```
/*-------------------------------------------------------------------------------------------------
 * Author:      Dan Cassidy
 * Date:        2015-07-18
 * Assignment:  HW4-1
 * Source File: GameView.java
 * Language:    Java
 * Course:      CSCI-C 490, Android Programming, MoWe 08:00
 -------------------------------------------------------------------------------------------------*/
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;

/**
 * View for the Tic-Tac-Toe game. Handles the visual representation.
 *
 * @author Dan Cassidy
 */
@SuppressWarnings("serial")
public class GameView extends JFrame
{
    private JLabel statusLabel;
    private JButton[][] board;
    private JLabel winningConditionsLabel;
    private JButton resetButton = new JButton("Reset");

    /**
     * 2-parameter constructor.
     *
     * @param numRows The number of rows of buttons the game board will have.
     * @param numColumns The number of columns of buttons the game board will have.
     */
    public GameView(int numRows, int numColumns)
    {
        // General window options and layout.
        super("Tic Tac Toe");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setResizable(false);
        setSize(numColumns * 50, (numRows + 2) * 50);
        setLayout(new BorderLayout());

        // NORTH. Create and add the game status label.
        statusLabel = new JLabel();
        statusLabel.setHorizontalAlignment(SwingConstants.CENTER);
        add(statusLabel, BorderLayout.NORTH);

        // CENTER. Create and add the buttons for the board.
        JPanel boardPanel = new JPanel(new GridLayout(numRows, numColumns));
        board = new JButton[numRows][numColumns];
        for (int row = 0; row < numRows; row++)
            for (int column = 0; column < numColumns; column++)
            {
                board[row][column] = new JButton();
                board[row][column].setActionCommand("" + (row * numColumns + column));
```

```
61                  boardPanel.add(board[row][column]);
62              }
63          add(boardPanel, BorderLayout.CENTER);
64
65          // SOUTH. Create and add a label detailing the winning conditions and a reset button.
66          winningConditionsLabel = new JLabel();
67          winningConditionsLabel.setHorizontalAlignment(SwingConstants.CENTER);
68          JPanel resetButtonPanel = new JPanel();
69          resetButtonPanel.setLayout(new FlowLayout());
70          resetButtonPanel.add(resetButton);
71          JPanel bottomPanel = new JPanel(new GridLayout(2,1));
72          bottomPanel.add(winningConditionsLabel);
73          bottomPanel.add(resetButtonPanel);
74          add(bottomPanel, BorderLayout.SOUTH);
75      }
76
77      /**
78       * Add an action listener to all of the board buttons.
79       *
80       * @param listener The ActionListener to add to the buttons.
81       */
82      public void addBoardButtonActionListener(ActionListener listener)
83      {
84          for (JButton[] buttonRow : board)
85              for (JButton button : buttonRow)
86                  button.addActionListener(listener);
87      }
88
89      /**
90       * Add an action listener to the reset button.
91       *
92       * @param listener The ActionListener to add to the reset button.
93       */
94      public void addResetButtonActionListener(ActionListener listener)
95      {
96          resetButton.addActionListener(listener);
97      }
98
99      /**
100      * Resets the status text and the board buttons to default.
101      */
102     public void reset()
103     {
104         statusLabel.setText("");
105         setBoardEnabled(true);
106
107         for (JButton[] buttonRow : board)
108             for (JButton button : buttonRow)
109                 button.setText("");
110     }
111
112     /**
113      * Interface method to set the text of a given board button.
114      *
115      * @param row The board row of the button.
116      * @param column The board column of the button.
117      * @param text The String to set the text to.
118      */
119     public void setBoardButtonText(int row, int column, String text)
120     {
```

```
121             board[row][column].setText(text);
122         }
123
124         /**
125          * Enables (or disables) the button board.
126          *
127          * @param b true to enable the button board, otherwise false.
128          */
129         public void setBoardEnabled(boolean b)
130         {
131             for (JButton[] buttonRow : board)
132                 for (JButton button : buttonRow)
133                     button.setEnabled(b);
134         }
135
136         /**
137          * Interface method to update the status label.
138          *
139          * @param text The String to set the text to.
140          */
141         public void setStatusLabelText(String text)
142         {
143             statusLabel.setText(text);
144         }
145
146         /**
147          * Interface method to update the winning conditions label.
148          *
149          * @param text The String to set the text to.
150          */
151         public void setWinningConditionsLabelText(String text)
152         {
153             winningConditionsLabel.setText(text);
154         }
155     }
156
```