```java
package kalkulator;

import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.text.NumberFormat;
import java.util.StringTokenizer;
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import java.util.ArrayList;

public class KalkulatorMatriks {
    private boolean CEK = true;
    private boolean INFO = true;
    private static int max = 100;
    private static int decimals =3;
    private int iDF = 0;
    private JTextArea taA, taB, taC;
    private JLabel statusBar;
    private int n = 4;
    private static NumberFormat nf;

    public Component buatTampilan(){
        //TEXT AREA UNTUK MASUKAN ANGKA
        taA = new JTextArea();
        taB = new JTextArea();
        taC = new JTextArea();

        //membuat panel
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
        panel.add(MatrixPane("Matrix A", taA));
        panel.add(Box.createRigidArea(new Dimension(10, 0)));
        panel.add(MatrixPane("Matrix B", taB));
        panel.add(Box.createRigidArea(new Dimension(10, 0)));
        panel.add(MatrixPane("Hasil", taC));;

        //Membuat dan Menambahkan button
        JPanel paneBtn = new JPanel();
        paneBtn.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
        paneBtn.setLayout(new GridLayout(3, 3));
        JButton btnApB = new JButton("A + B ");
        JButton btnAmB = new JButton("A - B ");
        JButton btnAkB = new JButton("A * B ");
        JButton btnInvA = new JButton("invers(A) ");
        JButton btnInvB = new JButton("invers(B) ");
        JButton btnTrnsA = new JButton("transpose(A) ");
                JButton btnTrnsB = new JButton("transpose(B) ");
```

```
        JButton btnDetA = new JButton("|A| ");
        JButton btnDetB = new JButton("|B| ");
                JButton btnDelA = new JButton ("Hapus Matrix A");
                JButton btnDelB = new JButton ("Hapus Matrix B");
                JButton btnDelC = new JButton ("Hapus Hasil");
                JButton btnCls = new JButton ("Hapus Semua" );
        paneBtn.add(btnApB);
        paneBtn.add(btnAmB);
        paneBtn.add(btnAkB);
        paneBtn.add(btnInvA);
        paneBtn.add(btnInvB);
        paneBtn.add(btnTrnsA);
                paneBtn.add(btnTrnsB);
        paneBtn.add(btnDetA);
        paneBtn.add(btnDetB);
                paneBtn.add(btnDelA);
                paneBtn.add(btnDelB);
                paneBtn.add(btnDelC);
                paneBtn.add(btnCls);

        /* == Menambahkan BUTTON Listeners untuk memanggil suatu fungsi atau method ==
           */
        btnApB.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                try {
                    MenampilkanMatrix(Plus(BacaMatrix(taA),
                            BacaMatrix(taB)), taC);
                } catch (Exception e) {
                    System.err.println("Error: " + e);
                }
            }
        });
                btnAmB.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                try {
                    MenampilkanMatrix(min(BacaMatrix(taA),
                            BacaMatrix(taB)), taC);
                } catch (Exception e) {
                    System.err.println("Error: " + e);
                        }
                    }
                });
                     btnAkB.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        try {
                            MenampilkanMatrix(kali(BacaMatrix(taA),
                                    BacaMatrix(taB)), taC);
                        } catch (Exception e) {
                            System.err.println("Error: " + e);
                        }
                    }
                });
                       btnInvA.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        try {
                            MenampilkanMatrix(Invers(BacaMatrix(taA)),taC);
                        } catch (Exception e) {
                            System.err.println("Error: " + e);
                        }
                    }
                });
```

```
                        btnInvB.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        try {
                            MenampilkanMatrix(Invers(BacaMatrix(taB)),taC);
                        } catch (Exception e) {
                            System.err.println("Error: " + e);
                        }
                    }
                });
                        btnTrnsA.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        try {
                            MenampilkanMatrix(Transpose(BacaMatrix(taA)),taC);
                        } catch (Exception e) {
                            System.err.println("Error: " + e);
                        }
                    }
                });
                        btnTrnsB.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        try {
                            MenampilkanMatrix(Transpose(BacaMatrix(taB)),taC);
                        } catch (Exception e) {
                            System.err.println("Error: " + e);
                        }
                    }
                });
                         btnDetA.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        try {
                            taC.setText("Determinant A: " + nf.format(determinant(BacaMa
    trix(taA))));
                        } catch (Exception e) {
                            System.err.println("Error: " + e);
                        }
                    }
                });
                         btnDetB.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        try {
                            taC.setText("Determinant B: " + nf.format(determinant(BacaMa
    trix(taB))));
                        } catch (Exception e) {
                            System.err.println("Error: " + e);
                        }
                    }
                });
                         btnDelA.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        taA.setText("");
                    }
                });
                         btnDelB.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        taB.setText("");
                    }
                });
                         btnDelC.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent evt) {
                        taC.setText("");
                    }
```

```
181.             });
182.                     btnCls.addActionListener(new ActionListener() {
183.                 public void actionPerformed(ActionEvent evt) {
184.                                             taA.setText("");
185.                                             taB.setText("");
186.                                             taC.setText("");
187.                 }
188.             });
189.
190.                     // Main Panel
191.                   JPanel pane = new JPanel();
192.                     pane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
193.             pane.setLayout(new BoxLayout(pane, BoxLayout.X_AXIS));
194.             pane.add(panel);
195.             pane.add(paneBtn);
196.
197.                     JPanel fpane = new JPanel();
198.                     fpane.setLayout(new BorderLayout());
199.             fpane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
200.             fpane.add("Center", pane);
201.             statusBar = new JLabel("");
202.             fpane.add("South", statusBar);
203.                     return fpane;
204.         }
205.
206.             //pengaturan penyesuaian ukuran matrix panel
207.                 private JPanel MatrixPane(String str, JTextArea ta) {
208.             JScrollPane scrollPane = new JScrollPane(ta);
209.             int size = 200;
210.
211.                     scrollPane.setPreferredSize(new Dimension(size, size));
212.             JLabel label = new JLabel(str);
213.             label.setLabelFor(scrollPane);
214.
215.             JPanel pane = new JPanel();
216.             pane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
217.             pane.setLayout(new BoxLayout(pane, BoxLayout.Y_AXIS));
218.             pane.add(label);
219.             pane.add(scrollPane);
220.             return pane;
221.                 }
222.
223.             public static void main(String[] args) {
224.                 JFrame frame= new JFrame("Kalkulator Matrix");
225.                 frame.setSize(new Dimension(800, 200));
226.                 KalkulatorMatriks app = new KalkulatorMatriks();
227.                 Component contents = app.buatTampilan();
228.                 frame.getContentPane().add(contents, BorderLayout.CENTER);
229.                 frame.addWindowListener(new WindowAdapter() {
230.                     public void windowClosing(WindowEvent e) {
231.                         System.exit(0);
232.                     }
233.                 });
234.                 frame.pack();
235.                 frame.setVisible(true);
236.                 nf = NumberFormat.getInstance();
237.                 nf.setMinimumFractionDigits(1);
238.                 nf.setMaximumFractionDigits(decimals);
239.             }
240.     //    -----------------------------------------------------
241.                   //akhir dari tampilan
```

```
242.       //    ----------------------------------------------------
243.
244.               public double[][] BacaMatrix (JTextArea ta) throws Exception{
245.               if(CEK){
246.                   System.out.println("Membaca Matriks");
247.               }
248.               // Menguraikan Text Area
249.                   String kosong = ta.getText();
250.                   String pisah  = "";
251.                   int i =0;
252.                   int j =0;
253.                   int [] rsize = new int [max];
254.
255.               // Memecah Sring
256.                   StringTokenizer ts = new StringTokenizer(kosong, "\n");
257.               while (ts.hasMoreTokens()) {
258.                           StringTokenizer ts2 = new StringTokenizer(ts.nextToken());
259.                           while (ts2.hasMoreTokens()) {
260.                               ts2.nextToken();
261.                               j++;
262.                           }
263.                           rsize[i] = j;
264.                           i++;
265.                           j = 0;
266.                       }
267.
268.               statusBar.setText("Ukuran Matriks : " + i + "x" + i);
269.                       if ((CEK) || (INFO)) {
270.                           System.out.println("Ukuran Matriks : " + i);
271.               }
272.               for (int c = 0; c < i; c++) {
273.                           if (CEK) {
274.                               System.out.println("i=" + i + "  j=" +rsize[c] + "   Kol
    om : " + c);
275.                           }
276.                           if (rsize[c] != i) {
277.                               statusBar.setText("Ukuran Matriks yang Dimasukan Tidak S
    ama");
278.                               throw new Exception("Ukuran Matriks yang Dimasukan Tidak
     Sama");
279.                           }
280.                       }
281.
282.               /* == Mengatur Ukuran Matriks == */
283.               n = i;
284.               double matrix[][] = new double[n][n];
285.               i = j = 0;
286.               pisah = "";
287.
288.               // == Melakukan pengecekan ukuran matriks kemudian menampilkanya di stat
    us bar
289.               StringTokenizer st = new StringTokenizer(kosong, "\n");
290.               while (st.hasMoreTokens()) {
291.                           StringTokenizer st2 = new StringTokenizer(st.nextToken());
292.                           while (st2.hasMoreTokens()) {
293.                               pisah = st2.nextToken();
294.                               try {
295.                                   matrix[i][j] = Double.valueOf(pisah).doubleValue();

296.                               } catch (Exception exception) {
297.                                   statusBar.setText("Angka Tidak Valid ");
```

```
                                }
                                j++;
                            }
                            i++;
                            j = 0;
                        }

                if (CEK) {
                            System.out.println("Membaca Matriks:");
                            System.out.println("Ukuran Matriks :" + i);
                    for (i = 0; i < n; i++) {
                                for (j = 0; j < n; j++) {
                                    System.out.print("m[" + i + "][" + j + "] = "+ m
    atrix[i][j] + "    ");
                                }
                                System.out.println();
                    }
                }
                        return matrix;
            }
                // Menampilkan Matriks di Text Area
            public void MenampilkanMatrix(double [][] matrix, JTextArea ta){
                if( CEK){
                }
                String rstr = "";
                String dv = "";

                for (int i = 0; i < matrix.length; i++) {
                    for (int j = 0; j < matrix[i].length; j++) {
                        dv = nf.format(matrix[i][j]);
                        rstr = rstr.concat(dv + "  ");
                    }
                    rstr = rstr.concat("\n");
                }
                ta.setText(rstr);
            }

           //Set Penghitungan rumus
                public double [][] Plus (double [][] a, double [][] b) {
                int tmpa = a.length;
                int tmpb = b.length;
                if (tmpa != tmpb) {
                            statusBar.setText("Ukuran Matriks Tidak Sama");
                }
                double matrix[][] = new double[tmpa][tmpb];
                        for (int i = 0; i < tmpb; i++)
                            for (int j = 0; j < tmpb; j++) {
                                matrix[i][j] = a[i][j] + b[i][j];
                            }
                        return matrix;
                }

                public double [][] min (double [][]a, double [][] b) {
                    int tmpa = a.length;
                    int tmpb = b.length;
                    if (tmpa != tmpb) {
                    statusBar.setText("Ukuran Matriks Tidak Sama");
                    }
                double matrix[][] = new double[tmpa][tmpb];
                for (int i = 0; i < tmpb; i++)
                            for (int j = 0; j < tmpb; j++) {
```

```
                                matrix[i][j] = a[i][j] - b[i][j];
                            }
                        return matrix;
                }

                public double [][] kali (double [][]a, double [][] b) {
                    int tmpa = a.length;
                    int tmpb = b.length;
                    if (tmpa != tmpb) {
                        statusBar.setText("Ukuran Matriks Tidak Sama");
                    }
                    double matrix[][] = new double[tmpa][tmpb];
                for (int i = 0; i < a.length; i++)
                            for (int j = 0; j < b[i].length; j++)
                                matrix[i][j] = 0;
                        for(int i = 0; i < matrix.length; i++){
                            for(int j = 0; j < matrix[i].length; j++){
                                matrix[i][j] =bariskolom(a,i,b,j);
                            }
                        }
                        return matrix;
            }
                public double bariskolom(double [][] A, int row, double [][] B, int col)
    {
                    double perkalian = 0;
                for(int i = 0; i < A[row].length; i++)
                            perkalian +=A[row][i]*B[i][col];
                return perkalian;
            }

                public double[][] Invers (double [][]a) {
                // rumus untuk menghitung matriks
                // inv(A) = 1/det(A) * adj(A)

                        if (INFO) {
                    System.out.println("Mencari Invers...");
                }
                int tma = a.length;

                double m[][] = new double[tma][tma];
                double mm[][] = Adjoint(a);

                double det = determinant(a);
                double dd = 0;

                if (det == 0) {
                            statusBar.setText("Determinan sama dengan 0, tidak bisa diba
    lik.");
                            if (INFO) {
                                System.out.println("Determinant sama dengan 0, tidak bis
    a dibalik.");
                            }
                        } else {
                            dd = 1 / det;
                }
                for (int i = 0; i < tma; i++)
                            for (int j = 0; j < tma; j++) {
                                m[i][j] = dd * mm[i][j];
                            }
                return m;
                }
```

```
416.
417.                 public double[][] Adjoint(double[][] a) {
418.                 if (INFO) {
419.                             System.out.println("Mencari Adjoint...");
420.                 }
421.                         int tma = a.length;
422.                 double m[][] = new double[tma][tma];
423.                 int ii, jj, ia, ja;
424.                 double det;
425.
426.                 for (int i = 0; i < tma; i++)
427.                             for (int j = 0; j < tma; j++) {
428.                                 ia = ja = 0;
429.                                 double ap[][] = new double[tma - 1][tma - 1];
430.                                 for (ii = 0; ii < tma; ii++) {
431.                                     for (jj = 0; jj < tma; jj++) {
432.                                         if ((ii != i) && (jj != j)) {
433.                                             ap[ia][ja] = a[ii][jj];
434.                                             ja++;
435.                                         }
436.                                     }
437.                                     if ((ii != i) && (jj != j)) {
438.                                         ia++;
439.                                     }
440.                                     ja = 0;
441.                                 }
442.                                 det = determinant(ap);
443.                                 m[i][j] = (double) Math.pow(-1, i + j) * det;
444.                             }
445.                 m = Transpose(m);
446.                 return m;
447.             }
448.
449.                 public double[][] SegitigaAtas(double[][] m) {
450.                     if (INFO) {
451.                         System.out.println("Mengubah Bentuk Ke segitia Atas...");
452.                     }
453.                         double f1 = 0;
454.                 double temp = 0;
455.                 int tma = m.length;
456.                 int v = 1;
457.
458.                 iDF = 1;
459.             for (int kol = 0; kol < tma - 1; kol++) {
460.                     for (int bar = kol + 1; bar < tma; bar++) {
461.                         v = 1;
462.                         luar: while (m[kol][kol] == 0) // cek jika 0 di diagonal
463.                         {
464.                             if (kol + v >= tma) // cek jika mengganti semua baris
465.                             {
466.                                 iDF = 0;
467.                                 break luar;
468.                             } else {
469.                                 for (int c = 0; c < tma; c++) {
470.                                     temp = m[kol][c];
471.                                     m[kol][c] = m[kol + v][c]; // switch rows
472.                                     m[kol + v][c] = temp;
473.                                 }
474.                                 v++; // mennghitung baris yang diganti
475.                                 iDF = iDF * -1; // setiap ganti mengubah determinan
476.                             }
```

```
                        }
                        if (m[kol][kol] != 0) {
                            if (CEK) {
                                System.out.println("Ukuran Matriks = " + tma + "   kolom
    = " + kol + "   baris= " + bar);
                            }
                            try {
                                f1 = (-1) * m[bar][kol] / m[kol][kol];
                                for (int i = kol; i < tma; i++) {
                                    m[bar][i] = f1 * m[kol][i] + m[bar][i];         }
                            } catch (Exception e) {
                                System.out.println("Maaf Masih Sampai Disini");
                            }
                        }
                    }
                }
                return m;
                }

                public double determinant(double[][] matrix) {
                    if (INFO) {
                        System.out.println("Mencari Determinan...");
                    }
                    int tma = matrix.length;
                    double det = 1;
                    matrix = SegitigaAtas(matrix);
                    for (int i = 0; i < tma; i++) {
                        det = det * matrix[i][i];
                    } // Mengalikan diagonal bawah

                    det = det * iDF;
                    if (INFO) {
                        System.out.println("Determinant: " + det);
                    }
                    return det;
                }

                public double[][] Transpose(double[][] a) {
                    if (INFO) {
                        System.out.println("Mencari Transpose...");
                    }
                    double m[][] = new double[a[0].length][a.length];
                    for (int i = 0; i < a.length; i++)
                        for (int j = 0; j < a[i].length; j++)
                            m[j][i] = a[i][j];
                    return m;
                }
        }
```