

```

1. package kalkulator;
2.
3. import java.awt.BorderLayout;
4. import java.awt.Component;
5. import java.awt.Dimension;
6. import java.awt.GridLayout;
7. import java.awt.event.ActionEvent;
8. import java.awt.event.ActionListener;
9. import java.awt.event.WindowAdapter;
10. import java.awt.event.WindowEvent;
11. import java.text.NumberFormat;
12. import java.util.StringTokenizer;
13. import javax.swing.BorderFactory;
14. import javax.swing.Box;
15. import javax.swing.BoxLayout;
16. import javax.swing.JButton;
17. import javax.swing.JFrame;
18. import javax.swing.JLabel;
19. import javax.swing.JPanel;
20. import javax.swing.JScrollPane;
21. import javax.swing.JTextArea;
22. import java.util.ArrayList;
23.
24. public class KalkulatorMatriks {
25.     private boolean CEK = true;
26.     private boolean INFO = true;
27.     private static int max = 100;
28.     private static int decimals = 3;
29.     private int idF = 0;
30.     private JTextArea taA, taB, taC;
31.     private JLabel statusBar;
32.     private int n = 4;
33.     private static NumberFormat nf;
34.
35.     public Component buatTampilan(){
36.         //TEXT AREA UNTUK MASUKAN ANGKA
37.         taA = new JTextArea();
38.         taB = new JTextArea();
39.         taC = new JTextArea();
40.
41.         //membuat panel
42.         JPanel panel = new JPanel();
43.         panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
44.         panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
45.         panel.add(MatrixPane("Matrix A", taA));
46.         panel.add(Box.createRigidArea(new Dimension(10, 0)));
47.         panel.add(MatrixPane("Matrix B", taB));
48.         panel.add(Box.createRigidArea(new Dimension(10, 0)));
49.         panel.add(MatrixPane("Hasil", taC));
50.
51.         //Membuat dan Menambahkan button
52.         JPanel paneBtn = new JPanel();
53.         paneBtn.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
54.         paneBtn.setLayout(new GridLayout(3, 3));
55.         JButton btnApB = new JButton("A + B ");
56.         JButton btnAmB = new JButton("A - B ");
57.         JButton btnAkB = new JButton("A * B ");
58.         JButton btnInvA = new JButton("invers(A) ");
59.         JButton btnInvB = new JButton("invers(B) ");
60.         JButton btnTrnsA = new JButton("transpose(A) ");
61.         JButton btnTrnsB = new JButton("transpose(B) ");

```

```

62.     JButton btnDetA = new JButton("|A| ");
63.     JButton btnDetB = new JButton("|B| ");
64.         JButton btnDelA = new JButton ("Hapus Matrix A");
65.         JButton btnDelB = new JButton ("Hapus Matrix B");
66.         JButton btnDelC = new JButton ("Hapus Hasil");
67.         JButton btnCls = new JButton ("Hapus Semua" );
68.     paneBtn.add(btnApB);
69.     paneBtn.add(btnAmB);
70.     paneBtn.add(btnAkB);
71.     paneBtn.add(btnInvA);
72.     paneBtn.add(btnInvB);
73.     paneBtn.add(btnTrnsA);
74.         paneBtn.add(btnTrnsB);
75.     paneBtn.add(btnDetA);
76.     paneBtn.add(btnDetB);
77.         paneBtn.add(btnDelA);
78.         paneBtn.add(btnDelB);
79.         paneBtn.add(btnDelC);
80.         paneBtn.add(btnCls);
81.
82.     /* == Menambahkan BUTTON Listeners untuk memanggil suatu fungsi atau method ==
    */
83.     btnApB.addActionListener(new ActionListener() {
84.         public void actionPerformed(ActionEvent evt) {
85.             try {
86.                 MenampilkanMatrix(Plus(BacaMatrix(taA),
87.                     BacaMatrix(taB)), taC);
88.             } catch (Exception e) {
89.                 System.err.println("Error: " + e);
90.             }
91.         }
92.     });
93.     btnAmB.addActionListener(new ActionListener() {
94.         public void actionPerformed(ActionEvent evt) {
95.             try {
96.                 MenampilkanMatrix(min(BacaMatrix(taA),
97.                     BacaMatrix(taB)), taC);
98.             } catch (Exception e) {
99.                 System.err.println("Error: " + e);
100.            }
101.        }
102.    });
103.    btnAkB.addActionListener(new ActionListener() {
104.        public void actionPerformed(ActionEvent evt) {
105.            try {
106.                MenampilkanMatrix(kali(BacaMatrix(taA),
107.                    BacaMatrix(taB)), taC);
108.            } catch (Exception e) {
109.                System.err.println("Error: " + e);
110.            }
111.        }
112.    });
113.    btnInvA.addActionListener(new ActionListener() {
114.        public void actionPerformed(ActionEvent evt) {
115.            try {
116.                MenampilkanMatrix(Invers(BacaMatrix(taA)),taC);
117.            } catch (Exception e) {
118.                System.err.println("Error: " + e);
119.            }
120.        }
121.    });

```

```

122.         btnInvB.addActionListener(new ActionListener() {
123.             public void actionPerformed(ActionEvent evt) {
124.                 try {
125.                     MenampilkanMatrix(Invers(BacaMatrix(taB)),taC);
126.                 } catch (Exception e) {
127.                     System.err.println("Error: " + e);
128.                 }
129.             }
130.         });
131.         btnTrnsA.addActionListener(new ActionListener() {
132.             public void actionPerformed(ActionEvent evt) {
133.                 try {
134.                     MenampilkanMatrix(Transpose(BacaMatrix(taA)),taC);
135.                 } catch (Exception e) {
136.                     System.err.println("Error: " + e);
137.                 }
138.             }
139.         });
140.         btnTrnsB.addActionListener(new ActionListener() {
141.             public void actionPerformed(ActionEvent evt) {
142.                 try {
143.                     MenampilkanMatrix(Transpose(BacaMatrix(taB)),taC);
144.                 } catch (Exception e) {
145.                     System.err.println("Error: " + e);
146.                 }
147.             }
148.         });
149.         btnDetA.addActionListener(new ActionListener() {
150.             public void actionPerformed(ActionEvent evt) {
151.                 try {
152.                     taC.setText("Determinant A: " + nf.format(determinant(BacaMa
trix(taA))));
153.                 } catch (Exception e) {
154.                     System.err.println("Error: " + e);
155.                 }
156.             }
157.         });
158.         btnDetB.addActionListener(new ActionListener() {
159.             public void actionPerformed(ActionEvent evt) {
160.                 try {
161.                     taC.setText("Determinant B: " + nf.format(determinant(BacaMa
trix(taB))));
162.                 } catch (Exception e) {
163.                     System.err.println("Error: " + e);
164.                 }
165.             }
166.         });
167.         btnDelA.addActionListener(new ActionListener() {
168.             public void actionPerformed(ActionEvent evt) {
169.                 taA.setText("");
170.             }
171.         });
172.         btnDelB.addActionListener(new ActionListener() {
173.             public void actionPerformed(ActionEvent evt) {
174.                 taB.setText("");
175.             }
176.         });
177.         btnDelC.addActionListener(new ActionListener() {
178.             public void actionPerformed(ActionEvent evt) {
179.                 taC.setText("");
180.             }

```

```

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 String
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
275.              om : " + c);
276.          }
277.          if (rsize[c] != i) {
278.              statusBar.setText("Ukuran Matriks yang Dimasukan Tidak S
279.              ama");
280.              throw new Exception("Ukuran Matriks yang Dimasukan Tidak
281.              Sama");
282.          }
283.      }
284.
285.      /* == Mengatur Ukuran Matriks == */
286.      n = i;
287.      double matrix[][] = new double[n][n];
288.      i = j = 0;
289.      pisah = "";
290.
291.      // == Melakukan pengecekan ukuran matriks kemudian menampilkanya di stat
292.      us bar
293.
294.      StringTokenizer st = new StringTokenizer(kosong, "\n");
295.      while (st.hasMoreTokens()) {
296.          StringTokenizer st2 = new StringTokenizer(st.nextToken());
297.          while (st2.hasMoreTokens()) {
298.              pisah = st2.nextToken();
299.              try {
300.                  matrix[i][j] = Double.valueOf(pisah).doubleValue();
301.              } catch (Exception exception) {
302.                  statusBar.setText("Angka Tidak Valid ");
303.              }
304.          }
305.          i++;
306.          j = 0;
307.      }

```

```

298.         }
299.         j++;
300.     }
301.     i++;
302.     j = 0;
303. }
304.
305.     if (CEK) {
306.         System.out.println("Membaca Matriks:");
307.         System.out.println("Ukuran Matriks : " + i);
308.         for (i = 0; i < n; i++) {
309.             for (j = 0; j < n; j++) {
310.                 System.out.print("m[" + i + "][" + j + "] = " + m
atrix[i][j] + " ");
311.             }
312.             System.out.println();
313.         }
314.     }
315.     return matrix;
316. }
317. // Menampilkan Matriks di Text Area
318. public void MenampilkanMatrix(double [][] matrix, JTextArea ta){
319.     if(CEK){
320.     }
321.     String rstr = "";
322.     String dv = "";
323.
324.     for (int i = 0; i < matrix.length; i++) {
325.         for (int j = 0; j < matrix[i].length; j++) {
326.             dv = nf.format(matrix[i][j]);
327.             rstr = rstr.concat(dv + " ");
328.         }
329.         rstr = rstr.concat("\n");
330.     }
331.     ta.setText(rstr);
332. }
333.
334. //Set Penghitungan rumus
335. public double [][] Plus (double [][] a, double [][] b) {
336.     int tmpa = a.length;
337.     int tmpb = b.length;
338.     if (tmpa != tmpb) {
339.         statusBar.setText("Ukuran Matriks Tidak Sama");
340.     }
341.     double matrix[][] = new double[tmpa][tmpb];
342.     for (int i = 0; i < tmpb; i++)
343.         for (int j = 0; j < tmpb; j++) {
344.             matrix[i][j] = a[i][j] + b[i][j];
345.         }
346.     return matrix;
347. }
348.
349. public double [][] min (double [][]a, double [][] b) {
350.     int tmpa = a.length;
351.     int tmpb = b.length;
352.     if (tmpa != tmpb) {
353.         statusBar.setText("Ukuran Matriks Tidak Sama");
354.     }
355.     double matrix[][] = new double[tmpa][tmpb];
356.     for (int i = 0; i < tmpb; i++)
357.         for (int j = 0; j < tmpb; j++) {

```

```

358.             matrix[i][j] = a[i][j] - b[i][j];
359.         }
360.         return matrix;
361.     }
362.
363.     public double [][] kali (double [][]a, double [][] b) {
364.         int tmpa = a.length;
365.         int tmpb = b.length;
366.         if (tmpa != tmpb) {
367.             statusBar.setText("Ukuran Matriks Tidak Sama");
368.         }
369.         double matrix[][] = new double[tmpa][tmpb];
370.         for (int i = 0; i < a.length; i++)
371.             for (int j = 0; j < b[i].length; j++)
372.                 matrix[i][j] = 0;
373.         for(int i = 0; i < matrix.length; i++){
374.             for(int j = 0; j < matrix[i].length; j++){
375.                 matrix[i][j] = bariskolom(a,i,b,j);
376.             }
377.         }
378.         return matrix;
379.     }
380.     public double bariskolom(double [][] A, int row, double [][] B, int col)
381.     {
382.         double perkalian = 0;
383.         for(int i = 0; i < A[row].length; i++)
384.             perkalian +=A[row][i]*B[i][col];
385.         return perkalian;
386.     }
387.     public double[][] Invers (double [][]a) {
388.         // rumus untuk menghitung matriks
389.         // inv(A) = 1/det(A) * adj(A)
390.
391.         if (INFO) {
392.             System.out.println("Mencari Invers...");
393.         }
394.         int tma = a.length;
395.
396.         double m[][] = new double[tma][tma];
397.         double mm[][] = Adjoint(a);
398.
399.         double det = determinant(a);
400.         double dd = 0;
401.
402.         if (det == 0) {
403.             statusBar.setText("Determinan sama dengan 0, tidak bisa diba
lik.");
404.             if (INFO) {
405.                 System.out.println("Determinant sama dengan 0, tidak bis
a dibalik.");
406.             }
407.             } else {
408.                 dd = 1 / det;
409.             }
410.             for (int i = 0; i < tma; i++)
411.                 for (int j = 0; j < tma; j++) {
412.                     m[i][j] = dd * mm[i][j];
413.                 }
414.             return m;
415.         }

```

```

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.                     }

```



```

477.         }
478.         if (m[kol][kol] != 0) {
479.             if (CEK) {
480.                 System.out.println("Ukuran Matriks = " + tma + " kolom
= " + kol + " baris= " + bar);
481.             }
482.             try {
483.                 f1 = (-1) * m[bar][kol] / m[kol][kol];
484.                 for (int i = kol; i < tma; i++) {
485.                     m[bar][i] = f1 * m[kol][i] + m[bar][i];
486.                 } catch (Exception e) {
487.                     System.out.println("Maaf Masih Sampai Disini");
488.                 }
489.             }
490.         }
491.     }
492.     return m;
493. }
494.
495.     public double determinant(double[][] matrix) {
496.         if (INFO) {
497.             System.out.println("Mencari Determinan...");
498.         }
499.         int tma = matrix.length;
500.         double det = 1;
501.         matrix = SegitigaAtas(matrix);
502.         for (int i = 0; i < tma; i++) {
503.             det = det * matrix[i][i];
504.         } // Mengalikan diagonal bawah
505.
506.         det = det * iDF;
507.         if (INFO) {
508.             System.out.println("Determinant: " + det);
509.         }
510.         return det;
511.     }
512.
513.     public double[][] Transpose(double[][] a) {
514.         if (INFO) {
515.             System.out.println("Mencari Transpose...");
516.         }
517.         double m[][] = new double[a[0].length][a.length];
518.         for (int i = 0; i < a.length; i++)
519.             for (int j = 0; j < a[i].length; j++)
520.                 m[j][i] = a[i][j];
521.         return m;
522.     }
523. }

```