

```

1  /**
2  * <h1>CatifyGUI.java</h1>
3  * <p>
4  * A GUI that allows one player to play the Catify Game at three different level, with
5  * instruction provided.<br>
6  * Connects with the backend program CatifyGame.java and provides pleasant game playing
7  * experience.
8  *
9  * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
10 * @since 12-08-2017
11 */
12 import javax.swing.*;
13 import java.awt.*;
14
15 public class CatifyGUI {
16
17     /**
18     * Main method, used to test the program.
19     * @param args unused.
20     */
21     public static void main (String[] args) {
22         // creates and shows a frame
23         JFrame frame = new JFrame("CatifyGUI");
24         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25
26         // assigns the frame with a static size
27         frame.setPreferredSize(new Dimension(800,600));
28         frame.setMaximumSize(new Dimension(800,600));
29         frame.setMinimumSize(new Dimension(800,600));
30
31         // creates an instance of CatifyGame
32         CatifyGame catify = new CatifyGame();
33
34         // a cardHolder panel with CardLayout that allows user to switch between different
35         // panels by clicking certain buttons
36         JPanel cardHolder = new JPanel();
37         cardHolder.setLayout(new CardLayout());
38
39         // all panels in this GUI
40         MainPanel mainP = new MainPanel(cardHolder);
41         GamePanel gameP = new GamePanel(cardHolder, catify);
42         SettingPanel settingP = new SettingPanel(cardHolder, catify, gameP);
43         ResultPanel resultP = new ResultPanel(cardHolder, catify, gameP);
44         HelpPanel helpP = new HelpPanel(cardHolder);
45
46         cardHolder.add(mainP, "Main Panel");
47         cardHolder.add(settingP, "Setting Panel");
48         cardHolder.add(gameP, "Game Panel");
49         cardHolder.add(resultP, "Result Panel");
50         cardHolder.add(helpP, "Help Panel");
51
52         frame.getContentPane().add(cardHolder);
53
54         frame.pack();
55         frame.setVisible(true);
56     }
57 }

```

```

1  /**
2   * <h1>MainPanel.java</h1>
3   *
4   * <p>
5   * The Home Screen for the game, contains the name of a game, a very cute picture, and
6   * three buttons: New Game, ? Help,
7   * and Quit.
8   *
9   * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
10  * @since 12-08-2017
11  */
12  import java.awt.*;
13  import java.awt.event.*;
14  import javax.swing.*;
15  import javax.swing.event.*;
16
17  public class MainPanel extends JPanel {
18
19      // instance variables
20      private JLabel l1;
21      private JLabel l2;
22      private JLabel img;
23      private JButton newGame;
24      private JButton help;
25      private JButton quit;
26      private JPanel myParent;
27
28      /**
29       * Constructor
30       * @param parent the parent panel (cardHolder/container) of this panel
31       */
32      public MainPanel (JPanel parent) {
33
34          // the parent panel (the card holder) of this panel
35          myParent = parent;
36
37          l1 = new JLabel ("Catify ");
38          l2 = new JLabel ("The next killer app designed by Lisa Huang, Huihan Li and Tina
39  Zhang");
40          img = new JLabel (new ImageIcon("../cat_pics/pic3.jpg"));
41
42          l1.setAlignmentX(Component.CENTER_ALIGNMENT);
43          l1.setFont(new Font("Chalkduster", Font.PLAIN, 80));
44          this.add(l1);
45
46          this.add(Box.createVerticalStrut(20));
47
48          l2.setAlignmentX(Component.CENTER_ALIGNMENT);
49          l2.setFont(new Font("Arial", Font.PLAIN, 18));
50          this.add(l2);
51
52          this.add(Box.createVerticalStrut(20));
53
54          img.setAlignmentX(Component.CENTER_ALIGNMENT);
55          this.add(img);
56
57          this.add(Box.createVerticalStrut(20));
58
59          // A panel with merely buttons
60          JPanel buttonsPanel = ButtonsPanel();
61          this.add(buttonsPanel);
62
63          this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
64          this.setBorder(BorderFactory.createEmptyBorder(20,50,20,50));
65      }
66
67      /**
68       * Returns a JPanel with three buttons.
69       * @return the result JPanel with three buttons
70       */
71      private JPanel ButtonsPanel(){
72
73          JPanel result = new JPanel();
74

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/MainPanel.java1

```

75     newGame = new JButton ("New Game");
76     help = new JButton ("? Help");
77     quit = new JButton ("Quit");
78
79     newGame.setAlignmentX(Component.CENTER_ALIGNMENT);
80     newGame.setPreferredSize(new Dimension(200, 100));
81     newGame.setFont(new Font("Arial", Font.PLAIN, 24));
82     newGame.addActionListener(new ButtonListener());
83     // add a switcher to the newGame button that guides the user to the setting panel
    before starting the game
84     newGame.addActionListener(new Switcher("Setting Panel", myParent));
85     result.add(newGame);
86
87     help.setAlignmentX(Component.CENTER_ALIGNMENT);
88     help.setPreferredSize(new Dimension(200, 100));
89     help.setFont(new Font("Arial", Font.PLAIN, 24));
90     help.addActionListener(new ButtonListener());
91     // add a switcher to the newGame button that guides the user to the setting panel
    before starting the game
92     help.addActionListener(new Switcher("Help Panel", myParent));
93     result.add(help);
94
95
96     quit.setAlignmentX(Component.CENTER_ALIGNMENT);
97     quit.setPreferredSize(new Dimension(200, 100));
98     quit.setFont(new Font("Arial", Font.PLAIN, 24));
99     quit.addActionListener(new ButtonListener());
100    result.add(quit);
101
102    result.setLayout(new FlowLayout());
103
104    return result;
105
106 }
107
108
109 /**
110  * A private ButtonListener class used as ActionListeners for the buttons in this
    class.
111  */
112    private class ButtonListener implements ActionListener {
113
114        /**
115         * Invoked when an action occurs.
116         * @param e The action event that occurs
117         */
118        public void actionPerformed (ActionEvent e) {
119            if (e.getSource() == quit) {
120                System.out.println("Goodbye!");
121                System.exit(0);
122            }
123            if (e.getSource() == newGame) {
124                System.out.println("Setting up the new game");
125            }
126            if (e.getSource() == help) {
127                System.out.println("Showing instructions of the game");
128            }
129        }
130    }
131 }
132 }

```

```

1  /**
2   * <h1>HelpPanel.java</h1>
3   *
4   * <p>
5   * A panel that shows instructions on playing the game.
6   *
7   * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
8   * @since 12-08-2017
9   */
10
11 import java.awt.*;
12 import java.awt.event.*;
13 import javax.swing.*;
14 import javax.swing.event.*;
15
16 public class HelpPanel extends JPanel {
17
18     // instance variables
19     private JPanel parent;
20     private JTextPane text;
21     private JPanel buttonPanel;
22     private JButton home;
23
24     /**
25     * Constructor
26     * @param parent the parent panel (cardHolder/container) of this panel
27     */
28     public HelpPanel(JPanel parent){
29
30         this.parent = parent;
31         this.setLayout(new BorderLayout());
32         this.setBorder(BorderFactory.createEmptyBorder(20,50,20,50));
33
34         // A panel that contains a Home button that guides user back to the home screen
35         buttonPanel = new JPanel();
36         home = new JButton("Home");
37         home.setPreferredSize(new Dimension(120, 60));
38         home.setFont(new Font("Arial", Font.PLAIN, 18));
39         home.addActionListener(new Switcher("Main Panel", this.parent));
40         buttonPanel.add(home);
41
42         // A JTextPane that contains textual instructions, written in html style
43         text = new JTextPane();
44         text.setContentType("text/html");
45         text.setText("<html><body bgcolor=\"#eeeeee\"><div align=\"center\">
46 ><h1>Instructions</h1></div>" +
47 "<font size=5><p><b>Catify</b> is a memory game where you will see a grid of cat
48 pictures, all faced down, and " +
49 "try to find all the pairs of identical pictures within the least possible clicks.
50 <br><br>" +
51 "The game contains three levels of difficulty: EASY (8-pairs), MEDIUM (18-pairs),
52 and HARD (32-pairs). <br><br>" +
53 "At the start of the game, you have as many seconds as you want to memorize the
54 location of all the pairs. Then click " +
55 "<b>Start</b>, and the pictures will turn around. When you click on two pictures,
56 they must form a pair to stay " +
57 "faced up. Otherwise, the pictures " +
58 "turn around again. <br><br> Your total attempts and valid attempts (those find
59 pairs) are tracked. When all the " +
60 "pairs are found, you win the game!<br><br>If you want to play another game within
61 the same level of difficulty, " +
62 "simply press <b>Restart</b>. The <b>Home</b> button will guide you back to the
63 home screen. Also, please feel " +
64 "free to restart the game or go back to the home screen in the middle of a game " +
65 "if you find it too challenging. <br><br>Enjoy!</p></font></body>");
66
67         this.add(text, BorderLayout.NORTH);
68         this.add(buttonPanel);
69     }
70 }

```

```

1  /**
2   * <h1>SettingPanel.java</h1>
3   *
4   * <p>
5   * Allows user to choose from three levels of difficulty of the game by clicking on three
   different buttons.
6   *
7   * @author Lisa Huang (rhuang2), Huihan Li (MAIN AUTHOR) (hli3) and Tina Zhang (yzhang16)
8   * @since 12-08-2017
9   */
10
11 import java.awt.*;
12 import java.awt.event.*;
13 import javax.swing.*;
14 import javax.swing.event.*;
15
16 public class SettingPanel extends JPanel {
17     //instance variables
18     private JButton easyButton, mediumButton, hardButton;
19     private JLabel info;
20     private JPanel parent;
21     private JPanel gridPanel;
22     private CatifyGame catify;
23     private GamePanel gp;
24
25     /**
26     * Constructor
27     * Needs these parameter to establish connections between this panel and other panels
28     * @param parent the parent panel (cardHolder/container) of this panel
29     * @param game the CatifyGame instance
30     * @param gp the Game Panel in the GUI
31     */
32     public SettingPanel(JPanel parent, CatifyGame game, GamePanel gp){
33         this.gp = gp;
34         this.parent = parent;
35         this.catify = game;
36
37         // gets specifically the grid panel in the game panel
38         gridPanel = this.gp.getGridPanel();
39
40         GridBagLayout gridbag = new GridBagLayout();
41         this.setLayout(gridbag);
42         GridBagConstraints c = new GridBagConstraints();
43
44         info = new JLabel("HOW MANY CATS CAN YOU FIND?");
45         info.setFont(new Font("Algerian", Font.BOLD, 44));
46         info.setHorizontalAlignment(JLabel.CENTER);
47         c.gridwidth = GridBagConstraints.REMAINDER;
48         c.gridheight = 3;
49         gridbag.setConstraints(info, c);
50         this.add(info);
51
52         // a JPanel that contains merely JButtons
53         JPanel buttonPanel = ButtonPanel();
54
55         c.gridwidth = GridBagConstraints.REMAINDER;
56         c.gridheight = GridBagConstraints.REMAINDER;
57         gridbag.setConstraints(buttonPanel, c);
58
59         this.add(ButtonPanel());
60     }
61
62     /**
63     * Returns a JPanel that contains merely JButtons
64     * @return the result panel with buttons
65     */
66     private JPanel ButtonPanel(){
67         JPanel result = new JPanel();
68         result.setPreferredSize(new Dimension(400, 300));
69         result.setMaximumSize(new Dimension(400, 300));
70         result.setMinimumSize(new Dimension(400, 300));
71
72         easyButton = new JButton(" EASY ");
73         easyButton.setFont(new Font("Forte", Font.PLAIN, 24));

```

Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/SettingPanel.java

```

76     easyButton.addActionListener (new ButtonListener());
77     easyButton.addActionListener(new Switcher("Game Panel", this.parent));
78
79     mediumButton = new JButton("MEDIUM");
80     mediumButton.setFont(new Font("Forte", Font.PLAIN, 24));
81     mediumButton.addActionListener (new ButtonListener());
82     mediumButton.addActionListener(new Switcher("Game Panel", this.parent));
83
84     hardButton = new JButton("    HARD    ");
85     hardButton.setFont(new Font("Forte", Font.PLAIN, 24));
86     hardButton.addActionListener (new ButtonListener());
87     hardButton.addActionListener(new Switcher("Game Panel", this.parent));
88
89     result.setLayout(new BoxLayout(result, BoxLayout.Y_AXIS));
90
91     easyButton.setAlignmentX(result.CENTER_ALIGNMENT);
92     mediumButton.setAlignmentX(result.CENTER_ALIGNMENT);
93     hardButton.setAlignmentX(result.CENTER_ALIGNMENT);
94
95     result.add(Box.createRigidArea(new Dimension(0,60)));
96     result.add(easyButton);
97     result.add(Box.createRigidArea(new Dimension(0,50)));
98
99     result.add(mediumButton);
100    result.add(Box.createRigidArea(new Dimension(0,50)));
101
102    result.add(hardButton);
103
104    return result;
105 }
106
107 /**
108  * A private ButtonListener class used as ActionListeners for the buttons in this
class.
109  */
110 private class ButtonListener implements ActionListener {
111
112     /*
113     * Invoked when an action occurs.
114     * @param event The ActionEvent
115     */
116     public void actionPerformed (ActionEvent event) {
117
118         // start the game both in the backend and in the frontend
119         if (event.getSource() == easyButton){
120             catify.setGame(8);
121             gp.setBoard(8);
122             System.out.println("Starting the game with 8 pairs");
123             System.out.println(catify.getBoard());
124         }
125         if (event.getSource() == mediumButton){
126             catify.setGame(18);
127             gp.setBoard(18);
128             System.out.println("Starting the game with 18 pairs");
129             System.out.println(catify.getBoard());
130         }
131         if (event.getSource() == hardButton){
132             catify.setGame(32);
133             gp.setBoard(32);
134             System.out.println("Starting the game with 32 pairs");
135             System.out.println(catify.getBoard());
136         }
137     }
138 }
139 }

```

```

1  /**
2   * GamePanel.java
3   * The JPanel where the catify game takes place
4   * BUG REPORT: If a button whose picture is revealed is clicked on again, the program
   will find a wrong pair.<br>
5   * This bug only exists when playing the game with GUI. Only playing on backend is fine.
   (Details see line 273)
6   *
7   * @author Lisa Huang (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
8   * @since 12-08-2017
9   */
10
11  import java.awt.*;
12  import java.awt.event.*;
13  import javax.swing.*;
14  import javax.swing.event.*;
15  import javax.swing.border.*;
16  import javax.swing.Timer;
17  import java.util.LinkedList;
18
19  public class GamePanel extends JPanel {
20      // instance variables
21      private int size;
22      // of infoPanel
23      private JButton start;
24      private JButton home;
25      private JLabel info;
26      private JLabel attempt;
27      private JLabel pairsFound;
28      // of gridPanel
29      private JPanel gridPanel;
30      private JPanel parent;
31      private CatifyGame catify;
32      private OneLayeredPanel[][] grids;
33      private LinkedList<LinkedList<Integer>> clicks;
34
35
36
37      /*
38       * Constructor
39       * Constructs a JPanel with infoPanel and gridPanel as subpanels
40       * @param JPanel parent - to which panel it belongs
41       * @param CatifyGame game
42       */
43      public GamePanel (JPanel parent, CatifyGame game) {
44          // instantiate variables
45          this.catify = game;
46          this.parent = parent;
47          this.clicks = new LinkedList<LinkedList<Integer>>();
48
49          // set layout
50          this.setLayout(new FlowLayout(0, 25, 0));
51          this.setBorder(BorderFactory.createEmptyBorder(0,25,25,25));
52
53          JPanel infoPanel = infoPanel();
54          gridPanel = gridPanel();
55
56          this.add(gridPanel);
57          this.add(infoPanel);
58      }
59
60
61      /*
62       * Contains start button, home button, and information about attempts made and pairs
   found
63       * @returns JPanel infoPanel
64       */
65      private JPanel infoPanel() {
66          JPanel result = new JPanel();
67          result.setPreferredSize(new Dimension(200,550));
68
69          start = new JButton("Start");
70          start.setAlignmentX(Component.CENTER_ALIGNMENT);
71          start.addActionListener(new ButtonListener());
72
73          home = new JButton("Home");

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/GamePanel.java1

```

74     home.setAlignmentX(Component.CENTER_ALIGNMENT);
75     home.addActionListener(new Switcher("Main Panel", this.parent));
76
77     info = new JLabel("Find all the pairs!");
78     info.setAlignmentX(Component.CENTER_ALIGNMENT);
79
80     attempt = new JLabel("Attempts made: 0");
81     attempt.setAlignmentX(Component.CENTER_ALIGNMENT);
82
83     pairsFound = new JLabel("Pairs found: 0");
84     pairsFound.setAlignmentX(Component.CENTER_ALIGNMENT);
85
86     // set layout
87     result.setLayout(new BoxLayout(result, BoxLayout.Y_AXIS));
88     result.add(Box.createVerticalStrut(100));
89     result.add(start);
90     result.add(Box.createVerticalStrut(20));
91     result.add(home);
92     result.add(Box.createVerticalStrut(20));
93     result.add(info);
94     result.add(Box.createVerticalStrut(20));
95     result.add(attempt);
96     result.add(Box.createVerticalStrut(20));
97     result.add(pairsFound);
98
99     return result;
100 }
101
102 /*
103  * Initializes an empty JPanel at the start of the program
104  * Will add buttons to it when the level is set
105  * @returns JPanel gridPanel
106  */
107 private JPanel gridPanel(){
108     JPanel result = new JPanel();
109     result.setPreferredSize(new Dimension(475,475));
110     return result;
111 }
112
113 /*
114  * Getter method for the board
115  * @returns JPanel gridPanel
116  */
117 public JPanel getGridPanel(){
118     return gridPanel;
119 }
120
121 /*
122  * Set the board according to the level selected
123  * Use GridLayout to hold JLayeredPane's
124  * Clear up the board and reset infoPanel before adding JLayeredPane's
125  * @param int numOfPics - how many pairs of pictures in one game
126  */
127 public void setBoard(int numOfPics){
128     // first clear the panel
129     gridPanel.removeAll();
130
131     // calculate the side of grid matrix
132     size = (int)Math.sqrt(numOfPics*2);
133     gridPanel.setLayout(new GridLayout(size, size));
134
135     // initialize the matrix of OneLayeredPane
136     grids = new OneLayeredPane[size][size];
137
138     // initialize each of the OneLayeredPane's
139     for (int i = 0; i < size; i++) {
140         for (int j = 0; j < size; j++){
141             grids[i][j] = new OneLayeredPane(i,j,size);
142             gridPanel.add(grids[i][j]);
143         }
144     }
145
146     // set informations
147     start.setText("Start");
148     attempt.setText("Attempts made: 0");
149     pairsFound.setText("Pairs found: 0");

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/GamePanel.java2


```

150 }
151
152
153 /*
154  * A JLayeredPane that contains a JLabel with ImageIcon and a JButton
155  */
156 private class OneLayeredPane extends JLayeredPane{
157     // instance variables
158     private int x, y, size;
159     private JLabel image;
160     private JButton button;
161
162     /*
163     * Constructs a OneLayeredPane at the x row y column of the grids with a side of
164     calculated length
165     * @param int x - which row
166     * @param int y - which column
167     * @param int size - will divide the size of the gridPanel to calculate the side of
168     one OneLayeredPane
169     */
170     public OneLayeredPane(int x, int y, int size){
171         super();
172         this.x = x;
173         this.y = y;
174         this.size = size;
175
176         // set the size and place of the OneLayeredPane
177         this.setBounds((475/size)*y,(475/size)*x,475/size,475/size);
178         this.setBorder(new MatteBorder(1, 1, 1, 1, Color.black));
179         this.setVisible(true);
180         this.setOpaque(true);
181         this.setLayout(null);
182
183         // create a JLabel with an ImageIcon
184         ImageIcon rawImage = new ImageIcon(new ImageIcon(catify.getBoard().getElement(x,y).
185         getFileName()).getImage()
186         .getScaledInstance(475/size, 475/size, Image.
187         SCALE_DEFAULT));
188         image = new JLabel(rawImage);
189         image.setOpaque(true);
190         image.setBounds(0, 0, 475/size, 475/size);
191
192         // create a JButton with ActionListener
193         button = new JButton();
194         button.setPreferredSize(new Dimension(475/size, 475/size));
195         button.setBackground(new Color(254, 238, 206));
196         button.setBorder(new MatteBorder(1, 1, 1, 1, new Color(253, 166, 122)));
197         button.setBounds(0, 0, 475/size,475/size);
198         button.setOpaque(true);
199         button.addActionListener(new ButtonListener());
200
201         // add image and button to OneLayeredPane
202         this.add(image, new Integer(2));
203         this.add(button,new Integer(1));
204     }
205
206     /*
207     * @returns the JLabel containing the image
208     */
209     public JLabel getImage(){
210         return image;
211     }
212
213     /*
214     * @returns the JButton
215     */
216     public JButton getButton(){
217         return button;
218     }
219
220     /*
221     * Set the JLabel to the desired layer
222     * @param int layer
223     */
224     public void flip(int layer){
225         this.setLayer(image, new Integer(layer));
226     }

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/GamePanel.java3

```

222     }
223 }
224 }
225
226 /*
227  * Assigns actions to each button
228  * */
229 private class ButtonListener implements ActionListener {
230     /*
231      * Invoked when an action occurs.
232      * @param ActionEvent event The ActionEvent
233      */
234     public void actionPerformed (ActionEvent event) {
235         // remember the which button is clicked on
236         JButton source = (JButton) event.getSource();
237
238         // if Start button is clicked
239         if (source == start) {
240
241             // Hide the picture if the button says "Start"
242             if (start.getText().equals("Start")) {
243                 // flip all JLabels to lower layer of the layout
244                 for (int i = 0; i < size; i++) {
245                     for (int j = 0; j < size; j++){
246                         grids[i][j].flip(0);
247                     }
248                 }
249                 start.setText("Restart");
250             }
251
252             // Reset the board and information if the button says "Restart"
253             else if (start.getText().equals("Restart")) {
254                 catify.setGame(size*size/2);
255                 setBoard(size*size/2);
256             }
257         }
258
259         // if the buttons of gridPanel are clicked
260         else {
261             // get OneLayeredPane in which the clicked button is located
262             OneLayeredPane container = (OneLayeredPane) source.getParent();
263
264             // look for which button is clicked
265             for (int i = 0; i < size; i++){
266                 for (int j = 0; j < size; j++){
267                     if (event.getSource() == grids[i][j].getButton()){
268                         // show the image of this button
269                         grids[i][j].flip(2);
270                         System.out.println("Clicked on Button (" + i + "," + j + ")");
271
272                         /**
273                          * THE FOLLOWING LINES INVOLVE A BUG
274                          * If the user click on an unrevealed picture as well as a revealed picture
275                          (which is really abnormal
276                          * as nobody would pair a new picture up with a revealed picture),
277                          * only the click on the unrevealed picture will be added to the
278                          LinkedList<LinkedList<Integer>> clicks
279                          * because we actually click on the "picture" rather than the "button" of
280                          the revealed picture.
281                          * Therefore, since only one click is tracked, the program will prompt the
282                          user to make a second click.
283                          */
284                         // Store the ID of the Picture of the clicked Button
285                         LinkedList<Integer> oneClick = new LinkedList<Integer>();
286                         oneClick.add(new Integer(i));
287                         oneClick.add(new Integer(j));
288                         clicks.add(oneClick);
289
290                         // Call takeOneClick in CatifyGame
291                         catify.takeOneClick(i,j);
292                         System.out.println("You have clicked " + catify.getClick() + " picture(s)

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/GamePanel.java4

```

293         // check if the user found one pair if they made two clicks
294         if (catify.getClick() == 2){
295             boolean foundOnePair = catify.findOnePair();
296             if (foundOnePair)
297                 pairsFound.setText("Pairs found: " + catify.getPairsFound());
298             else {
299                 // retrieve the information of the clicked button
300                 int x1 = clicks.get(0).get(0);
301                 int y1 = clicks.get(0).get(1);
302                 int x2 = clicks.get(1).get(0);
303                 int y2 = clicks.get(1).get(1);
304
305                 //wait for one second and hide the image
306                 Timer flipTimer = new Timer(1000, new Flipper(grid[x1][y1], grid[x2][y2]));
307                 flipTimer.setRepeats(false);
308                 flipTimer.restart();
309             }
310
311             // prepare for a new round
312             clicks.clear();
313             catify.clearChoices();
314             //increment attempts
315             attempt.setText("Attempts made: " + catify.getAttempts());
316         }
317
318         // wait for 1.5 seconds and turn to the ResultPanel if the game is finished
319         if (catify.isFinish()) {
320             Timer stayTimer = new Timer(1500, new StayAndSwitch());
321             stayTimer.setRepeats(false);
322             stayTimer.restart();
323         }
324     }
325 }
326
327
328
329 /*
330  * Helps hide the images of the two clicked buttons if they are different
331  */
332 private class Flipper implements ActionListener {
333     private OneLayeredPane pane1, pane2;
334     /*
335     * Constructor
336     * @param OneLayeredPane p1
337     * @param OneLayeredPane p2
338     */
339     public Flipper (OneLayeredPane p1, OneLayeredPane p2) {
340         super();
341         pane1 = p1;
342         pane2 = p2;
343     }
344     /*
345     * Flip both pictures
346     * @param ActionEvent event The ActionEvent
347     */
348     public void actionPerformed(ActionEvent e) {
349         pane1.flip(0);
350         pane2.flip(0);
351     }
352 }
353
354
355 /*
356  * Helps switch to ResultPanel after the game has finished
357  */
358 private class StayAndSwitch implements ActionListener {
359     /*
360     * Invoked when the last pair of images is found.
361     * @param ActionEvent event The ActionEvent
362     */
363     public void actionPerformed(ActionEvent e) {
364         CardLayout cards = (CardLayout)parent.getLayout();
365         //show the ResultPanel
366         cards.show(parent, "Result Panel");
367     }
368 }

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/GamePanel.java5

369 }

```

1  /**
2   * <h1>ResultPanel.java</h1>
3   *
4   * <p>
5   * Displays information that congratulates the user on winning the game, and provides the
6   * user with options to play
7   * again, to return to the home screen and to quit the game.
8   *
9   * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (MAIN AUTHOR) (hli3) and Tina
10  * Zhang (yzhang16)
11  * @since 12-08-2017
12  */
13
14  import java.awt.*;
15  import java.awt.event.*;
16  import javax.swing.*;
17  import javax.swing.event.*;
18
19  public class ResultPanel extends JPanel {
20
21      // instance variables
22      private JLabel l1;
23      private JLabel l2;
24      private JButton restart;
25      private JButton home;
26      private JButton quit;
27      private JPanel parent;
28      private CatifyGame catify;
29      private GamePanel gp;
30
31      /**
32       * Constructor
33       * Needs these parameter to establish connections between the ResultPanel and other
34       * panels
35       * @param parent the parent panel (cardHolder/container) of this panel
36       * @param game the CatifyGame instance
37       * @param gp the Game Panel in the GUI
38       */
39      public ResultPanel (JPanel parent, CatifyGame game, GamePanel gp) {
40          this.catify = game;
41          this.parent = parent;
42          this.gp = gp;
43
44          l1 = new JLabel("Congratulations!");
45          l2 = new JLabel("You found all pairs!");
46
47          l1.setAlignmentX(Component.CENTER_ALIGNMENT);
48          l1.setFont(new Font("Chalkduster", Font.PLAIN, 60));
49          this.add(l1);
50
51          this.add(Box.createVerticalStrut(20));
52
53          l2.setAlignmentX(Component.CENTER_ALIGNMENT);
54          l2.setFont(new Font("Chalkduster", Font.PLAIN, 40));
55          this.add(l2);
56
57          this.add(Box.createVerticalStrut(150));
58
59          // a JPanel that contains merely buttons
60          JPanel buttonsPanel = ButtonsPanel();
61
62          this.add(buttonsPanel);
63          this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
64          this.setBorder(BorderFactory.createEmptyBorder(50,50,50,50));
65
66      }
67
68      /**
69       * Creates a JPanel with three buttons
70       * @return the created JPanel with three buttons
71       */
72      private JPanel ButtonsPanel () {
73          JPanel result = new JPanel();
74
75          restart = new JButton("Restart");
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/ResultPanel.java

```

74     // the restart and home buttons need two listeners, respectively, to not only restart
the game but also switch
75     // to a different panel outside of this panel
76     restart.addActionListener(new ButtonListener());
77     restart.addActionListener(new Switcher("Game Panel", this.parent));
78     home = new JButton("Home");
79     home.addActionListener(new Switcher("Main Panel", this.parent));
80     home.addActionListener(new ButtonListener());
81     // the quit button only needs one ButtonListener because it only performs one action
- quit
82     quit = new JButton("Quit");
83     quit.addActionListener(new ButtonListener());
84
85     restart.setAlignmentX(Component.CENTER_ALIGNMENT);
86     restart.setPreferredSize(new Dimension(200, 100));
87     restart.setFont(new Font("Arial", Font.PLAIN, 24));
88     result.add(restart);
89
90     home.setAlignmentX(Component.CENTER_ALIGNMENT);
91     home.setPreferredSize(new Dimension(200, 100));
92     home.setFont(new Font("Arial", Font.PLAIN, 24));
93     result.add(home);
94
95     quit.setAlignmentX(Component.CENTER_ALIGNMENT);
96     quit.setPreferredSize(new Dimension(200, 100));
97     quit.setFont(new Font("Arial", Font.PLAIN, 24));
98     result.add(quit);
99
100    result.setLayout(new FlowLayout());
101
102    return result;
103 }
104
105 /**
106  * A private ButtonListener class used as ActionListeners for the buttons in this
class.
107  */
108 private class ButtonListener implements ActionListener {
109
110     /**
111      * Invoked when an action occurs.
112      * @param e The action event that occurs
113      */
114     public void actionPerformed (ActionEvent e) {
115         if (e.getSource() == quit) {
116             System.out.println("Goodbye!");
117             System.exit(0);
118         }
119         if (e.getSource() == restart) {
120             // restarts the game in the back end
121             catify.setGame(catify.getNumPics());
122             // restarts the game in the front end (the Game Panel)
123             gp.setBoard(catify.getNumPics());
124             System.out.println("Restart a new game");
125         }
126         if (e.getSource() == home) {
127             System.out.println("Going back to main panel");
128         }
129     }
130 }
131 }
132 }

```

```

1  /**
2   * <h1>Switcher.java</h1>
3   *
4   * <p>
5   * An ActionListener that switch between panels in the parent panel (the cardHolder) with
6   * a CardLayout when an event
7   * is invoked.
8   *
9   * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
10  * @since 12-16-2017
11  */
12  import java.awt.*;
13  import java.awt.event.*;
14  import javax.swing.*;
15
16  public class Switcher implements ActionListener {
17
18      // instance variables
19      private JPanel cardHolder;
20      private CardLayout cards;
21      private String card;
22
23      public Switcher (String card, JPanel holder) {
24          this.card = card;
25          cardHolder = holder;
26          cards = (CardLayout)cardHolder.getLayout();
27      }
28
29      public void actionPerformed(ActionEvent e) {
30          // shows a certain card in the cardHolder
31          cards.show(cardHolder, card);
32      }
33  }

```

```

1  /*
2  * Picture.java
3  * Represents a picture object that contains a picture ID and a picture URL
4  *
5  * @author Huihan Li(hli3), Lisa Huang(rhuang2), Tina Zhang (MAIN AUTHOR) (yzhang16)
6  * @since 12/17/2017
7  *
8  */
9
10 public class Picture implements Comparable<Picture>{
11
12     // Instance variables
13     String picFileName;
14     int picID;
15
16     /* Constructor, create a picture object
17     * @param an integer that represents the picture's ID
18     * @param picture's URL
19     */
20     public Picture (int picID, String picFileName){
21         this.picFileName = picFileName;
22         this.picID = picID;
23     }
24
25     /* get method that return picture's ID
26     * @return picture's ID
27     */
28     public int getID(){
29         return picID;
30     }
31
32     /* get method that return picture's URL
33     * @return picture's URL
34     */
35     public String getFileName(){
36         return picFileName;
37     }
38
39     /* Implements comparable interface, compare two picture's ID
40     * @override
41     * @param comparable picture
42     * @return 1 if pic1 ID > pic2 ID;
43     *         0 if pic1 ID = pic2 ID;
44     *        -1 if pic1 ID < pic2 ID;
45     */
46     public int compareTo(Picture Pic2){
47         if(this.getID()>Pic2.getID()){
48             return 1;
49         }
50         else if(this.getID()==Pic2.getID()){
51             return 0;
52         }
53         else{
54             return -1;
55         }
56     }
57
58     //main method to test
59     public static void main (String args[]){
60         Picture testpic = new Picture (3, "../cat_pics/pic3.jpg");
61         System.out.println("Pic N0.3's ID is " + testpic.getID());
62         System.out.println("Pic N0.3's URL is " + testpic.getFileName());
63     }
64 }
65 }

```



```

1  /*
2  * ImageCollection.java
3  * Create a hashtable for all the images
4  *
5  * @author Huihan Li(hli3), Lisa Huang(rhuang2), Tina Zhang (MAIN AUTHOR) (yzhang16)
6  * @since 12/17/2017
7  *
8  */
9  import java.util.*;
10
11 public class ImageCollection{
12
13     //instance variable
14     Hashtable<Integer, Picture> images;
15     final static int MAX_CAPACITY = 32;
16
17     /*Constructor constructs a hashtable*/
18     public ImageCollection(){
19         //create a hash table that has integer as key and picture object as element
20         images = new Hashtable<Integer, Picture>(MAX_CAPACITY*2);
21         //put the pictures into the hashtable, the key is equal to the picture ID
22         for(int i=1;i<=MAX_CAPACITY;i++){
23             Picture pic = new Picture (i,"../cat_pics/pic"+i+".jpg");
24             images.put(i, pic);
25         }
26     }
27
28     /* get method that return a picture according to its picture ID
29     * @param picture ID
30     * @return picture
31     */
32     public Picture getPic(int picID){
33         return images.get(picID);
34     }
35
36     //main method to test
37     public static void main(String args[]){
38         ImageCollection test = new ImageCollection();
39         System.out.println ("Printing out the hashtable in the format of *Picture ID: URL*");
40         for(int i=1;i<=MAX_CAPACITY;i++){
41             Picture pic = test.getPic(i);
42             System.out.println(i+": "+pic.getFileName());
43         }
44     }
45 }

```

```

1  /*
2  * Board.java
3  * Represents a board object that creates a board for the game
4  *
5  * @author Huihan Li(hli3), Lisa Huang(rhuang2), Tina Zhang (MAIN AUTHOR) (yzhang16)
6  * @since 12/17/2017
7  *
8  */
9
10 import java.util.*;
11
12 public class Board{
13
14     // instance variables
15     private final ImageCollection pictureSource =new ImageCollection() ; //create the
hashtable from ImageCollection
16     private Picture[][] gameBoard; //2-D array
17     private int numPics;
18     private int size;
19
20     /*Constructor that creates the game board that puts random pictures into random
positions in the board
21     * @param number of pictures needed for the game (easy-8, medium-18, hard-32)
22     * @return board
23     */
24     public Board(int numPics){
25         //calcuat board's (2-D array) size
26         this.size=(int)Math.sqrt(numPics*2);
27         //create empty board
28         gameBoard=new Picture[size][size];
29         //generate a list of numbers from 0-32 in random order
30         ArrayList<Integer> whichPicList = uniqueRandom(32);
31         //generate a list of numbers in randm order
32         ArrayList<Integer> positionList = uniqueRandom(numPics*2);
33         //randomly select numPics of pictures from the pictureSource and put it in a random
position in the board
34         for(int i=0; i<numPics; i++){
35             //get a non-repeated random picture
36             int picID = whichPicList.get(i).intValue() + 1;
37             Picture pic= pictureSource.getPic(picID);
38             // generate two random number
39             int firstPos = positionList.get(2*i);
40             int secondPos = positionList.get(2*i+1);
41             //convert the two random numbers to two positions in the board
42             int x1,y1,x2,y2;
43             x1 = firstPos/size;
44             // debug: System.out.println("x1:"+ x1);
45             y1 = firstPos%size;
46             // debug: System.out.println("y1:"+ y1);
47             x2 = secondPos/size;
48             // debug: System.out.println("x2:"+ x2);
49             y2 = secondPos%size;
50             // debug: System.out.println("y2:"+ y2);
51             gameBoard[x1][y1] = pic;
52             gameBoard[x2][y2] = pic;
53         }
54     }
55
56     /* helper function to generate a list of number in random order
57     * @param number of pictures
58     * @return an arrayList
59     */
60     private ArrayList<Integer> uniqueRandom(int numPics) {
61         //use ArrayList in order to shuffle in next step
62         ArrayList<Integer> list = new ArrayList<Integer>();
63         for (int i=0; i<numPics; i++) {
64             list.add(new Integer(i));
65         }
66         //generate random order
67         Collections.shuffle(list);
68         return list;
69     }
70
71     /*get method to return the picture stored the position
72     * @param x, y that indicates which position
73     * @return the picture

```

```

74     */
75     public Picture getElement (int x, int y) {
76         return gameBoard[x][y];
77     }
78
79     /*toString method that create a string of the board
80     *@return string representation of the board
81     */
82     public String toString() {
83         String s = "";
84         for (int i = 0; i < size; i++) {
85             for (int j=0; j < size; j++)
86                 s+=gameBoard[i][j].getID() + " ";
87             s+="\n";
88         }
89         return s;
90     }
91
92     //main method to test
93     public static void main (String args[]){
94         Board test = new Board (8);
95         System.out.println("Easy game board (4*4 board, randomly assigned position)");
96         System.out.println("_____");
97         System.out.println(test);
98         System.out.println("\nMedium game board (6*6 board, randomly assigned position)");
99         System.out.println("_____");
100        Board test2 = new Board (18);
101        System.out.println(test2);
102        System.out.println("\nHard game board (8*8 board, randomly assigned position)");
103        System.out.println("_____");
104        Board test3 = new Board (32);
105        System.out.println(test3);
106        System.out.println("_____");
107        System.out.println("The element at (3,5) is "+test3.getElement(3,5).getID());
108    }
109 }

```

```

1  /*
2  * CatifyGame.java
3  * Back-end version of Catify game can be played in this class
4  * Bugs in the GUI version have been fixed in this version
5  *
6  * @author Tina Zhang (MAIN AUTHOR) (yzhang16) , Huihan Li (MAIN AUTHOR) (hli3)
7  * @since 12/17/2017
8  *
9  */
10 import java.util.*;
11
12 public class CatifyGame{
13
14     //Instance variables
15     private Board board;
16     private int[][] found; //a 2-D array to check if the picture has been found
17     private int click, attempts, pairsFound, numPics;
18     private int x1, x2, y1, y2;
19     private LinkedList<Picture> userClicks; //a LinkdList to store the pictures that users
        have clicked in one round
20
21     /* Constructor: create & initialize instance variables */
22     public CatifyGame(){
23         //Instance variables
24         numPics = 0;
25         click = 0;
26         attempts = 0;
27         pairsFound = 0;
28         x1 = x2 = y1 = y2 = -1;
29         userClicks = new LinkedList<Picture>();
30     }
31
32     /* Set the game by creating the board and other instance variables
33     * @param number of pictures
34     */
35     public void setGame(int numPictures){
36         //create instance variables again (for restart function)
37         numPics = numPictures;
38         click = 0;
39         attempts = 0;
40         pairsFound = 0;
41         x1 = x2 = y1 = y2 = -1;
42         userClicks = new LinkedList<Picture>();
43         int size = (int)Math.sqrt(numPics*2);
44         board = new Board(numPics);
45         found = new int[size][size];
46     }
47
48     /*getter method to return number of pictures
49     * @return numPics
50     */
51     public int getNumPics(){
52         return numPics;
53     }
54
55     /*getter method to return attempts
56     * @return attempts
57     */
58     public int getAttempts(){
59         return attempts;
60     }
61
62     /*getter method to return how many pairs the user has found
63     * @return pairFound
64     */
65     public int getPairsFound(){
66         return pairsFound;
67     }
68
69     /*getter method to reuturn how many clicks the user has used
70     * @reutrn click
71     */
72     public int getClick(){
73         return click;
74     }
75

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/CatifyGame.java

```

76  /*getter method to return the LinkedList that store the user clicked pictures
77  *@return userClicks
78  */
79  public LinkedList<Picture> getUserClicks() {
80      return userClicks;
81  }
82
83  /*takeOneClick method that store the selected picture in the queue
84  *it throws ArrayIndexOutOfBoundsException if the user's input is bigger than size
85  *@param x & y that indicates the picture that user selected
86  */
87  public void takeOneClick(int x, int y) throws ArrayIndexOutOfBoundsException{
88      //Do nothing if the user click on the picture that already found or if the first
picture & second are the same
89      if (!(found[x][y]!=0 || (x1 == x && y1 == y))){
90          Picture pic = board.getElement(x,y);
91          //add the selected picture into the LinkedList
92          userClicks.add(pic);
93          click ++;
94          //store the position of the selected picture
95          if(click==1){
96              x1 = x; y1=y;
97          }
98          else if (click ==2){
99              x2=x; y2=y;
100          }
101      }
102  }
103
104
105  /*findOnePair check if the two pictures in the LinkedList are the same
106  *@return T/F if the two pictures are the same
107  */
108  public boolean findOnePair(){
109      boolean isFound = false;
110      if (click == 2){
111          Picture pic1 = userClicks.getFirst();
112          Picture pic2 = userClicks.getLast();
113          //if pic1 and pic2 are the same, change the the number in found[][] from 0 to the
pictures' ID number & update
114          //the instance variables
115          if (pic1.compareTo(pic2) == 0){
116              found[x1][y1] = pic1.getID();
117              found[x2][y2] = pic2.getID();
118              pairsFound++;
119              isFound = true;
120          }
121          //if two pictures are not the same, reset click & update attempts
122          click = 0;
123          attempts++;
124          //debug
125          //System.out.println(pic1.getID());
126          //System.out.println(pic2.getID());
127      }
128      return isFound;
129  }
130
131  /*clearChoices reset clear the LinkedList in order to prepare for next round*/
132  public void clearChoices(){
133      click = 0;
134      x1 = x2 = y1 = y2 = -1;
135      userClicks.clear();
136  }
137
138  /*isFinish checks if the game is finished
139  *@return T/F if the game is finished
140  */
141  public boolean isFinish(){
142      return pairsFound == numPics;
143  }
144
145  /*printBoard prints the gameboard (all pictures' locations)
146  *@return string
147  */
148  public String printBoard(){
149      return board.toString();
150  }

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/CatifyGame.java2

```

150 }
151
152 /*getter method that return the board
153 *@return board
154 */
155 public Board getBoard(){
156     return board;
157 }
158
159 /*toString method represents the string for the found[][] board. if not found, the
position appears 0, if found the
160 *position appears as the picture ID
161 *@return string
162 */
163 public String toString(){
164     String s = "Pictures found in positions:\n";
165     int size = (int)Math.sqrt(numPics*2);
166     for (int i = 0; i < size; i++) {
167         for (int j=0; j < size; j++)
168             s += found[i][j]+" ";
169         s+="\n";
170     }
171     return s;
172 }
173
174
175 /*play method can be invoked to play the game
176 *@param number of pictures
177 */
178 public void play(int i){
179     //create a game
180     this.setGame(i);
181     //show the board for testing purpose
182     System.out.println(this.printBoard());
183     int size = (int)Math.sqrt(numPics*2);
184
185     Scanner scan = new Scanner(System.in);
186     int X1,Y1,X2,Y2;
187
188     while (!this.isFinish()){
189         do{
190             //user type in the x,y coordinators for the first picture she selects
191             System.out.println("Type x for the first picture: ");
192             X1 = scan.nextInt();
193             System.out.println("Type y for the first picture: ");
194             Y1 = scan.nextInt();
195             }while(X1 > size-1 || Y1 > size-1 || found[X1][Y1] != 0); //if the user type in a
number that bigger than the size
196         //or if the user click on a picture that has already been found, then the game will
keep asking the user for
197         //correct inputs
198
199         //if user's input is valid, take one click
200         this.takeOneClick(X1,Y1);
201
202         do{
203             System.out.println("Type x for the second picture: ");
204             X2 = scan.nextInt();
205             System.out.println("Type y for the second picture: ");
206             Y2 = scan.nextInt();
207             }while(X2 > size-1 || Y2 > size-1 || found[X2][Y2] != 0 || (X1 == X2 && Y1 ==
Y2)); //same as the previous step
208         // if the user click on the same picture as last time, it counts as invalid and the
system will keep asking for
209         // her input
210
211         this.takeOneClick(X2,Y2);
212
213         //debug: System.out.println("x1: "+x1+ "y1: "+y1+"x2: "+x2+"y2 "+y2);
214         System.out.println("Picture1 and Picture2 are same? " + this.findOnePair());
215         this.clearChoices();
216         System.out.println("You took " + this.getAttempts() + " attempts.");
217         System.out.println("You found " + this.getPairsFound() + " pairs.");
218         System.out.println("\nBoard right now: \n" + this);
219     }
220 }

```

/Users/air/Desktop/Fall 2017/CS 230/FinalProject/cs-230-final-project/project/CatifyGame.java

```

221
222     System.out.println("Congratulations! You found all pairs!");
223     System.out.println("You took " + this.getAttempts() + " attempts.");
224     System.out.println("You found " + this.getPairsFound() + " pairs.");
225     scan.close();
226
227 }
228
229 //main method to play the game
230 public static void main(String args[]){
231     CatifyGame test= new CatifyGame();
232     test.setGame(18);
233     System.out.println("Created a medium game of 18 pictures");
234     System.out.println("The correct positions of the game is printed in the following
game board");
235     System.out.println(test.printBoard());
236     System.out.println("Your progress right now is printed below (0: not found)");
237     System.out.println(test);
238     System.out.println("get number of pictures (18): "+test.getNumPics());
239     test.takeOneClick(0,0);
240     test.takeOneClick(2,3);
241     System.out.println("Clicked on picture " + test.getBoard().getElement(0,0).getID());
242     System.out.println("Clicked on picture " + test.getBoard().getElement(2,3).getID());
243     System.out.println("Are the two pictures the same? " + test.findOnePair());
244     System.out.println("You made " + test.getAttempts() + " attempts");
245     System.out.println("You found "+ test.getPairsFound() +" paris");
246     test.clearChoices();
247     System.out.println(" Clearchoices. click now is (0)" + test.getClick());
248     System.out.println("Is the game over (False) "+test.isFinish());
249 }
250
251 }

```

```

1  /*
2  * CatifyGameDriver.java
3  * Driver for the Catify game to be played in backend
4  *
5  * @author Tina Zhang (MAIN AUTHOR) (yzhang16)
6  * @since 12/17/2017
7  *
8  */
9
10 public class CatifyGameDriver{
11     public static void main(String args[]){
12         CatifyGame testGame= new CatifyGame();
13         //easy-level game starts
14         testGame.play(8);
15     }
16 }

```