Advanced Programming Assignment 1 - Exercise 1

Diego Arcelli - 647979

January 2023

Tiles movement

In order to implement the the change of the `label` property of a tile when it is clicked, all the tiles register as `PropertyChangeListener` of their adjacent tiles (for example tile 1 will only register as listener of tile 2 and 4), while the controller registers as `VetoableChangeListener` of all the tiles. In order to understand whether a move is legal or not, the controller maintains a list of integers of 9 elements, where each position of the list corresponds to a tile of the board, and the value in that position represent the value of the `label` property of the tile. When a tile is clicked and it tries to set its `label` value to 9, the controller uses its internal representation of the board to determine if the tile which has been clicked is adjacent to the hole. In case it is adjacent, it updates it's internal representation of the board, otherwise it throws a `PropertyVetoException` to forbid the movement. If the movement is allowed, the tile which has been pressed change its `label` value to 9 and it fires the property change event to its adjacent tiles. Each tile that listens to the event, check if its current value of `label` is 9, and if this condition is true, the tile sets its `label` value to the old value of the tile which has been clicked. The adjacent tiles can get the old value of the clicked tile from the object of type `PropertyChangeEvent`, which is the only parameter of the `propertyChange` method.

Restart

In order to handle the restart of the board I created a class called `RestartEvent`, which represent the event which is fired when the restart button is clicked. This class has two attributes:

- `listeners`: of type `ArrayList<RestartListener>`, which is used to maintain the list of all the listeners of the event
- `permutation`, of type `List<Integer>`, which is a list of 9 elements where each element of the list corresponds to the `label` value of a tile

`RestartListener` is the interface which a class must implement in order to allow its instances to register as listeners of the `Restart` event. The interface has one abstract of signature:

```
public abstract void onRestart(RestartEvent evt);
```

All the listeners must provide a concrete implementation of this method, in order to define their behavior once the restart event is fired.
The class `RestartEvent` has two methods:

- `addRestartListener(RestartListener l)`: which is used to add a new event listener
- `update()`: which generates a random permutation that is assigned to the `permutation` attribute, and then it fires the event calling the `onRestart` method of all the listeners

The class `EightTile` implements the `RestartListener` interface. When the event is fired every tile use its value of the `position` property to get the new randomly generated value for its `label` property.
Also the `EightController` implements the `RestartListener` interface. When the restart event is fired the `EightController` updates its internal representation of the board, with the one generated by the restart event.
The event is fired by the `Restart` button, whenever the button is clicked.

Flip

In order to handle the flipping of the first two tiles, I created a new event that is represented by the `FlipEvent` class, which has an attribute of type `List<FlipListener>`, that is used to maintain a list of all the listeners of the event.
`FlipListener` is the interface which a class must implement in order to allow its instances to register as listeners of the flip event. The interface has one abstract of method of signature:

```
public abstract void onFlip(FlipEvent evt);
```

All the listeners must provide a concrete implementation of this method. The method can throw a `FlipForbiddenException`, which is an exception I defined and it can be thrown in order to forbid the flip (similarly to what happens for a vetoable property).
The `FlipListener` interface is implemented by the `EightController`, which redefines the `onFlip` to check if the hole is in position 9. If this condition doesn't hold, then the controller throws the `FlipForbiddenException`, otherwise the controller updates its internal representation of the board, flipping the elements in the first two positions of the list representing the board.
The flip button is defined in the class `Flip` (that extends `JButton`). When the button is clicked the `update` method of `Flip` is called. This method first fires the flip event, then, if the controller doesn't throw the `FlipForbiddenException`, it flips the two tiles in position 1 and 2, calling on them the `flip` method of the `EightTile` class. The `flip` method allows to set the value of the `label` property of a tile regardless the position of the hole.