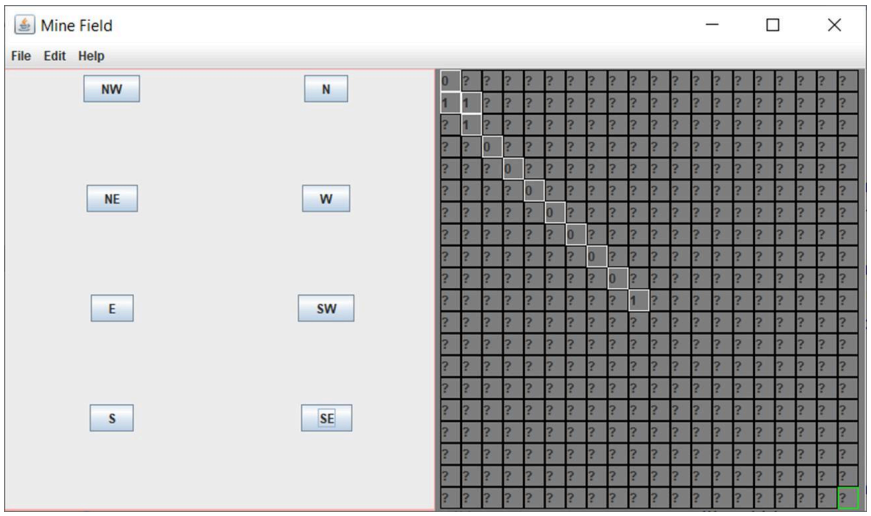
Mine Field

Starting in the upper-left corner, Sargent Rock must tiptoe through a mine field trying to reach the bottom-right corner (the green cell). With each step a mine detector he carries tells him how many neighboring cells are mined, but not which ones. He must use logical deduction to figure this out.

In the screenshot below, we see the path Rock has travelled (in white), one step at a time, by using the direction buttons (which are repeated under the Edit menu). He has paused near the center of the minefield because he knows one of the neighboring cells contains a mine, but which one?



Implement Mine Field as a customization of the MVC framework.

Hints

- Your model can throw exceptions in the following situations:
  - The player tries to move off of the grid. (Screen wrapping is not allowed).
  - The player steps on a mine.
  - The player reaches the goal.
  - The player attempts to move after the game has ended.
- Patches are mined randomly. I use a static variable to determine the percentage of mined patches:

```
public static int percentMined = 5;
```

- I only needed a single move command instead of one for each heading.

```
class MoveCommand extends Command {
   Heading heading;
```

- When the user selects New or Open from the File menu, the MVC framework calls AppPanel.setModel(newModel) so that the AppPanel can unsubscribe from the old model and subscribe to the new one. It also calls View.setModel(newModel) so that the view can do the same thing. I overrode this in MineFieldView as follows:

```
public void setModel(Model newModel) {
super.setModel(newModel);
   initView();
   repaint();
}
```

Where initView repopulates the view's grid with fresh cells, similar to what's done in the view's constructor.