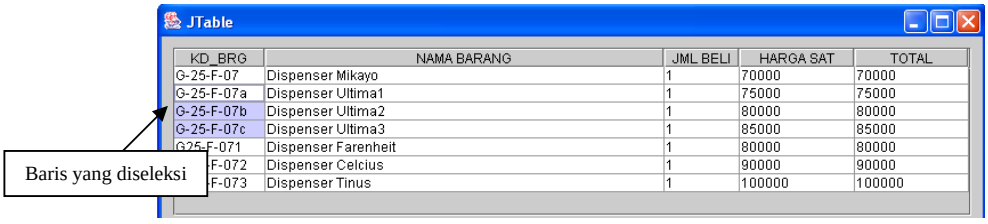
Kondisi Awal



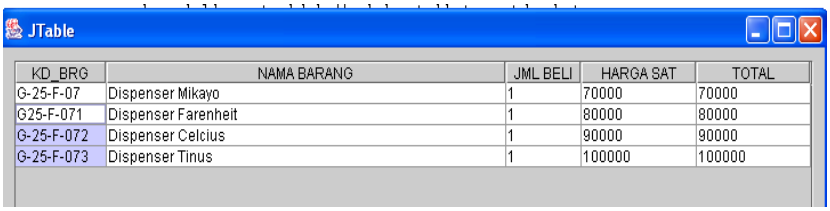
Setelah ditekan tombol F1

2. Penghapusan baris (1 baris atau lebih baris)

Untuk melakukan penghapusan 1 atau lebih baris, seleksi baris seperti pada operasi penambahan baris, lalu tekan tombol DELETE. Baris dapat diseleksi sebanyak 1 baris atau lebih. Hasil akan terlihat seperti berikut ini :



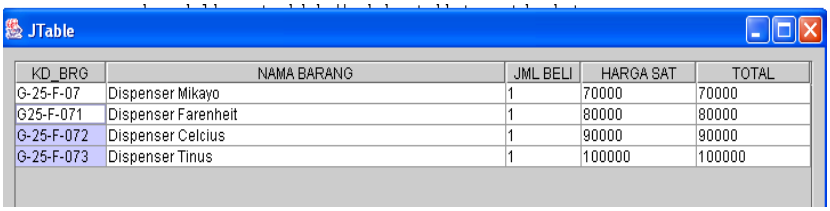
Kondisi Awal



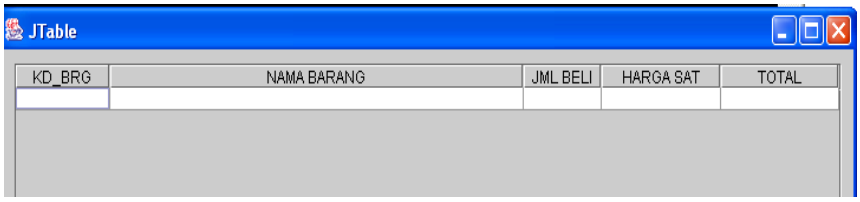
Setelah ditekan tombol DELETE

3. Refresh table

   Operasi refresh berguna untuk mengembalikan tabel kedalam keadaan awal, yang dalam hal ini seperti yang terlihat pada Gambar 2.Untuk melakukannya adalah dengan menekan tombol ESCAPE (ESC).



Kondisi Awal



Setelah ditekan tombol ESCAPE

Pendefinisian JTable dan Statement Operasi Baris JTable

Untuk melakukan operasi baris seperti yang telah dijelaskan di atas, terdapat beberapa prosedur dan statement yang harus didefinisikan terlebih dahulu yang akan dijabarkan seperti berikut ini :

1. Pembuatan JTable

   Statement berikut akan mendeklarasikan pembuatan JTable :

```
1-- String[] columnNames = {"KD_BRG","NAMA BARANG","JML
2-- BELI","HARGA SAT","TOTAL"};
3-- JTable table ;
4-- DefaultTableModel tabModel = new DefaultTableModel(){
5--    public boolean isCellEditable(int row, int column) {
6--    return true; }
7--    };
8-- Vector rows = new Vector();
9-- Vector header = new Vector();
10- addColumns(columnNames);
11- tabModel.setDataVector(rows,header);
12- table=new JTable(tabModel);
```

   String columnNames merupakan header dari kolom tabel yang akan dibuat. Tabel ini diberi nama table melalui deklarasi statement pada baris ke 3. Baris 4 – 7 mendeklarasikan model tabel yang digunakan dan diberi nama tabModel.

Selanjutnya baris 8 dan 9 menyatakan pembuatan vector rows dan header yang akan mengisi table. Baris 11 memasukkan vector pengisi table ke dalam tabModel dan selanjutnya pada baris 12 mendeklarasikan pembuatan table yang sesuai dengan model yang dibuat (tabModel) dan berisi vector rows dan header.

2. buatRowKosong()

   Prosedur ini berfungsi untuk mengisi baris yang ditambahkan dengan nilai kosong. Adapun *statement*-nya adalah seperti berikut ini :

```
public Vector buatRowKosong()
{
     Vector t = new Vector();
     t.addElement((String) "");
     t.addElement((String) "");
     t.addElement((String) "");
     t.addElement((String) "");
     t.addElement((String) "");
     return t;
 }
```

   Karena tabel berbasis vector, maka harus dideklarasikan pembuatan vektor baru terlebih dahulu yang diberi nama t. Dalam vector tersebut kemudian diisi nilai kosong dengan statement addElement yang bertipe String. Jumlah penambahan element disesuaikan dengan jumlah kolom pada tabel.

3. addRow()

   Prosedur ini berfungsi untuk menambah 1 baris pada tabel yang berisi baris kosong. Adapun *statement*-nya adalah seperti berikut ini :

```
public void addRow()
{
    Vector r=new Vector();
    r=buatRowKosong();
    rows.addElement(r);
    table.addNotify();
}
```

   Pembuatan vector r mengawali prosedur addRow(). Pada baris kedua, memerintahkan pemanggilan prosedur buarRowKosong(). Baris ketiga mengisi variabel rows (lihat pada poin no 1 – Pembuatan JTable). Statement table.addNotify() akan melakukan eksekusi perubahan pada tabel.

4. insertRow()

   Prosedur ini berfungsi untuk melakukan operasi penyisipan 1 baris pada tabel. Adapun *statement*-nya adalah seperti berikut ini :

```
public void insertRow() //Add Row
{
   Vector r=new Vector();
   r=buatRowKosong();
   int ind = table.getSelectedRow() + 1;
   for (int i = ind; i <= ind; i++){
     rows.insertElementAt(r,i);
     table.addNotify();
   }
}
```

   Yang perlu diperhatikan pada prosedur ini adalah pernyataan variabel ind yang berguna untuk memperoleh index baris yang dipilih. Selanjutnya berdasarkan index ini akan mengarahkan posisi baris yang akan disisipi dengan penggunaan statement rows.insertElementAt(r,i). Variabel r adalah pemanggilan prosedur pembuatan row kosong.

5. insertMultiRow()

   Prosedur ini berfungsi untuk melakukan penyisipan untuk 1 atau lebih baris pada tabel. Adapun statement-nya adalah seperti berikut ini :

```
public void insertMultiRow()
{

     int y = table.getSelectionModel().getMinSelectionIndex();
     int z = table.getSelectionModel().getMaxSelectionIndex();
     for (int i = y; i <= z; i++){
         Vector r=new Vector();
         r=buatRowKosong();
         rows.insertElementAt(r,i);
         table.addNotify();
     }
 }
```

   Yang perlu diperhatikan di sini adalah pendefinisian variabel y dan z. Dengan getMin/MaxSelectionIndex(), akan didapat index baris awal dan akhir dari beberapa baris yang diseleksi. Selanjutnya proses dilakukan seperti halnya pada prosedur insertRow().

6. deleteMultiRow()

   Prosedur ini berfungsi untuk melakukan penghapusan 1 atau lebih baris pada tabel. Adapun *statement*-nya adalah seperti berikut ini :

```
void deletemultiRow()
{
     int y = table.getSelectionModel().getMinSelectionIndex();
     int z = table.getSelectionModel().getMaxSelectionIndex();
     for (int i = z; i >= y; i--){
         rows.removeElementAt(i);
         table.addNotify();
     }
    }
```

   Seperti halnya prosedur insertMultiRow(), prosedur ini bekerja berdasarkan getMin/MaxSelectionIndex() sebagai indeks untuk menyeleksi baris mana yang akan dihapus.

7. refreshTable()

   Prosedur ini berfungsi untuk mengembalikan tabel pada kondisi awal sama seperti saat form pertama kali dibuka. Adapun *statement*-nya adalah seperti berikut ini :

```
void refreshtable()
{
     int y = table.getRowCount()-1;
     for (int i = y; i >= 0; i--){
         rows.removeElementAt(i);
         table.addNotify();
     }
     addRow();
}
```

   Tabel terlebih dahulu dihitung jumlah barisnya dengan mendeklarasikan variabel y. Setelah itu, dengan removeElementAt() satu persatu baris akan dihapus yang dimulai dari baris terakhir.

8. Prosedur untuk merespon tombol ENTER, ESC, DELETE, F1

   Agar tombol-tombol ENTER, ESC, DELETE dan F1 pada keyboard dapat berfungsi untuk melakukan operasi tabel, maka diperlukan statement yang dapat ditulis seperti di bawah ini :

```
//Statement tekantombol
KeyStroke tekandelete =
KeyStroke.getKeyStroke(KeyEvent.VK_DELETE, 0, false);
table.getInputMap().put(tekandelete, "DELETE");
table.getActionMap().put("DELETE", deleteRowAction);

KeyStroke tekanenter =
KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0, false);
table.getInputMap().put(tekanenter, "ENTER");
table.getActionMap().put("ENTER", insertRowAction);
```

```
KeyStroke tekanesc =
KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0, false);
table.getInputMap().put(tekanesc, "ESCAPE");
table.getActionMap().put("ESCAPE", refreshRowAction);

KeyStroke tekanf1 =
KeyStroke.getKeyStroke(KeyEvent.VK_F1, 0, false);
table.getInputMap().put(tekanf1, "F1");
table.getActionMap().put("F1", insertMultiRowAction);
```

Perhatikan huruf yang tercetak tebal, yang merupakan penghubung dengan prosedur yang akan merespon aksi apabila tombol ditekan. Berikut adalah prosedur yang dipanggil oleh huruf yang tercetak tebal tersebut.

```
//Aksi tekan tombol
Action deleteRowAction = new AbstractAction()
{
    public void actionPerformed(ActionEvent e)
    {
          deletemultiRow();
    }
 };
Action insertRowAction = new AbstractAction()
{
    public void actionPerformed(ActionEvent e)
    {
         insertRow();
    }
 };
Action refreshRowAction = new AbstractAction()
{
    public void actionPerformed(ActionEvent e)
    {
          refreshtable();
    }
};
Action insertMultiRowAction = new AbstractAction()
{
    public void actionPerformed(ActionEvent e)
    {
        insertMultiRow();
    }
 };
```

Masing-masing prosedur di atas selanjutnya dihubungkan dengan operasi-operasi tabel yang telah dideklarasikan sebelumnya.

Membuat tabel.java

File dengan ekstension java merupakan file hasil penulisan program pada pemrograman java yang setelah di-*compile* akan membentuk file dengan ekstension class. Berikut adalah contoh pemrograman java yang akan menghasilkan form dengan operasi tabel yang telah dijabarkan di atas.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.table.*;
import java.util.*;


public class tabel9 extends JFrame
{

   String[] columnNames =
   {"KD_BRG","NAMA BARANG","JML BELI","HARGA SAT","TOTAL"};
   JTable table ;
   DefaultTableModel tabModel = new DefaultTableModel(){
      public boolean isCellEditable(int row, int column)
      { return true; }
   };
   Vector rows = new Vector();
   Vector header = new Vector();

public static void main(String args[])
    {
      new tabel().show();
    }


public tabel9()
   {
      super("JTable");
      setLocation(300,400);
      Container c = this.getContentPane();
      GridBagConstraints d = new GridBagConstraints();
      c.setLayout(new GridBagLayout());
      addColumns(columnNames);
      tabModel.setDataVector(rows,header);
      table=new JTable(tabModel);

      TableColumn column0 = table.getColumnModel().getColumn(0);
      TableColumn column1 = table.getColumnModel().getColumn(1);
      TableColumn column2 = table.getColumnModel().getColumn(2);
      TableColumn column3 = table.getColumnModel().getColumn(3);
      TableColumn column4 = table.getColumnModel().getColumn(4);
      column0.setPreferredWidth(30);
      column1.setPreferredWidth(300);
      column2.setPreferredWidth(10);
      column3.setPreferredWidth(50);
      column4.setPreferredWidth(50);
      JScrollPane sp = new JScrollPane(table);
      sp.setPreferredSize(new Dimension(700,150));
      table.setCellSelectionEnabled(true);
```

```
      table.setRowSelectionAllowed(true);

      addRow();
      d.insets = new Insets(10,10,10,10);
      c.add(sp, d);

      //masukkan statement tekantombol



       this.addWindowListener(new WindowAdapter()
       {
            public void windowClosing(WindowEvent ev)
            {
                exitDialog();
            }
        });

       this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
       this.pack();
   }

//Masukkan prosedur-prosedur operasi tabel
private void exitDialog()
     {
      this.dispose();
      System.exit(0);
     }

}
```

Kesimpulan

Kunci utama dalam operasi baris pada JTable adalah penentuan index baris. Apabila operasi bekerja untuk *multiple-row* maka dapat dipergunakan getMinSelectionIndex() atau getMaxSelectionIndex(). Namun bila hanya 1 baris saja dapat menggunakan getSelectedRow(). Selanjutnya untuk mendeklarasikan apakah operasi tersebut merupakan operasi penambahan, penghapusan, atau penyisipan dapat digunakan addElement(), insertElementAt() atau removeElementAt(). Sedangkan syntax addNotify() akan menjalankan operasi yang telah didefinisikan.

Daftar Pustaka

Anonymous, *Setting the Height and Width of Rows and Columns in JTable*, www.roseindia.net, 2007

Kadir, Abdul, *Dasar Pemrograman JavaTM 2*, ANDI Yogyakarta, Yogyakarta : 2004-2005

Wicaksono, Ady, *Dasar Pemrograman Java 2*, PT Elex Media Komputindo, Jakarta : 2002

Banerjee, Ashook & Mehta, Jignesh, *Enable Copy and Paste Functionality between Swing's JTables and Excel*, javaworld.com, 1999

Hermawan, Benny, *Menguasai Java 2 & Object Oriented Programming*, ANDI Yogyakarta, Yogyakarta, 2004

Kaufhold, Christian, *JTable and TableModel Layout*, www.chka.de, 2001

Deshpande, Kanad, *A Simple Way to JTable*, FindMyHosting.com, 2005