S509/ICS4U1-11
Software Design Specification
Java Game Application

Connect Four

By Kevin Xie
Teacher: Mr. PC

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a high-level description of the software architecture and design.

1.2 Scope

This project implements Java programing language through the GUI-based Connect Four game. The application development uses SDLC Waterfall Model.

1.3 Application Description

"Connect Four (also known as Four Up, Plot Four, Find Four, Captain's Mistress, Four in a Row, Drop Four, and Gravitrips in the Soviet Union) is a two-player connection board game, in which the players choose a color and then take turns dropping colored discs into a seven-column, six-row vertically suspended grid. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs." - *Wikipedia.*

This specific program modifies the traditional Connect Four game. The following are the list of features implemented in the modified Connect Four game application,

- 6x7 grid board
- 2 players (Red and Yellow)
- Players are able to freely choose where to place their coloured disks on the grid
- The players take turns to place colored discs on the gaming grid board
- The winner is declared if either player forms a horizontal, vertical or diagonal line of four of his/her own discs

1.4 References

[1] Wikipedia Connect Four https://en.wikipedia.org/wiki/Connect_Four
[2] Textbook - A Guide to Programming in Java
https://bbarrettchs.weebly.com/uploads/3/7/7/8/37782575/lvp_java_text.pdf
[3] Course notes - ICS4U1-11 Introduction to Computer Studies - Christian

2 Algorithm

Step 1: Load the game board

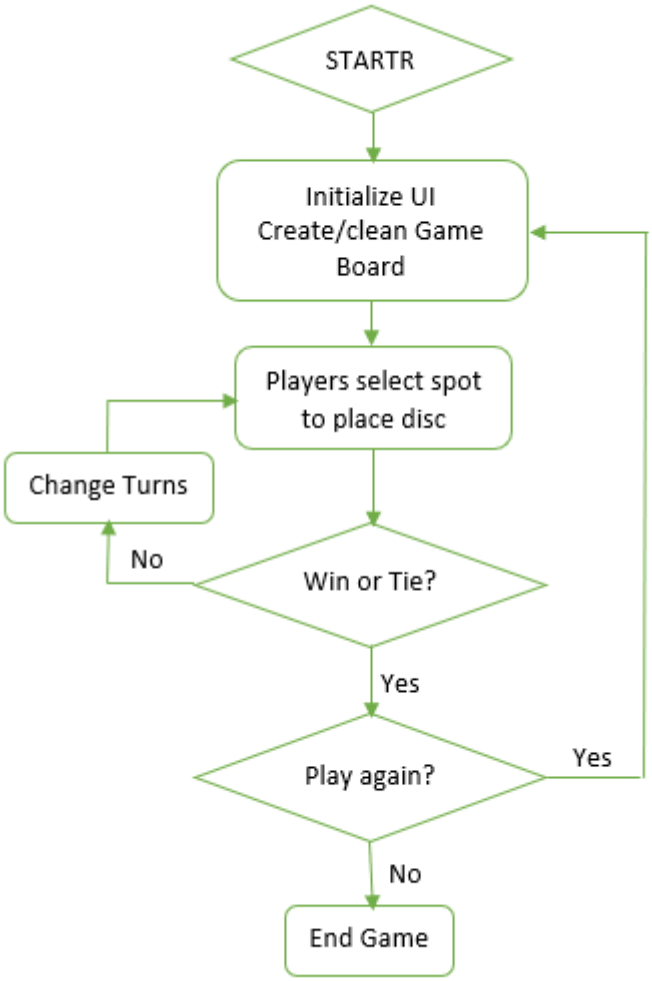
Step 2: While there is no winner or tie

1) Player 1 's turn to pick a free spot to place the disc. After the disc is placed , check if player 1 win or game tie
2) If not, set the turn to player 2's, player 2's turn to pick a free spot to place the disc. After the disc is placed , check if player 2 win or game tie
3) If no winner or tie, go back step 2, alternatively taking turns, repeating 1) and 2)

Step 3: Game terminates if win or tie condition is reached, the winner is declared and asked whether to start a new game or quit

Step 4: if the user selects to play again, go back Step 1, else program is terminated

2.1 Flowchart



2.2 Pseudocode

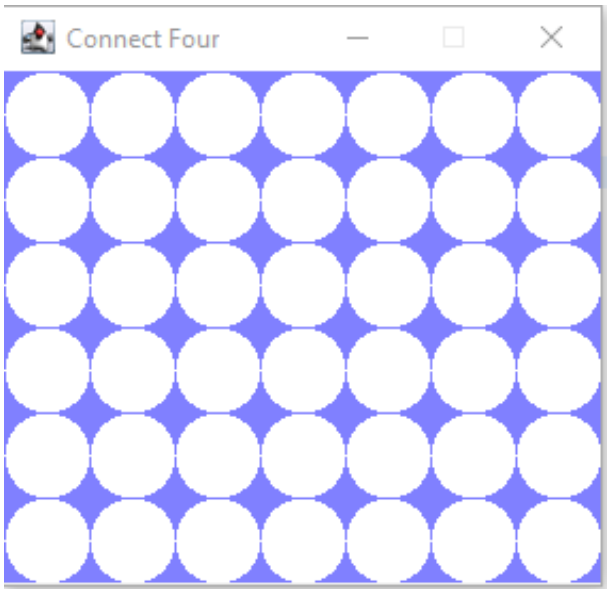
1.

```
1. initUI() //define and create the frame, add player Grid object to the frame
2. drawGrid(Graphics2D g2d) // draw and initialize the grid
3. mouseClicked(MouseEvent e) //Grid object is listening mouse event and will
   interact with the user input, if a click event happened,
   if(playerGrid is not full) {
                if(Red player places disc) {
                      cell.setColor(red)); //paint the cell in red
                      if(if game.tie()) {
                            showMsg("Game tie!! Do you like to play again?")

                      }
```

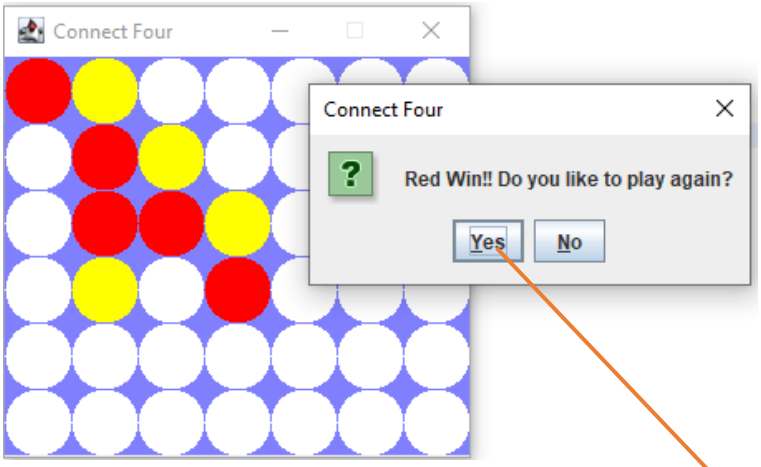
```
                        if(if game.win(red)){
                                showMsg("Red Win!! Do you like to play again?")
                        }

                        player = Yellow; //switch the turn

                }else { // player 2 - yellow player

                        if(if game.tie()) {
                                showMsg("Game tie! Do you like to play again?")

                        }

                        if(if game.win(Yellow)){
                                showMsg("Yellow Win!! Do you like to play again?")
                        }

                        player = Red; //switch the turn
}
```

4. When the game is over (one player wins or tie is reached), either restart the game or quit the game based on the user input (play again or quit)

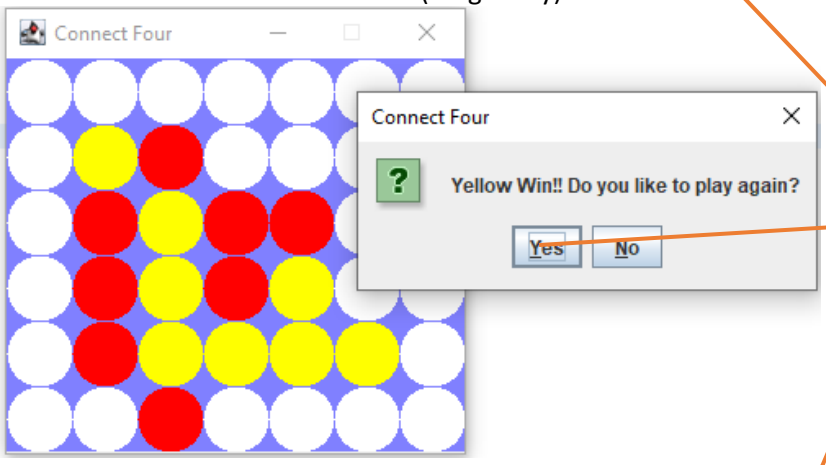
   here are some snapshots of the game board,



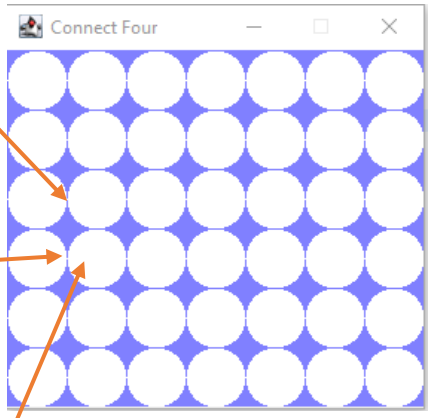
Picture 2.1 Blank game board – prior to player action



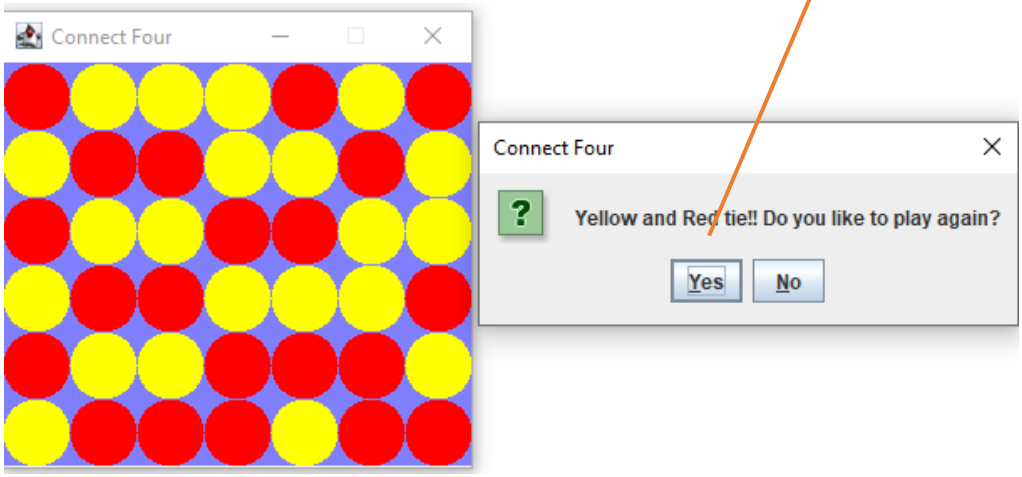
Picture 2. 2 Game board – Red win (Diagonally)



Picture 2.3 Game board – Yellow win (Horizontally)



Picture 2.4 New game begins



Picture 2.5 Game board – Game tie

3 Implementation

The Connect Four application is modeled with player object, grid object and game object. The player object stores player identification(Red or Yellow) and each disc placement(row and column in the grid). The game object contains methods to determine whether either win or tie is reached. The grid class extends ***Jpanel*** to shape the game board, which is a 6x7 grid and implements ***MouseListener*** to interact with the user. The grid object receives inputs from players to transfer the data to the game object in order to determine the outcome of each round. The application source code will be submitted with this specification documentation.

3.1 ConnectFour.java

The aspects for ConnectFour classes are:

ConnectFour *(extends JFrame)*

Variables: gameBoard (Grid Class)

Methods:

InitUI – defines the frame size and shows gameboard on the frame

Main – creates and starts the game

3.2 GameRun.java

This class is a helper for checking whether win or tie condition is reached.

GameRun

Variables: none

Methods:

win - checks whether any win condition is reached

tie - checks whether any tie condition is reached

3.3 Grid.java

This class implements MouseListener to interact with the user, receive the input from players. When a mouse event occurs, the relevant method will be invoked.

Grid *(extends Jpanel implements mouseListener)*

Variables:

ROWS, COLS – rows and columns of the grid

startX, StartY – the start position of the grid

cellWidth – cell width of the grid, cell height is the same as cell width

player – 2 players, red and yellow

playAgain – stores the user selection when a game completed

Methods:

paintComponent – Override parent method; call drawGrid method to create the grid

drawGrid - create and draw the grid in the frame

mouseClicked – interact with the players and update the game grid

showMsg – when a game completed, pop up a dialog window declaring the winner, and ask if the user whether continue or quit the game

4 Application Testing and Debugging

Test ID	Description	Expected Results (R – Red Player; Y – Yellow Player)
1	UI check	Check if the grid is 6 rows by 7 columns as required
2	Horizontal check	If 4 discs of the same colour is adjacent to one another in a row then winner is declared (check for both players Y and R)
3	Vertical check	If 4 discs of the same colour is adjacent to one another in a column then winner is declared (check for both players Y and R)
4	diagonally check (1)	If 4 discs of the same colour is adjacent to one another diagonally from top left to bottom right then winner is declared (check for both players Y and R)
5	diagonally check (2)	If 4 discs of the same colour adjacent to one another diagonally from top right to bottom left then winner is declared (check for both players Y and R)
6	Game tie	No free spot for player to place next disc, declare game tie
7	Continue to play	When win or tie occurs, pop dialog window to ask the user whether to continue or quit the game. The user selects to play next game, the game grid board should clear and refresh for players to restart next game
8	Exit game	When win or tie occurs, pop dialog window to ask the user whether to continue or quit the game. The user selects to quit the game, the application stops and exits

Other than the above functionality testing, the project source codes are also performed unit testing to examine code path (e.g condition coverage, decision coverage etc) to make sure each entry and exit point is invoked and shows the expected outcome.