```java
package package1;

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

/**
 * Created by jon on 1/27/14.
 */
public class MineSweeperPanel extends JPanel {
    private JButton[][] board;
    private JButton quitButton;
    private JButton resetButton;
    private Cell iCell;
    private MineSweeperGame game;
    private static int boardSize = 10;
    private static int mineCount = 10;
    private ImageIcon flagIcon;
    private ImageIcon mineIcon;
    private ImageIcon unselectedIcon;
    private ImageIcon[] mineCountIcons;
    private int winCount;
    private int loseCount;
    private int flagCount;
    private JPanel gamePanel;
    private JPanel buttonPanel;
    private JLabel lCounter;
    private JLabel wCounter;
    private JLabel fCounter;
    private MouseListener mouseListener;
    private boolean cheat;

    /**
     * constructor for the minesweeper panel
     */
    public MineSweeperPanel() {
        init();
    }

    /**
     * removes panels to avoid weird redrawing bug, reinits everything
     */
    private void reset() {
        this.remove(gamePanel);
        this.remove(buttonPanel);
        init();
        this.revalidate();
        SwingUtilities.updateComponentTreeUI(this.getParent());
    }

    /**
     * construct in separate method to allow easy resetting
     */
    public void init() {
        // icons
        flagIcon = new ImageIcon("icons/flag.png");
        mineIcon = new ImageIcon("icons/mine.png");
        unselectedIcon = new ImageIcon("icons/unselected.png");
        mineCountIcons = new ImageIcon[9];
        for (int i = 0; i < 9; i++) mineCountIcons[i] = new ImageIcon("mine" + i + ".png");
        // show mines in the beginning?
        if (JOptionPane.showConfirmDialog(null,
                "Are you a cheater?", "Show mines?",
                JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION) cheat = true;
        // / handle input for size/mines, 10 for default
        try {
            boardSize = Integer.parseInt(JOptionPane.showInputDialog("How large should the board be? (3-30)"));
            mineCount = Integer.parseInt(JOptionPane.showInputDialog("How many mines should there be?"));
        } catch (Exception ignored) {
        }
        if (boardSize > 30 || boardSize < 3) boardSize = 10;
        if (mineCount > boardSize * boardSize - 1 || mineCount < 1) mineCount = 10;
        flagCount = 0;
        // create the game, board, listeners, components
        game = new MineSweeperGame(boardSize, mineCount);
        mouseListener = new MouseListener();
        buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(1, 5));
        gamePanel = new JPanel();
        quitButton = new JButton("Quit");
        resetButton = new JButton("Reset");
        wCounter = new JLabel("W: " + winCount);
        lCounter = new JLabel("L: " + loseCount);
        fCounter = new JLabel("Flags: " + flagCount);
        //add things
        quitButton.addMouseListener(mouseListener);
        resetButton.addMouseListener(mouseListener);
        buttonPanel.add(quitButton);
        buttonPanel.add(lCounter);
        buttonPanel.add(wCounter);
        buttonPanel.add(fCounter);
        buttonPanel.add(resetButton);
        gamePanel.setLayout(new GridLayout(boardSize, boardSize));
        board = new JButton[boardSize][boardSize];
        // loop that sets all the buttons,icons,listeners, adds to panel
        for (int row = 0; row < boardSize; row++) {
            for (int col = 0; col < boardSize; col++) {
                board[row][col] = new JButton("");
                board[row][col].setPreferredSize(new Dimension(25, 25));
                board[row][col].addMouseListener(mouseListener);
                board[row][col].setIcon(unselectedIcon);
                gamePanel.add(board[row][col]);
            }
        }
        //put it all together
        gamePanel.setPreferredSize(new Dimension(boardSize * 25, boardSize * 25));
        add(buttonPanel);
        add(gamePanel);
        displayBoard();
    }

    /**
     * renders the board by setting the icons
     */
    private void displayBoard() {
        flagCount = 0;
        for (int row = 0; row < boardSize; row++) {
            for (int col = 0; col < boardSize; col++) {
                iCell = game.getCell(row, col);
                if (iCell.isExposed()) {
                    board[row][col].setIcon(mineCountIcons[iCell.getMineCount()]);
                    continue;
                }
                if (iCell.isFlagged()) {
                    board[row][col].setIcon(flagIcon);
                    flagCount++;
                } else {
                    board[row][col].setIcon(unselectedIcon);
                }
                if (cheat) if (iCell.isMine() && !iCell.isFlagged()) board[row][col].setIcon(mineIcon);
            }
        }
        //labels
        wCounter.setText("W:  " + winCount);
        lCounter.setText("L:  " + loseCount);
        fCounter.setText("Flags:  " + flagCount);
    }

    /**
     * shows everything on lose
     */
    private void displayAllMines() {
        for (int row = 0; row < boardSize; row++) {
            for (int col = 0; col < boardSize; col++) {
                game.select(row, col);
                iCell = game.getCell(row, col);
                if (iCell.isMine()) {
                    board[row][col].setIcon(mineIcon);
                } else if (!iCell.isFlagged()) {
                    board[row][col].setIcon(mineCountIcons[iCell.getMineCount()]);
                }
            }
        }
    }



    /**
     * listener to detect clicks, handle actions
     */
    class MouseListener extends MouseAdapter {
        public void mouseReleased(MouseEvent e) {
            if (e.getSource() == quitButton) System.exit(0);
            if (e.getSource() == resetButton) reset();
            if (e.getButton() == 3) {
                for (int row = 0; row < boardSize; row++)
                    for (int col = 0; col < boardSize; col++)
                        if (board[row][col] == e.getSource()) {
                            game.flag(row, col);
                        }
            } else {
                for (int row = 0; row < boardSize; row++)
                    for (int col = 0; col < boardSize; col++)
                        if (board[row][col] == e.getSource()) {
                            game.select(row, col);
                        }
            }
            //game statuses
            if (game.getGameStatus() == GameStatus.LOST) {
                loseCount++;
                displayAllMines();
                return;
            }
            displayBoard();
            if (game.getGameStatus() == GameStatus.WON) {
                if (!cheat) winCount++;
                JOptionPane.showMessageDialog(null, "You Win!");
            }
        }
    }
}
```