```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;
    }

}
```