

**LAPORAN PROJECT**  
**PEMROGRAMAN BERORIENTASI OBJEK**



**Dosen Pembimbing :**  
**SLAMET TRIYANTO, S. ST**

Disusun Oleh :

**MUHAMMAD FAUZAN**

**NIM : 202013012**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**POLITEKNIK KAMPAR**

**2021**

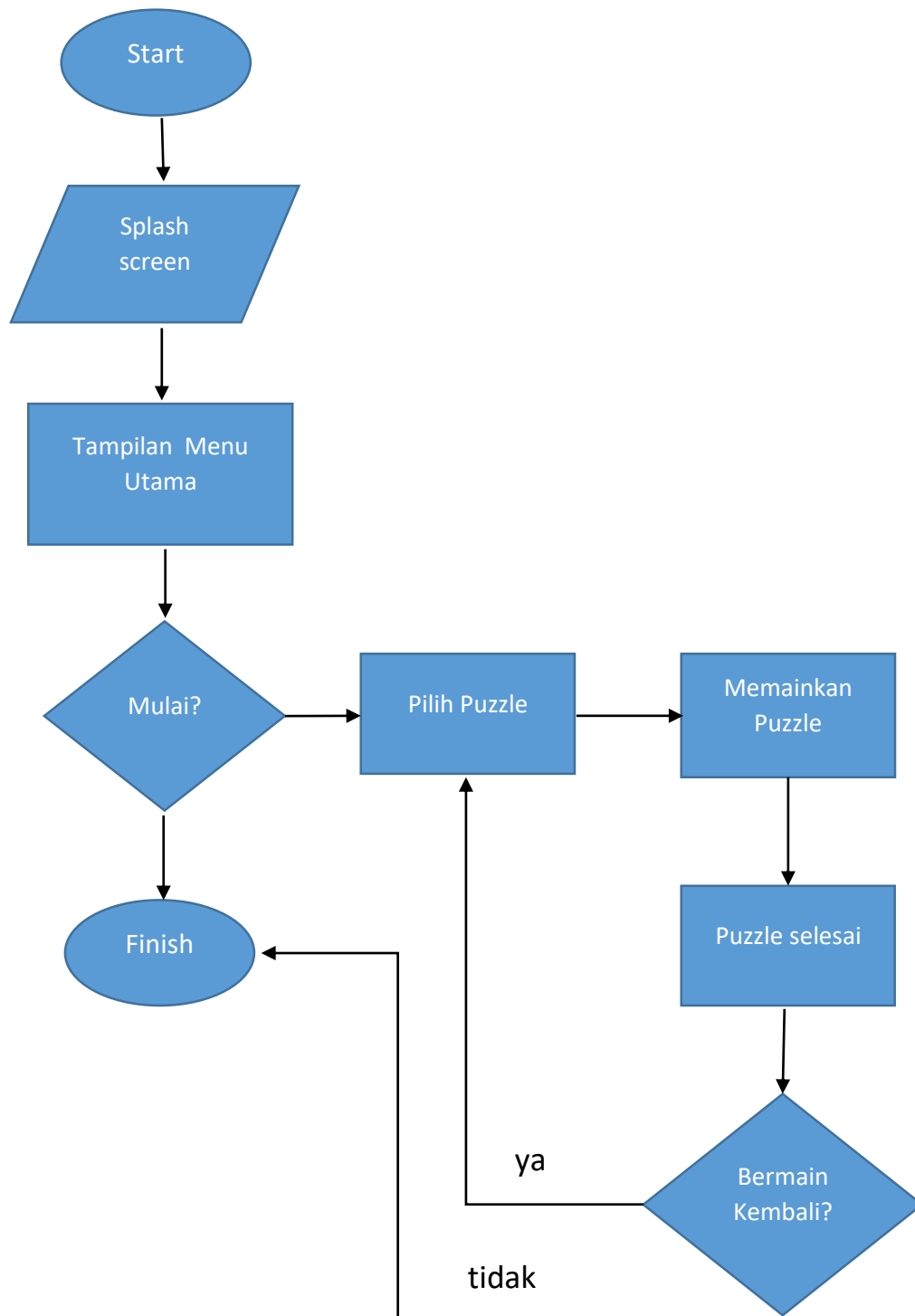
## **Perancangan Aplikasi Game Puzzle Tokoh Sejarah**

1. Tujuan dan manfaat game
  - a. Mempelajari dan mengenal lebih dalam tentang pembuatan game puzzle
  - b. Mengetahui lebih dalam tentang bahasa pemrograman yang di pakai dalam pembuatan game puzzle
  - c. Melatih otak dalam memecahkan suatu masalah
  - d. Mempertajam daya ingat
  - e. Mampu berfikir dan bertindak lebih cepat dalam mengambil suatu keputusan

### **2. Gambaran Aplikasi**

Game puzzle merupakan permainan menyusun potongan gambar dengan aturan sebuah potongan potongan dapat dipindahkan dengan menggesernya ke ruang kosong. Umumnya orang yang memainkan puzzle butuh waktu lama dalam menyelesaikan permainan ini. Hal ini disebabkan karena pada permainan ini tidak ada informasi tambahan yang dimiliki untuk membantu melakukan pencarian solusi, sehingga saat proses penyusunan potongan-potongan menjadi suatu tantangan bagi kita yang memainkannya.

### 3. Flowchart



## 4. Source Code

### a. Source Code Tampilan Utama

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package project.neatbeans;
7
8  import javax.swing.JOptionPane;
9
10 /**
11  *
12  * @author HP
13  */
14 public class MENU_UTAMA extends javax.swing.JFrame {
15
16     // private Object PilihanGambar;
17
18     /**
19      * Creates new form MENU_UTAMA
20      */
21     public MENU_UTAMA() {
22         initComponents();
23     }
24
25     /**
26      * This method is called from within the constructor to initialize the form.
27      * WARNING: Do NOT modify this code. The content of this method is always
28      * regenerated by the Form Editor.
29      */
30     @SuppressWarnings("unchecked")
31     Generated Code
107
108 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
109     System.exit(0);
110 }
111
112 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
113     new PilihGambar().show();
114     this.dispose();
115 }
116
117 /**
118  * @param args the command line arguments
119  */
120 public static void main(String args[]) {
121     /* Set the Nimbus look and feel */
122     Look and feel setting code (optional)
123
124     /* Create and display the form */
125     java.awt.EventQueue.invokeLater(() -> {
126         new MENU_UTAMA().setVisible(true);
127     });
128 }
129
130 // Variables declaration - do not modify
131 private javax.swing.JButton jButton1;
132 private javax.swing.JButton jButton2;
133 private javax.swing.JLabel jLabel1;
134 private javax.swing.JPanel jPanel1;
135 // End of variables declaration
136
137 }
138
```

## Gambar Tampilan Utama



### b. Source Code Tampilan Pilihan Gambar

```
1  /*  
2  * To change this license header, choose License Headers in Project Properties.  
3  * To change this template file, choose Tools | Templates  
4  * and open the template in the editor.  
5  */  
6  package project.neatbeans;  
7  
8  /**  
9   *  
10  * @author HP  
11  */  
12  public class PilihGambar extends javax.swing.JFrame {  
13  
14      /**  
15       * Creates new form PilihGambar  
16       */  
17      public PilihGambar() {  
18          initComponents();  
19      }  
20  
21      /**  
22       * This method is called from within the constructor to initialize the form.  
23       * WARNING: Do NOT modify this code. The content of this method is always  
24       * regenerated by the Form Editor.  
25       */  
26      @SuppressWarnings("unchecked")  
27      // <editor-fold defaultstate="collapsed" desc="Generated Code">  
28      private void initComponents() {  
29  
30          jLabel1 = new javax.swing.JLabel();  
31          jButton1 = new javax.swing.JButton();  
32          jButton2 = new javax.swing.JButton();  
33          jButton3 = new javax.swing.JButton();  
34      }  
35  }
```

```

160 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
161     new Soekarno().show();
162     this.dispose();
163 }
164
165 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
166     new Hatta().show();
167     this.dispose();
168 }
169
170 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
171     new AgusSalim().show();
172     this.dispose();
173 }
174
175 private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
176     new Soepomo().show();
177     this.dispose();
178 }
179
180 private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
181     new Yamin().show();
182     this.dispose();
183 }
184
185 private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
186     new Kartini().show();
187     this.dispose();
188 }
189
190 private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
191     new KHDewantara().show();
192 }
193
194 private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
195     new KHDewantara().show();
196     this.dispose();
197 }
198
199 private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
200     new Sudirman().show();
201     this.dispose();
202 }
203
204 /**
205  * @param args the command line arguments
206  */
207 public static void main(String args[]) {
208     /* Set the Nimbus look and feel */
209     //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
210     /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
211      * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
212      */
213     try {
214         for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
215             if ("Nimbus".equals(info.getName())) {
216                 javax.swing.UIManager.setLookAndFeel(info.getClassName());
217                 break;
218             }
219         }
220     } catch (ClassNotFoundException ex) {
221         java.util.logging.Logger.getLogger(PilihGambar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
222     } catch (InstantiationException ex) {
223         java.util.logging.Logger.getLogger(PilihGambar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
224     } catch (IllegalAccessException ex) {
225         java.util.logging.Logger.getLogger(PilihGambar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
226     } catch (IOException ex) {
227         java.util.logging.Logger.getLogger(PilihGambar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
228     } catch (ClassNotFoundException ex) {
229         java.util.logging.Logger.getLogger(PilihGambar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
230     } catch (InstantiationException ex) {
231         java.util.logging.Logger.getLogger(PilihGambar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
232     } catch (IllegalAccessException ex) {
233         java.util.logging.Logger.getLogger(PilihGambar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
234     } catch (IOException ex) {
235         java.util.logging.Logger.getLogger(PilihGambar.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
236     }
237 }
238
239 /* Create and display the form */
240 java.awt.EventQueue.invokeLater(new Runnable() {
241     public void run() {
242         new PilihGambar().setVisible(true);
243     }
244 });
245
246 // Variables declaration - do not modify
247 private javax.swing.JButton jButton1;
248 private javax.swing.JButton jButton2;
249 private javax.swing.JButton jButton3;
250 private javax.swing.JButton jButton4;
251 private javax.swing.JButton jButton5;
252 private javax.swing.JButton jButton6;
253 private javax.swing.JButton jButton7;
254 private javax.swing.JButton jButton8;
255 private javax.swing.JLabel jLabel1;
256 // End of variables declaration

```

## Gambar Tampilan Pilihan Gambar



### c. Source Code Game Puzzle

```
1 package project.neatbeans;
2
3
4 import java.awt.BorderLayout;
5 import java.awt.Color;
6 import java.awt.EventQueue;
7 import java.awt.Graphics2D;
8 import java.awt.GridLayout;
9 import java.awt.Image;
10 import java.awt.Point;
11 import java.awt.event.ActionEvent;
12 import java.awt.event.MouseAdapter;
13 import java.awt.event.MouseEvent;
14 import java.awt.image.BufferedImage;
15 import java.awt.image.CropImageFilter;
16 import java.awt.image.FilteredImageSource;
17 import java.io.File;
18 import java.io.IOException;
19 import java.util.ArrayList;
20 import java.util.Collections;
21 import java.util.List;
22 import java.util.logging.Level;
23 import java.util.logging.Logger;
24 import javax.imageio.ImageIO;
25 import javax.swing.AbstractAction;
26 import javax.swing.BorderFactory;
27 import javax.swing.ImageIcon;
28 import javax.swing.JButton;
29 import javax.swing.JComponent;
30 import javax.swing.JFrame;
31 import javax.swing.JOptionPane;
32 import javax.swing.JPanel;
```

```

31 import javax.swing.JOptionPane;
32 import javax.swing.JPanel;
33
34 class MyButton extends JButton {
35
36     private boolean isLastButton;
37
38     public MyButton() {
39
40         super();
41
42         initUI();
43     }
44
45     public MyButton(Image image) {
46
47         super(new ImageIcon(image));
48
49         initUI();
50     }
51
52     private void initUI() {
53
54         isLastButton = false;
55         BorderFactory.createLineBorder(Color.gray);
56
57         addMouseListener(new MouseAdapter() {
58
59             @Override
60             public void mouseEntered(MouseEvent e) {
61                 setBorder(BorderFactory.createLineBorder(Color.yellow));
62             }
63
64             @Override
65             public void mouseExited(MouseEvent e) {
66                 setBorder(BorderFactory.createLineBorder(Color.gray));
67             }
68         });
69     }
70
71     public void setLastButton() {
72
73         isLastButton = true;
74     }
75
76     public boolean isLastButton() {
77
78         return isLastButton;
79     }
80 }
81
82 public class Soekarno extends JFrame {
83
84     private JPanel panel;
85     private BufferedImage source;
86     private BufferedImage resized;
87     private Image image;
88     private MyButton lastButton;
89     private int width, height;
90
91     private List<MyButton> buttons;
92     private List<Point> solution;

```



```

91 private List<MyButton> buttons;
92 private List<Point> solution;
93
94 private final int NUMBER_OF_BUTTONS = 12;
95 private final int DESIRED_WIDTH = 300;
96
97 public Soekarno() {
98     initUI();
99 }
100
101
102 private void initUI() {
103
104     solution = new ArrayList<>();
105
106     solution.add(new Point(0, 0));
107     solution.add(new Point(0, 1));
108     solution.add(new Point(0, 2));
109     solution.add(new Point(1, 0));
110     solution.add(new Point(1, 1));
111     solution.add(new Point(1, 2));
112     solution.add(new Point(2, 0));
113     solution.add(new Point(2, 1));
114     solution.add(new Point(2, 2));
115     solution.add(new Point(3, 0));
116     solution.add(new Point(3, 1));
117     solution.add(new Point(3, 2));
118
119     buttons = new ArrayList<>();
120
121     panel = new JPanel();
122     panel.setBorder(BorderFactory.createLineBorder(Color.gray));
123
124     panel = new JPanel();
125     panel.setBorder(BorderFactory.createLineBorder(Color.gray));
126     panel.setLayout(new GridLayout(4, 3, 0, 0));
127
128     try {
129         source = loadImage();
130         int h = getNewHeight(source.getWidth(), source.getHeight());
131         resized = resizeImage(source, DESIRED_WIDTH, h,
132             BufferedImage.TYPE_INT_ARGB);
133     } catch (IOException ex) {
134         Logger.getLogger(Soekarno.class.getName()).log(
135             Level.SEVERE, null, ex);
136     }
137
138     width = resized.getWidth(null);
139     height = resized.getHeight(null);
140
141     add(panel, BorderLayout.CENTER);
142
143     for (int i = 0; i < 4; i++) {
144         for (int j = 0; j < 3; j++) {
145             image = createImage(new FilteredImageSource(resized.getSource(),
146                 new CropImageFilter(j * width / 3, i * height / 4,
147                     (width / 3), height / 4)));
148             MyButton button = new MyButton(image);
149             button.putClientProperty("position", new Point(i, j));
150
151             if (i == 3 && j == 2) {
152                 if (i == 3 && j == 2) {
153                     lastButton = new MyButton();
154                     lastButton.setBorderPainted(false);
155                     lastButton.setContentAreaFilled(false);
156                     lastButton.setLastButton();
157                     lastButton.putClientProperty("position", new Point(i, j));
158                 } else {
159                     buttons.add(button);
160                 }
161             }
162         }
163     }
164
165     Collections.shuffle(buttons);
166     buttons.add(lastButton);
167
168     for (int i = 0; i < NUMBER_OF_BUTTONS; i++) {
169         MyButton btn = buttons.get(i);
170         panel.add(btn);
171         btn.setBorder(BorderFactory.createLineBorder(Color.gray));
172         btn.addActionListener(new ClickAction());
173     }
174
175     pack();
176     setTitle("Game Puzzle Tokoh Sejarah");
177     setResizable(false);
178     setLocationRelativeTo(null);
179     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
180 }
181
182 private int getNewHeight(int w, int h) {

```

```

181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

private int getNewHeight(int w, int h) {
    double ratio = DESIRED_WIDTH / (double) w;
    int newHeight = (int) (h * ratio);
    return newHeight;
}

private BufferedImage loadImage() throws IOException {
    BufferedImage bimg = ImageIO.read(new File("src/gambar/Soekarno.jpg"));
    return bimg;
}

private BufferedImage resizeImage(BufferedImage originalImage, int width,
    int height, int type) throws IOException {
    BufferedImage resizedImage = new BufferedImage(width, height, type);
    Graphics2D g = resizedImage.createGraphics();
    g.drawImage(originalImage, 0, 0, width, height, null);
    g.dispose();
    return resizedImage;
}

private class ClickAction extends AbstractAction {
    @Override
    public void actionPerformed(ActionEvent e) {
        checkButton(e);
        checkButton(e);
        checkSolution();
    }

    private void checkButton(ActionEvent e) {
        int lidx = 0;
        for (MyButton button : buttons) {
            if (button.isLastButton()) {
                lidx = buttons.indexOf(button);
            }
        }
        JButton button = (JButton) e.getSource();
        int bidx = buttons.indexOf(button);
        if ((bidx - 1 == lidx) || (bidx + 1 == lidx)
            || (bidx - 3 == lidx) || (bidx + 3 == lidx)) {
            Collections.swap(buttons, bidx, lidx);
            updateButtons();
        }
    }

    private void updateButtons() {
        panel.removeAll();
        for (JComponent btn : buttons) {
            panel.add(btn);
        }
        panel.add(btn);
        panel.validate();
    }

    private void checkSolution() {
        List<Point> current = new ArrayList<>();
        for (JComponent btn : buttons) {
            current.add((Point) btn.getClientProperty("position"));
        }
        if (compareList(solution, current)) {
            JOptionPane.showMessageDialog(panel, "Finished",
                "Congratulation", JOptionPane.INFORMATION_MESSAGE);
            new MENU_UTAMA().show();
            this.dispose();
        }
    }

    public static boolean compareList(List ls1, List ls2) {
        return ls1.toString().equals(ls2.toString());
    }

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {

```

```

262     }
263 }
264
265 public static boolean compareList(List ls1, List ls2) {
266     return ls1.toString().equals(ls2.toString());
267 }
268
269
270 public static void main(String[] args) {
271     EventQueue.invokeLater(new Runnable() {
272
273         @Override
274         public void run() {
275             Soekarno puzzle = new Soekarno();
276             puzzle.setVisible(true);
277         }
278     });
279 }
280
281 }
282

```

Gambar Tampilan Game Puzzle

