PSA 6: 2048 GUI

Read all instructions in this document before starting any coding!

In this assignment, we wrap up the 2048 project. In the third assignment, you implemented the backend of the game. Now, we will implement the front-end to add some pretty visual effects.

Get started early! The GUI implementation can be quite tricky.

Helpful Information:

- Online Communication: Using Piazza, Opening Regrade Requests
- Getting Help from Tutor, TA, and Professor's Hours
    - Lab and Office Hours (Always refer to this calendar before posting.)
- Academic Integrity: What You Can and Can't Do in CSE 8B
- Setting Up the PSA
- How to Use Vim
- Style Guidelines
- Submitting on Vocareum and Grading Guidelines

Table of Contents:

Part 1: GUI 2048 [90 Points]

After completing this assignment you will have a fully functioning, graphical 2048 game which you can show off to all of your friends and family. You may even want to show a recruiter or two. You will be graded on look and feel of your final game. For this assignment you will be utilizing the 2048 backend which you completed in PSA3 (Board.java). We encourage you to fix any bugs you may still have so that you can have a bug free 2048 game by the end of this PSA.

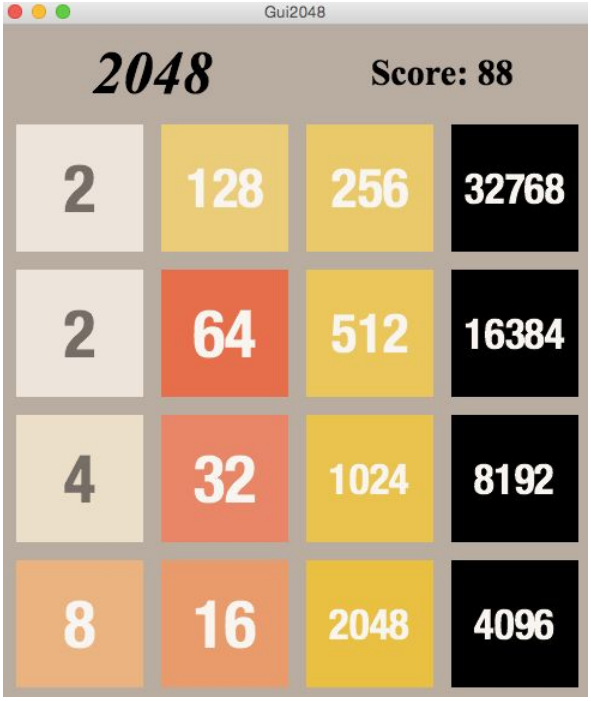
Note: We will be grading your GUI using your Board.java. If there are any errors in your Board.java that impede us from grading your GUI, then you will lose points.

One thing you'll have to make sure you do is place the call to your board's saveBoard() method inside of a try catch statement (since it currently throws an IOException).

```
try {
        board.saveBoard(outputFile);
} catch (IOException e) {
        System.out.println("saveBoard threw an Exception");
}
```

Gui2048.java

You will implement the GUI and the event handling code in this file. We have provided a few methods for processing the command line arguments and creating an instance of the Board class.



Constants2048.java

We have provided constants for the Colors for each different tile type based on the original 2048 game. *You are free to use the colors we provided or you can be different and creative and come up with your own color scheme.* It cannot be the default GUI scheme that has no modifications to it.

GUI (50 Points)

Your GUI is required to show the score, the name of the game (2048), the tiles with values (and colors which change with values). It is also required to display "Game Over" when the game ends.

For the GUI portion of the assignment we recommend you refer to Chapter 14 in your textbook. It has wonderful examples of how to add gui components to a Scene. There are a few classes which you should familiarize yourself with and which you will need to use to complete the assignment: Pane (more specifically GridPane and/or BorderPane), Text or Label, Rectangle, Color, and Font.

Pane

A Pane is used as a container for all of the objects that are in a window. It is essentially a way to organize the various components that you will be displaying. We recommend that you use a GridPane, since it is naturally suited to our needs as it organizes objects in a grid like pattern. You can use the **add(Node, x, y)** method to add a component like a Rectangle to the (x,y) location on the grid. (x represents the column and y represents the row of the grid). All other questions regarding Panes and GridPanes can be answered by refering to Section 14.10 in your text and the following links: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/Pane.html and http://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/GridPane.html.

Text/Label

The Text and Label components can be used to display text in the GUI. You can use these classes for displaying the values of each of the tiles as well as the score. You can use the Font class to set the font of a Text or Label object by doing the following:

```
textExample.setFont(Font.font("Times New Roman", FontWeight.BOLD, 30));
```

This will change the font of the Text element **textExample** to be Times New Roman, Bold and font size 30. For information on Text, Label, and Font can be found online or in Sections 14.8 and 14.11.1.

Java Documentation:
https://docs.oracle.com/javafx/2/api/javafx/scene/text/Font.html

Rectangle

We recommend that you use the Rectangle class from the JavaFX library to make the tiles for Board. Section 14.11.3 discusses how to use the Rectangle class.

Color

You will be making extensive use of the Color class. You can use the setFill(Color c) method for Text and Rectangle objects to change their color. Your tiles must change color based on the value of the tile. There is more information on the color class in section 14.7 of your textbook.

Example

Let's take a look at an example to get started so open your book to section 14.10.2 on GridPanes. We'll be working in the start method to initialize the GUI. We first need to create a pane that we can add elements to, while we're at it let's do a bit of formatting:

```
GridPane pane = new GridPane();
pane.setAlignment(Pos.CENTER);
pane.setPadding(new Insets(11.5,12.5,13.5,14.5));
pane.setHgap(5.5);
pane.setVgap(5.5);
```

Now we have a simple gridpane set up and ready for use. Feel free to mess with the values to get the spacing and formatting better. You can also use the **setStyle()** method to color the background.

```
pane.setStyle("-fx-background-color: rgb(187, 173, 160)");
```

Java Documentation:
https://docs.oracle.com/javafx/2/css_tutorial/jfxpub-css_tutorial.htm
The next step is to add our newly constructed pane to a scene, and then the scene to the stage so that we can display our pane.

```
Scene scene = new Scene(pane);
primaryStage.setTitle("Gui2048");
primaryStage.setScene(scene);
primaryStage.show();
```

Now if we run this we'll have a window with nothing but a colored background (you may have to resize the window since it's empty):

Now that we know how to get a blank window to show up let's try and create a Tile that will be used to represent a section of the 2048 grid. To do this we'll need to add a Rectangle (to represent the tile) and

a Text element (to show the value). We'll be using a few instance methods to format the rectangle. We can use the **setHeight()** and **setWidth()** methods to specify the dimensions and the **setFill()** method to set the color. For the Text element we can use **setText()** to change the text of the object and **setFill()** to change the color of the text. If we want to change the size of the font then we'll need to use the **setFont()** method.
Java Documentation:
https://docs.oracle.com/javafx/2/api/javafx/scene/text/Font.html

```
Rectangle aLonelySquare = new Rectangle();
aLonelySquare.setWidth(100);
aLonelySquare.setHeight(100);
aLonelySquare.setFill(Color.BLACK);

Text someText = new Text();
someText.setText("2048");
someText.setFont(Font.font("Times New Roman", FontWeight.BOLD, 30));
someText.setFill(Color.WHITE);
```

Now that we have our Text and Rectangle we need to add them to the Grid so that it can be displayed.

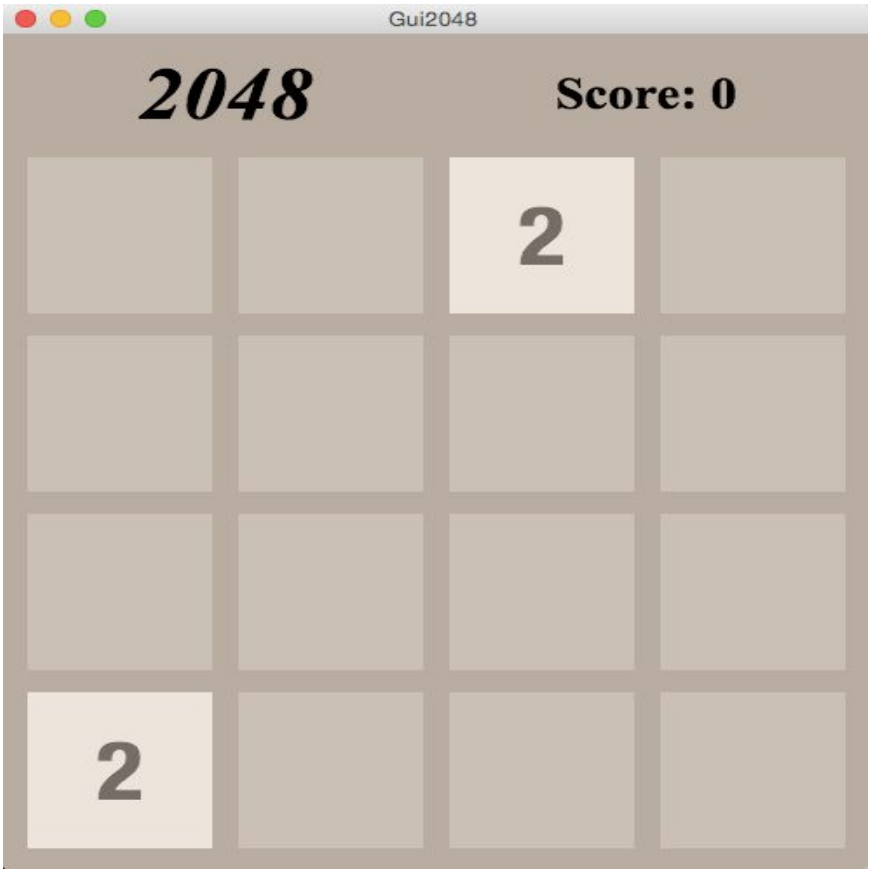
```
pane.add(aLonelySquare, 0, 0);
pane.add(someText, 0, 0);
```

Hmm that doesn't quite look right. We want the text to be centered in the middle of the Rectangle. Let's use the static method setHalignment() to do this.

```
GridPane.setHalignment(someText, HPos.CENTER);
```

There we go, that looks much better. Now you'll need to go and add the rest of the tiles to the board. You may want to save the references to the Rectangle and Text objects you create because you'll need to update the text and color of them later.

Here's what a brand new game would look like:



Events (40 Points)

You will need to be able to handle keyboard events that occur while focused on your Gui Window. Based on the key pressed you will need to either move (if one of the arrow keys is pressed) or **save the board if the "s" key is pressed**. Here is the relevant documentation on key code:
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/input/KeyEvent.html
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/input/KeyCode.html

In addition to either moving or saving, you will need to print out **(to the terminal)** that you are saving by printing:

> **Saving Board to <filename>**

When an arrow key is pressed **AND the move was successful** then you will print:

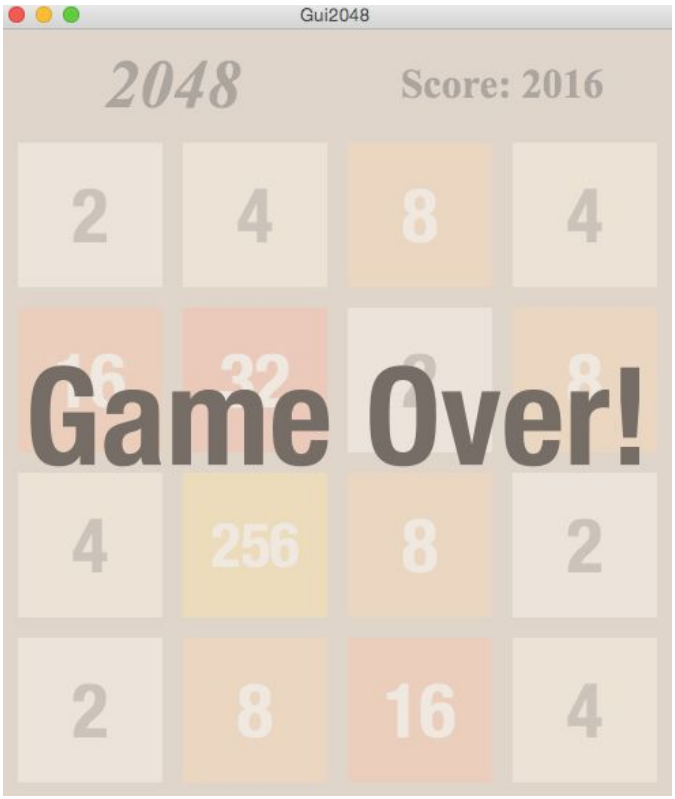
> **Moving <Direction>**

For example, if the right key was pressed but none of the tiles could move right then "Moving Right" would not print.

Note:
Please note that, in this assignment we are using the **ARROW KEYS** for moving the tiles on the board. This is different from PSA3 where you used the **keyboard keys (w, a, s, d)** to indicate movement. To implement this, you do not have to make any change in your PSA3 code.

When a move was successful you will need to perform the move on the game board and then update the GUI components to reflect the resultant board and score. This means updating the Score text that is displayed as well as changing the text on the tiles with their new values, and changing the colors on each tile.

After each valid move you will need to check to see if the game is over or not. If the game is over then you will need to place a semi transparent overlay over the whole window with the Text "Game Over" in the center of the window. Once the game is over, the Game Over! panel should remain on the GUI, instead of disappearing if another key is pressed, and remain so until the user closes the window.



To be able to handle keypresses you'll need to create a private inner class that implements the **EventHandler<KeyEvent>** interface.

```
private class myKeyHandler implements EventHandler<KeyEvent>
{
        @Override
        public void handle(KeyEvent e)
        {/* KeyEvent Processing Code Goes Here */}
}
```

You can then add an instantiation of your key handler to the scene (back up in the start method) so that whenever you press a key, while focused on the scene, it will be processed by your key handler.

```
scene.setOnKeyPressed(new myKeyHandler());
```

Part 2: README.md & Short Response [10 Points]

Remember that the audience of the README description is anyone who knows no programming or computer science. This means use absolutely no Java or CSE terms but high-level terms are fine. Ensure you discuss how you tested the program to ensure it was working as expected.

Short Response

Unix/Linux Questions:

1) Suppose you are currently inside a directory and in there you want to make a new directory called fooDir. And inside fooDir, you want another directory called barDir. Using only a single mkdir command, how can you create a directory called fooDir with a directory called barDir inside it?
2) Give an example of how you would use a wildcard character (you can use it with whichever command you want like rm, cat, ls, cp, etc). Describe what happens after you use it.
3) How can you run gvim through the command line to open all Java source code files in the current directory, each file in its own tab?

Java Question:

4) What does the keyword static mean in regards to methods? Provide an example use of a static method.
5) A high school student is trying to write a Java program that will draw different shapes and in different possible colors. To do this, she has written just one Java class called ShapeDrawer, which contains all the necessary methods like drawRedCircle(), drawBlueCircle(), drawYellowSquare(), drawGreenSquare(), and so on. Using object-oriented terminology, describe how you can help the student improve her design.

Style Guidelines (Link)

Refer to the website with a more complete guideline and set of examples of what your style should look like. There are ten possible points that can be deducted.

Extra Credit [+10 Points]

Extra credit has two parts, You can earn up to maximum of 10 points (10%) extra credit.

Visual Additions

Part 1: Resizeable Grid (+5 points)

Extra credit will be given for making your GUI resizeable based on the size of the grid (for grid sizes greater than 4) and by making your GUI resizeable based on the window size. The first requirement means if your window size stays the same, and your grid is larger (e.g. changed from 4 by 4 to 8 by 8), your tile size will shrink.

Part 2: Resizeable Window (+5 Points)

The second requirement means that if your window is made bigger, the tiles will be bigger or if you make the window smaller the tiles will get smaller. Think about using Bindings. **Note: The font size doesn't have to change since this can't easily be done using bindings.**

Submitting the Assignment (Link)

Note: Submitted code that does not compile will receive a zero without any chance of partial credit. Make sure your code compiles before submitting!

Submission Files (Make sure your submission contains all of the files and that they work on the ieng6 lab machines!)

- Gui2048.java
- README.md
- Board.java
- Constants2048.java

Maximum Score Possible: 110/100 Points