Holden Dalton, Shay Snyder, & Hannah Taylor

Honors 1260 – Spring 2020

**SudokuWars Design Document**

Problem: 1260 Honors Capstone Project

### **List of Inputs, Outputs, and Processing Required**

Inputs

* + Button clicks from GUI
  + Menu item choices
  + Choose difficulty
  + Player name

Outputs

* + GUI window
    - Sudoku board
    - Move options
    - Menu bar
  + Intro screen
  + Game screen

Processing

* + User click
    - Open screen based on menu choice
    - Open/import board based on difficulty choice and random number
    - Add number to board based on button click
  + Import board/ read txt file
  + Determine if move is valid
  + Submit the move to the board
  + Create clock to keep time of how long the game takes
  + Run GUI windows

### **Identification of Classes and Their Responsibilities**

**Class name: BGPanel**

Responsibilities

Create content /base panel for the window

Test Case

Pull image from file, set size to window

\*Override method in JPanel to allow add of different background

**Class name: Board**

Responsibilities

Control sudoku playing surface

Test Case

* + If error reading txt file, exit game
  + Loop through file to initialize board
  + Comparing times of games
    - Return values similar to the compareTo()
    - Since lesser time means a better time positive and negative values need to be reversed

**Class name: Difficulty**

Responsibilities

Enum of different levels of difficulty

**Class name: Element**

Responsibilities

Manage each element of the board/ 9x9 grid

Test Case

* + Check if moves made are correct
  + Return boolean for correct value

**Class name: GameGridElement**

Responsibilities

Create JButtons to populate the 9x9 grid

**Class name: GameGridPanel**

Responsibilities

Build the 9x9 grid of GridElements

Test Case

* + Don’t allow the user to resize the screen to keep proportions

**Class name: GameLowerPanel**

Responsibilities

Create lower panel of game window

**Class name: GameLowerPanelElement**

Responsibilities

Create elements/buttons to go in lower panel

**Class name: GameMenu**

Responsibilities

Manage ‘game’ menu int the MenuBar

**Class name: GamePanel**

Responsibilities

Manage playing surface panel

Test Case

\*add items to panel, should not be any issues

**Class name: GamePanelElement**

Responsibilities

Create button elements in GamePanel

**Class name: HelpMenuItem**

Responsibilities

Allows user to learn about the game and how to play sudoku

Test Case

* + If click help button in menu, display the rules of the game

**Class name: HelpPanel**

Responsibilities

Build panel to hold items that teach user how to play

**Class name: IntroLowerPanel**

Responsibilities

Initialize lower panel of intro screen

**Class name: IntroPanel**

Responsibilities

Manage functions of intro screen

Test Case

* + If start button clicked, ask for difficulty
  + Else exit the game

**Class name: IntroUpperPanel**

Responsibilities

Manage upper panel of intro screen

**Class name: Leaderboard**

Responsibilities

Contain and manage all previous boards

Test Case

* + Handle any I/O file exceptions

**Class name: LeaderboardMenuItem**

Responsibilities

Create the menu item to access the leaderboard

**Class name: LeaderboardPanel**

Responsibilities

Panel to visualize and hold the leaderboard

**Class name: MenuBar**

Responsibilities

Manage game’s menu bar

**Class name: NewGameMenuItem**

Responsibilities

Allow user to start a new game

Test Case

* + Confirm new game
  + Ask for difficulty
  + If close button hit generate default board

**Class name: Piece**

Responsibilities

Represent possible values in a sudoku cell

**Class name: Settings**

Responsibilities

Manage all settings of SudokuWars

**Class name: Sudoku**

Responsibilities

Control the sudoku game/ solving of the puzzle operations

Test Case

* + Interact with GUI to get inputs from buttons

**Class name: SudokuDriver**

Responsibilities

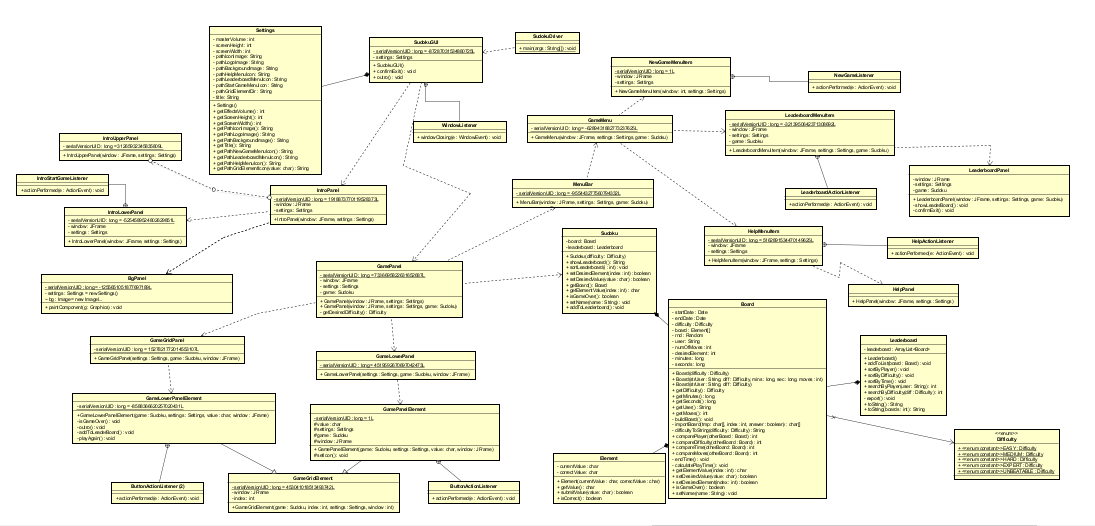
Driver for the sudoku game, instantiates the GUI

**Class name: SudokuGUI**

Responsibilities

Initialize the graphical windows of the application/ game

### **UML Class Diagram**



### **Algorithms**

**Class:** BgPanel.java

**Method:** paintComponent(Graphics g)

**Description:**

Start

Do: draw image of the background

End

**Class:** Board.java

**Method:** Board(String strUser, Difficulty diff,long min,long sec,int moves)

**Description:**

Start

Do: set the appropriate values

End

**Class:** Board.java

**Method:** Board(Difficulty difficulty)

**Description:**

Start

Do: set the appropriate values

End

**Class:** Board.java

**Method:** buildBoard()

**Description:**

Start

Do: get the index

Do: create array of characters

Do: importBoard method

Do: for loop to fill the board

End

**Class:** Board.java

**Method:** importBoard(char[] tmp,int index, boolean answer)

**Description:**

Start

Do: set the type

Do: set the difficulty

Do: get the directory of the boards

Do: create the path

Do: Try to create the file with the path

Do: create scanner

Do: go through the file getting each value and store in array

Do: close the scanner

Do: catch any exceptions

Do: return array tmp

End

Test Case

* Handle any I/O exceptions thrown by import in try/catch

**Class:** Difficulty.java

**Method:** Difficulty

**Description:** holds the enumerated types for difficulty

**Class:** Element.java

**Method:** Element(char currentValue, char correctValue)

**Description:**

Start

Do: set the current value that user has entered in the cell

Do: set the correct value that cell should be

End

**Class:** GameGridElement.java

**Method:** GameGridElement(Sudoku game, int index, Settings settings, JFrame window)

**Description:**

Start

Do: constructor for the parent class to set up game panel

Do: set the value of the window

Do: set the value of the index

Do: add and action listener

End

**Class:** ButtonActionListener (inside GameGridElement.java)

**Method:** actionPerformed(ActionEvent e)

**Description:**

Start

Do: if the cell already has the correct answer

Do: display message telling the user the answer for that cell is already correct

End

Test Case

* Listen for button click, and perform operation

**Class:** GameGridPanel.java

**Method:** GameGridPanel(Settings settings, Sudoku game, JFrame window)

**Description:**

Start

Do: create a JPanel

Do: set the layout of the panel

Do: set the opacity to false

Do: set the border settings

Do: add each element to each cell in the grid that will be the board

End

**Class:** GameLowerPanel.java

**Method:** GameLowerPanel(Settings settings, Sudoku game, JFrame window)

**Description:**

Start

Do: set the layout for the lower panel for user input

Do: set opacity to false

Do: set the preferred size

Do: create an array of values for input by user

Do: add those values to the panel

End

**Class:** GameLowerPanelElement.java

**Method:** GameLowerPanelElement(Sudoku game, Settings settings, char value, JFrame window)

**Description:**

Start

Do: parent class constructor to generate the lower panel

End

**Class:** GameMenu.java

**Method:** GameMenu(JFrame window, Settings settings)

**Description:**

Start

Do: create a new JMenu

Do: set the text

Do: add a new game item to the menu

Do: add a separator to the menu

Do: add a leaderboard item to the menu

Do: add a separator to the menu

Do: add a help menu item to the menu

End

**Class:** GamePanel.java

**Method:** GamePanel(JFrame window, Settings settings)

**Description:**

Start

Do: set the window value

Do: set the settings value

Do: clear the window except for the background

Do: set the new content pane

Do: add the menu bar

Do: get the difficulty the user wishes to play

Do: generate the sudoku game based on difficulty

Do: set opacity to false

Do: set the layout

Do: add the panels to the window

Do: set the layout for the window

Do: center the panel on the frame

Do: validate the game

Do: repaint the frame

End

Test Case

* In order to keep correct visibiity, don’t let user resize the window frame

**Class:** GamePanel.java

**Method:** getDesiredDifficulty()

**Description:**

Start

Do: create an array of difficulty values

Do: get the option from the user

Do: switch statement that will return the user option

End

Test Case

* If no difficulty chosen generate default board to be completed

**Class:** GamePanelElement.java

**Method:** GamePanelElement(Sudoku game, Settings settings, char value, JFrame window)

**Description:**

Start

Do: set the game value

Do: set the settings value

Do: set the value

Do: set the window value

Do: set the opacity to true

Do: set the background color

Do: set the icon using icon method

End

**Class:** GamePanelElement.java

**Method:** setIcon()

**Description:**

Start

Do: set the icon that is gotten from the getPathGridElementIcon method

End

**Class:** HelpMenuItem.java

**Method:** HelpMenuItem(JFrame window, Settings settings)

**Description:**

Start

Do: set the window value

Do: set the settings value

Do: create new JMenuItem

Do: set text

Do: Try to set an icon

Do: catch any exception

Do: add an action listener

End

Test Case

* Exceptions thrown to be handled in try/catch blocks

**Class:** HelpActionListener (inside HelpMenuItem.java)

**Method:** actionPerformed(ActionEvent e)

**Description:**

Start

Do: create new HelpPanel

End

**Class:** HelpPanel.java

**Method:** HelpPanel(JFrame window, Settings settings)

**Description:**

Start

Do: set the window value

Do: set the settings value

Do: create the help panel

End

**Class:** IntroLowerPanel.java

**Method:** IntroLowerPanel(JFrame window, Settings settings)

**Description:**

Start

Do: set the window value

Do: set the settings value

Do: set the opacity to false

Do: add a start button

Do: add the action listener for the start button

Do: add the lower panel to the window

Do: set the border to a border factory setting

End

**Class:** IntroStartGameListener (inside IntroLowerPanel.java)

**Method:** actionPerformed(ActionEvent e)

**Description:**

Start

Do: add the menu bar to the window

End

**Class:** IntroPanel.java

**Method:** IntroPanel(JFrame window, Settings settings)

**Description:**

Start

Do: set the window value

Do: set the settings value

Do: create a new BgPanel

Do: set the layout

Do: add panels to the window

Do: set the opacity to false

Do: add item to the window

Do: revalidate window

Do: repaint the window

End

**Class:** IntroUpperPanel.java

**Method:** IntroUpperPanel(JFrame window, Settings settings)

**Description:**

Start

Do: set the opacity to false

Do: set the border

Do: try getting the icon for the logo

Do: try adding the logo

Do: catch any exception

End

**Class:** Leaderboard.java

**Method:** Leaderboard()

**Description:**

Start

Do: create array list

Do: try getting the leaderboard text file to read from

Do: if file exists then create a scanner

Do: while the file has a next line with content

Do: take the line and create an array with the content and split the content up

Do: store each value in variables

Do: convert each string to the appropriate type for the board constructor

Do: create a new board

Do: add the board to the leaderboard array list

Do: close the scanner

Do: catch any exceptions

End

**Class:** Leaderboard.java

**Method:** addToList(Board board)

**Description:**

Start

Do: add a board to the list

Do: save the list to the text file

End

Test Case

* If error saving list to txt file display error and exit

**Class:** Leaderboard.java

**Method:** sortByPlayer()

**Description:**

Start

Do: standard for loop sorting array list alphabetically by user name

End

**Class:** Leaderboard.java

**Method:** sortByDifficulty()

**Description:**

Start

Do: standard for loop sorting array list alphabetically by difficulty

End

**Class:** Leaderboard.java

**Method:** sortByTime

**Description:**

Start

Do: standard for loop sorting array list by the time

Do: inside the for loop the minutes and seconds are just being turned

Into seconds for easier sorting

End

**Class:** Leaderboard.java

**Method:** searchByPlayer(String user)

**Description:**

Start

Do: create a new array list to hold each board that belongs to a certain user

Do: for loop to check to see which boards have the same user name that player is searching for

Do: return the array list containing each board by the searched user name

End

**Class:** Leaderboard.java

**Method:** searchByDifficulty(Difficulty diff)

**Description:**

Start

Do: create an array list to hold the boards that have the search difficulty

Do: for loop to check which boards have the searched difficulty

Do: add each board to that list

Do: return the array list of the search difficulty boards

End

**Class:** Leaderboard.java

**Method:** export()

**Description:**

Start

Do: Try creating a printwriter using the leaderboard text file

Do: using a for loop go through the the leaderboard list and get each specified value

Do: convert each value to the appropriate string value

Do: Save each value to the text file and splitting them with | symbol

Do: close the print writer

Do: catch any exceptions

End

**Class:** LeaderboardMenuItem.java

**Method:** LeaderboardMenuItem(JFrame window, Settings settings)

**Description:**

Start

Do: set the window value

Do: set the settings value

Do: create new JMenuItem

Do: set text

Do: add action listener

Do: try setting leaderboard icon

Do: catch any exceptions

End

**Class:** LeaderBoardActionListener (inside LeaderboardMenuItem.java)

**Method:** actionPerformed(ActionEvent e)

**Description:**

Start

Do: create new leaderboard panel

End

**Class:** LeaderboardPanel.java

**Method:** LeaderboardPanel(JFrame window, Settings settings)

**Description:**

Start

Do: set the window value

Do: set the settings value

Do: create the panel

End

**Class:** MenuBar.java

**Method:** MenuBar(JFrame window, Settings settings)

**Description:**

Start

Do: create new JMenuBar

Do: add game menu to the JMenuBar

End

**Class:** NewGameMenuItem.java

**Method:** NewGameMenuItem(JFrame window, Settings settings)

**Description:**

Start

Do: set the window value

Do: set the settings value

Do: create new MenuItem

Do: set text

Do: add action listener

Do: try to set icon

Do: catch any exceptions

End

**Class:** NewGameListener (Inside NewGameMenuItem.java)

**Method:** actionPerformed(ActionEvent e)

**Description:**

Start

Do: check if user does want a new game

Do: if yes the generate new Game Panel

End

**Class:** SudokuGUI.java

**Method:** SudokuGUI()

**Description:**

Start

Do: set the settings value

Do: set the content pane

Do: set the title

Do: set the layout

Do: set the close operation

Do: add window listener

Do: set the size of the screen

Do: set resizable to false

Do: set the location of the game in the middle of the screen

Do: try to set the icon

Do: set visible to true

Do: create intro panel

End