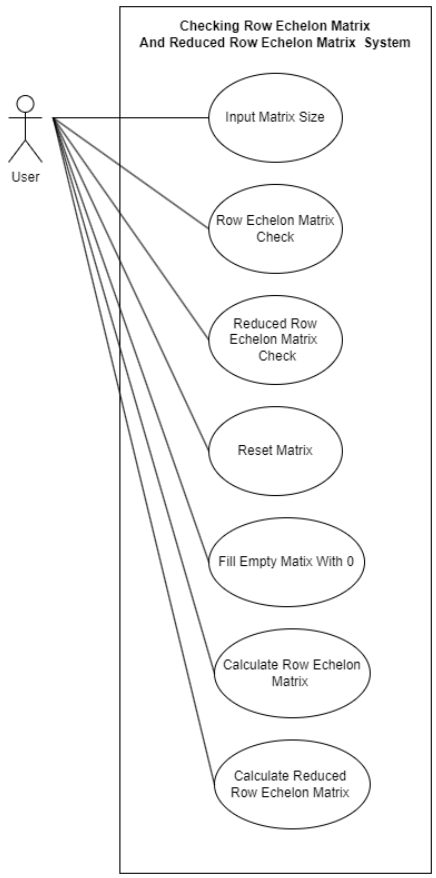
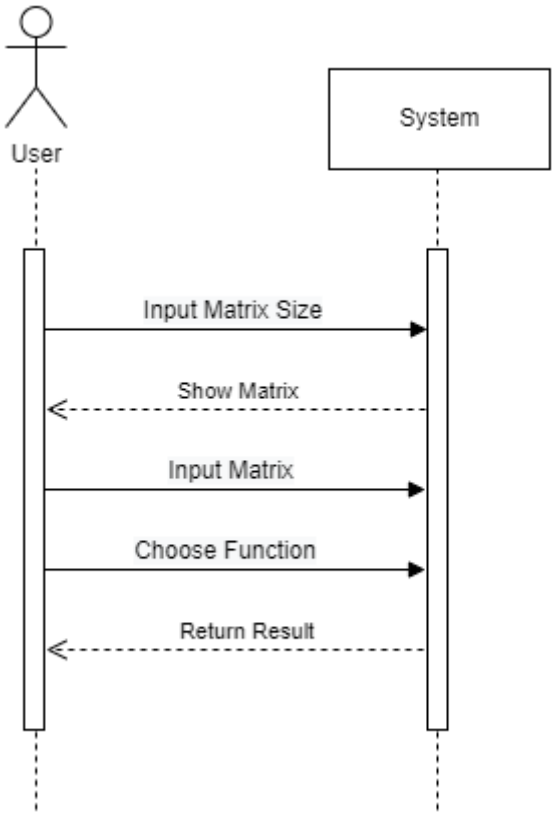
อธิบายแนวคิด/ ไอเดีย พร้อมทั้งวาด Use case ,Sequence และ Class diagram

ตอบ แนวคิด มาจากการเรียนวิชาพีชคณิตเชิงเส้นพื้นฐาน แล้วเกิดปัญหาในการแก้โจทย์ปัญหา การตรวจคำตอบ เกี่ยวกับ Reduced row echelon matrix และ Row echelon matrix ทำให้อยากสร้างโปรแกรมที่ช่วยในการเรียนรู้เกี่ยวกับเรื่อง Reduced row echelon matrix และ Row echelon matrix

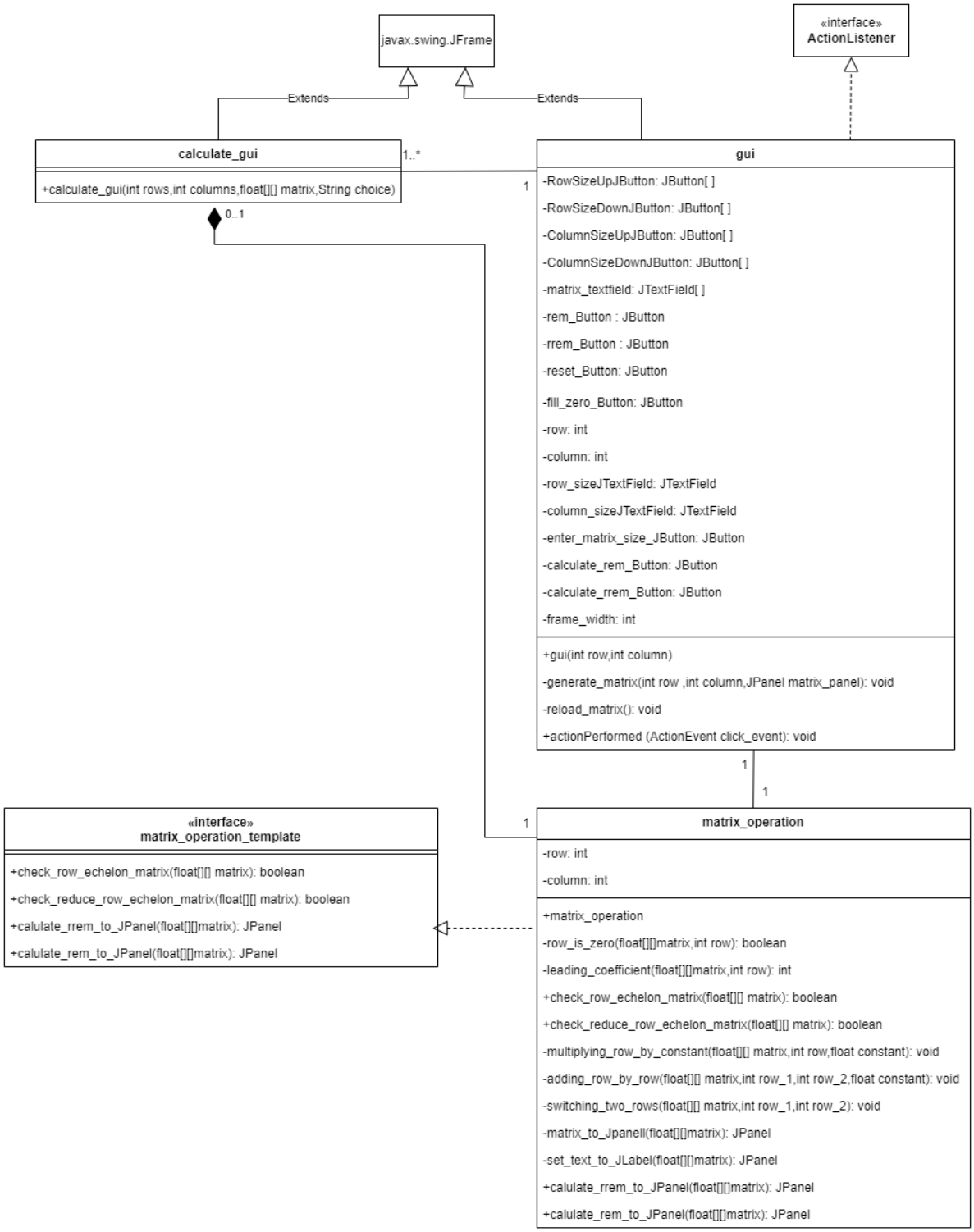
Use case diagram



Sequence diagram



Class diagram



```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class main_final {
6      public static void main(String[] args) {
7          new gui(1, 1);
8      }
9  }
10
11 class gui extends javax.swing.JFrame implements ActionListener { //เพิ่มให้ class extends javax.swing.JFrame
12
13     private int max = 10;
14
15     private JButton RowSizeUpJButton = new JButton();
16     private JButton RowSizeDownJButton = new JButton();
17     private JButton ColumnSizeUpJButton = new JButton();
18     private JButton ColumnSizeDownJButton = new JButton();
19     private JTextField matrix_textfield[][] = new JTextField[max][max];
20     private JButton rem_Button = new JButton();
21     private JButton rrem_Button = new JButton();
22     private JButton reset_Button = new JButton();
23     private JButton fill_zero_Button = new JButton();
24     private int row, column;
25
26     JFrame frame = new JFrame();
27     Container container = frame.getContentPane();
28
29     private JPanel main_panel = new JPanel();
30     private JPanel matrix_panel = new JPanel();
31     private JPanel button_panel = new JPanel();
32     private JPanel top_panel = new JPanel();
33
34     private JPanel matrix_panel_center = new JPanel();
35     private JTextField row_sizeJTextField = new JTextField("", 3);
36     private JTextField column_sizeJTextField = new JTextField("", 3);
37     private JButton enter_matrix_size_JButton = new JButton("Enter Matrix Size");
38
39     private JButton calculate_rem_Button = new JButton();
40     private JButton calculate_rrem_Button = new JButton();
41
42     GridBagConstraints c = new GridBagConstraints();
43
44     private int frame_width;
45
46     gui(int row, int column) {
47         this.row = row;
```

```
48          this.column = column;
49
50          ////////////////////////////////////////////////////////////////////////////////ทำให้ปุ่มใช้ feature ที่มีใน  javax.swing มาใช้
51          RowSizeUpJButton = new javax.swing.JButton();
52          RowSizeDownJButton = new javax.swing.JButton();
53          ColumnSizeUpJButton = new javax.swing.JButton();
54          ColumnSizeDownJButton = new javax.swing.JButton();
55          rem_Button = new javax.swing.JButton("REM check");
56          rrem_Button = new javax.swing.JButton("RREM check");
57          reset_Button = new javax.swing.JButton("reset matrix");
58          fill_zero_Button = new javax.swing.JButton("fill empty matix with 0");
59          calculate_rem_Button = new javax.swing.JButton("calculate REM");
60          calculate_rrem_Button = new javax.swing.JButton("calculate RREM");
61          ////////////////////////////////////////////////////////////////////////// set รูป/ลักษณะปุ่มให้ปุ่ม , กำหนดตัวอักษร
62
63          RowSizeUpJButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("up.png")));
64          RowSizeUpJButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
65          RowSizeDownJButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("down.png")));
66          RowSizeDownJButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
67          ColumnSizeUpJButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("right.png")));
68          ColumnSizeUpJButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
69          ColumnSizeDownJButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("left.png")));
70          ColumnSizeDownJButton.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
71
72          rem_Button.setFont(new java.awt.Font("Tahoma", 1, 13));
73          rrem_Button.setFont(new java.awt.Font("Tahoma", 1, 13));
74          reset_Button.setFont(new java.awt.Font("Tahoma", 1, 13));
75          fill_zero_Button.setFont(new java.awt.Font("Tahoma", 1, 13));
76          calculate_rem_Button.setFont(new java.awt.Font("Tahoma", 1, 13));
77          calculate_rrem_Button.setFont(new java.awt.Font("Tahoma", 1, 13));
78
79          row_sizeJTextField.setHorizontalAlignment(SwingConstants.CENTER);
80          column_sizeJTextField.setHorizontalAlignment(SwingConstants.CENTER);
81
82          ////////////////////////////////////////////////////////////////////////////////////////////////////////
83          container.setLayout(new BorderLayout());
84
85          ////////////////////////////////////////////////////ทำ panel ย่อย
86          main_panel.setLayout(new GridBagLayout());
87
88          matrix_panel.setLayout(new GridBagLayout());
89
90          button_panel.setLayout(new GridBagLayout());
91
92          top_panel.setLayout(new GridBagLayout());
93
94          matrix_panel_center.setLayout(new GridBagLayout());
```

```
95          ////////////////////////////////////////////////////////
96
97          ////////////////////////////////////////////////////////สร้างช่อง matrix
98          for (int i = 0; i < max; i++) {
99              for (int l = 0; l < max; l++) {
100                 matrix_textfield[i][l] = new JTextField("", 3);
101                 matrix_textfield[i][l].setHorizontalAlignment(SwingConstants.CENTER);
102             }
103         }
104         ////////////////////////////////////////////////////////
105
106         ////////////////////////////////////////////////////////จัดองค์ประกอบต่างๆ
107         c.insets = new Insets(5, 5, 5, 5);
108
109         for (int i = 0; i < row; i++) {
110             for (int l = 0; l < column; l++) {
111                 c.gridx = l;
112                 c.gridy = i;
113                 matrix_panel.add(matrix_textfield[i][l], c);
114             }
115         }
116
117         c.gridx = 1;
118         c.gridy = 1;
119         button_panel.add(rem_Button, c);
120         c.gridx = 2;
121         c.gridy = 1;
122         button_panel.add(rrem_Button, c);
123
124         c.gridx = 1;
125         c.gridy = 2;
126         button_panel.add(calculate_rem_Button, c);
127         c.gridx = 2;
128         c.gridy = 2;
129         button_panel.add(calculate_rrem_Button, c);
130
131         c.gridx = 1;
132         c.gridy = 1;
133         matrix_panel_center.add(matrix_panel, c);
134
135         c.gridx = 1;
136         c.gridy = 2;
137         main_panel.add(reset_Button, c);
138         c.gridx = 1;
139         c.gridy = 3;
140         main_panel.add(fill_zero_Button, c);
141
```

```
142         c.gridx = 1;
143         c.gridy = 4;
144         main_panel.add(button_panel, c);
145
146         c.gridx = 2;
147         c.gridy = 1;
148         top_panel.add(RowSizeUpJButton, c);
149         c.gridx = 2;
150         c.gridy = 2;
151         top_panel.add(RowSizeDownJButton, c);
152         c.gridx = 6;
153         c.gridy = 1;
154         top_panel.add(ColumnSizeUpJButton, c);
155         c.gridx = 5;
156         c.gridy = 1;
157         top_panel.add(ColumnSizeDownJButton, c);
158
159         c.gridx = 1;
160         c.gridy = 1;
161         top_panel.add(row_sizeJTextField, c);
162         row_sizeJTextField.setText(row + "");
163
164         c.gridx = 4;
165         c.gridy = 1;
166         top_panel.add(column_sizeJTextField, c);
167         column_sizeJTextField.setText(column + "");
168
169         c.gridx = 3;
170         c.gridy = 1;
171         top_panel.add(enter_matrix_size_JButton, c);
172
173         ////////////////////////////////////////////////////
174         container.add(top_panel, BorderLayout.NORTH);
175         container.add(matrix_panel_center, BorderLayout.CENTER);
176         container.add(main_panel, BorderLayout.SOUTH);
177
178         fill_zero_Button.addActionListener(this);
179         reset_Button.addActionListener(this);
180         rem_Button.addActionListener(this);
181         rrem_Button.addActionListener(this);
182
183         RowSizeUpJButton.addActionListener(this);
184         RowSizeDownJButton.addActionListener(this);
185         ColumnSizeUpJButton.addActionListener(this);
186         ColumnSizeDownJButton.addActionListener(this);
187
188         enter_matrix_size_JButton.addActionListener(this);
```

```
189
190         calculate_rem_Button.addActionListener(this);
191         calculate_rrem_Button.addActionListener(this);
192
193         frame.pack();
194         //frame.setSize(500,550);
195         frame.setVisible(true);
196         //frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
197         frame.setLocationRelativeTo(null);
198         frame_width = frame.getWidth();
199     }
200
201     private void generate_matrix(int row, int column, JPanel matrix_panel) {//เอาไว้สร้าง matrix
202         matrix_panel.setLayout(new GridBagLayout());
203         GridBagConstraints c = new GridBagConstraints();
204         c.insets = new Insets(5, 5, 5, 5);
205         for (int i = row; i < max; i++) {
206             for (int l = 0; l < max; l++) {
207                 matrix_textfield[i][l].setText("");
208             }
209         }
210         for (int i = 0; i < max; i++) {
211             for (int l = column; l < max; l++) {
212                 matrix_textfield[i][l].setText("");
213             }
214         }
215         for (int i = 0; i < row; i++) {
216             for (int l = 0; l < column; l++) {
217                 c.gridx = l;
218                 c.gridy = i;
219                 matrix_panel.add(matrix_textfield[i][l], c);
220             }
221         }
222     }
223
224     private void reload_matrix() {//เอาไว้ reload matrix เมื่อเกิดการเพิ่มขนาด matrix
225         matrix_panel.setLayout(new GridBagLayout());
226         matrix_panel.removeAll();
227         generate_matrix(row, column, matrix_panel);
228         frame.revalidate();
229         row_sizeJTextField.setText(row + "");
230         column_sizeJTextField.setText(column + "");
231         frame.pack();
232
233         Point frame_location = frame.getLocation();
234         frame.setLocation((int) frame_location.getX() - (frame.getWidth() - frame_width) / 2, (int) frame_location.getY());
235
```

```
236         frame_width = frame.getWidth();
237     }
238 
239     @Override
240     public void actionPerformed(ActionEvent click_event) {
241         matrix_operation m_o = new matrix_operation(row, column);
242         try {
243             if (click_event.getSource() == reset_Button) {
244                 for (int i = 0; i < row; i++) {
245                     for (int l = 0; l < column; l++) {
246                         matrix_textfield[i][l].setText("");
247                     }
248                 }
249             } else if (click_event.getSource() == rem_Button) {
250                 float[][] matrix = new float[row][column];
251                 for (int i = 0; i < row; i++) {
252                     for (int l = 0; l < column; l++) {
253                         matrix[i][l] = Float.parseFloat(matrix_textfield[i][l].getText());
254                     }
255                 }
256                 JOptionPane.showMessageDialog(null, "Row Echelon Matrix : " + m_o.check_row_echelon_matrix(matrix));
257             } else if (click_event.getSource() == rrem_Button) {
258                 float[][] matrix = new float[row][column];
259                 for (int i = 0; i < row; i++) {
260                     for (int l = 0; l < column; l++) {
261                         matrix[i][l] = Float.parseFloat(matrix_textfield[i][l].getText());
262                     }
263                 }
264                 JOptionPane.showMessageDialog(null, "Reduce Row Echelon Matrix : " + m_o.check_reduce_row_echelon_matrix(matrix));
265             } else if (click_event.getSource() == fill_zero_Button) {
266                 for (int i = 0; i < row; i++) {
267                     for (int l = 0; l < column; l++) {
268                         if ((matrix_textfield[i][l].getText().isBlank())) {
269                             matrix_textfield[i][l].setText("0");
270                         }
271                     }
272                 }
273             } else if (click_event.getSource() == RowSizeUpJButton) {
274                 if (row < max) {
275                     row++;
276                     reload_matrix();
277                 }
278 
279             } else if (click_event.getSource() == RowSizeDownJButton) {
280                 if (row > 1) {
281                     row--;
282                     reload_matrix();
```

```
283                 }
284
285             } else if (click_event.getSource() == ColumnSizeUpJButton) {
286                 if (column < max) {
287                     column++;
288                     reload_matrix();
289                 }
290
291             } else if (click_event.getSource() == ColumnSizeDownJButton) {
292                 if (column > 1) {
293                     column--;
294                     reload_matrix();
295                 }
296
297             } else if (click_event.getSource() == enter_matrix_size_JButton) {
298                 try {
299                     if (Integer.parseInt(column_sizeJTextField.getText()) >= 1 && Integer.parseInt(column_sizeJTextField.getText()) <= max
300                             && Integer.parseInt(row_sizeJTextField.getText()) >= 1 && Integer.parseInt(row_sizeJTextField.getText()) <= max) {
301                         column = Integer.parseInt(column_sizeJTextField.getText());
302                         row = Integer.parseInt(row_sizeJTextField.getText());
303                         reload_matrix();
304                     } else {
305                         JOptionPane.showMessageDialog(null, "Pls Matrix Size in 1 to " + max);
306                     }
307                 } catch (Exception e) {
308                     JOptionPane.showMessageDialog(null, "Pls Input Only Integer In Matrix Size"/*+e*/);
309                 }
310
311             } else if (click_event.getSource() == calculate_rem_Button) {
312                 float[][] matrix = new float[row][column];
313                 for (int i = 0; i < row; i++) {
314                     for (int l = 0; l < column; l++) {
315                         matrix[i][l] = Float.parseFloat(matrix_textfield[i][l].getText());
316                     }
317                 }
318                 if (m_o.check_row_echelon_matrix(matrix) == true) {
319                     JOptionPane.showMessageDialog(null, "It Already Is Row Echelon Matrix");
320                 } else {
321                     new calculate_gui(row, column, matrix, "rem");
322                 }
323
324             } else if (click_event.getSource() == calculate_rrem_Button) {
325                 float[][] matrix = new float[row][column];
326                 for (int i = 0; i < row; i++) {
327                     for (int l = 0; l < column; l++) {
328                         matrix[i][l] = Float.parseFloat(matrix_textfield[i][l].getText());
329                     }
```

```
330                 }
331                 if (m_o.check_reduce_row_echelon_matrix(matrix) == true) {
332                     JOptionPane.showMessageDialog(null, "It Already Is Reduce Row Echelon Matrix");
333                 } else {
334                     new calculate_gui(row, column, matrix, "rrem");
335                 }
336 
337             }
338 
339         } catch (Exception e) {
340             JOptionPane.showMessageDialog(null, "Pls Input Only Float In Matrix"/*+e*/);
341         }
342 
343     }
344 }
345 
346 class calculate_gui extends javax.swing.JFrame {
347 
348     private JFrame frame = new JFrame();
349     private Container container = frame.getContentPane();
350 
351     calculate_gui(int rows, int columns, float[][] matrix, String choice) {
352         matrix_operation m_o = new matrix_operation(rows, columns);
353 
354         container.setLayout(new BorderLayout());
355         JPanel calculate_panel = new JPanel();
356         JPanel subPanel = new JPanel();
357 
358         subPanel.setLayout(new GridBagLayout());
359 
360         GridBagConstraints c = new GridBagConstraints();
361 
362         JLabel titleLabel = new JLabel();
363 
364         titleLabel.setFont(new java.awt.Font("Tahoma", 1, 13));
365         if (choice.equals("rem")) {
366             titleLabel.setText("Calculate Row Echelon Matrix");
367             c.gridx = 0;
368             c.gridy = 0;
369             subPanel.add(titleLabel, c);
370             c.gridx = 0;
371             c.gridy = 1;
372             subPanel.add(m_o.calulate_rem_to_JPanel(matrix), c);
373             calculate_panel.add(subPanel, BorderLayout.CENTER);
374         } else if (choice.equals("rrem")) {
375             titleLabel.setText("Calculate Reduced Row Echelon Matrix");
376             c.gridx = 0;
```

```
377             c.gridy = 0;
378             subPanel.add(titleLabel, c);
379             c.gridx = 0;
380             c.gridy = 1;
381             subPanel.add(m_o.calulate_rrem_to_JPanel(matrix), c);
382             calculate_panel.add(subPanel, BorderLayout.CENTER);
383         }
384         JScrollPane ScrollPane = new JScrollPane(calculate_panel, JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
    JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
385         container.add(ScrollPane, BorderLayout.CENTER);
386
387         //frame.pack();
388         frame.setSize(500, 550);
389         frame.setVisible(true);
390         //frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
391         frame.setLocationRelativeTo(null);
392
393     }
394 }
395
396 interface matrix_operation_template {
397     boolean check_row_echelon_matrix(float[][] matrix);
398
399     boolean check_reduce_row_echelon_matrix(float[][] matrix);
400
401     JPanel calulate_rrem_to_JPanel(float[][] matrix);
402
403     JPanel calulate_rem_to_JPanel(float[][] matrix);
404 }
405
406 class matrix_operation implements matrix_operation_template {
407     /*
408     rows คือ จำนวนแถว
409     columns คือ จำนวนหลัก
410     ปัญหาคือ เวลาเก็บใน array มันเริ่มที่ 0 แต่ใน matrix มันเริ่มที่ 1 เวลาใช้ method เลยนับว่าให้เริ่มที่ 1 ตาม matrix
411      */
412     private int rows, columns;
413
414     matrix_operation(int rows, int columns) {//กำหนดจำนวนแถวและหลัก
415         this.rows = rows;
416         this.columns = columns;
417     }
418
419     private boolean row_is_zero(float[][] matrix, int row) {//return ว่าแถวเป็น 0 ทั้งแถวหรือเปล่า
420         boolean row_is_zero = true;
421         for (int i = 0; i < columns; i++) {
422             if (matrix[row - 1][i] == 0) {
```

```
423             } else {
424                 row_is_zero = false;
425             }
426         }
427         if (row_is_zero == true) {
428             return true;
429         } else {
430             return false;
431         }
432     }
433 
434     private int leading_coefficient(float[][] matrix, int row) {//return ตำแหน่งของตัวนำ 1 ในแถว ถ้าไม่มีจะ return -1
435         for (int i = 0; i < columns; i++) {
436             if (matrix[row - 1][i] == 1) {
437                 return i + 1;
438             } else if (matrix[row - 1][i] == 0) {
439             } else {
440                 return -1;
441             }
442         }
443         return -1;
444     }
445 
446     public boolean check_row_echelon_matrix(float[][] matrix) {//เช็คว่าเป็น row echelon matrix เปล่า
447         if (rows == 1) {
448             if ((row_is_zero(matrix, 1)) || leading_coefficient(matrix, 1) != -1) {
449                 return true;
450             }//ถ้า matrix มี 1 แถวแล้วแถว 1 เป็น 0 ทั้งแถวหรือมีตัวนำ 1
451             else {
452                 return false;
453             }
454         }
455         for (int i = 2; i <= rows; i++) {
456             if ((leading_coefficient(matrix, i - 1) != -1 || row_is_zero(matrix, i - 1)) && (leading_coefficient(matrix, i) != -1 || row_is_zero(matrix,
    i))) {//ถ้า (matrix ตัวก่อนหน้ามีตัวนำ 1 หรือเป็น 0 ทั้งแถว) และ (matrix ตัวปัจจุบันมีตัวนำ 1 หรือเป็น 0 ทั้งแถว)
457                 if (leading_coefficient(matrix, i - 1) != -1 && leading_coefficient(matrix, i) != -1) {
458                     if (leading_coefficient(matrix, i - 1) < leading_coefficient(matrix, i)) {
459                     } else {
460                         return false;
461                     }
462                 } else if (leading_coefficient(matrix, i - 1) != -1 && row_is_zero(matrix, i)) {
463                 }//matrix ตัวก่อนหน้าต้องมี ตัวนำ 1 และ ตัวปัจจุบันต้องเป็น 0 ทั้งแถว
464                 else if (row_is_zero(matrix, i - 1) && row_is_zero(matrix, i)) {
465                 }///matrix ตัวก่อนหน้าต้องเป็น 0 ทั้งแถวและ ตัวปัจจุบันต้องเป็น 0 ทั้งแถว
466                 else {
467                     return false;
468                 }
```

```
            } else {
                return false;
            }
        }
        return true;
    }

    public boolean check_reduce_row_echelon_matrix(float[][] matrix) {//เช็คว่าเป็น reduce row echelon matrix เปล่า
        if (check_row_echelon_matrix(matrix)) {//เช็คว่าเป็น rem เปล่าถ้าไม่เป็นก็ไม่มีทางเป็น rrem ได้
            for (int i = 1; i <= rows; i++) {
                if (leading_coefficient(matrix, i) != -1) {//เลือกม่าเฉพาะแถวที่มีตัวนำ 1
                    for (int l = 1; l <= rows; l++) {
                        if (matrix[l - 1][leading_coefficient(matrix, i) - 1] == 0) {
                        }//ในหลักของตัวนำ 1 ตัวที่ไม่ใช่ตัวนำ 1 ต้องเป็น 0 เท่านั้น ทั้งข้างบนข้างล่าง
                        else if (l == i) {
                        }//กัน for loop แล้วเจอตัวนำ 1 เพราะตัวนำ 1 ไม่ใช่ 0
                        else {
                            return false;
                        }
                    }
                }
            }
            return true;
        }
        return false;
    }

    private void multiplying_row_by_constant(float[][] matrix, int row, float constant) {
        if (constant == 0) {
            return;
        } else {
            for (int i = 0; i < columns; i++) {
                if (matrix[row - 1][i] == 0) {
                } else {
                    matrix[row - 1][i] = matrix[row - 1][i] * constant;
                }
            }
        }
    }

    private void adding_row_by_row(float[][] matrix, int row_1, int row_2, float constant) {//row_1 คือตัวตั้ง row_2 คือตัวบวก
        if (constant == 0) {
            return;
        } else {
            for (int i = 0; i < columns; i++) {
                matrix[row_1 - 1][i] = matrix[row_1 - 1][i] + (matrix[row_2 - 1][i] * constant);
            }
```

```
516         }
517     }
518 
519     private void switching_two_rows(float[][] matrix, int row_1, int row_2) {//สลับแถว
520         for (int i = 0; i < columns; i++) {
521             float tmp = matrix[row_1 - 1][i];
522             matrix[row_1 - 1][i] = matrix[row_2 - 1][i];
523             matrix[row_2 - 1][i] = tmp;
524         }
525     }
526 
527     private JPanel matrix_to_Jpanel(float[][] matrix) {//เปลี่ยน matrix เป็น JPanel
528         JPanel main_panel = new JPanel();
529         JPanel matrix_panel = new JPanel();
530         JLabel[][] matrix_Jlabel = new JLabel[rows][columns];
531         matrix_panel.setLayout(new GridBagLayout());
532         GridBagConstraints c = new GridBagConstraints();
533         for (int i = 0; i < rows; i++) {
534             for (int l = 0; l < columns; l++) {
535                 matrix_Jlabel[i][l] = new JLabel();
536                 matrix_Jlabel[i][l].setText(matrix[i][l] + "    ");
537                 matrix_Jlabel[i][l].setFont(new java.awt.Font("Tahoma", 1, 13));
538                 c.gridx = l;
539                 c.gridy = i;
540                 matrix_panel.add(matrix_Jlabel[i][l], c);
541             }
542         }
543         main_panel.add(matrix_panel);
544         return main_panel;
545     }
546 
547     private JLabel set_text_to_JLabel(String tmp) {
548         JLabel text_Label = new JLabel();
549         text_Label.setText(tmp);
550         text_Label.setFont(new java.awt.Font("Tahoma", 1, 13));
551         return text_Label;
552     }
553 
554     public JPanel calulate_rrem_to_JPanel(float[][] matrix) {//คำนวณ rrem และแปลงเป็น JPanel
555         JPanel panel = new JPanel();
556         panel.setLayout(new GridBagLayout());
557         GridBagConstraints c = new GridBagConstraints();
558         c.gridx = 1;
559         int leading_coefficient_position_in_row = 0;//คือ ค่าของแถวของตัวนำ 1 ที่คาดหวัง ใช้หาว่าตัวนำต้องอยู่ในแถวไหน
560         String tmp = "";
561         int JPanel_sqsequence = 1;
562         for (int i = 0; i < columns; i++) {
```

```
563             for (int l = 0; l < rows; l++) {
564                 if (leading_coefficient_position_in_row != -1) {//ใช้กันไม่ให้เกิดเหตุการณ์ leading_coefficient_position_in_row มีค่ามากกว่าจำนวนแถวสูงสุด
565                     if (matrix[leading_coefficient_position_in_row][i] == 1) {
566                         if (leading_coefficient_position_in_row != l && matrix[l][i] != 0) {//ถ้าในหลักนั้นมีตัวนำ 1 ให้เปลี่ยนตัวที่เหลือในหลักนั้นเป็น 0
567                             tmp = "R" + (l + 1) + "+(" + (-matrix[l][i]) + ")R" + (leading_coefficient_position_in_row + 1) + "\n\n";
568                             c.gridy = JPanel_sqsequence;
569                             adding_row_by_row(matrix, l + 1, leading_coefficient_position_in_row + 1, -matrix[l][i]);
570                             panel.add(matrix_to_Jpanel(matrix), c);
571 
572                             JPanel_sqsequence++;
573                             c.gridy = JPanel_sqsequence;
574                             panel.add(set_text_to_JLabel(tmp), c);
575                             JPanel_sqsequence++;
576                             ////////////////////////////////สร้างเว้นวรรคด้วย JLabel
577                             c.gridy = JPanel_sqsequence;
578                             panel.add(set_text_to_JLabel(" "), c);
579                             JPanel_sqsequence++;
580                             ////////////////////////////////
581                         }
582                     } else if (matrix[l][i] != 0 && leading_coefficient_position_in_row <= l) {//ใช้หาตัวนำ 1 โดยต้องอยู่ในแถวที่น้อยกว่าหรือเท่ากับตำนำ1ที่คาดหวัง
583                         if (l != leading_coefficient_position_in_row) {//อย่างแรกสลับแถวปัจจุบันไปแถวที่ควรมีตัวนำ 1
584                             tmp = "R" + (l + 1) + "↔R" + (leading_coefficient_position_in_row + 1) + "\n\n";
585                             c.gridy = JPanel_sqsequence;
586                             switching_two_rows(matrix, l + 1, leading_coefficient_position_in_row + 1);
587                             panel.add(matrix_to_Jpanel(matrix), c);
588                             JPanel_sqsequence++;
589 
590                             c.gridy = JPanel_sqsequence;
591                             panel.add(set_text_to_JLabel(tmp), c);
592                             JPanel_sqsequence++;
593                             ////////////////////////////////สร้างเว้นวรรคด้วย JLabel
594                             c.gridy = JPanel_sqsequence;
595                             panel.add(set_text_to_JLabel(" "), c);
596                             JPanel_sqsequence++;
597                             ////////////////////////////////
598                         }
599                         if (matrix[leading_coefficient_position_in_row][i] != 1) {//เปลี่ยนช่องที่อยู่ให้กลายเป็น 1 โดยการหารตัวมันเองทั้งแถว
600                             tmp = "R" + (leading_coefficient_position_in_row + 1) + "/" + matrix[leading_coefficient_position_in_row][i] + "\n\n";
601                             c.gridy = JPanel_sqsequence;
602                             multiplying_row_by_constant(matrix, leading_coefficient_position_in_row + 1, 1 / matrix[leading_coefficient_position_in_row][i]);
603                             panel.add(matrix_to_Jpanel(matrix), c);
604                             JPanel_sqsequence++;
605 
606                             c.gridy = JPanel_sqsequence;
```

```
607                             panel.add(set_text_to_JLabel(tmp), c);
608                             JPanel_sqsequence++;
609                             ////////////////////////////////สร้างเว้นวรรคด้วย JLabel
610                             c.gridy = JPanel_sqsequence;
611                             panel.add(set_text_to_JLabel(" "), c);
612                             JPanel_sqsequence++;
613                             ////////////////////////////////
614                         }
615                         l = -1;//ต้องเป็น -1 เพราะว่า อยากให้ l=0 เพื่อที่มันจะได้วน loop แต่พอจบ if มันจะ +1 เลยต้องเป็น -1
616                     }
617                 }
618             }
619             if (leading_coefficient_position_in_row == -1) {
620             }//ใช้กันไม่ให้เกิดเหตุการณ์ leading_coefficient_position_in_row มีค่ามากกว่าจำนวนแถวสูงสุด
621             else if (matrix[leading_coefficient_position_in_row][i] == 1) {
622                 leading_coefficient_position_in_row++;
623                 if (leading_coefficient_position_in_row == rows) {//ถ้า leading_coefficient_position_in_row มีค่าเท่ากับจำนวนแถวสูงสุดแล้วปรับ
    ให้เป็น -1 ซะ เพื่อจะไม่ต้องทำต่อ
624                     leading_coefficient_position_in_row = -1;
625                 }
626             }
627         }
628 
629         for (int i = 0; i < columns; i++) {
630             for (int l = 0; l < rows; l++) {
631                 if (matrix[l][i] == -0) {
632                     matrix[l][i] = 0;
633                 }
634             }
635         }
636         return panel;
637     }
638 
639     public JPanel calulate_rem_to_JPanel(float[][] matrix) {//คำนวณ rem และแปลงเป็น JPanel
640         JPanel panel = new JPanel();
641         panel.setLayout(new GridBagLayout());
642         GridBagConstraints c = new GridBagConstraints();
643         c.gridx = 1;
644         int leading_coefficient_position_in_row = 0;//คือ ค่าของแถวของตัวนำ 1 ที่คาดหวัง ใช้หาว่าตัวนำต้องอยู่ในแถวไหน
645         String tmp = "";
646         int JPanel_sqsequence = 1;
647         for (int i = 0; i < columns; i++) {
648             for (int l = 0; l < rows; l++) {
649                 if (leading_coefficient_position_in_row != -1) {//ใช้กันไม่ให้เกิดเหตุการณ์ leading_coefficient_position_in_row มีค่ามากกว่าจำนวน
    แถวสูงสุด
650                     if (matrix[leading_coefficient_position_in_row][i] == 1) {
651 
```

```
652                         if (leading_coefficient_position_in_row != l && matrix[l][i] != 0 && leading_coefficient_position_in_row < l) {//ถ้าในหลักนั้นมีตัวนำ 1 ให้เปลี่ยนตัวที่เหลือในหลักนั้นเป็น 0
653                             tmp = "R" + (l + 1) + "+(" + (-matrix[l][i]) + ")R" + (leading_coefficient_position_in_row + 1) + "\n\n";
654
655                             c.gridy = JPanel_sqsequence;
656                             adding_row_by_row(matrix, l + 1, leading_coefficient_position_in_row + 1, -matrix[l][i]);
657                             panel.add(matrix_to_Jpanel(matrix), c);
658                             JPanel_sqsequence++;
659
660                             c.gridy = JPanel_sqsequence;
661                             panel.add(set_text_to_JLabel(tmp), c);
662                             JPanel_sqsequence++;
663                             ////////////////////////////////สร้างเว้นวรรคด้วย JLabel
664                             c.gridy = JPanel_sqsequence;
665                             panel.add(set_text_to_JLabel(" "), c);
666                             JPanel_sqsequence++;
667                             ////////////////////////////////////
668                         }
669                     } else if (matrix[l][i] != 0 && leading_coefficient_position_in_row <= l) {//ใช้หาตัวนำ 1 โดยต้องอยู่ในแถวที่น้อยกว่าหรือเท่ากับตำนำ1ที่คาดหวัง
670                         if (l != leading_coefficient_position_in_row) {//อย่างแรกสลับแถวปัจจุบันไปแถวที่ควรมีตัวนำ 1
671                             tmp = "R" + (l + 1) + "↔R" + (leading_coefficient_position_in_row + 1) + "\n\n";
672                             c.gridy = JPanel_sqsequence;
673                             switching_two_rows(matrix, l + 1, leading_coefficient_position_in_row + 1);
674                             panel.add(matrix_to_Jpanel(matrix), c);
675                             JPanel_sqsequence++;
676
677                             c.gridy = JPanel_sqsequence;
678                             panel.add(set_text_to_JLabel(tmp), c);
679                             JPanel_sqsequence++;
680                             ////////////////////////////////สร้างเว้นวรรคด้วย JLabel
681                             c.gridy = JPanel_sqsequence;
682                             panel.add(set_text_to_JLabel(" "), c);
683                             JPanel_sqsequence++;
684                             ////////////////////////////////////
685                         }
686                         if (matrix[leading_coefficient_position_in_row][i] != 1) {//เปลี่ยนช่องที่อยู่ให้กลายเป็น 1 โดยการหารตัวมันเองทั้งแถว
687                             tmp = "R" + (leading_coefficient_position_in_row + 1) + "/" + matrix[leading_coefficient_position_in_row][i] + "\n\n";
688                             c.gridy = JPanel_sqsequence;
689                             multiplying_row_by_constant(matrix, leading_coefficient_position_in_row + 1, 1 / matrix[leading_coefficient_position_in_row][i]);
690                             panel.add(matrix_to_Jpanel(matrix), c);
691                             JPanel_sqsequence++;
692
693                             c.gridy = JPanel_sqsequence;
694                             panel.add(set_text_to_JLabel(tmp), c);
695                             JPanel_sqsequence++;
```

```
696                         ////////////////////////////////สร้างเว้นวรรคด้วย JLabel
697                         c.gridy = JPanel_sqsequence;
698                         panel.add(set_text_to_JLabel(" "), c);
699                         JPanel_sqsequence++;
700                         ////////////////////////////////
701                     }
702                     l = -1;//ต้องเป็น -1 เพราะว่า อยากให้ l=0 เพื่อที่มันจะได้วน loop แต่พอจบ if มันจะ +1 เลยต้องเป็น -1
703                 }
704             }
705         }
706         if (leading_coefficient_position_in_row == -1) {
707         }//ใช้กันไม่ให้เกิดเหตุการณ์ leading_coefficient_position_in_row มีค่ามากกว่าจำนวนแถวสูงสุด
708         else if (matrix[leading_coefficient_position_in_row][i] == 1) {
709             leading_coefficient_position_in_row++;
710             if (leading_coefficient_position_in_row == rows) {//ถ้า leading_coefficient_position_in_row มีค่าเท่ากับจำนวนแถวสูงสุดแล้วปรับ
    ให้เป็น -1 ซะ เพื่อจะไม่ต้องทำต่อ
711                 leading_coefficient_position_in_row = -1;
712             }
713         }
714     }
715
716     for (int i = 0; i < columns; i++) {
717         for (int l = 0; l < rows; l++) {
718             if (matrix[l][i] == -0) {
719                 matrix[l][i] = 0;
720             }
721         }
722     }
723     return panel;
724 }
725 }
```