Java Game Suite Phase 2

Revision 9

CMSC 495 6382

September 23, 2021

Group Charlie

Sherry Funches, Oyewole Sanusi, Janee’ Jones, Wayne Mack, Jeffrey McGlinn

## **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number** | **Date** | **Description** | **Name** |
| 1 | 9/22 | Creation of Doc and Outline | Oyewole |
| 2 | 9/23 | Overview and Milestone | Oyewole |
| 3 | 9/23 | Phase 2 created/ Modified | Oyewole |
| 4 | 9/26 | Slider Puzzle Phase 2 Description | Jeff |
| 5 | 9/26 | Phase 2 Documenation for Sudoku | Janee’ |
| 6 | 9/26 | Phase 2 Documentation for Word Search | Sherry |
| 7 | 9/26 | Phase 2 Maze | Wayne |
| 8 | 9/26 | Phase 2 Snakes | Oyewole |
| 9 | 9/26 | Combine Files, Formatting, Add Table of Contents | Sherry |

**Table of Contents**

I. Overview………………………………………………………………………………………..4

1.1 Overview of Project…………………………………………………………………………...4

II. Milestones………………………………………………………………………….…………..4

2.1 Projected Team Milestones……………………………………………………………………4

2.2 Current Status on Phase 2 …………………………………………………………………….6

A. Maze………………………………………………………………………………...….6

B. Snakes…………………………………………………………………………………..7

C. Sudoku …………………………………………………………………………………7

D. Word Search……………………………………………………………………………9

E. Slider Puzzle…………………………………………………………………………..10

## **I. Overview**

**1.1 Overview of project**

The Java Game Suite program is a simple GUI-based application that provides entertainment to users by allowing them to play several simple single player computer games. The user will be presented with an interface that indicates the available games: a maze, snakes, Sudoku, word search, and a slider puzzle. The user will be able to select a game from the suite and launch it from that interface.

## **II. Milestones**

* 1. **Milestone**

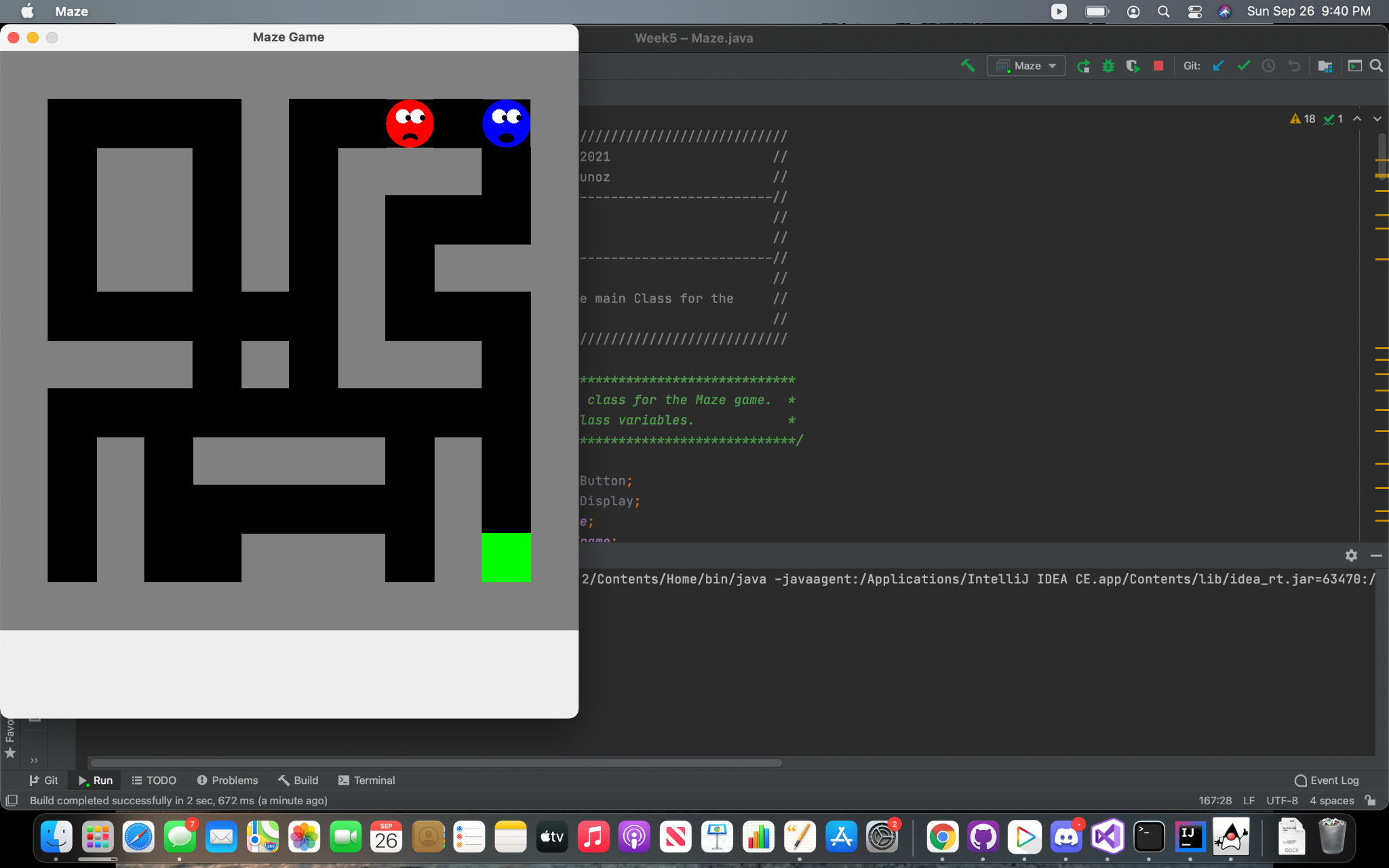
The following table shows our completed and pending goals. The timeline of certain goals have changed based on changes made to the implementation process.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Projected Date of  Accomplished Goal | Name of the Goal | Was the Goal  Accomplished |
| Phase 1 | 9/13 | Design what the system does and how the different pieces fit together. | Yes |
| 9/20 | Start GUI coding | Yes |
| 9/20 | Start other required classes coding | Yes |
| Phase 2 | **9/26** | **Continue GUI and other required classes coding** | **Yes** |
| **9/26** | **Finish and share prototypes of working games with group** | **Yes** |
| **9/26** | **Start Testing** | **Yes** |
| Phase 3 | 10/1 | Integrate all games and main menu into package | Pending |
| 10/1 | Testing | Pending |
| 10/3 | Revise code and implement additional features | Pending |
| Phase 4 | Final Sprint and Delivery | | |
| 10/6 | Finishing Touches | Pending |
| 10/6 | Update all documents with any changes | Pending |
| 10/8 | Group Revision | Pending |

**2.2 Current Status on Phase 2**

**A. Maze**

For this phase in the project, I am creating 9 additional levels and beta testing the levels for glitches and inconsistencies. I am also test running levels to make sure they are winnable and not very difficult. The levels are to get progressively harder and, I am also trying to make sure there is enough to keep the players interested without potentially frustrating them.



**B. Snakes**

Phase 2 is completed on the snake game app. The snake will consume an apple and grow in length, with each end of the snaking being a different color so that users can differentiate the head from the tail. Like many games, the levels become more challenging the further the players progress. I’ve also settled on a grid size for the player. I realized that the larger the grid size, the longer the game will take to switch from easy to difficult. Phase 3 will be to polish my code and fix any remaining bugs in my game.

**C. Sudoku**

For Phase II, the Sudoku puzzle is a working prototype. The production of the Sudoku game having one difficulty level across various games was a better option for the game development. The code now has zero errors and runs. Once the code is executed, the first initial game is displayed. The user will have the option to proceed with that game and filling in the missing numbers. If the user chooses not to complete the first initial game that is generated, then there will be an option to choose a new game. As they keep choosing the new game option, a new game will be loaded and displayed to the user. One change that I am trying to configure code for would be to add a timer. The timer would show the user how much time it took them to complete the puzzle. The timer would automatically stop and calculate the time once the puzzle has been completed. Throughout the upcoming week I will be working on the timer portion of the code to see if it could be incorporated. The timer would be a great feature, just in case the user would like to know the time it took for them to complete and if they wanted to improve on their time.

Proceeding with what I have for this phase, I continued minor testing to ensure that the Sudoku was still producing the product that I needed it to. I would like the timer to be displayed beside the ‘Exit’ button. At the initial start of the game, it will start as 0:00 until the user makes the first move then the timer will begin to count upward. At the end of the completed game the timer will show how much time it took for the user to complete, location still beside the exit button.

Calendar

Description automatically generated

Timer Location start time 0:00

Figure 1: User is presented once Sudoku is chosen from the main menu. Timer will be displayed as 0:00 location beside the ‘Exit’ button.

Table

Description automatically generated

Timer Location: completed time

Figure 2: User has completed the game. User can select New Game to play again (new game board generated) or Exit Game to be directed to the main menu. Game completed the timer will display the time taken to complete the puzzle.

**D. Word Search**

I successfully completed the word search program’s phase two goal, which was to produce a complete functioning prototype where the user can select a puzzle, the puzzle loads from a file, the user can interact with the GUI to find the words, and the program displays a score upon clicking the submit button. Since I already implemented the GUI interface in phase one, phase two focused on backend development in which I created the PuzzleGenerator class, which reads in a puzzle file and creates objects to be used by the GUI layer. Phase two also involved adjustments to the GUI display as well as implementing and testing the scoring function.

First, I implemented the Word and PuzzleGenerator classes. The programming of both was straightforward as I already decided the structure of the test files and devised the algorithm for reading them in the design phase. After testing that the PuzzleGenerator read in the files and assigned them to data structures as intended, I linked it to the GUI. This was relatively simple and involved removing the hard coding I used to get the GUI display up and running and finishing implementations of the load file function that reads in data from the PuzzleGenerator class.

The first design challenge I encountered was that hard coding the size of the letter fonts on the search grid could be problematic. I have a small laptop screen and too large of a font with too many rows made part of the Jframe not visible. I also did not want to fix the JFrame’s size in case someone was using a smaller or far larger screen. My solution was to first determine the user’s screen size, then calculate what I felt was an acceptable font size based on those measurements, and pass that to the puzzle generator to set the font size. I also decided a maximum number of columns and rows for puzzles that would work with this ratio. In phase three, I may want to allow the user to increase font sizes of the letters on the grid and words on the word list panel separately to make the program more accessible to the vision impaired.

The next challenge was efficiently scoring the words that the user found correctly. I originally planned on using set operations. First I would create a set containing all the sets of letters that each line intersected. I thought the most efficient approach would be to do perform a set difference between this set and the set of hidden words, the difference producing the unfound words. However, I realized that I could not create word objects from the sets of letters that intersected by the user’s highlights that would match the hidden words easily. In phase 3 I may try to create a custom comparator. In the interest of time and considering that the number of hidden words and potentially found words was always going to be small, I decided it was not that inefficient to just iterate over the hidden words and compare their inner set of letters to the each of the sets of letters intersected by the user’s highlights. As seen below, I implemented turning the font of the unfound words to red and displayed a JDialog box with the score. I also implemented clearing the entire panel when changing puzzles, which I found did not work as expected when tested.

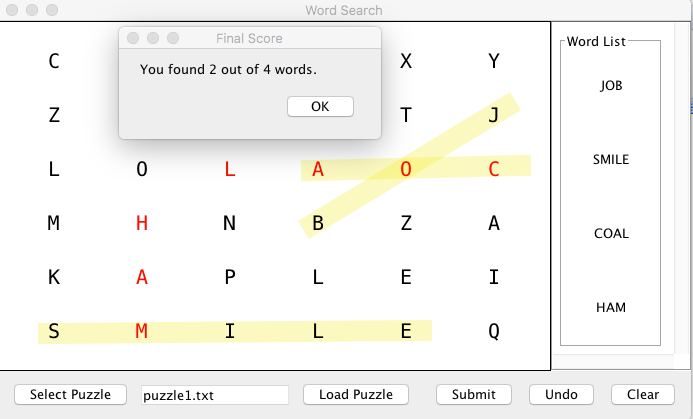


Figure A. Implementation of word search scoring and display of unfound words. Note that “coal” was not completely highglighted so correctly not counted as found.

Phase three will involve minor tweaks to the program, such as fixing the display in which the user has to drag the JFrame in order to reveal the puzzle when it is loaded and adding additional puzzle files.

**E. Slider Puzzle**

The goal for phase 2 was to continue coding the GUI for the slider puzzle, implementing the required methods, adding action listeners, begin testing, and integrating the slider puzzle into the main menu. Below is a screen capture demonstrating the launch of the GUI from the main menu:

Graphical user interface, website

Description automatically generated

All the required methods and action listeners were successfully implemented in phase 2. The program now allows the user to swap the image tiles and detects when the puzzle is complete. A message is displayed that informs the user that the puzzle has been completed. I also adjusted the code so that the main menu does not close when the slider puzzle is exited. This will allow the user to launch multiple games at once. Moving forward into phase 3, the goal will be to integrate the slider puzzle into a package containing all the java games and to continue testing.