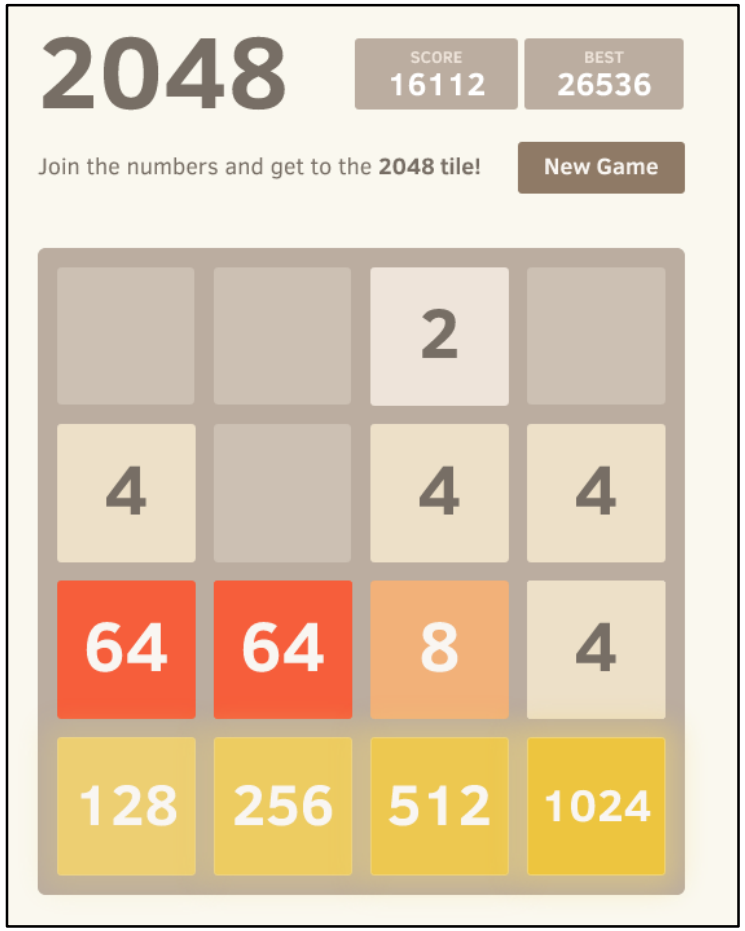
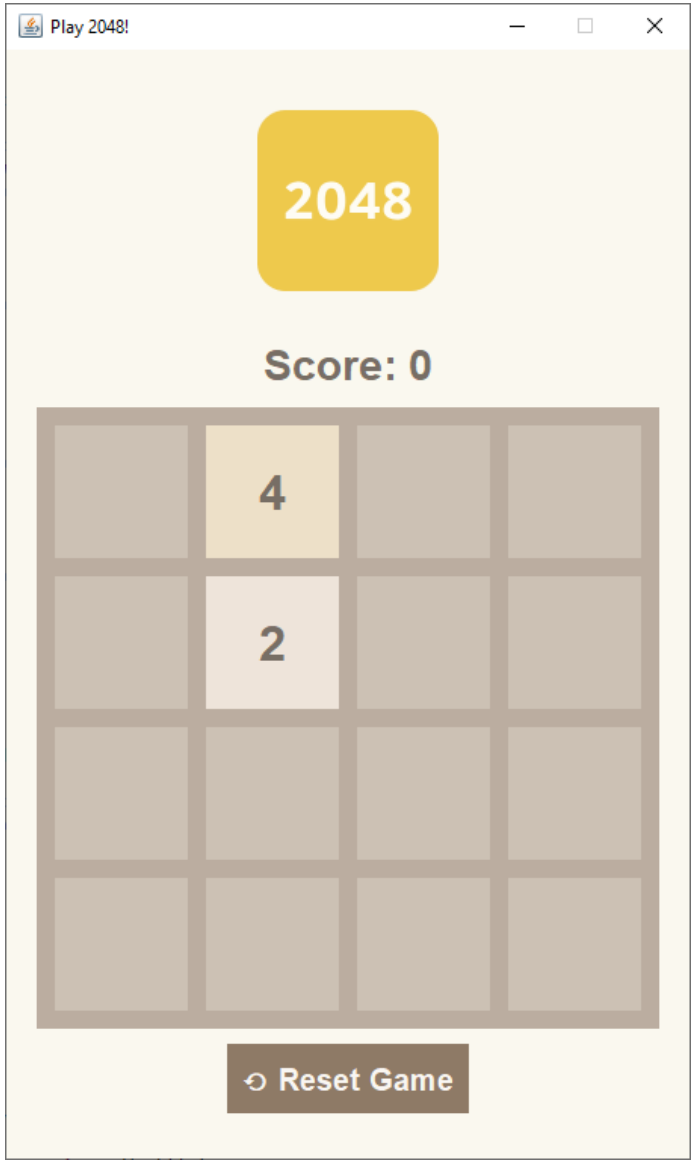
2048 Game Design with Java Swing

Aaron Smith

November 23, 2019

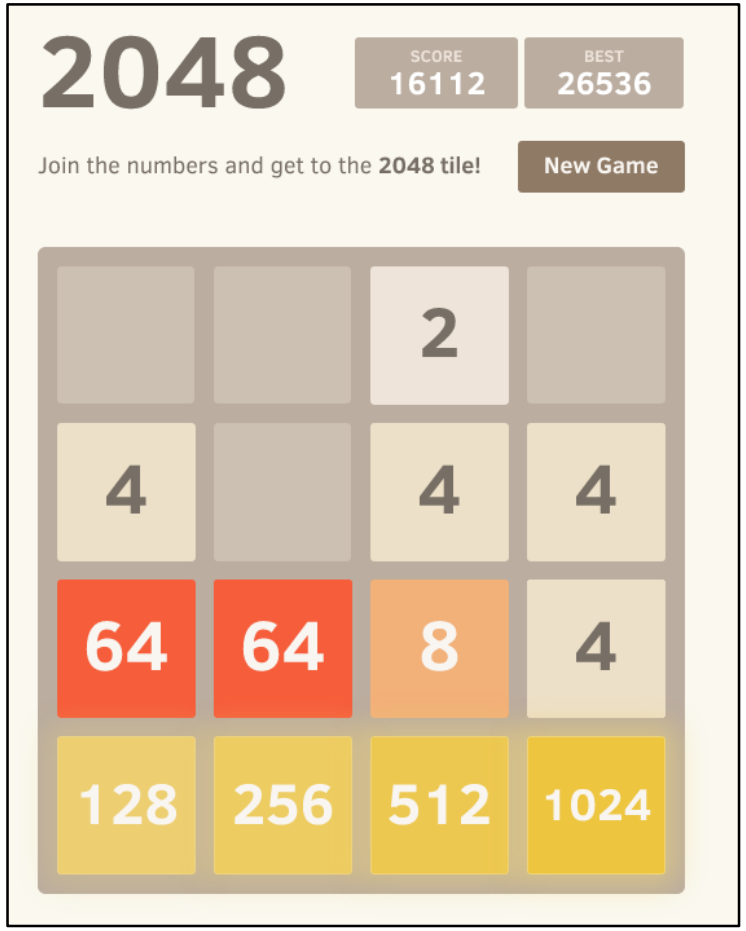


play2048.co



Our version!

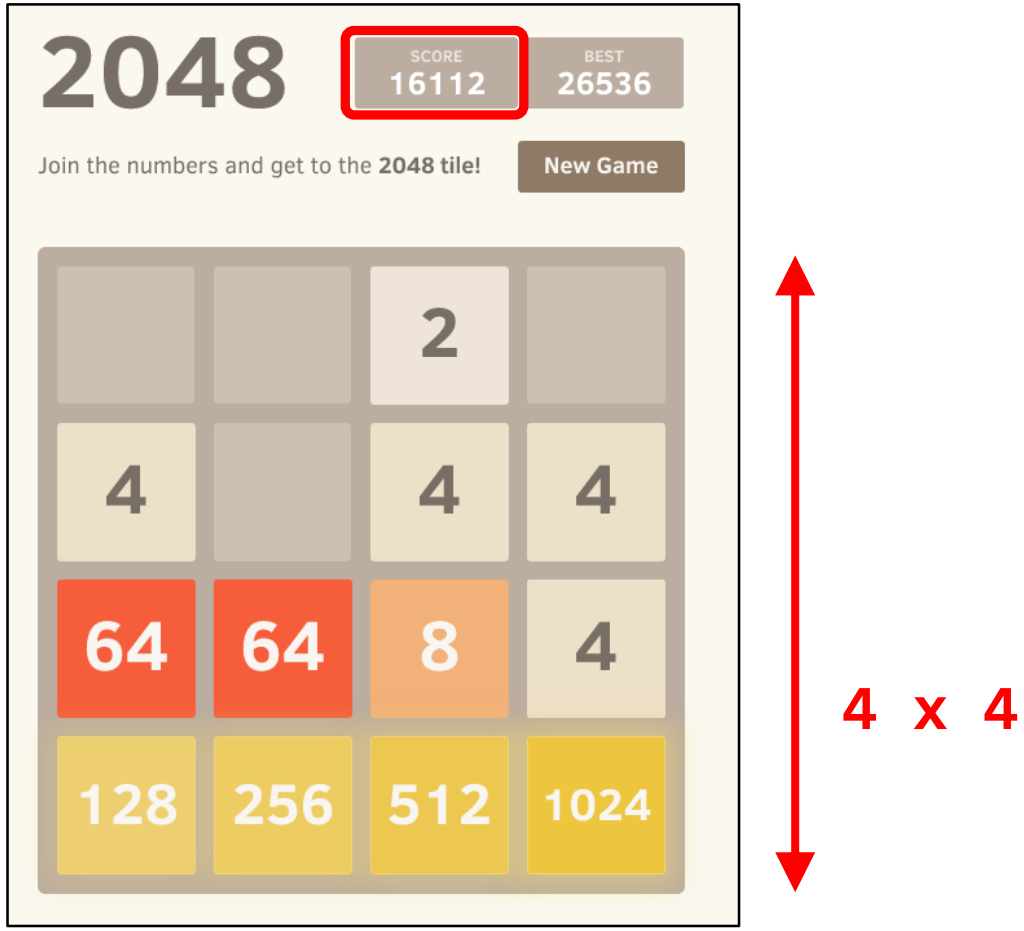
How should we represent the board?




```
private int[][] board;
```

```
[
    [   0,    0,    2,    0],
    [   4,    0,    4,    4],
    [  64,   64,    8,    4],
    [128, 256, 512, 1024]
]
```

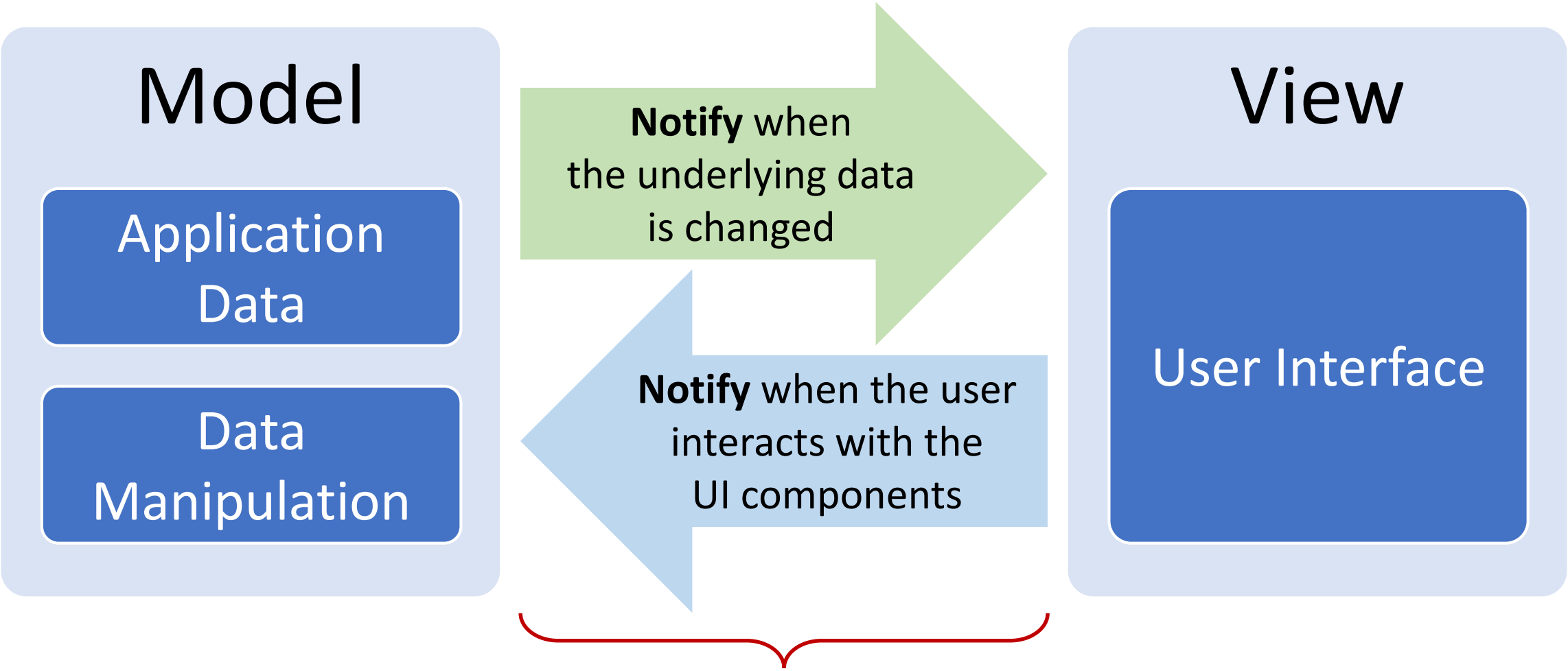
What else do we need to represent?



4 x 4

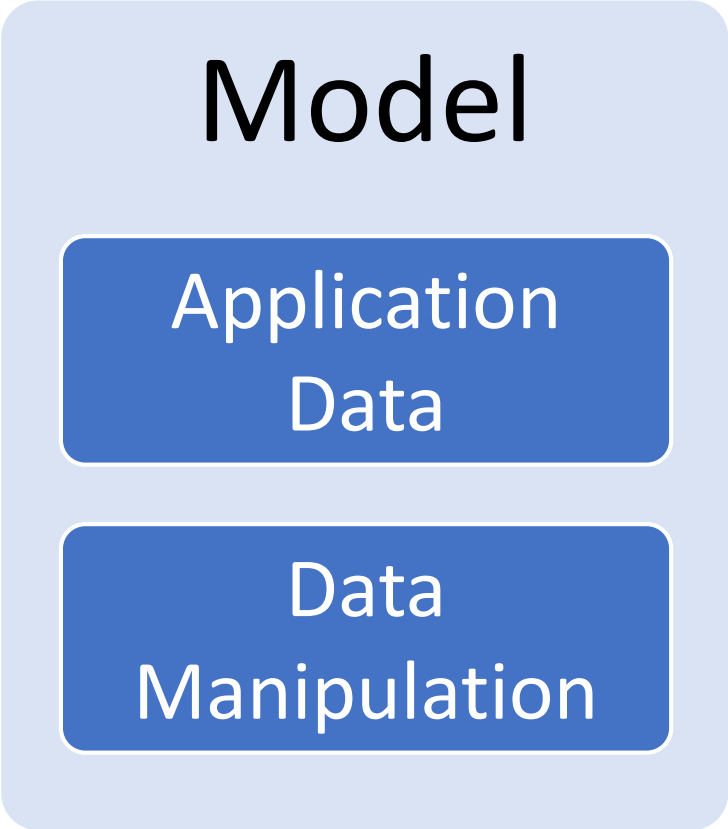
```
private int[][] board;

private int score;

private int size;
```

Model-View Design Pattern



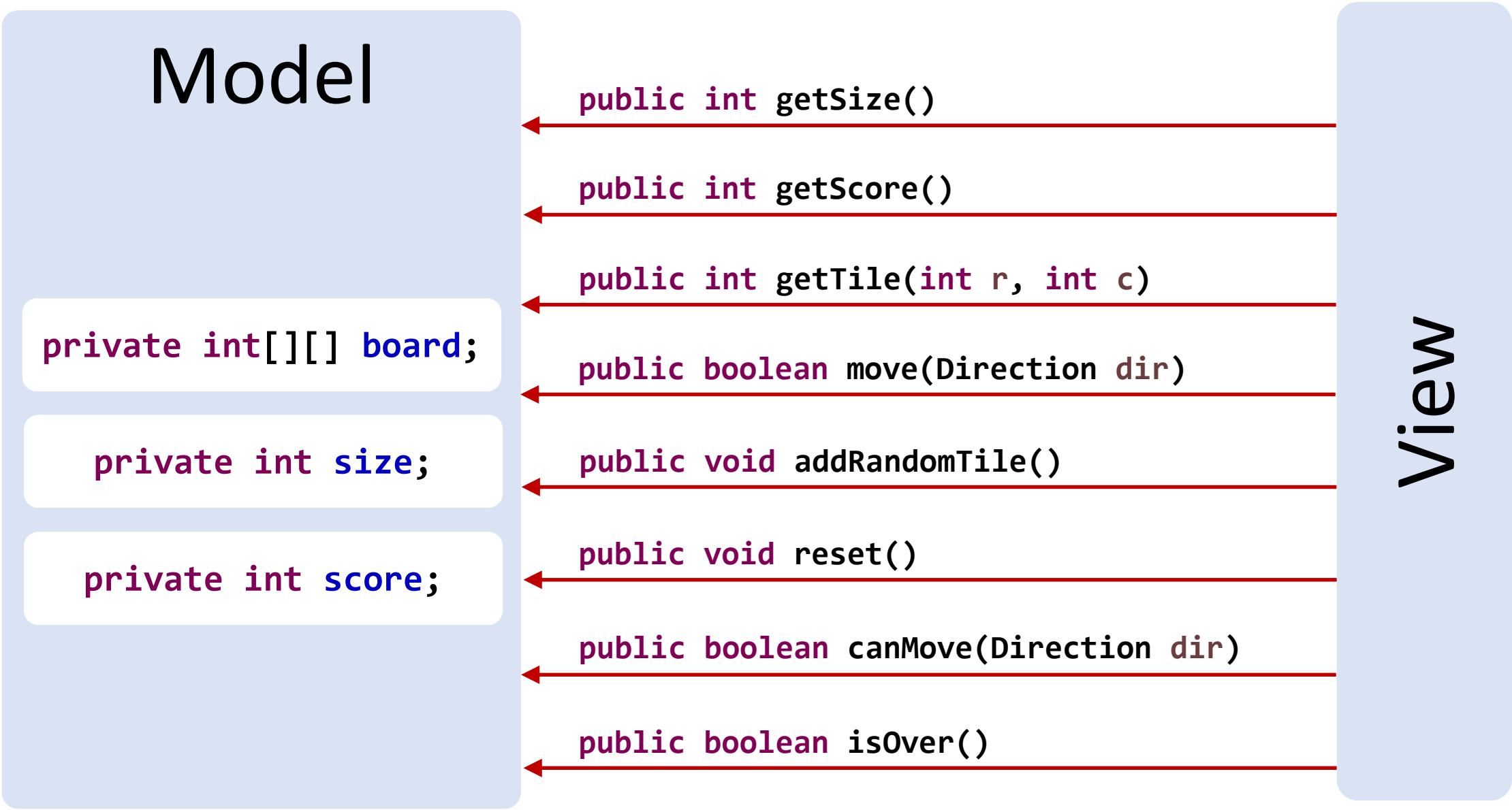
This is the **observer** design pattern

The model **exposes methods** to the data



The purpose of the model is to…

1. Store the data
2. Provide **methods** for data **access** and **manipulation**


Model
private int[][] board;
private int size;
private int score;
public int getSize()
public int getScore()
public int getTile(int r, int c)
public boolean move(Direction dir)
public void addRandomTile()
public void reset()
public boolean canMove(Direction dir)
public boolean isOver()
View

Workshop Resources

github.com/onsmith/comp401-2048

1. Eclipse Java project code
   - "**master**" branch: starter code
   - "**solution**" branch: completed code
2. Slides

Swing Documentation

Lists Swing's **interfaces** and **classes**

Provides descriptions of each

More details accessible by clicking the links

javax.swing (Java Platform SE 7)

docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

Overview | Package | Class | Use | Tree | Deprecated | Index | Help

Java™ Platform Standard Ed. 7

Prev Package | Next Package | Frames | No Frames | All Classes

Package javax.swing

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

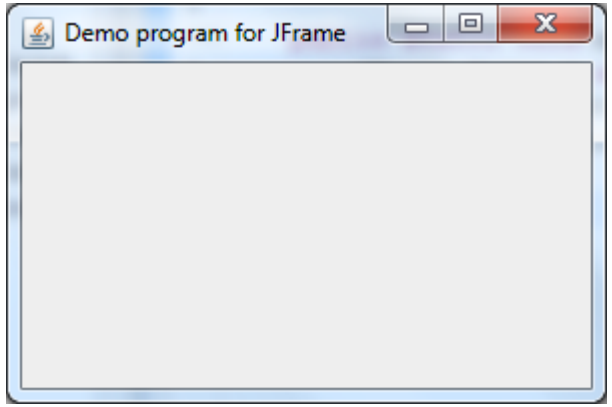
See: Description

Interface Summary

Interface	Description
Action	The `Action` interface provides a useful extension to the `ActionListener` interface in cases where the same functionality may be accessed by several controls.
BoundedRangeModel	Defines the data model used by components like `Sliders` and `ProgressBars`.
ButtonModel	State model for buttons.
CellEditor	This interface defines the methods any general editor should be able to implement.
ComboBoxEditor	The editor component used for JComboBox components.
ComboBoxModel<E>	A data model for a combo box.
DesktopManager	DesktopManager objects are owned by a JDesktopPane object.
Icon	A small fixed size picture, typically used to decorate components.
JComboBox.KeySelectionManager	The interface that defines a `KeySelectionManager`.
ListCellRenderer<E>	Identifies components that can be used as "rubber stamps" to paint the cells in a JList.
ListModel<E>	This interface defines the methods components like JList use to get the value of each cell in a list and the length of the list.
ListSelectionModel	This interface represents the current state of the selection for any of the components that display a list of values with stable indices.

JFrame

Represents a window



JFrame

```
my_frame.setTitle("Window title");
```

– **Sets the title of the window (shown on tab)**

```
my_frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

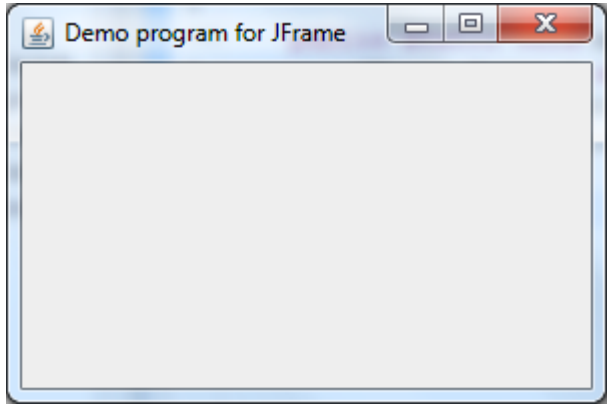
– **Makes the window close when you press X**

```
my_frame.setResizable(false);
```

– **Makes the window so you can’t resize it**

JPanel

Represents a grouping of multiple Components



JPanel

```
my_panel.add(              );
```

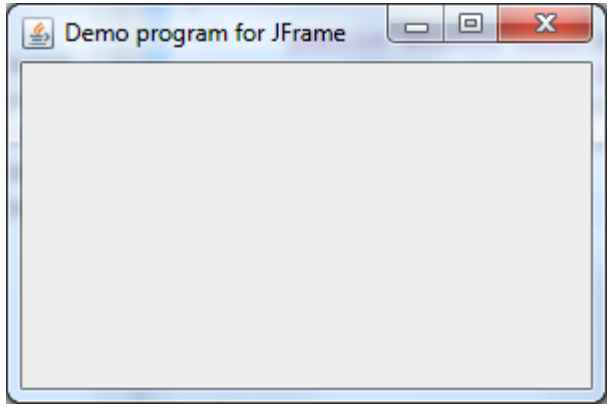
– **Adds a component to the group**

```
my_panel.setLayout(              );
```

– **Specifies how the components should be arranged**

JPanel

Represents a grouping of multiple Components



JPanel

```
my_panel.removeAll(          );
```

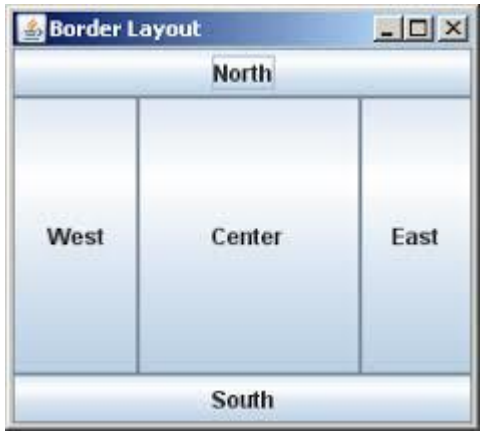
– **Removes all components from the group**

```
my_panel.revalidate(          );
```

– **Refreshes the component after you change it**

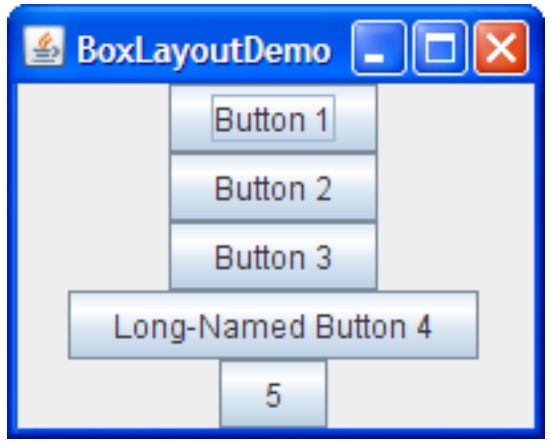
This will come in handy when we update the UI components!

JPanel Layouts



BorderLayout

GridLayout

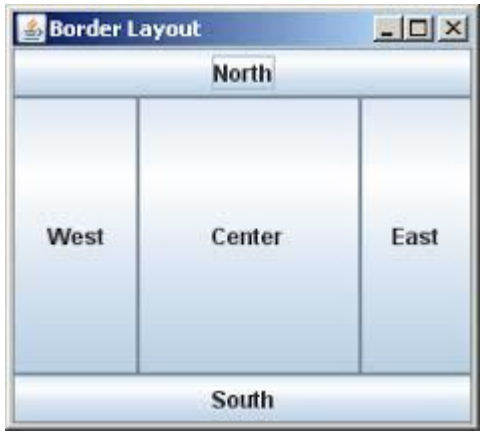


BoxLayout

```
my_panel.setLayout(          );
```

What goes here?

BorderLayout




```
new BorderLayout()
```

- Creates a new `BorderLayout` object
- Pass this into `my_panel.setLayout()`

SOUTH, NORTH, WEST, CENTER, EAST

```
my_panel.add(my_component, BorderLayout.SOUTH);
```

Extra argument required when adding components to the panel

GridLayout

Rows Columns

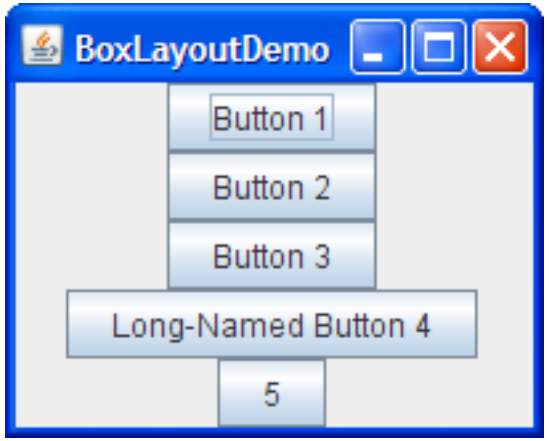
```
new GridLayout(3, 3)
```

- Creates a new `GridLayout` object
- Pass this into `my_panel.setLayout()`

Either "**rows**" or "**columns**" can be **0**, indicating that it should be automatically calculated—but not both!

Components will be added left-to-right, top-to-bottom

BoxLayout



The JPanel object

Dimension for insertion

```
new BoxLayout(my_panel, BoxLayout.Y_AXIS)
```

- **Creates a new `BoxLayout` object**
- **Pass this into `my_panel.setLayout()`**

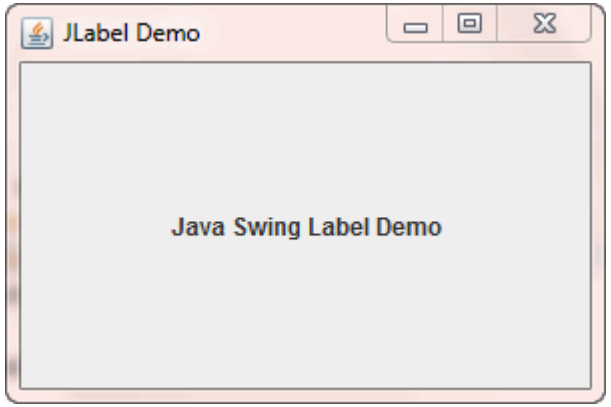
Y_AXIS aligns objects vertically; **X_AXIS** aligns them horizontally

```
my_component.setAlignmentX(Component.CENTER_ALIGNMENT);
my_panel.add(my_component);
```

- **Must call `setAlignmentX` or `setAlignmentY` on every component added to the `JPanel`**

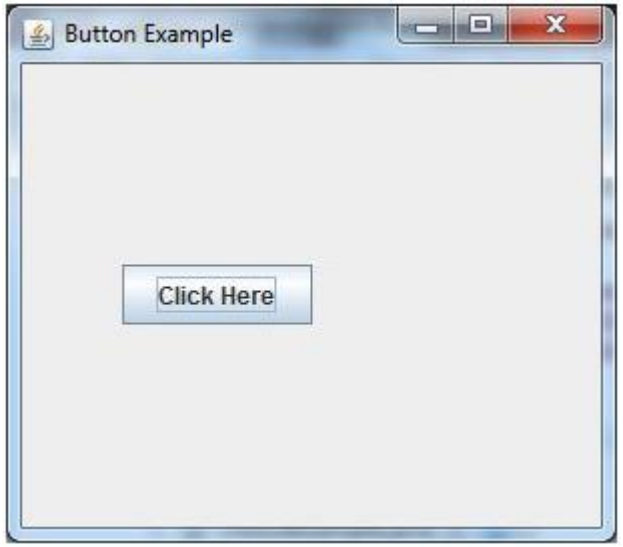
Also supported:
RIGHT_ALIGNMENT
and **LEFT_ALIGNMENT**

Components



JLabel

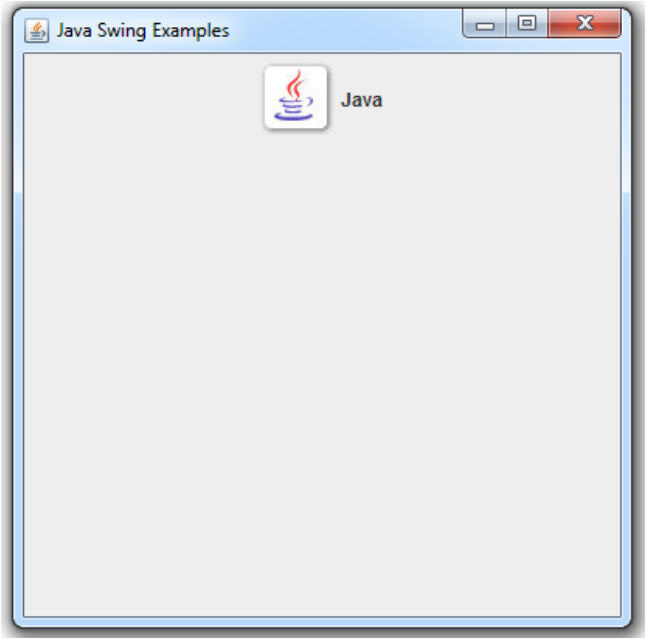
`setText()`



JButton

`addActionListener()`

`setActionCommand()`



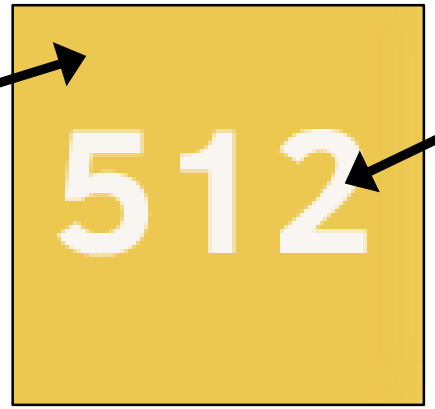
ImageIcon

Changing the color of a component

What goes in here?

```
my_component.setForeground(                    );

my_component.setBackground(                    );
```

Background color (yellow)



Foreground color (white)

For `JLabels`, the text color is the foreground color!

Colors

Red component
(0-255)

Blue component
(0-255)

```
new Color(187, 173, 160)
```

Green component
(0-255)

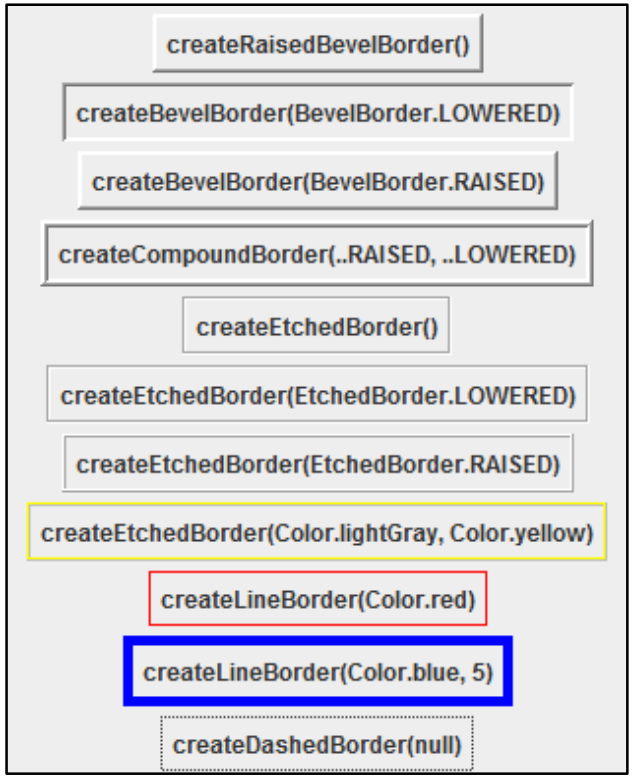
Tools to help you pick colors:

- coolors.co
- colorhunt.co
- Google “color picker”
- The one we made in COMP 401!

Adding a border to a component

```
my_component.setBorder(                    );
```

What goes in here?



There are lots of border choices/styles to choose from! Google "**Java BorderFactory**" for a list!

EmptyBorder

Adds a border with the same color as the **background**

```
my_component.setBorder(                    );
```

Size of
top, left, bottom, right

```
BorderFactory.createEmptyBorder(20, 20, 20, 20)
```

LineBorder

Lets you specify the color of the border

```
my_component.setBorder(                    );
```

```
BorderFactory.createLineBorder(
    new Color(187, 173, 160), // border color
    6                         // border size
)
```

CompoundBorder

Lets you combine two borders!

```
my_component.setBorder(                    );
```

```
BorderFactory.createCompoundBorder(
    BorderFactory.createLineBorder( … ),
    BorderFactory.createEmptyBorder( … )
)
```

Changing the font

What goes in here?

```
my_component.setFont(                    );
```

```
new Font("Dialog", Font.BOLD, 28)
```

A Java Font object

Changing the font

What goes in here?

```
my_component.setFont(                    );
```

```
new Font("Dialog", Font.BOLD, 28)
```

font family

(Google search “Java fonts”)

font style

(BOLD, ITALIC, PLAIN)

font size

Specifying the size of a component

```
my_component.setMinimumSize(new Dimension(100, 100));
my_component.setPreferredSize(new Dimension(100, 100));
my_component.setMaximumSize(new Dimension(100, 100));
```

width

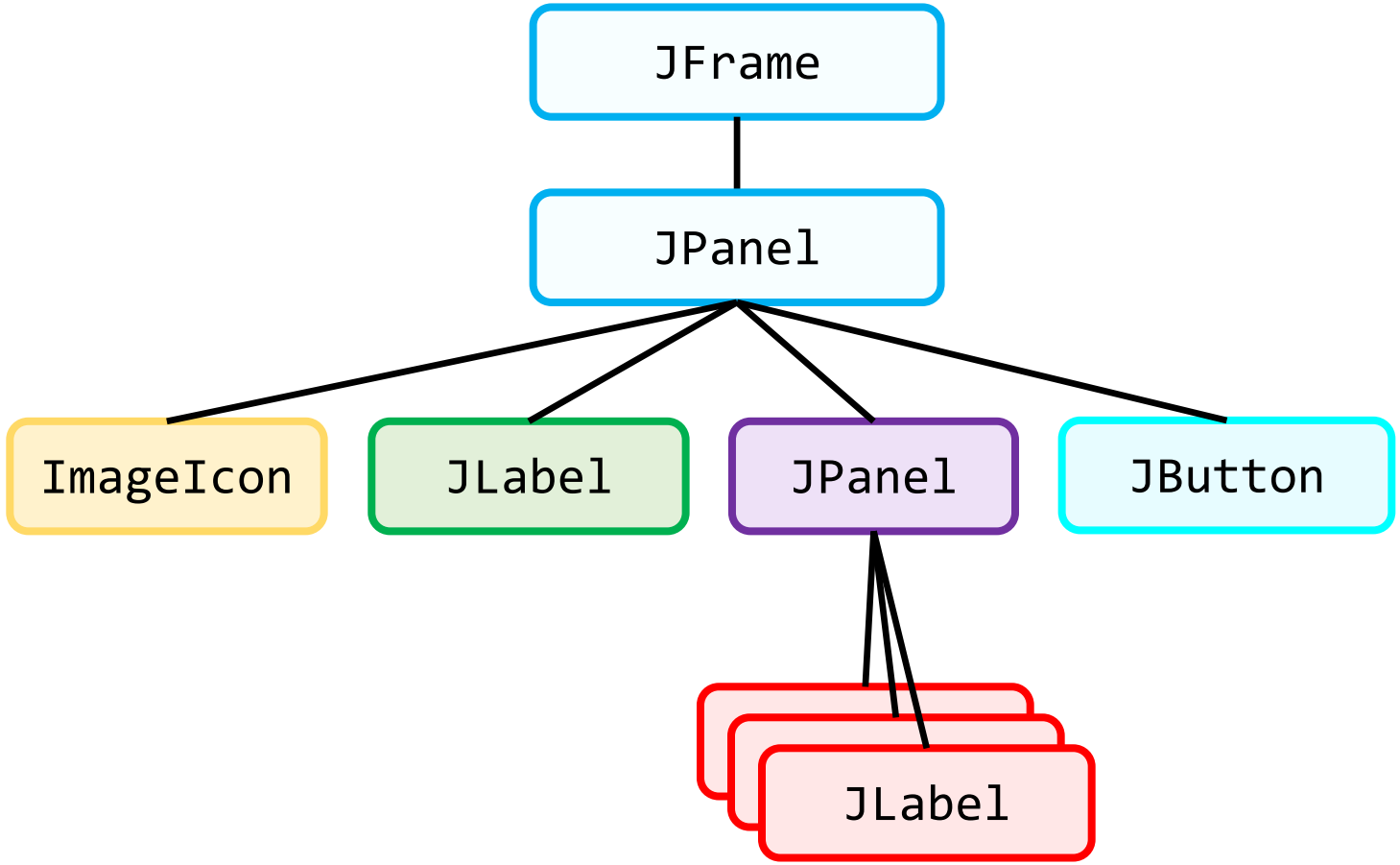
height

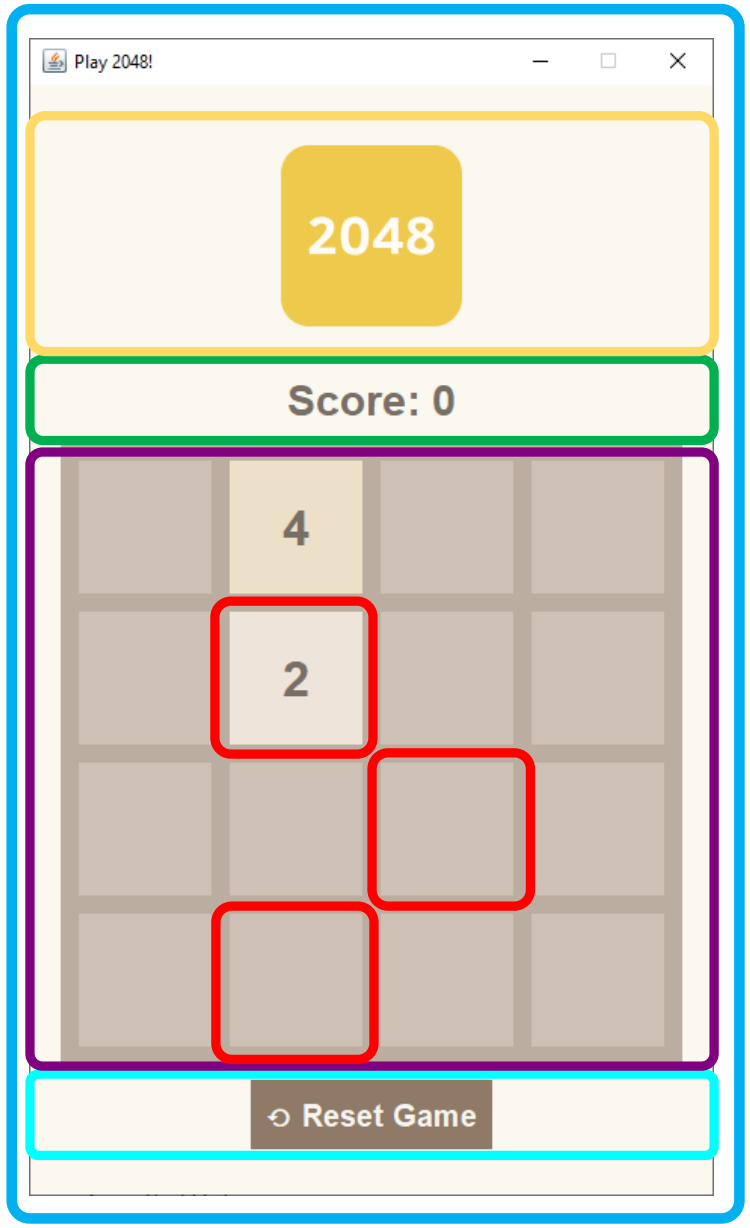
Text alignment

```
my_jlabel.setHorizontalAlignment(SwingConstants.CENTER);
my_jlabel.setVerticalAlignment(SwingConstants.CENTER);
```

Can set alignment to LEFT, CENTER, or RIGHT

UI Organization





User Events

When a keyboard arrow key is pressed…

1. Figure out which arrow key was pressed
2. Call `model.move(DIRECTION)`
3. Repaint UI

When reset is clicked…

1. Call `model.reset()`
2. Repaint UI