Advanced Programming
*Assignment **n.1** - Exercise **n.1***

Antonio Strippoli

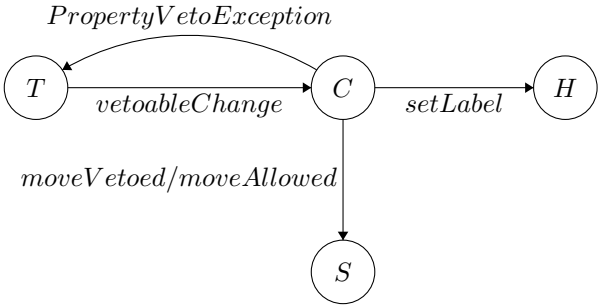
The project has been structured in **four files**:

- `EightController.java`: it implements all of the game logic, while being agnostic of the GUI components. It doesn't know about the tiles, and it does only keep track of the position of the *Hole*. Differently from the original request, it does not have the graphical appearance of a label, which is instead implemented by `EightStatus.java`. This allows the game logic and the game GUI to be loosely coupled, allowing to individually update them more easily in the future;
- `EightStatus.java`: a Bean inheriting from `JLabel`. It displays a different label depending on the current status of the game;
- `EightTile.java`: a Bean inheriting from `JButton`. Differently from the original request, the field `label` is named `tileLabel`, since the first one was a deprecated field of `JButton` and would have lead to different naming for its getter/setter, thus generating confusion;
- `EightBoard.java`: entry point spawning all the GUI elements and setting their properties and listeners;

Concerning the GUI, I've decided to employ a `GridLayout`, enabling responsiveness and significantly shortening the boilerplate code, while not changing the appearance too much from the request.

For the game logic, the start/restart of the game is handled by the `EightController`: it fires a `PropertyChange` event to all the `EightTiles` (giving them the permutation array) and to the `EightStatus` (in order to update it with "START").

Next, there is the **click of an** `EightTile`. This event sees four GUI elements communicating with each other as per the picture:



- **T**: an *EightTile*. It interacts with the *Controller* (C) when it is clicked, firing a `vetoableChange` event. If the *Controller* does not veto, the tile becomes the *Hole*, otherwise it blinks red;
- **C**: the *EightController*. When fired by a `vetoableChange` event, if the click on the *Tile* is not legal it throws a `PropertyVetoException`. The decision made here is also notified to the label *Status* (S). Finally, the *Controller* fires a `PropertyChange` event to the *Hole* (H), requesting it to set its label to the old value of the newly became *Hole*;

Finally, we have the **click on the** `Flip` **button**, which is implemented by an handler in `EightBoard`. The label of the 1st and the 2nd *Tile* are passed to the *Controller*, which after checking if the flip is possible, it fires two `PropertyChange` events to the tiles, in order to swap the two labels.