1. **import** java.awt.BorderLayout;
2. **import** java.awt.Component;
3. **import** java.awt.Dimension;
4. **import** java.awt.GridLayout;
5. **import** java.awt.event.ActionEvent;
6. **import** java.awt.event.ActionListener;
7. **import** java.awt.event.WindowAdapter;
8. **import** java.awt.event.WindowEvent;
9. **import** java.text.NumberFormat;
10. **import** java.util.ArrayList;
11. **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;
23. **public** **class** KalkulatorMatriks {
24. **private** **boolean** CEK = **true**;
25. **private** **boolean** INFO = **true**;
26. **private** **static** **int** max = 100;
27. **private** **static** **int** decimals =3;
28. **private** **int** iDF = 0;
29. **private** JTextArea taA, taB, taC;
30. **private** JLabel statusBar;
31. **private** **int** n = 4;
32. **private** **static** NumberFormat nf;
34. **public** Component buatTampilan(){
35. //TEXT AREA UNTUK MASUKAN ANGKA
36. taA = **new** JTextArea();
37. taB = **new** JTextArea();
38. taC = **new** JTextArea();
40. //membuat panel
41. JPanel panel = **new** JPanel();
42. panel.setLayout(**new** BoxLayout(panel, BoxLayout.Y\_AXIS));
43. panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
44. panel.add(MatrixPane("Matrix A", taA));
45. panel.add(Box.createRigidArea(**new** Dimension(10, 0)));
46. panel.add(MatrixPane("Matrix B", taB));
47. panel.add(Box.createRigidArea(**new** Dimension(10, 0)));
48. panel.add(MatrixPane("Hasil", taC));;

51. //Operasi 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. paneBtn.add(btnApB);
65. paneBtn.add(btnAmB);
66. paneBtn.add(btnAkB);
67. paneBtn.add(btnInvA);
68. paneBtn.add(btnInvB);
69. paneBtn.add(btnTrnsA);
70. paneBtn.add(btnTrnsB);
71. paneBtn.add(btnDetA);
72. paneBtn.add(btnDetB);

75. /\* == Menambahkan BUTTON Listeners untuk memanggil suatu fungsi atau method == \*/
76. btnApB.addActionListener(**new** ActionListener() {
77. **public** **void** actionPerformed(ActionEvent evt) {
78. **try** {
79. MenampilkanMatrix(Plus(BacaMatrix(taA),
80. BacaMatrix(taB)), taC);
81. } **catch** (Exception e) {
82. System.err.println("Error: " + e);
83. }
84. }
85. });
87. btnAmB.addActionListener(**new** ActionListener() {
88. **public** **void** actionPerformed(ActionEvent evt) {
89. **try** {
90. MenampilkanMatrix(min(BacaMatrix(taA),
91. BacaMatrix(taB)), taC);
92. } **catch** (Exception e) {
93. System.err.println("Error: " + e);
94. }
95. }
96. });
98. btnAkB.addActionListener(**new** ActionListener() {
99. **public** **void** actionPerformed(ActionEvent evt) {
100. **try** {
101. MenampilkanMatrix(kali(BacaMatrix(taA),
102. BacaMatrix(taB)), taC);
103. } **catch** (Exception e) {
104. System.err.println("Error: " + e);
105. }
106. }
107. });
108. btnInvA.addActionListener(**new** ActionListener() {
109. **public** **void** actionPerformed(ActionEvent evt) {
110. **try** {
111. MenampilkanMatrix(Invers(BacaMatrix(taA)),taC);
112. } **catch** (Exception e) {
113. System.err.println("Error: " + e);
114. }
115. }
116. });
117. btnInvB.addActionListener(**new** ActionListener() {
118. **public** **void** actionPerformed(ActionEvent evt) {
119. **try** {
120. MenampilkanMatrix(Invers(BacaMatrix(taB)),taC);
121. } **catch** (Exception e) {
122. System.err.println("Error: " + e);
123. }
124. }
125. });
126. btnTrnsA.addActionListener(**new** ActionListener() {
127. **public** **void** actionPerformed(ActionEvent evt) {
128. **try** {
129. MenampilkanMatrix(Transpose(BacaMatrix(taA)),taC);
130. } **catch** (Exception e) {
131. System.err.println("Error: " + e);
132. }
133. }
134. });
135. btnTrnsB.addActionListener(**new** ActionListener() {
136. **public** **void** actionPerformed(ActionEvent evt) {
137. **try** {
138. MenampilkanMatrix(Transpose(BacaMatrix(taB)),taC);
139. } **catch** (Exception e) {
140. System.err.println("Error: " + e);
141. }
142. }
143. });
144. btnDetA.addActionListener(**new** ActionListener() {
145. **public** **void** actionPerformed(ActionEvent evt) {
146. **try** {
147. taC.setText("Determinant A: " + nf.format(determinant(BacaMatrix(taA))));
148. } **catch** (Exception e) {
149. System.err.println("Error: " + e);
150. }
151. }
152. });
153. btnDetB.addActionListener(**new** ActionListener() {
154. **public** **void** actionPerformed(ActionEvent evt) {
155. **try** {
156. taC.setText("Determinant B: " + nf.format(determinant(BacaMatrix(taB))));
157. } **catch** (Exception e) {
158. System.err.println("Error: " + e);
159. }
160. }
161. });




167. // Main Panel
168. JPanel pane = **new** JPanel();
169. pane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
170. pane.setLayout(**new** BoxLayout(pane, BoxLayout.X\_AXIS));
171. pane.add(panel);
172. pane.add(paneBtn);
174. JPanel fpane = **new** JPanel();
175. fpane.setLayout(**new** BorderLayout());
176. fpane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
177. fpane.add("Center", pane);
178. statusBar = **new** JLabel("Siap");
179. fpane.add("South", statusBar);
181. **return** fpane;
182. }
183. //pengaturan matrix panel
184. **private** JPanel MatrixPane(String str, JTextArea ta) {
185. JScrollPane scrollPane = **new** JScrollPane(ta);
186. **int** size = 200;
188. scrollPane.setPreferredSize(**new** Dimension(size, size));
189. JLabel label = **new** JLabel(str);
190. label.setLabelFor(scrollPane);
192. JPanel pane = **new** JPanel();
193. pane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
194. pane.setLayout(**new** BoxLayout(pane, BoxLayout.Y\_AXIS));
195. pane.add(label);
196. pane.add(scrollPane);
198. **return** pane;
199. }
200. **public** **static** **void** main(String[] args) {
201. JFrame frame= **new** JFrame("Kalkulator Matrix");
202. frame.setSize(**new** Dimension(800, 200));
203. KalkulatorMatriks app = **new** KalkulatorMatriks();
204. Component contents = app.buatTampilan();
205. frame.getContentPane().add(contents, BorderLayout.CENTER);
206. frame.addWindowListener(**new** WindowAdapter() {
207. **public** **void** windowClosing(WindowEvent e) {
208. System.exit(0);
209. }
210. });
211. frame.pack();
212. frame.setVisible(**true**);
214. nf = NumberFormat.getInstance();
215. nf.setMinimumFractionDigits(1);
216. nf.setMaximumFractionDigits(decimals);
218. }
219. //    ----------------------------------------------------
220. //akhir dari tampilan
221. //    ----------------------------------------------------
223. **public** **double**[][] BacaMatrix (JTextArea ta) **throws** Exception{
224. **if**(CEK){
225. System.out.println("Membaca Matriks");
226. }
227. // Menguraikan Text Area
228. String kosong = ta.getText();
229. String pisah  = "";
230. **int** i =0;
231. **int** j =0;
232. **int** [] rsize = **new** **int** [max];
234. // mendefinisikan ukuran matriks agar valid
235. StringTokenizer ts = **new** StringTokenizer(kosong, "\n");
236. **while** (ts.hasMoreTokens()) {
237. StringTokenizer ts2 = **new** StringTokenizer(ts.nextToken());
238. **while** (ts2.hasMoreTokens()) {
239. ts2.nextToken();
240. j++;
241. }
242. rsize[i] = j;
243. i++;
244. j = 0;
245. }
246. statusBar.setText("Ukuran Matriks : " + i + "x" + i);
247. **if** ((CEK) || (INFO)) {
248. System.out.println("Ukuran Matriks : " + i);
249. }
251. **for** (**int** c = 0; c < i; c++) {
252. **if** (CEK) {
253. System.out.println("i=" + i + "  j=" +rsize[c] + "   Kolom : " + c);
254. }
256. **if** (rsize[c] != i) {
257. statusBar.setText("Ukuran Matriks yang Dimasukan Tidak Sama");
258. **throw** **new** Exception("Ukuran Matriks yang Dimasukan Tidak Sama");
259. }
260. }
261. /\* == Mengatur Ukuran Matriks == \*/
262. n = i;
264. **double** matrix[][] = **new** **double**[n][n];
265. i = j = 0;
266. pisah = "";
268. /\* == Melakukan penguraian teks yang sebenarnya sekarang == \*/
269. StringTokenizer st = **new** StringTokenizer(kosong, "\n");
270. **while** (st.hasMoreTokens()) {
271. StringTokenizer st2 = **new** StringTokenizer(st.nextToken());
272. **while** (st2.hasMoreTokens()) {
273. pisah = st2.nextToken();
274. **try** {
275. matrix[i][j] = Double.valueOf(pisah).doubleValue();
276. } **catch** (Exception exception) {
277. statusBar.setText("Angka Tidak Valid ");
278. }
279. j++;
280. }
281. i++;
282. j = 0;
283. }
285. **if** (CEK) {
286. System.out.println("Membaca Matriks:");
287. System.out.println("Ukuran Matriks :" + i);
288. **for** (i = 0; i < n; i++) {
289. **for** (j = 0; j < n; j++) {
290. System.out.print("m[" + i + "][" + j + "] = "
291. + matrix[i][j] + "   ");
292. }
293. System.out.println();
294. }
295. }
296. **return** matrix;
297. }
298. // Menampilkan Matriks di Text Area
299. **public** **void** MenampilkanMatrix(**double** [][] matrix, JTextArea ta){
300. **if**( CEK){
301. }
302. String rstr = "";
303. String dv = "";
305. **for** (**int** i = 0; i < matrix.length; i++) {
306. **for** (**int** j = 0; j < matrix[i].length; j++) {
307. dv = nf.format(matrix[i][j]);
308. rstr = rstr.concat(dv + "  ");
309. }
310. rstr = rstr.concat("\n");
311. }
312. ta.setText(rstr);
313. }
314. //Set Penghitungan rumus
315. **public** **double** [][] Plus (**double** [][] a, **double** [][] b) {
316. **int** tmpa = a.length;
317. **int** tmpb = b.length;
318. **if** (tmpa != tmpb) {
319. statusBar.setText("Ukuran Matriks Tidak Sama");
320. }
322. **double** matrix[][] = **new** **double**[tmpa][tmpb];
324. **for** (**int** i = 0; i < tmpb; i++)
325. **for** (**int** j = 0; j < tmpb; j++) {
326. matrix[i][j] = a[i][j] + b[i][j];
327. }
329. **return** matrix;
330. }
331. **public** **double** [][] min (**double** [][]a, **double** [][] b) {
332. **int** tmpa = a.length;
333. **int** tmpb = b.length;
334. **if** (tmpa != tmpb) {
335. statusBar.setText("Ukuran Matriks Tidak Sama");
336. }
338. **double** matrix[][] = **new** **double**[tmpa][tmpb];
340. **for** (**int** i = 0; i < tmpb; i++)
341. **for** (**int** j = 0; j < tmpb; j++) {
342. matrix[i][j] = a[i][j] - b[i][j];
343. }
345. **return** matrix;
346. }
347. **public** **double** [][] kali (**double** [][]a, **double** [][] b) {
348. **int** tmpa = a.length;
349. **int** tmpb = b.length;
350. **if** (tmpa != tmpb) {
351. statusBar.setText("Ukuran Matriks Tidak Sama");
352. }
354. **double** matrix[][] = **new** **double**[tmpa][tmpb];
356. **for** (**int** i = 0; i < a.length; i++)
357. **for** (**int** j = 0; j < b[i].length; j++)
358. matrix[i][j] = 0;
360. **for**(**int** i = 0; i < matrix.length; i++){
361. **for**(**int** j = 0; j < matrix[i].length; j++){
362. matrix[i][j] =bariskolom(a,i,b,j);
363. }
364. }
365. **return** matrix;
366. }
367. **public** **double** bariskolom(**double** [][] A, **int** row, **double** [][] B, **int** col){
368. **double** perkalian = 0;
369. **for**(**int** i = 0; i < A[row].length; i++)
370. perkalian +=A[row][i]\*B[i][col];
371. **return** perkalian;
372. }
374. **public** **double**[][] Invers (**double** [][]a) {
375. // rumus untuk menghitung matriks
376. // inv(A) = 1/det(A) \* adj(A)
378. **if** (INFO) {
379. System.out.println("Mencari Invers...");
380. }
381. **int** tma = a.length;
383. **double** m[][] = **new** **double**[tma][tma];
384. **double** mm[][] = Adjoint(a);
386. **double** det = determinant(a);
387. **double** dd = 0;
389. **if** (det == 0) {
390. statusBar.setText("Determinan sama dengan 0, tidak bisa dibalik.");
391. **if** (INFO) {
392. System.out.println("Determinant sama dengan 0, tidak bisa dibalik.");
393. }
394. } **else** {
395. dd = 1 / det;
396. }
398. **for** (**int** i = 0; i < tma; i++)
399. **for** (**int** j = 0; j < tma; j++) {
400. m[i][j] = dd \* mm[i][j];
401. }
402. **return** m;
403. }
404. **public** **double**[][] Adjoint(**double**[][] a) {
405. **if** (INFO) {
406. System.out.println("Mencari Adjoint...");
407. }
408. **int** tma = a.length;
410. **double** m[][] = **new** **double**[tma][tma];
412. **int** ii, jj, ia, ja;
413. **double** det;
415. **for** (**int** i = 0; i < tma; i++)
416. **for** (**int** j = 0; j < tma; j++) {
417. ia = ja = 0;
419. **double** ap[][] = **new** **double**[tma - 1][tma - 1];
420. **for** (ii = 0; ii < tma; ii++) {
421. **for** (jj = 0; jj < tma; jj++) {
423. **if** ((ii != i) && (jj != j)) {
424. ap[ia][ja] = a[ii][jj];
425. ja++;
426. }
427. }
428. **if** ((ii != i) && (jj != j)) {
429. ia++;
430. }
431. ja = 0;
432. }
433. det = determinant(ap);
434. m[i][j] = (**double**) Math.pow(-1, i + j) \* det;
435. }
437. m = Transpose(m);
438. **return** m;
439. }
440. **public** **double**[][] SegitigaAtas(**double**[][] m) {
441. **if** (INFO) {
442. System.out.println("Mengubah Bentuk Ke segitia Atas...");
443. }
444. **double** f1 = 0;
445. **double** temp = 0;
446. **int** tma = m.length;
447. **int** v = 1;
448. iDF = 1;
449. **for** (**int** kol = 0; kol < tma - 1; kol++) {
450. **for** (**int** bar = kol + 1; bar < tma; bar++) {
451. v = 1;
452. luar: **while** (m[kol][kol] == 0) // cek jika 0 di diagonal
453. {
454. **if** (kol + v >= tma) // cek jika mengganti semua baris
455. {
456. iDF = 0;
457. **break** luar;
458. } **else** {
459. **for** (**int** c = 0; c < tma; c++) {
460. temp = m[kol][c];
461. m[kol][c] = m[kol + v][c]; // switch rows
462. m[kol + v][c] = temp;
463. }
464. v++; // mennghitung baris yang diganti
465. iDF = iDF \* -1; // setiap ganti mengubah determinan
466. }
467. }
468. **if** (m[kol][kol] != 0) {
469. **if** (CEK) {
470. System.out.println("Ukuran Matriks = " + tma + "   kolom = " + kol + "   baris= " + bar);
471. }
473. **try** {
474. f1 = (-1) \* m[bar][kol] / m[kol][kol];
475. **for** (**int** i = kol; i < tma; i++) {
476. m[bar][i] = f1 \* m[kol][i] + m[bar][i];
477. }
478. } **catch** (Exception e) {
479. System.out.println("Maaf Masih Sampai Disini");
480. }
481. }
482. }
483. }
484. **return** m;
485. }
486. **public** **double** determinant(**double**[][] matrix) {
487. **if** (INFO) {
488. System.out.println("Mencari Determinan...");
489. }
490. **int** tma = matrix.length;
492. **double** det = 1;
494. matrix = SegitigaAtas(matrix);
496. **for** (**int** i = 0; i < tma; i++) {
497. det = det \* matrix[i][i];
498. } // Mengalikan diagonal bawah
500. det = det \* iDF;
502. **if** (INFO) {
503. System.out.println("Determinant: " + det);
504. }
505. **return** det;
506. }
508. **public** **double**[][] Transpose(**double**[][] a) {
509. **if** (INFO) {
510. System.out.println("Mencari Transpose...");
511. }
513. **double** m[][] = **new** **double**[a[0].length][a.length];
515. **for** (**int** i = 0; i < a.length; i++)
516. **for** (**int** j = 0; j < a[i].length; j++)
517. m[j][i] = a[i][j];
518. **return** m;
519. }
520. }