```java
package hu.elte.prt.eightqueens.model;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class Engine {

    private static final int SIZE = 8;

    private boolean[][] queens;
    private List<Position> queensPositions;
    private boolean paused;

    public void startNewGame() {
    paused = false;
    queensPositions = new ArrayList<>();
    queens = new boolean[SIZE][SIZE];
    for (int i = 0; i < SIZE; ++i) {
        for (int j = 0; j < SIZE; ++j) {
        queens[i][j] = false;
        }
    }
    }

    public int getSize() {
    return SIZE;
    }

    public void put(int i, int j) {
    if (!paused) {
        getFirstEmptyColumn().ifPresent(col -> putIfNotInScope(col, i, j));
    }
    }

    private Optional<Integer> getFirstEmptyColumn() {
    for (int j = 0; j < SIZE; ++j) {
        if (columnIsEmpty(j)) {
        return Optional.of(j);
        }
    }
    return Optional.empty();
    }

    private void putIfNotInScope(Integer col, int i, int j) {
    if (j == col && isNotInScope(i, j)) {
        queens[i][j] = true;
        queensPositions.add(new Position(i, j));
    }
    }

    public boolean isNotInScope(int i, int j) {
    for (Position p : queensPositions) {
        Position p2 = new Position(i, j);
        if (isTheSameRow(p, p2) || isTheSameColumn(p, p2) || isTheSameDiagonal(p, p2)) {
        return false;
        }
```

```
    }
    return true;
    }

    private boolean isTheSameRow(Position p, Position p2) {
    return p.getRow() == p2.getRow();
    }

    private boolean isTheSameColumn(Position p, Position p2) {
    return p.getColumn() == p2.getColumn();
    }

    private boolean isTheSameDiagonal(Position p, Position p2) {
    return Math.abs(p.getRow() - p2.getRow()) == Math.abs(p.getColumn() - p2.getColumn());
    }

    private boolean columnIsEmpty(int j) {
    for (int i = 0; i < SIZE; ++i) {
        if (queens[i][j]) {
        return false;
        }
    }
    return true;
    }

    public boolean isQueen(int i, int j) {
    return queens[i][j];
    }

    public boolean won() {
    if (SIZE == queensPositions.size()) {
        startNewGame();
        return true;
    }
    return false;
    }

    public void undo() {
    if (!paused && !queensPositions.isEmpty()) {
        Position lastQueen = getLastQueenPosition();
        removeQueen(lastQueen);
        queensPositions.remove(lastQueen);
    }
    }

    private Position getLastQueenPosition() {
    return queensPositions.get(queensPositions.size() - 1);
    }

    private void removeQueen(Position pos) {
    queens[pos.getRow()][pos.getColumn()] = false;
    }

    public boolean canPutHere(int i, int j) {
    return isNotInScope(i, j) && j == queensPositions.size();
    }
```

```
    public void togglePause() {
    paused = !paused;
    }

    public boolean isPaused() {
    return paused;
    }

}
```

```
package hu.elte.prt.eightqueens.view;

import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;

import hu.elte.prt.eightqueens.model.Engine;

public class Frame extends JFrame {

    private static final long serialVersionUID = 831657296117161624L;

    private Engine engine;
    private JMenuItem pauseMenuItem;

    public Frame(Engine engine) {
    super("8 Queens");
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    this.engine = engine;
    }

    public void showFrame() {
    createFields();
    setMenu();
    pack();
    setVisible(true);
    }

    private void createFields() {
    getContentPane().setLayout(new GridLayout(engine.getSize(), engine.getSize()));
    for (int i = 0; i < engine.getSize(); ++i) {
        for (int j = 0; j < engine.getSize(); ++j) {
        JButton field = new JButton();
        field.setBackground(getFieldBackground(i, j));
        field.setPreferredSize(new Dimension(80, 80));
        field.setFont(field.getFont().deriveFont(30.0f));
        addFieldActionListener(i, j, field);
        getContentPane().add(field);
        }
    }
    }

    private void addFieldActionListener(int i, int j, JButton field) {
    field.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
        engine.put(i, j);
```

```
        updateFields();
        checkVictoryCondition();
        }
    });
    }

    private void checkVictoryCondition() {
    if (engine.won()) {
        JOptionPane.showMessageDialog(this, "Oh nice.");
        updateFields();
    }
    }

    private void setMenu() {
    JMenuBar menuBar = new JMenuBar();
    JMenu menu = new JMenu("Game");
    menuBar.add(menu);
    addRestartMenuItem(menu);
    addUndoMenuItem(menu);
    addPauseMenuItem(menu);
    setJMenuBar(menuBar);
    }

    private void addRestartMenuItem(JMenu menu) {
    JMenuItem menuItem = new JMenuItem("Restart");
    menuItem.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
        engine.startNewGame();
        updateFields();
        updatePauseMenuItem();
        }
    });
    menu.add(menuItem);
    }

    private void addUndoMenuItem(JMenu menu) {
    JMenuItem menuItem = new JMenuItem("Undo");
    menuItem.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
        engine.undo();
        updateFields();
        }
    });
    menu.add(menuItem);
    }

    private void addPauseMenuItem(JMenu menu) {
    pauseMenuItem = new JMenuItem("Pause");
    pauseMenuItem.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
        engine.togglePause();
        updatePauseMenuItem();
        updateFields();
        }
```

```
    });
    menu.add(pauseMenuItem);
    }

    private void updatePauseMenuItem() {
    pauseMenuItem.setText(engine.isPaused() ? "Resume" : "Pause");
    }

    private void updateFields() {
    for (int i = 0; i < engine.getSize(); ++i) {
        for (int j = 0; j < engine.getSize(); ++j) {
        Component c = getContentPane().getComponent(i * engine.getSize() + j);
        JButton field = (JButton) c;
        field.setText(getFieldText(i, j));
        field.setBackground(getFieldBackground(i, j));
        }
    }
    }

    private String getFieldText(int i, int j) {
    return !engine.isPaused() && engine.isQueen(i, j) ? "♕" : "";
    }

    private Color getFieldBackground(int i, int j) {
    if (!engine.isPaused() && engine.canPutHere(i, j)) {
        return Color.RED;
    }
    return (i + j) % 2 == 0 ? Color.GRAY : Color.WHITE;
    }

}
```

```java
package hu.elte.prt.eightqueens;

import hu.elte.prt.eightqueens.model.Engine;
import hu.elte.prt.eightqueens.view.Frame;

public class Launcher {

    public static void main(String[] args) {
    Engine engine = new Engine();
    Frame frame = new Frame(engine);
    engine.startNewGame();
    frame.showFrame();
    }
}
```

```java
package hu.elte.prt.eightqueens.model;

public class Position {
    private int row;

    private int column;

    public Position(int row, int column) {
    this.row = row;
    this.column = column;
    }

    public int getRow() {
    return row;
    }

    public int getColumn() {
    return column;
    }

    @Override
    public boolean equals(Object o) {
    if (o instanceof Position) {
        Position p = (Position) o;
        return row == p.getRow() && column == p.getColumn();
    }
    return false;
    }

    @Override
    public int hashCode() {
    return row * 11 + column * 7;
    }
}
```