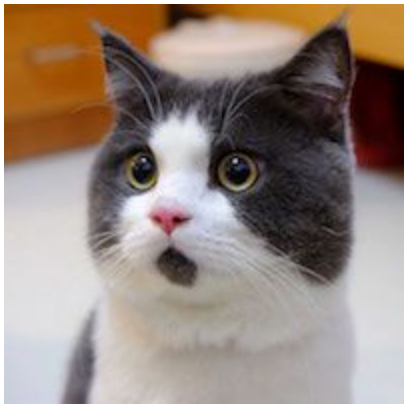
CATIFY

A memory game

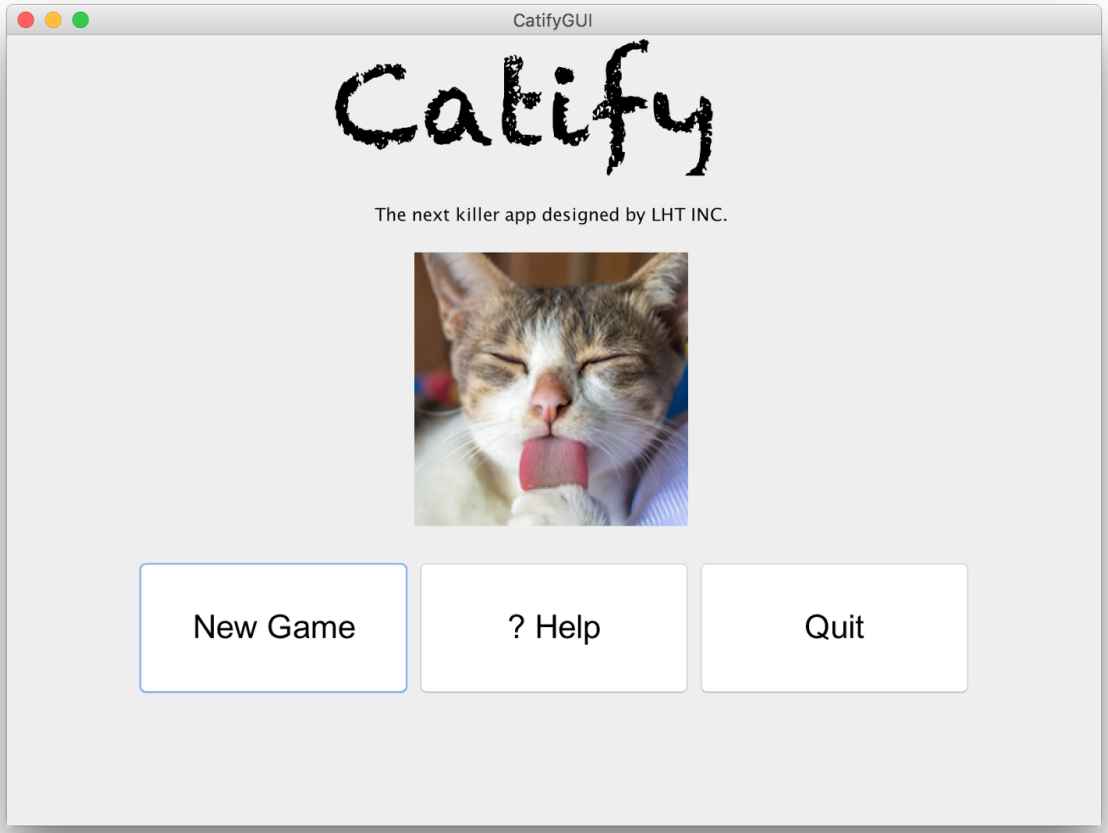




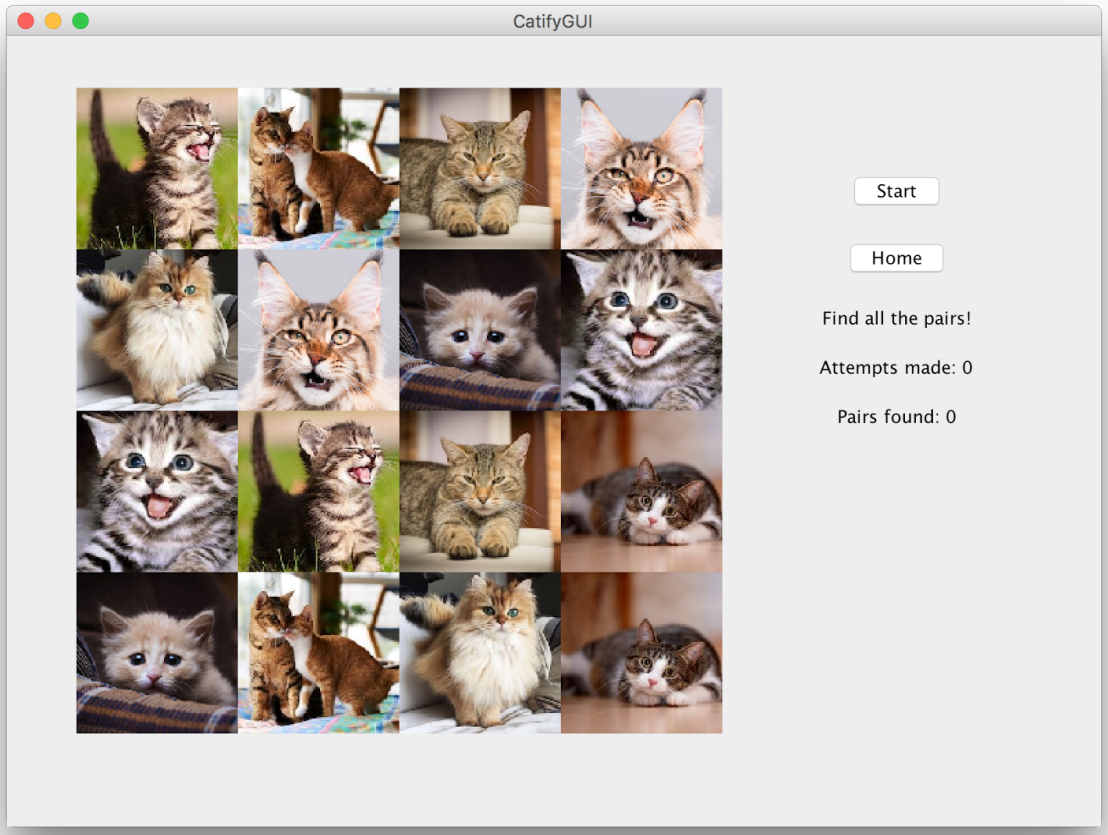
Authors: Lisa Huang, Huihan Li, Tina Zhang
Instructor: P. Takis Metaxas
Fall 2017
CS 230: Data Structures
Wellesley College

I. User Manual

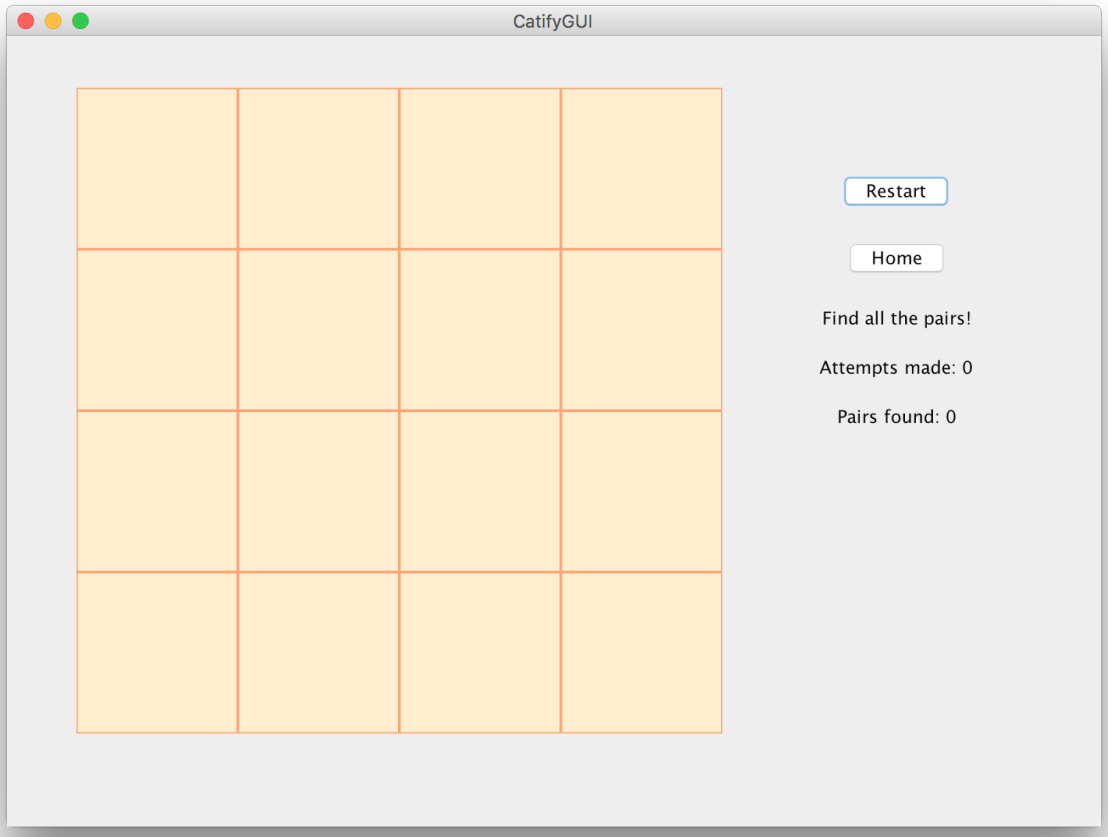
Catify is a memory game where you will see a grid of cat pictures, all faced down, and try to find all the pairs of identical pictures within the least possible clicks.



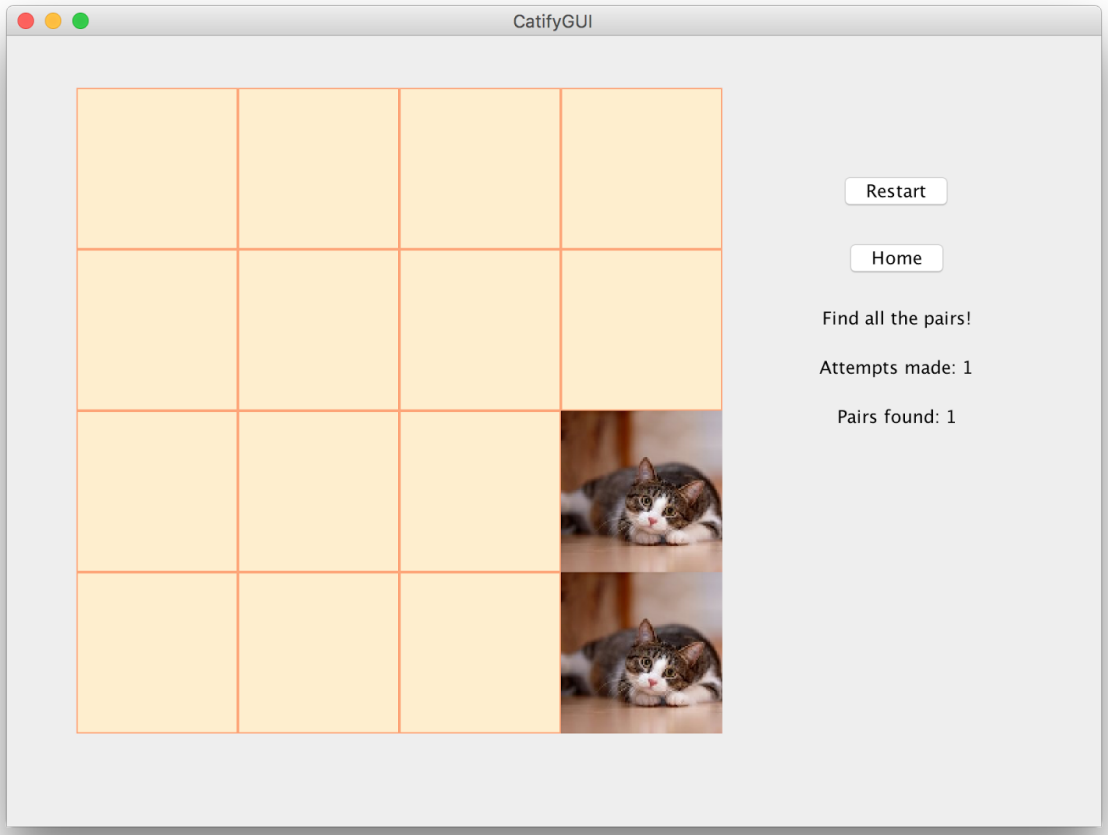
The game contains three levels of difficulty: **EASY** (8-pairs), **MEDIUM** (18-pairs), and **HARD** (32-pairs). At the start of the game, you have as many seconds as you want to memorize the location of all the pairs.



Then click **Start**, and the pictures will turn around.



When you click on two pictures, they must form a pair to stay faced up. Otherwise, the pictures turn around again.



Your total attempts and valid attempts (those find pairs) are tracked. When all the pairs are found, you win the game!

If you want to play another game within the same level of difficulty, simply press **Restart**. The **Home** button will guide you back to the home screen. Also, please feel free to restart the game or go back to the home screen in the middle of a game if you find it too challenging.

Enjoy!

II. Technical Report

ADTs

We used two ADTs in our game. The first one is a **`Hashtable`** which is used in **`ImageCollection`** class and **`CatifyGame`** class. Hashtable helps us store all the pictures used in the game. The second ADT is a **`LinkedList`** that is used in CatifyGame class and GamePanel. The LinkedList helps us to store the pictures that users clicked on in one round.

Classes and Methods

We also used 5 classes in backend and 5 panels in GUIs. The following list show the classes and their public methods.

BackEnd:

Class `Picture`: create a picture object

- `getID();`
- `getFileName();`
- `compareTo(Picture Pic2);`

Class `ImageCollection`: create a hashtable to store all the images

- `getPic(int picID);`

Class `Board`: generate the game board that put pictures in random positions

- `getElement(int x, int y);`
- `toString();`

Class `CatifyGame`: contain all methods needed for CatifyGame

- `setGame(int numPictures);`
- `getAttempts();`
- `getClick();`
- `takeOneClick(int x, int y);`
- `clearChoices();`
- `printBoard();`
- `toString();`
- `getNumPics();`
- `getPairsFound();`
- `getUserClicks();`
- `findOnePair();`
- `isFinish();`
- `getBoard();`
- `play (int i);`

Class `CatifyDriver`: CatifyGame can be played here

- `main() - creates a CatifyGame object and call play()`

GUI:

`CatifyGUI`: Contains a `JPanel` with **`CardLayout`**, which contains all JPanels of GUI
`MainPanel`: The home screen that contains three `JButtons`: New Game, Help, Quit
`SettingPanel`: Set the level of the game

- Contains three `JButtons`: EASY, MEDIUM, HARD
- Switches to `GamePanel` when any of the buttons is clicked on, and set the `GamePanel` according to the level

`GamePanel`: Displays the game board, keeps track of user's attempts and pairs found

- Contains Home button, Start/Restart button and a grid panel in which user actually plays the game
- Uses `JLayeredPane` to layer image and button for every grid in grid panel
- Displays the image when a button in grid panel is clicked on
- Two images stay when they are the same
- A `Timer` is set to 1 second if the two images are different, and the different images disappear when `Timer` is up
- When the last pair of images is found, switch to `ResultPanel` after waiting for 1.5 seconds

`ResultPanel`: Displays information that congratulates the user on winning the game

From this page, the user can:

- start a new game: switches to the `GamePanel` and starts a new game of the same level
- return to home: switches to the `MainPanel`
- Quit: exist the game

Known Bug(s)

So far we have detected one significant bug when playing the game in an abnormal way. This is a bug only in user interface but not in the backend program, and we suggest users follow the rules and **strongly discourage** users from playing the game in a quirky way.

In the backend, not only do we track the two clicks on pictures in an attempt, but we also keep track of all the locations of pictures that are visited in the grid. Therefore, if we want to "click" on a picture that is already revealed, nothing will happen and the program functions normally.

However, in the frontend, we are only able to keep track of the two clicks in an attempt, and thus if the user click on an unrevealed picture A as well as a revealed picture B (which is really abnormal as nobody would pair a new picture up with a revealed picture), only the click on the unrevealed picture B will be added to the **LinkedList<LinkedList<Integer>> clicks** because we actually click on the "picture" rather than the "button" of the revealed picture. Therefore, since only one click is tracked, the program will prompt the user to make a second click. When the second click is made, however, if the second picture revealed C doesn't pair up with picture A, **picture B will disappear instead**. We are still unable to explain why picture B would disappear.

III. Labor Report

Lisa Huang:

- Worked mainly on writing GUIs and connections between the frontend and the backend;
- Wrote CatifyGUI, GamePanel, HelpPanel, MainPanel, ResultPanel and Switcher;
- Applied CardLayout to a container JPanel that contains all the panels in the GUI and enabled users to switch between different panels in the GUI by clicking on buttons (had to use JTabbedPane before);
- Created timers for different actions in GamePanel that allow time gaps between certain actions;
- Debugged and tested on GUIs.

Huihan Li:

- Worked mainly on writing the backend and helped out with GUIs;
- Wrote SettingPanel, GamePanel, CatifyGame, ResultPanel;
- Applied JLayeredPane to each grid within the gridPanel to enable hiding and showing images on the game board;
- Wrote play() method in CatifyGame to facilitate playing on the backend;

Tina Zhang:

- Main role in backend and supported GUIs.
- Wrote Picture, ImageCollection, Board, CatifyGame, SettingPanel
- Applied algorithm to randomly generate unrepeated number to create the board
- Fixed the two bugs in backend
- Debugged and tested on backend codes

IV. How to Run the Catify Program

1. Download and unzip **rhuang2-hli3-yzhang16.zip** from the server
2. Make sure you have Java 8.0 or more installed on your computer
3. Go to cs230-final-project/project
4. Run **CatifyGUI.class**
5. Enjoy the game!

```java
/**
 * <h1>MainPanel.java</h1>
 *
 * <p>
 * The Home Screen for the game, contains the name of a game, a very cute picture, and three buttons: New Game, ? Help,
 * and Quit.
 *
 * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
 * @since 12-08-2017
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class MainPanel extends JPanel {

  // instance variables
  private JLabel l1;
  private JLabel l2;
  private JLabel img;
  private JButton newGame;
  private JButton help;
  private JButton quit;
  private JPanel myParent;

  /**
   * Constructor
   * @param parent the parent panel (cardHolder/container) of this panel
   */
  public MainPanel (JPanel parent) {

    // the parent panel (the card holder) of this panel
    myParent = parent;

    l1 = new JLabel ("Catify ");
    l2 = new JLabel ("The next killer app designed by LHT INC.");
    img = new JLabel (new ImageIcon("../cat_pics/pic3.jpg"));


    l1.setAlignmentX(Component.CENTER_ALIGNMENT);
    l1.setFont(new Font("Chalkduster", Font.PLAIN, 80));
    this.add(l1);

    this.add(Box.createVerticalStrut(20));

    l2.setAlignmentX(Component.CENTER_ALIGNMENT);
    this.add(l2);

    this.add(Box.createVerticalStrut(20));

    img.setAlignmentX(Component.CENTER_ALIGNMENT);
    this.add(img);

    this.add(Box.createVerticalStrut(20));

    // A panel with merely buttons
    JPanel buttonsPanel = ButtonsPanel();
    this.add(buttonsPanel);

    this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
  }

  /**
   * Returns a JPanel with three buttons.
   * @return the result JPanel with three buttons
   */
  private JPanel ButtonsPanel(){

    JPanel result = new JPanel();

    newGame = new JButton ("New Game");
    help = new JButton ("? Help");
    quit = new JButton ("Quit");
```

```
76
77      newGame.setAlignmentX(Component.CENTER_ALIGNMENT);
78      newGame.setPreferredSize(new Dimension(200, 100));
79      newGame.setFont(new Font("Arial", Font.PLAIN, 24));
80      newGame.addActionListener(new ButtonListener());
81      // add a switcher to the newGame button that guides the user to the setting panel
    before starting the game
82      newGame.addActionListener(new Switcher("Setting Panel", myParent));
83      result.add(newGame);
84
85      help.setAlignmentX(Component.CENTER_ALIGNMENT);
86      help.setPreferredSize(new Dimension(200, 100));
87      help.setFont(new Font("Arial", Font.PLAIN, 24));
88      help.addActionListener(new ButtonListener());
89      // add a switcher to the newGame button that guides the user to the setting panel
    before starting the game
90      help.addActionListener(new Switcher("Help Panel", myParent));
91      result.add(help);
92
93
94      quit.setAlignmentX(Component.CENTER_ALIGNMENT);
95      quit.setPreferredSize(new Dimension(200, 100));
96      quit.setFont(new Font("Arial", Font.PLAIN, 24));
97      quit.addActionListener(new ButtonListener());
98      result.add(quit);
99
100     result.setLayout(new FlowLayout());
101
102     return result;
103
104   }
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 e The action event that occurs
115      */
116     public void actionPerformed (ActionEvent e) {
117       if (e.getSource() == quit) {
118         System.out.println("Goodbye!");
119         System.exit(0);
120       }
121       if (e.getSource() == newGame) {
122         System.out.println("Setting up the new game");
123       }
124
125       if (e.getSource() == help) {
126         System.out.println("Showing instructions of the game");
127       }
128     }
129   }
130 }
```

```
76
77      newGame.setAlignmentX(Component.CENTER_ALIGNMENT);
78      newGame.setPreferredSize(new Dimension(200, 100));
79      newGame.setFont(new Font("Arial", Font.PLAIN, 24));
80      newGame.addActionListener(new ButtonListener());
81      // add a switcher to the newGame button that guides the user to the setting panel
    before starting the game
82      newGame.addActionListener(new Switcher("Setting Panel", myParent));
83      result.add(newGame);
84
85      help.setAlignmentX(Component.CENTER_ALIGNMENT);
86      help.setPreferredSize(new Dimension(200, 100));
87      help.setFont(new Font("Arial", Font.PLAIN, 24));
88      help.addActionListener(new ButtonListener());
89      // add a switcher to the newGame button that guides the user to the setting panel
    before starting the game
90      help.addActionListener(new Switcher("Help Panel", myParent));
91      result.add(help);
92
93
94      quit.setAlignmentX(Component.CENTER_ALIGNMENT);
95      quit.setPreferredSize(new Dimension(200, 100));
96      quit.setFont(new Font("Arial", Font.PLAIN, 24));
97      quit.addActionListener(new ButtonListener());
98      result.add(quit);
99
100     result.setLayout(new FlowLayout());
101
102     return result;
103
104   }
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 e The action event that occurs
115      */
116     public void actionPerformed (ActionEvent e) {
117       if (e.getSource() == quit) {
118         System.out.println("Goodbye!");
119         System.exit(0);
120       }
121       if (e.getSource() == newGame) {
122         System.out.println("Setting up the new game");
123       }
124
125       if (e.getSource() == help) {
126         System.out.println("Showing instructions of the game");
127       }
128     }
129   }
130 }
```

```java
/**
 * <h1>HelpPanel.java</h1>
 *
 * <p>
 * A panel that shows instructions on playing the game.
 *
 * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
 * @since 12-08-2017
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class HelpPanel extends JPanel {

  // instance variables
  private JPanel parent;
  private JTextPane text;
  private JPanel buttonPanel;
  private JButton home;

  /**
   * Constructor
   * @param parent the parent panel (cardHolder/container) of this panel
   */
  public HelpPanel(JPanel parent){

    this.parent = parent;
    this.setLayout(new BorderLayout());
    this.setBorder(BorderFactory.createEmptyBorder(20,50,20,50));

    // A panel that contains a Home button that guides user back to the home screen
    buttonPanel = new JPanel();
    home = new JButton("Home");
    home.setPreferredSize(new Dimension(120, 60));
    home.setFont(new Font("Arial", Font.PLAIN, 18));
    home.addActionListener(new Switcher("Main Panel", this.parent));
    buttonPanel.add(home);

    // A JTextPane that contains textual instructions, written in html style
    text = new JTextPane();
    text.setContentType("text/html");
    text.setText("<html><body bgcolor=\"#eeeeee\"><div align=\"center\"
><h1>Instructions</h1></div>" +
        "<font size=5><p><b>Catify</b> is a memory game where you will see a grid of cat
pictures, all faced down, and " +
        "try to find all the pairs of identical pictures within the least possible clicks.
<br><br>" +
        "The game contains three levels of difficulty: EASY (8-pairs), MEDIUM (18-pairs),
and HARD (32-pairs). <br><br>" +
        "At the start of the game, you have as many seconds as you want to memorize the
location of all the pairs. Then click " +
        "<b>Start</b>, and the pictures will turn around. When you click on two pictures,
they must form a pair to stay " +
        "faced up. Otherwise, the pictures " +
        "turn around again. <br><br> Your total attempts and valid attempts (those find
pairs) are tracked. When all the " +
        "pairs are found, you win the game!<br><br>If you want to play another game within
the same level of difficulty, " +
        "simply press <b>Restart</b>. The <b>Home</b> button will guide you back to the
home screen. Also, please feel " +
        "free to restart the game or go back to the home screen in the middle of a game " +
        "if you find it too challenging. <br><br>Enjoy!</p></font></body>");

    this.add(text, BorderLayout.NORTH);
    this.add(buttonPanel);
  }
}
```

```java
/**
 * <h1>SettingPanel.java</h1>
 *
 * <p>
 * Allows user to choose from three levels of difficulty of the game by clicking on three different buttons.
 *
 * @author Lisa Huang (rhuang2), Huihan Li (MAIN AUTHOR) (hli3) and Tina Zhang (yzhang16)
 * @since 12-08-2017
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class SettingPanel extends JPanel {
  //instance variables
  private JButton easyButton,mediumButton,hardButton;
  private JLabel info;
  private JPanel parent;
  private JPanel gridPanel;
  private CatifyGame catify;
  private GamePanel gp;

  /**
   * Constructor
   * Needs these parameter to establish connections between this panel and other panels
   * @param parent the parent panel (cardHolder/container) of this panel
   * @param game the CatifyGame instance
   * @param gp the Game Panel in the GUI
   */
  public SettingPanel(JPanel parent, CatifyGame game, GamePanel gp){
    this.gp = gp;
    this.parent = parent;
    this.catify = game;

    // gets specifically the grid panel in the game panel
    gridPanel = this.gp.getGridPanel();

    GridBagLayout gridbag = new GridBagLayout();
    this.setLayout(gridbag);
    GridBagConstraints c = new GridBagConstraints();

    info = new JLabel("HOW MANY CATS CAN YOU FIND?");
    info.setFont(new Font("Algerian", Font.BOLD, 44));
    info.setHorizontalAlignment(JLabel.CENTER);
    c.gridwidth = GridBagConstraints.REMAINDER;
    c.gridheight = 3;
    gridbag.setConstraints(info,c);
    this.add(info);

    // a JPanel that contains merely JButtons
    JPanel buttonPanel = ButtonPanel();

    c.gridwidth = GridBagConstraints.REMAINDER;
    c.gridheight = GridBagConstraints.REMAINDER;
    gridbag.setConstraints(buttonPanel,c);

    this.add(ButtonPanel());

  }

  /**
   * Returns a JPanel that contains merely JButtons
   * @return the result panel with buttons
   */
  private JPanel ButtonPanel(){

    JPanel result = new JPanel();
    result.setPreferredSize(new Dimension(400,300));
    result.setMaximumSize(new Dimension(400,300));
    result.setMinimumSize(new Dimension(400,300));

    easyButton = new JButton("   EASY   ");
    easyButton.setFont(new Font("Forte", Font.PLAIN, 24));
```

```
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 }
```

```java
/**
 * GamePanel.java
 * The JPanel where the catify game takes place
 * BUG REPORT: If a button whose picture is revealed is clicked on again, the program will find a wrong pair.<br>
 * This bug only exists when playing the game with GUI. Only playing on backend is fine. (Details see line 273)
 *
 * @author Lisa Huang (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
 * @since 12-08-2017
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;
import javax.swing.Timer;
import java.util.LinkedList;

public class GamePanel extends JPanel {
  // instance variables
  private int size;
  // of infoPanel
  private JButton start;
  private JButton home;
  private JLabel info;
  private JLabel attempt;
  private JLabel pairsFound;
  // of gridPanel
  private JPanel gridPanel;
  private JPanel parent;
  private CatifyGame catify;
  private OneLayeredPane[][] grids;
  private LinkedList<LinkedList<Integer>> clicks;


  /*
   * Constructor
   * Constructs a JPanel with infoPanel and gridPanel as subpanels
   * @param JPanel parent - to which panel it belongs
   * @param CatifyGame game
   * */
  public GamePanel (JPanel parent, CatifyGame game) {
    // instantiate variables
    this.catify = game;
    this.parent = parent;
    this.clicks = new LinkedList<LinkedList<Integer>>();

    // set layout
    this.setLayout(new FlowLayout(0, 25, 0));
    this.setBorder(BorderFactory.createEmptyBorder(0,25,25,25));

    JPanel infoPanel = infoPanel();
    gridPanel = gridPanel();

    this.add(gridPanel);
    this.add(infoPanel);
  }


  /*
   * Contains start button, home button, and information about attempts made and pairs found
   * @returns JPanel infoPanel
   * */
  private JPanel infoPanel() {
    JPanel result = new JPanel();
    result.setPreferredSize(new Dimension(200,550));

    start = new JButton("Start");
    start.setAlignmentX(Component.CENTER_ALIGNMENT);
    start.addActionListener(new ButtonListener());

    home = new JButton("Home");
```

```
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");
```

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

```
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
    (which is really abnormal
275                * as nobody would pair a new picture up with a revealed picture),
276                * only the click on the unrevealed picture will be added to the
    LinkedList<LinkedList<Integer>> clicks
277                * because we actually click on the "picture" rather than the "button" of
    the revealed picture.
278                * Therefore, since only one click is tracked, the program will prompt the
    user to make a second click.
279                */
280               // Store the ID of the Picture of the clicked Button
281               LinkedList<Integer> oneClick = new LinkedList<Integer>();
282               oneClick.add(new Integer(i));
283               oneClick.add(new Integer(j));
284               clicks.add(oneClick);
285
286               // Call takeOneClick in CatifyGame
287               catify.takeOneClick(i,j);
288               System.out.println("You have clicked " + catify.getClick() + " picture(s)
    so far");
289             }
290           }
291         }
292
```

```
293         // check if the user found one pair if they made two clickes
294         if (catify.getClick() == 2){
295           //increment attempts
296           attempt.setText("Attempts made: " + catify.getAttempts());
297 
298           boolean foundOnePair = catify.findOnePair();
299           if (foundOnePair)
300             pairsFound.setText("Pairs found: " + catify.getPairsFound());
301           else {
302             // retrieve the information of the clicked button
303             int x1 = clicks.get(0).get(0);
304             int y1 = clicks.get(0).get(1);
305             int x2 = clicks.get(1).get(0);
306             int y2 = clicks.get(1).get(1);
307 
308             //wait for one second and hide the image
309             Timer flipTimer = new Timer(1000, new Flipper(grids[x1][y1], grids[x2][y2]));
310             flipTimer.setRepeats(false);
311             flipTimer.restart();
312           }
313 
314           // prepare for a new round
315           clicks.clear();
316           catify.clearChoices();
317         }
318 
319         // wait for 1.5 seconds and turn to the ResultPanel if the game is finished
320         if (catify.isFinish()) {
321           Timer stayTimer = new Timer(1500, new StayAndSwitch());
322           stayTimer.setRepeats(false);
323           stayTimer.restart();
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 
334     private OneLayeredPane pane1, pane2;
335     /*
336      * Constructor
337      * @param OneLayeredPane p1
338      * @param OneLayeredPane p2
339      * */
340     public Flipper (OneLayeredPane p1, OneLayeredPane p2) {
341       super();
342       pane1 = p1;
343       pane2 = p2;
344     }
345     /*
346      * Flip both pictures
347      * @param ActionEvent event The ActionEvent
348      */
349     public void actionPerformed(ActionEvent e) {
350       pane1.flip(0);
351       pane2.flip(0);
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   }
```

```
369 }
```

```java
/**
 * <h1>ResutltPanel.java</h1>
 *
 * <p>
 * Displays information that congratulates the user on winning the game, and provides the user with options to play
 * again, to return to the home screen and to quit the game.
 *
 * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (MAIN AUTHOR) (hli3) and Tina Zhang (yzhang16)
 * @since 12-08-2017
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class ResultPanel extends JPanel {

  // instance variables
  private JLabel l1;
  private JLabel l2;
  private JButton restart;
  private JButton home;
  private JButton quit;
  private JPanel parent;
  private CatifyGame catify;
  private GamePanel gp;

  /**
   * Constructor
   * Needs these parameter to establish connections between the ResultPanel and other panels
   * @param parent the parent panel (cardHolder/container) of this panel
   * @param game the CatifyGame instance
   * @param gp the Game Panel in the GUI
   */
  public ResultPanel (JPanel parent, CatifyGame game, GamePanel gp) {
    this.catify = game;
    this.parent = parent;
    this.gp = gp;

    l1 = new JLabel("Congratulations!");
    l2 = new JLabel("You found all pairs!");

    l1.setAlignmentX(Component.CENTER_ALIGNMENT);
    l1.setFont(new Font("Chalkduster", Font.PLAIN, 60));
    this.add(l1);

    this.add(Box.createVerticalStrut(20));

    l2.setAlignmentX(Component.CENTER_ALIGNMENT);
    l2.setFont(new Font("Chalkduster", Font.PLAIN, 40));
    this.add(l2);

    this.add(Box.createVerticalStrut(150));

    // a JPanel that contains merely buttons
    JPanel buttonsPanel = ButtonsPanel();

    this.add(buttonsPanel);
    this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
    this.setBorder(BorderFactory.createEmptyBorder(50,50,50,50));


  }

  /**
   * Creates a JPanel with three buttons
   * @return the created JPanel with three buttons
   */
  private JPanel ButtonsPanel () {
    JPanel result = new JPanel();

    restart = new JButton("Restart");
```

```
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
127       if (e.getSource() == home) {
128         System.out.println("Going back to main panel");
129       }
130     }
131   }
132 }
```

```java
/**
 * <h1>Switcher.java</h1>
 *
 * <p>
 * An ActionListener that switch between panels in the parent panel (the cardHolder) with a CardLayout when an event
 * is invoked.
 *
 * @author Lisa Huang (MAIN AUTHOR) (rhuang2), Huihan Li (hli3) and Tina Zhang (yzhang16)
 * @since 12-16-2017
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Switcher implements ActionListener {

  // instance variables
  private JPanel cardHolder;
  private CardLayout cards;
  private String card;

  public Switcher (String card, JPanel holder) {
    this.card = card;
    cardHolder = holder;
    cards = (CardLayout)cardHolder.getLayout();
  }

  public void actionPerformed(ActionEvent e) {
    // shows a certain card in the cardHolder
    cards.show(cardHolder, card);
  }
}
```

```java
/*
 * CatifyGameDriver.java
 * Driver for the Catify game to be played in backend
 *
 * @author Tina Zhang (MAIN AUTHOR) (yzhang16)
 * @since 12/17/2017
 *
 */

public class CatifyGameDriver{
  public static void main(String args[]){
    CatifyGame testGame= new CatifyGame();
    //easy-level game starts
    testGame.play(8);
  }
}
```

```java
/*
 * CatifyGame.java
 * Back-end version of Catify game can be played in this class
 * Bugs in the GUI version have been fixed in this version
 *
 * @author Tina Zhang (MAIN AUTHOR) (yzhang16) , Huihan Li (MAIN AUTHOR) (hli3)
 * @since 12/17/2017
 *
 */
import java.util.*;

public class CatifyGame{

  //Instance variables
  private Board board;
  private int[][] found; //a 2-D array to check if the picture has been found
  private int click, attempts, pairsFound, numPics;
  private int x1, x2, y1, y2;
  private LinkedList<Picture> userClicks; //a LinkdList to store the pictures that users have clicked in one round

  /* Constructor: create & initialize instance variables */
  public CatifyGame(){
    //Instance variables
    numPics = 0;
    click = 0;
    attempts = 0;
    pairsFound = 0;
    x1 = x2 = y1 = y2 = -1;
    userClicks = new LinkedList<Picture>();
  }

  /* Set the game by creating the board and other instance variables
   * @param number of pictures
   */
  public void setGame(int numPictures){
    //create instance variables again (for restart function)
    numPics = numPictures;
    click = 0;
    attempts = 0;
    pairsFound = 0;
    x1 = x2 = y1 = y2 = -1;
    userClicks = new LinkedList<Picture>();
    int size = (int)Math.sqrt(numPics*2);
    board = new Board(numPics);
    found = new int[size][size];
  }

  /*getter method to return number of pictures
   * @return numPics
   */
  public int getNumPics(){
    return numPics;
  }

  /*getter method to return attempts
   * @return attempts
   */
  public int getAttempts(){
    return attempts;
  }

  /*getter method to return how many pairs the user has found
   *@return pairFound
   */
  public int getPairsFound(){
    return pairsFound;
  }

  /*getter method to reuturn how many clicks the user has used
   *@reutrn click
   */
  public int getClick(){
    return click;
  }

```

```
76    /*getter method to return the LinkedList that store the user clicked pictures
77     *@ reutrn 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   }
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     }
```

```
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 }
```

```java
/*
 * Board.java
 * Represents a board object that creates a board for the game
 *
 * @author Huihan Li(hli3), Lisa Huang(rhuang2), Tina Zhang (MAIN AUTHOR) (yzhang16)
 * @since 12/17/2017
 *
 */

import java.util.*;

public class Board{

  // instance variables
  private final ImageCollection pictureSource =new ImageCollection() ; //create the hashtable from ImageCollection
  private Picture[][] gameBoard; //2-D array
  private int numPics;
  private int size;

  /*Constructor that creates the game board that puts random pictures into random positions in the board
   * @param number of pictures needed for the game (easy-8, medium-18, hard-32)
   * @return board
   */
  public Board(int numPics){
    //calcuate board's (2-D array) size
    this.size=(int)Math.sqrt(numPics*2);
    //create empty board
    gameBoard=new Picture[size][size];
    //generate a list of numbers from 0-32 in random order
    ArrayList<Integer> whichPicList = uniqueRandom(32);
    //generate a list of numbers in randm order
    ArrayList<Integer> positionList = uniqueRandom(numPics*2);
    //randomly select numPics of pictures from the pictureSource and put it in a random position in the board
    for(int i=0; i<numPics; i++){
      //get a non-repeated random picture
      int picID = whichPicList.get(i).intValue() + 1;
      Picture pic= pictureSource.getPic(picID);
      // generate two random number
      int firstPos = positionList.get(2*i);
      int secondPos = positionList.get(2*i+1);
      //convert the two random numbers to two positions in the board
      int x1,y1,x2,y2;
      x1 = firstPos/size;
      // debug: System.out.println("x1:"+ x1);
      y1 = firstPos%size;
      // debug: System.out.println("y1:"+ y1);
      x2 = secondPos/size;
      // debug: System.out.println("x2:"+ x2);
      y2 = secondPos%size;
      // debug: System.out.println("y2:"+ y2);
      gameBoard[x1][y1] = pic;
      gameBoard[x2][y2] = pic;
    }
  }

  /* helper function to generate a list of number in random order
   * @param number of pictures
   * @return an arrayList
   */
  private ArrayList<Integer> uniqueRandom(int numPics) {
    //use ArrayList in order to shuffle in next step
    ArrayList<Integer> list = new ArrayList<Integer>();
    for (int i=0; i<numPics; i++) {
      list.add(new Integer(i));
    }
    //generate random order
    Collections.shuffle(list);
    return list;
  }

  /*get method to return the picture stored the position
   *@param x, y that indicates which position
   *@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 }
```

```java
/*
 * ImageCollection.java
 * Create a hashtable for all the images
 *
 * @author Huihan Li(hli3),Lisa Huang(rhuang2), Tina Zhang (MAIN AUTHOR) (yzhang16)
 * @since 12/17/2017
 *
 */
import java.util.*;

public class ImageCollection{

  //instance variable
  Hashtable<Integer, Picture> images;

  /*Constructor contructs a hashtable*/
  public ImageCollection(){
     //create a hash table that has integer as key and picture object as element
     images = new Hashtable<Integer, Picture>(43);
     //put the pictures into the hashtable, the key is equal to the picture ID
     for(int i=1;i<=32;i++){
       Picture pic = new Picture (i,"../cat_pics/pic"+i+".jpg");
       images.put(i, pic);
     }
  }

  /* get method that return a picture according to its picture ID
   * @param picture ID
   * @return picture
   */
  public Picture getPic(int picID){
    return images.get(picID);
  }

  //main method to test
  public static void main(String args[]){
    ImageCollection test = new ImageCollection();
    System.out.println ("Printing out the hashtable in the format of *Picture ID: URL*");
    for(int i=1;i<=32;i++){
      Picture pic = test.getPic(i);
      System.out.println(i+":"+pic.getFileName());
     }
  }
}
```

```java
/*
 * Picture.java
 * Represents a picture object that contains a picture ID and a picture URL
 *
 * @author Huihan Li(hli3),Lisa Huang(rhuang2), Tina Zhang (MAIN AUTHOR) (yzhang16)
 * @since 12/17/2017
 *
 */

public class Picture implements Comparable<Picture>{

  // Instance variables
  String picFileName;
  int picID;

  /* Constructor,create a picture object
   * @param an integer that represents the picture's ID
   * @param picture's URL
   */
  public Picture (int picID, String picFileName){
    this.picFileName = picFileName;
    this.picID = picID;
  }

  /* get method that return picture's ID
   * @return picture's ID
   */
  public int getID(){
    return picID;
  }

  /* get method that return picture's URL
   * @return picture's URL
   */
  public String getFileName(){
    return picFileName;
  }

  /* Implements comparable intereface, compare two picture's ID
   * @override
   * @param comparable picture
   * @return 1 if pic1 ID > pic2 ID;
   *         0 if pic1 ID = pic2 ID;
   *        -1 if pic1 ID < pic2 ID;
   */
  public int compareTo(Picture Pic2){
    if(this.getID()>Pic2.getID()){
      return 1;
    }
    else if(this.getID()==Pic2.getID()){
      return 0;
    }
    else{
      return -1;
    }
  }

  //main method to test
  public static void main (String args[]){
    Picture testpic = new Picture (3, "../cat_pics/pic3.jpg");
    System.out.println("Pic N0.3's ID is " + testpic.getID());
    System.out.println("Pic N0.3's URL is " + testpic.getFileName());
  }

}
```