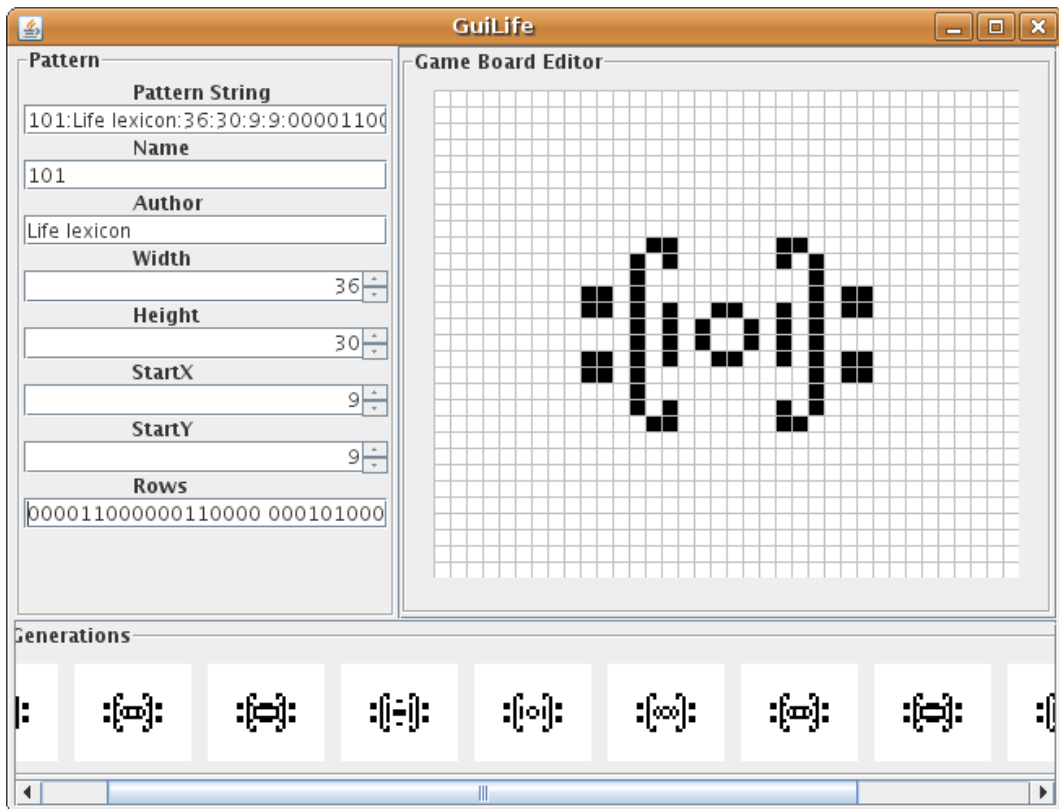


# Tick 8\*

In this exercise you will write a simple Game of Life game board editor. The editor allows the user to interactively build a Game of Life game board by either: (1) clicking the mouse on individual cells to invert their current state (dead to alive or alive to dead), or (2) editing the game state as recorded in a series of text fields describing the pertinent facts about the world. A "film strip" at the bottom of the editor should display the state of game board in successive generations. An example user interface which supports these features is shown in Figure 1, "The Game Board Editor".



**Figure 1. The Game Board Editor**

As you can see in the figure, the editor consists of three main components: the Pattern Panel, which displays a series of `JTextField` and `JSpinner` widgets; the Game Board Panel, which displays a visual representation of the pattern described in the Pattern Panel; and the Generations Panel, which should display the next twenty generations of the game board.

Your implementation of the editor should ensure that the information stored in the various fields inside the Pattern Panel is kept in strict synchrony with the information displayed in the Game Board Panel and the Generations Panel, regardless of whether the user edits the Game Board Panel, or the Pattern Panel. For example, if the user increases the value of the Width spinner in Pattern Panel by one, then the contents of the Pattern String text field should update appropriately, and the graphics in the Game Board Panel and all the Generation Panels should become one column wider; similarly, if the user clicks their mouse whilst over a dead cell in the Game Board Panel, then the cell should become a live cell and the contents of Pattern String and Rows in the Pattern Panel should be updated automatically, together with the contents of the Generation Panel. The Generation Panel itself is not editable.

The editor should not read or write to files on the hard disk. Instead, when the application is initially started, the Game Board Panel should display an 8-by-8 world with no live cells; the name of the author and the name of the pattern should be represented by the empty string. All other fields in Pattern Panel and Generation Panel should be initialised with values consistent with this configuration (for example, the Width spinner should contain the value "8").

One common usage of the editor is to allow the user to paste in a string describing an existing world into the Pattern String text field. The editor should then update the rest of the fields in the Pattern Panel, together with the contents of the Game Board Panel and the Generations Panel, using the information contained in the Pattern String.

## Hints 'n' Tips

Adding the following anonymous inner class to the constructor of `GamePanel` you wrote for Tick 8 will print out the position of the mouse relative to the top left corner of the game board to the command line:

```
addMouseListener(new MouseAdapter(){
    public void mousePressed(MouseEvent me) {
        java.awt.Point p = me.getPoint();
        System.out.println("X="+p.x+" Y="+p.y);
    }
});
```

You can create a new instance of `JSpinner` which has an initial value of 8, a minimum value of 8, a maximum value of 1024, and adjusts the contents of the spinner by 1 each time the user presses an arrow button as follows:

```
JSpinner spin = new JSpinner(new SpinnerNumberModel(8,8,1024,1));
```

Given a class called `FilmStripPanel` which extends `JPanel`, the following method will create the Generations Panel with only a horizontal scroll bar:

```
private JComponent createFilmStripPanel() {
    JPanel holder = new JPanel();
    addBorder(holder, "Future Generations"); //method as defined in tick8.GuiLife
    filmStripPanel = new FilmStripPanel();
    holder.add(filmStripPanel);
    JScrollPane sp = new JScrollPane(holder);
    sp.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_NEVER);
    sp.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
    return sp;
}
```

You will undoubtedly find it useful to reuse or modify classes you wrote for Tick 8, however please make sure that all your code is inside the package `uk.ac.cam.crsid.tick8star`.

## Tick submission

Once you are happy your application works as requested in this exercise, please package up your program into `crsid-tick8star.jar` and email it to `ticks1a-java@cl.cam.ac.uk`. Your application should start with the following invocation:

```
crsid@machine:~> java -cp world.jar:crsid-tick8star.jar \
uk.ac.cam.crsid.tick8star.Editor
```

It is not possible to perform extensive automated testing on your submission given the scant description provided, therefore you should not rely on the automated testing framework to check your program for correctness! Please test your program thoroughly before submission and do not include the contents of `world.jar` inside your jar file.