```java
/*

Jared Dyreson
CWID: 889546529
JDemoKey.java ->
csrc_compile: TRUE

*/

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

public class JDemoKey extends JFrame implements KeyListener{

	// the boundaries for the checker board
	private final int FRAME_HEIGHT = 500, FRAME_WIDTH = 500, ROWS = 16, COLS = 16;

	// we need two parallel arrays to keep track of the colors on the board
	// this one keeps track of the colors for the pane
	private JPanel[][] tpanel = new JPanel[ROWS][COLS];
	// this keeps track of what colors the tpanel object reflects
	private Color[][] color_panel = new Color[ROWS][COLS];
	// the main container for the checker board layout
	private JPanel pane = new JPanel(new GridLayout(ROWS, COLS,2, 2));

	private Color w = Color.WHITE;
	private Color b = Color.BLACK;
	// so we can keep track of what colors are around us
	private Color previous_color, current_color;

	private Cursor cursor = new Cursor(0, 0);

	// setters
	public void set_previous_color(Color c){ this.previous_color = c; }
	public void set_current_color(Color c){ this.current_color = c; }

	// getters
	public Color get_previous_color(){ return this.previous_color; }
	public Color get_current_color(){ return this.current_color; }

	public JDemoKey(){
		super("MS Paint");

		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setSize(FRAME_HEIGHT, FRAME_WIDTH);
		this.setLayout(new BorderLayout());

```

```java
                for(int x = 0; x < COLS; ++x){
                        for(int y = 0; y < ROWS; ++y){
                                // make the grid and corresponding parallel array with all white tiles
                                tpanel[x][y] = new JPanel();
                                color_panel[x][y] = this.w;
                                this.pane.add(tpanel[x][y]);
                                tpanel[x][y].setBackground(color_panel[x][y]);
                        }
                }

                this.add(pane, BorderLayout.CENTER);

                addKeyListener(this);
                // initialize the cursor
                this.tpanel[0][0].setBackground(this.cursor.get_color());
        }

        public void set_position(int x, int y, boolean coloring) throws IndexOutOfBoundsException{

                // initial position
                int x_naught = this.cursor.get_x();
                int y_naught = this.cursor.get_y();

                // move the cursor over
                this.tpanel[y][x].setBackground(this.cursor.get_color());
                // update the current and previous colors
                this.set_previous_color(this.color_panel[y_naught][x_naught]);
                this.set_current_color(this.color_panel[y][x]);

                // we pass in a boolean flag to allow us to use the same function
                // with different conditions

                if(coloring){
                        // erase black tiles
                        if(this.previous_color == Color.BLACK){
                                this.color_panel[y_naught][x_naught] = this.w;
                                this.tpanel[y_naught][x_naught].setBackground(this.w);
                                this.previous_color = Color.WHITE;
                        }
                        // mark tiles
                        else if(this.current_color == Color.WHITE){
                                this.color_panel[y_naught][x_naught] = this.b;
                                this.tpanel[y_naught][x_naught].setBackground(this.b);
                                this.previous_color = Color.BLACK;
                        }
                }
                // just continue as normal
                else{
```

```
                                this.tpanel[y_naught][x_naught].setBackground(this.previous_color);
                        }
                        // update cursor location
                        this.cursor.set_position(x, y);
                        // make the pane reload
                        this.pane.revalidate();
                        this.pane.repaint();
                        // we are only allowing for one pixel being marked
                        // at a time
                        this.cursor.toggle_marker(false);

        }

        @Override
        public void keyTyped(KeyEvent event){
                // only executes when char is typed
                char c = event.getKeyChar();
        }

        @Override
        public void keyPressed(KeyEvent event){

                int key_code = event.getKeyCode();

                // get the position of the cursor
                int x = this.cursor.get_x();
                int y = this.cursor.get_y();
                // we have a try catch block to basically ignore the fact that we bump against a wall
                // the catch does nothing
                try{
                        // if the space key is pressed, we can indicate we want to draw
                        if(key_code == KeyEvent.VK_SPACE){
                                this.cursor.toggle_marker(true);
                        }
                        // -/+ are flipped because of the frame of reference
                        // - means we want to go up/right because the indexes start from 0
                        // + means we want to go down/left because the indexes start from 0

                        if(key_code == KeyEvent.VK_UP){
                                this.set_position(x, y-1, this.cursor.get_space_flag());
                        }
                        else if(key_code == KeyEvent.VK_DOWN){
                                this.set_position(x, y+1, this.cursor.get_space_flag());
                        }
                        else if(key_code == KeyEvent.VK_LEFT){
                                this.set_position(x-1, y, this.cursor.get_space_flag());
                        }
                        else if(key_code == KeyEvent.VK_RIGHT){
```

```java
                                this.set_position(x+1, y, this.cursor.get_space_flag());
                        }
                }
                catch(Exception error){}
        }

        @Override
        public void keyReleased(KeyEvent event){ return; }
        public static void main(String[] args){
                // Auto generated with caffine and autovt@.service
                JDemoKey j = new JDemoKey();
                j.setVisible(true);
        }
}
```