`EightController` Bean

It contains the data structure used to logically represent the game movements. In particular, it includes a `List<List<Integer>> board` to represent the values for each slot in the grid. A `vetoableChange` method has been implemented so that whenever a tile is pressed, it first checks if that move is legal (by examining whether there's any hole at the left, right, up, and down using the `List<List<Integer>> board`). If the move is legal, the board values are updated based on which tile has been decided to be moved. `EightController` manages the flip move as well by ensuring that the tile in position 9 is a hole. This is possible because of the `propertyChange` method that first checks whether the incoming event is a `Constants.FLIP_EVT` (sent by the `flipButtonActionPerformed` in the `EightBoard` class) and then applies the logic as described earlier in the `checkFlipMove()` method. The same methodology has been applied for the restart functionality. Whenever the user presses the button, the `resetTilesPosition()` method in the `EightBoard` class is executed, firing a property change that is being listened to by the controller, applying the generated new permutation accordingly to the grid.

`EightBoard` Bean

It is a `JFrame` containing the tiles list, and its behavior consists of initializing and setting the graphics of the game and setting listeners for both tiles and the controller. It also handles the click events for the restart and flip button. When `EightBoard` is initialized, `addPropertyChangeListener(eightController)` is used so that the `EightController` object can initialize its `List<List<Integer>> board` due to the execution of `resetTilePosition()`. In this case, it is used to start up the tile positions at the beginning.

`EightTile` Bean

It is responsible for each tile UI based on the events that it receives, but it also fires `"tileLabelProperty"` events when it's clicked. `EightController` is the main receiver of such an event.