Grid Games

1 Overview

In this lab we will be practicing both interface based programing and GUI programing in order to create a small suite of games along with a graphical interface for them. By the end of this lab you will have implemented Tic Tac Toe, Minesweeper and Connect Four, and be able to play any of them using a Swing GUI. A single interface is provided for all three of these grid games. Each game you program must be programmed to that interface, and a single GUI frame will be responsible for playing any grid based game using that same interface. Additionaly, a small launcher will instantiate any of these three games into that GUI. Rules for each of the games is provided below, both as precise specification and for the benefit of those who may be unfamilar with these games.

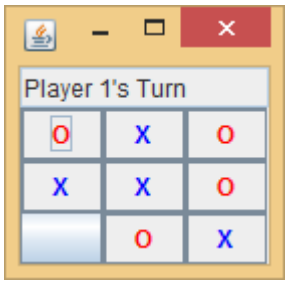


Fig. 1: Tic Tac Toe

Tic Tac Toe

Tic Tac Toe is a two player game played on a 3 by 3 grid. The X player goes first, followed by the O player. During each players turn they may leave their mark in an unmarked square of the game board. If that mark creates three of their mark in a row, that play is the winner. If the board is filled without any three being connected, the game is tied and both players lose. There is a maximum of 9 moves in any Tic Tac Toe game.

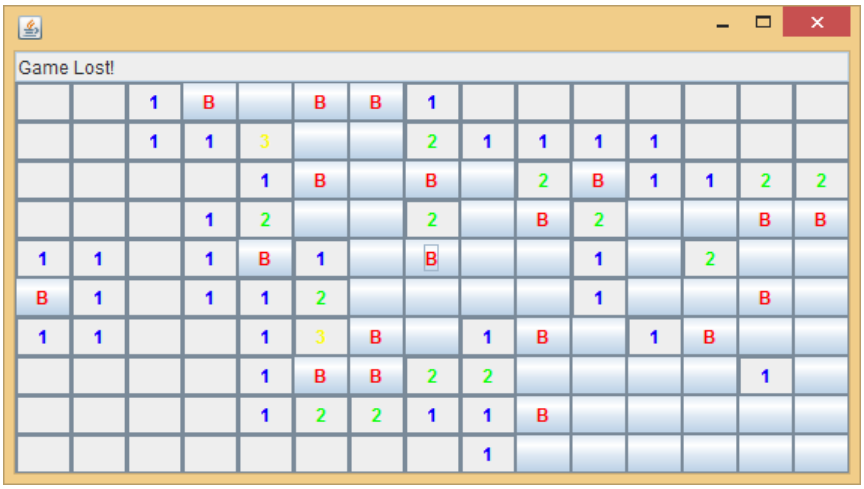


Fig. 2: Minesweeper

Minesweeper

Minesweeper is a one player game in which the goal is to uncover all squares on a large grid that do not contain a bomb. All locations on the grid start as hidden. When a player clear a space one of three cases occur. If that location held a bomb, the bomb explodes, revealing all bombs on the map and losing the game. If a space is a neighbor to a bomb it is cleared and is marked with a number showing the number of bombs that neighbor that square. If a square does not neighbor a bomb, it is revealed with no text, and autocratically clears all of its uncleared neighbors, which may in turn clear their neighbors if they do not neighbor bombs. When all spaces without bombs are cleared, the player wins, and if a bomb locations is cleared, the player loses.

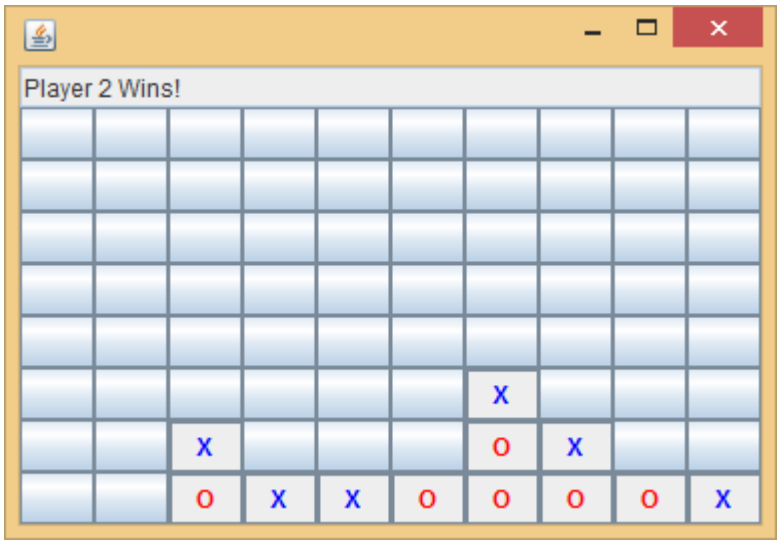


Fig. 3: Connect Four

Connect Four

Connect Four is a two player game in which players take turns leaving a mark on a grid 10 wide and 8 high. Players marks drop to the bottom of the board, either sitting at the bottom of their column or stacking on top of a mark beneath them. Trying to mark any square will result in placing the mark at the bottom of the column. If any player achieves four marks in a row, horizontally, vertically or diagonally, that player wins. If all 80 spaces are filled without getting four in a row, both players lose.

2 Learning Outcomes

By the end of this project students should be able to:

- design object oriented systems using interfaces and inheritance;
- utilize polymorphism;
- use interfaces and inheritance to reduce coupling in a system;
- use Java's Swing library for GUI programming;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem solving process.

3 Pre-Lab Instructions

Do this part before you come to lab:

This project will require you to make use of inheritance and interfaces.

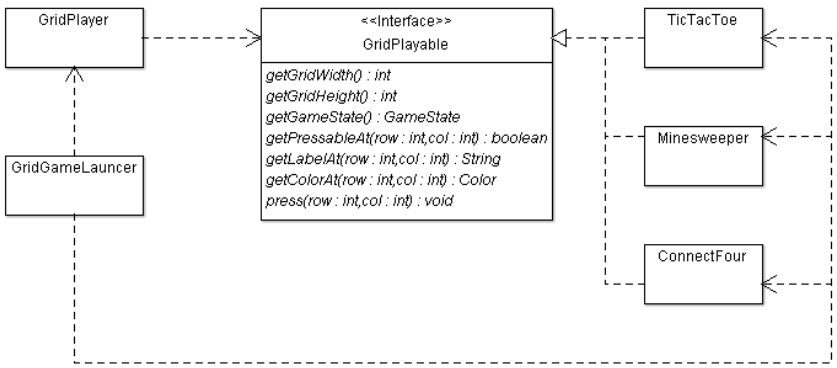
- Read Big Java chapters 10.7-10.10 and 19 on the topics of Graphical Programming.
- What classes and interfaces must be utilized to allow a Swing Button to execute code after the user clicks it? What methods must be defined? **How might one button function differently than other buttons using the same listener class?** Write down answers for all of these questions.
- In this lab we will need a grid of buttons as well as a status bar. What two layout managers might be useful for this lab? Write down your answer.

4 Lab Instructions

Do this part in lab:

Step 1

Download the provided GridPlayable.java file containing the GridPlayable interface. **Do not change this file!** Create the classes specified in the following UML Class diagram:



As a reminder for reading UML class diagrams, our dashed lines with open arrows indicate dependency. This class at the start of the arrow uses the class at the end of the arrow in some way. The dashed lines with the white triangle

arrow represents implementing an interface. TicTacToe, Minesweeper and ConnectFour all implement GridPlayable. Grid player can use any GridPlayable class, but should not know about the existence of ANY of the specific games. Our GridPlayer class should have no special code for any of the specific games that we make. GridGameLauncher is our starting frame that asks which game you want to play, and must therefor know about both the player and the specific games that could be played with it.

Notice that getGameState() returns a GameState this is an enumeration class defined in the interface java file. Enumerations are classes that can only contain one of a specified number of values. For a reminder on how enumerations work, refer to page 206 of the Big Java textbook, or research them online. This may also be the first time you have encountered an inner class, one class defined inside of another. This means that the scope of the enumeration is inside the scope of the containing class. If you were outside the playable interface and you wanted to address the player one win state, you might use the code "GridPlayable.GameState.WIN_P1".

I recommend starting with a single game such as TicTacToe. Do your best first attempt at satisfying the interface. Then create your GridPlayer and GridLauncher as extended JFrames. Make sure you can instantiate a GridPlayer frame from your GridLauncher frame using a button. Once your GridPlayer frame is coming up without errors, using the launcher to connect it to a TicTacToe game. Now develop both your player and TicTacToe classes together to test one another.

Step 2

Once your TicTacToe game is working in the player, implement the other two game types. Create additional buttons on the launcher frame to build players for the different games, and ensure that the player can handle each new game you make. Once you are done with these games, go back and make sure all three games still work! Again, there should be NO special code in the player for handling Minesweeper Connect Four or Tic Tac Toe specificity. Your class should be designed such that you could implement any number of grid based games without having to modify your player.

Extra Credit

For a full labs worth of extra credit, you can implement the game Snake using a similar method. Snake is a game in which a the player controls a long snake moving around a very large grid to eat food. Each time the snake eats food, it grows one grid space longer. The player loses whenever the snake runs into a wall or itself. The goal of the game is to eat as many items of food as possible before losing, which represents the players score. This game has several major differences from the previous grid based games. It advances over time without input. Also, rather than pressing locations on the grid, the player controls the game with the arrow keys. The game doesn't have a win state or turns, just

a score, and if the game is over or not. **With these differences in mind, what is the minimum number of changes that can be made to the GridPlayble interface?** Create a new interface named ActionGridPlayable that contains this new interface. Now create an ActionGridPlayer that utilizes this interface, and Snake game class that implements it. When it is done, add Snake to your GridGameLauncher. For full points create a separate report for the snake game and write it the same way you would a normal lab report with problem statement, planning, implementation and reflection sections.

5 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical. In order to recieve points, you MUST make a BBLearn submission.

Your lab report should begin with a preamble that contains:

- The lab assignment number and all partner names
- Your name(s)
- The date
- The lab section

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in **your** own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific funtions, classes or numeric requirements given to you, they should be represented in this bulleted list. It is recomended that you complete this section before you begin coding the problem, as gathering these requirements may help you organize your thoughts before you begin your soulation.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why. How do you plan on breaking up the problems into functions, classes and methods?

3. Implementation and Testing

In the third section you should demonstrate your implementation. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as .java files, should not be in your report document)
- a screen shot of your running application / solution. This should include a screenshot of the output, be it a terminal window or graphical interface
- results from testing. This is specific to the lab, and not every lab will include this element.

4. Reflection

In the last section you should reflect on the project. What part of the projec was the most difficult? Where there other solutions to this problem that you considered? Where there any new solutions you can think of now that you couldn't then? Do you think the way you chose was the best option, or would you go about the problem differently if you had to start over? How might the problem had been broken up into easier problems? If you had one more day to work this problem, what improvements might you make? Every problem has alternative solutions and solution has tradeoffs or improvements, you are required to identify some of these elements.

5. Partner Rating

Every assignment you are required to rate your partner with a rating between 1 and 10. For each patener a name and rating should be submitted in the comment section of the BBLearn submission, and not in the report document. You do not have to tell your partner the rating you assign them. A rating 8 or more indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 5 indcates that your partner met the class expectations of them. Rating your partner 3 or less means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. In the case where you give a very low rating, please explain why in the comment section of the submission. Those who recieve low scores may be asked to explain their actions to the lab staff, and additional low ratings after a warning could lead to losing a letter grade, or even failing the course.

Consistant high ratings from partners are noted durring the grading process, and may be taken into account when rounding your final grade.

Colophon

This project was developed by Richard Lester and Dr. James Dean Palmer of Northern Arizona University.