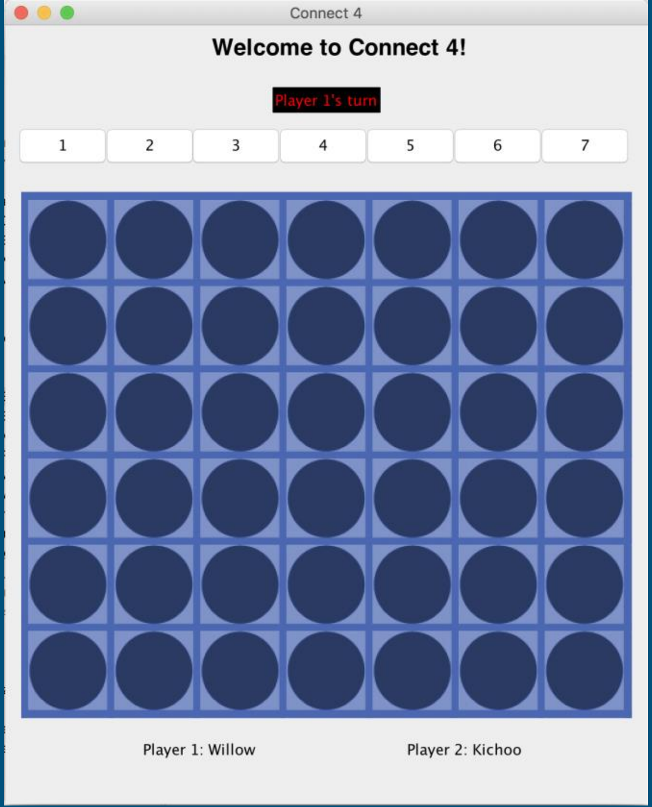
Team 5

Connect Four

Arushi Gupta | Jay Shah | Swati Sinha | Zhenyu Zhou

14th March, 2018

Demo



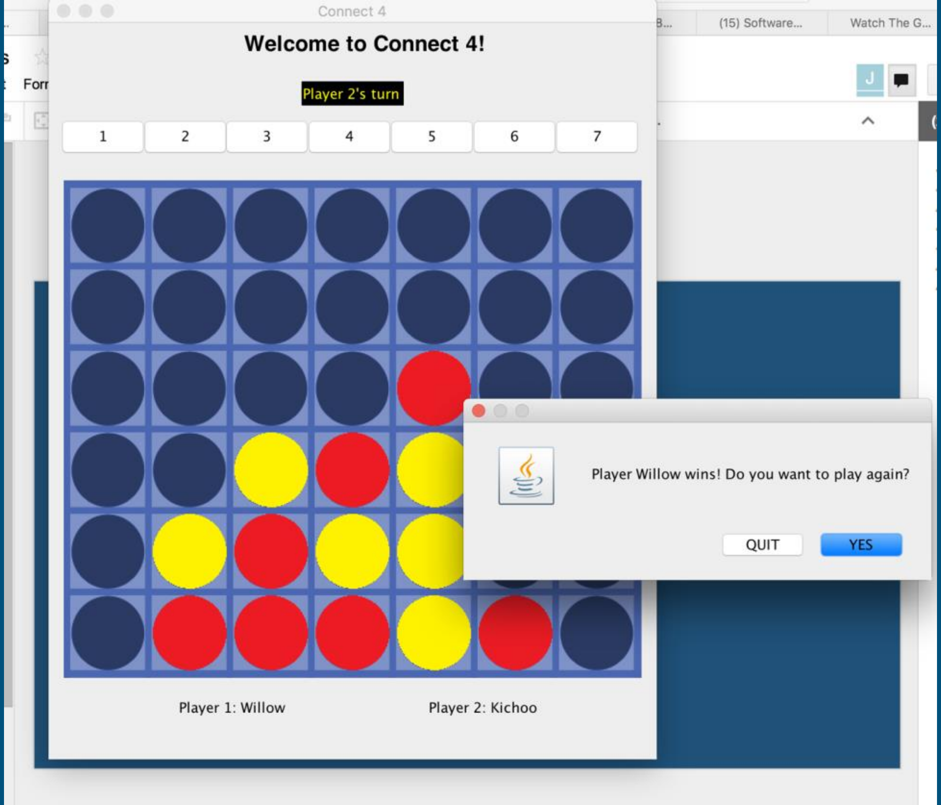
Connect 4

Welcome to Connect 4!

Player 1's turn

Player 1: Willow

Player 2: Kichoo



Connect 4

Welcome to Connect 4!

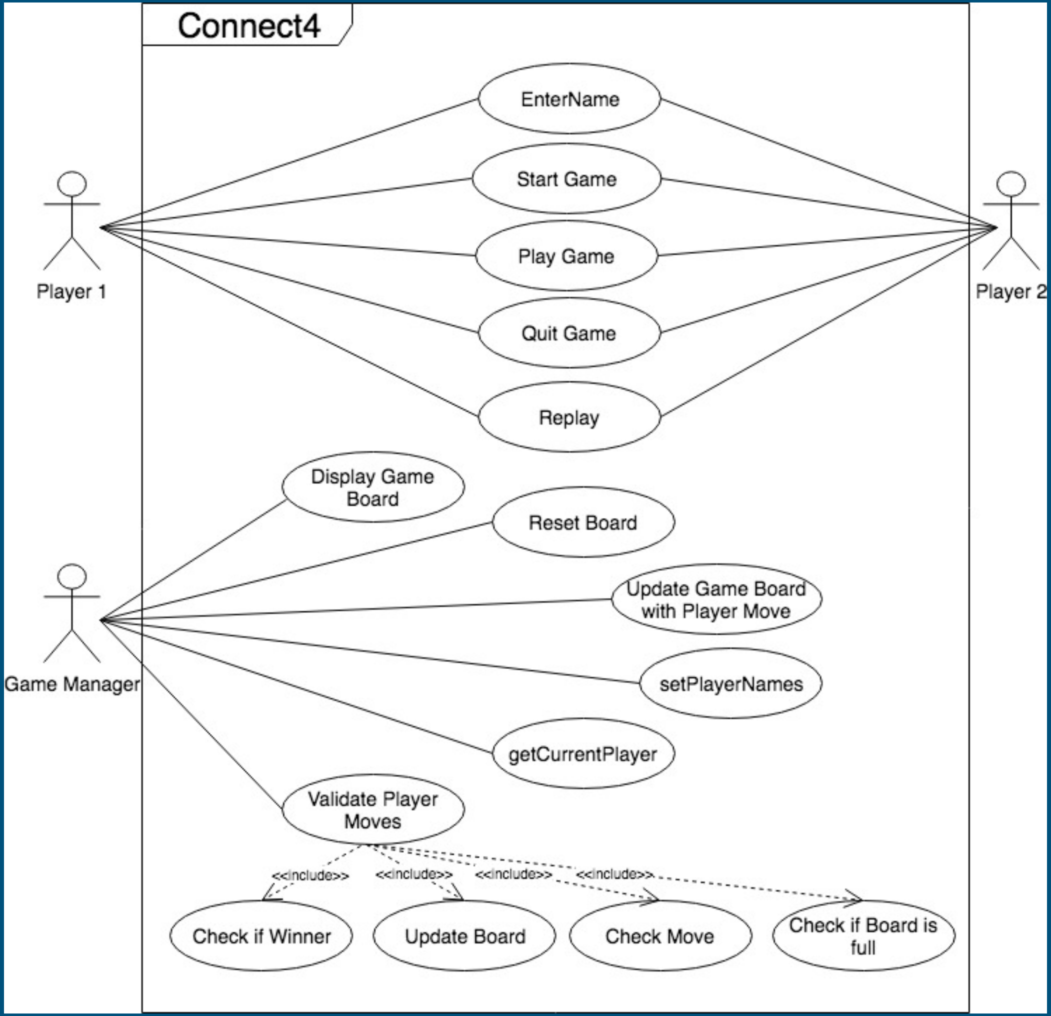
Player 2's turn

Player 1: Willow

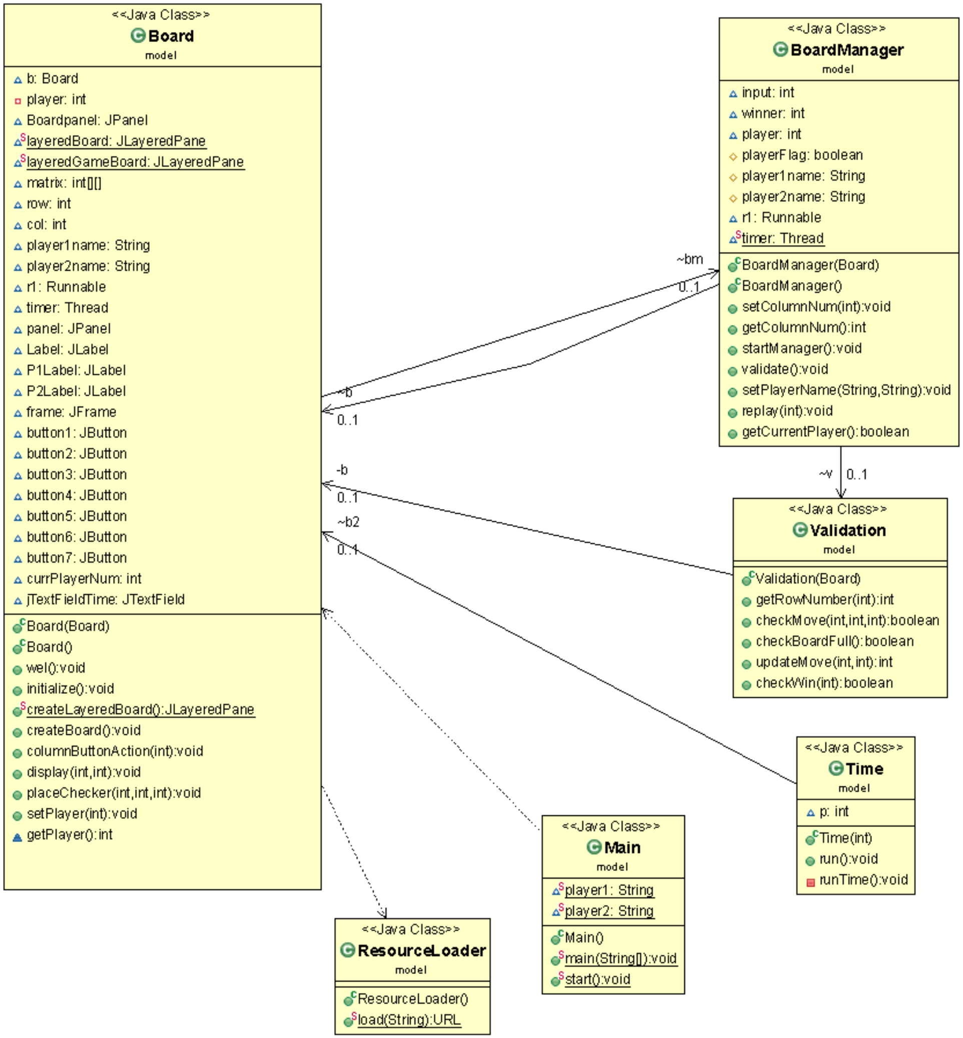
Player 2: Kichoo

Player Willow wins! Do you want to play again?

Use Cases


Connect4
Player 1
Player 2
Game Manager
EnterName
Start Game
Play Game
Quit Game
Replay
Display Game Board
Reset Board
Update Game Board with Player Move
setPlayerNames
getCurrentPlayer
Validate Player Moves
<<include>>
<<include>>
<<include>>
<<include>>
Check if Winner
Update Board
Check Move
Check if Board is full

Class Diagram



<<Java Class>>
Board
model

- b: Board
- player: int
- Boardpanel: JPanel
- layeredBoard: JLayeredPane
- layeredGameBoard: JLayeredPane
- matrix: int[][]
- row: int
- col: int
- player1name: String
- player2name: String
- r1: Runnable
- timer: Thread
- panel: JPanel
- Label: JLabel
- P1Label: JLabel
- P2Label: JLabel
- frame: JFrame
- button1: JButton
- button2: JButton
- button3: JButton
- button4: JButton
- button5: JButton
- button6: JButton
- button7: JButton
- currPlayerNum: int
- jTextFieldTime: JTextField

- Board(Board)
- Board()
- wel():void
- initialize():void
- createLayeredBoard():JLayeredPane
- createBoard():void
- columnButtonAction(int):void
- display(int,int):void
- placeChecker(int,int,int):void
- setPlayer(int):void
- getPlayer():int

<<Java Class>>
BoardManager
model

- input: int
- winner: int
- player: int
- playerFlag: boolean
- player1name: String
- player2name: String
- r1: Runnable
- timer: Thread

- BoardManager(Board)
- BoardManager()
- setColumnNum(int):void
- getColumnNum():int
- startManager():void
- validate():void
- setPlayerName(String,String):void
- replay(int):void
- getCurrentPlayer():boolean

<<Java Class>>
Validation
model

- Validation(Board)
- getRowNumber(int):int
- checkMove(int,int,int):boolean
- checkBoardFull():boolean
- updateMove(int,int):int
- checkWin(int):boolean

<<Java Class>>
Time
model

- p: int

- Time(int)
- run():void
- runTime():void

<<Java Class>>
Main
model

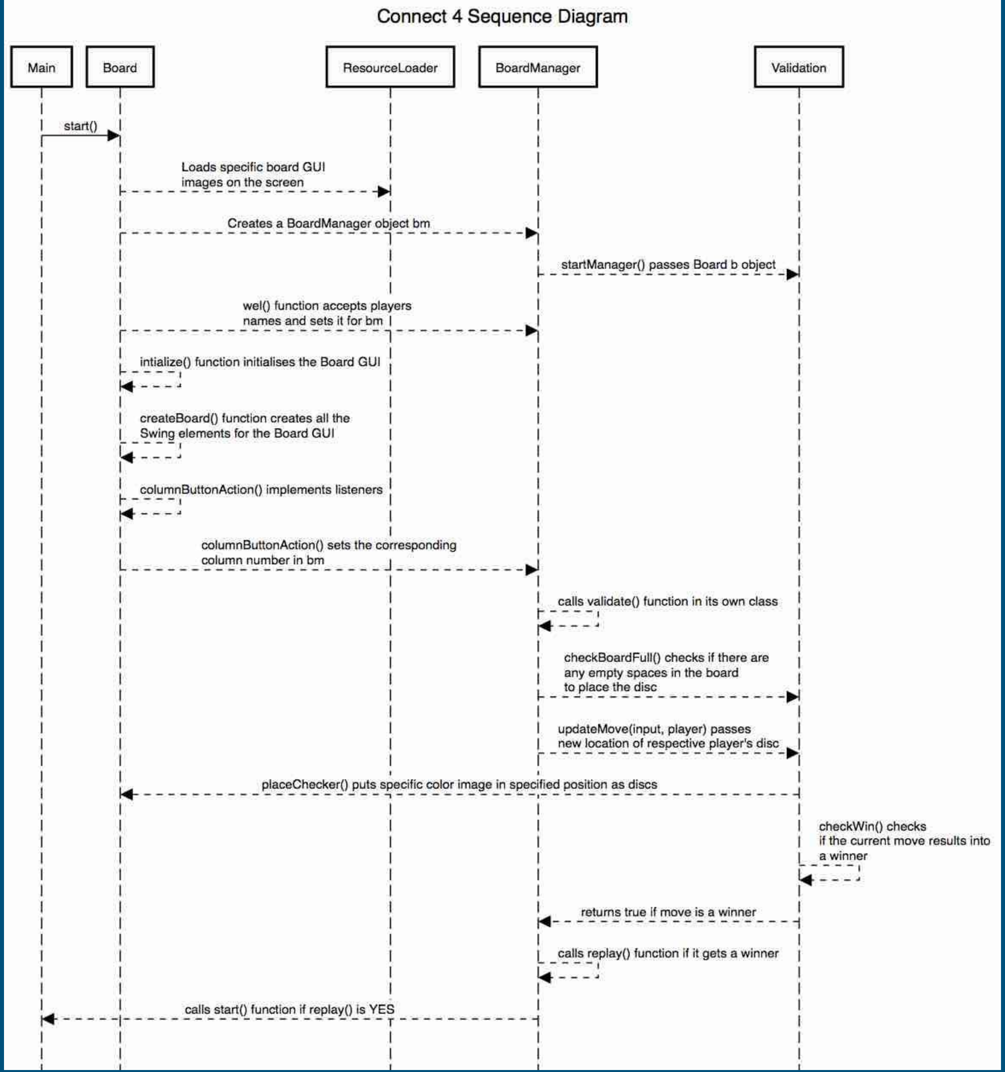
- player1: String
- player2: String

- Main()
- main(String[]):void
- start():void

<<Java Class>>
ResourceLoader
model

- ResourceLoader()
- load(String):URL

Sequence Diagram



Connect 4 Sequence Diagram

Main

Board

ResourceLoader

BoardManager

Validation

start()

Loads specific board GUI images on the screen

Creates a BoardManager object bm

startManager() passes Board b object

wel() function accepts players names and sets it for bm

intialize() function initialises the Board GUI

createBoard() function creates all the Swing elements for the Board GUI

columnButtonAction() implements listeners

columnButtonAction() sets the corresponding column number in bm

calls validate() function in its own class

checkBoardFull() checks if there are any empty spaces in the board to place the disc

updateMove(input, player) passes new location of respective player's disc

placeChecker() puts specific color image in specified position as discs

checkWin() checks if the current move results into a winner

returns true if move is a winner

calls replay() function if it gets a winner

calls start() function if replay() is YES

Traceability

1. Use class diagram/sequence diagram as link between requirements (uses cases) and code artefacts (or developers)
2. Use glossary of terms to describe classes, functions, states, and variables and documentate code descriptively
3. Use google doc to update, share the code and keep up versioning

Test

1. Separate testing of logic part and GUI
2. Examine every internal structure of program and attempt to test logical case - test the input causes the output of the program to be identical as the specifications would require
3. Integrate all dependent modules together

Test

A. “Good” input

B. “Bad” or malicious mistakes

a. For logic part:

i. Boundary conditions and values of different types (i.e. positive, negative)

ii. Logic based nested statements and case statements

iii. Entering condition and exit values of loop

b. For GUI

i. Everything display correctly

ii. Input has value

iii. Event listener and event handler

Review of Documentation

Future Scope

- Motion
- Countdown
- Load Game
- Undo/Reset

Lessons Learned

- Analysis phase helps in creating a functional model of the system regardless of constraints
- 5 C’s ~ Law of Demeter ~ Separation of concerns
- OO principles
- Saves time, easier to hand-off, less complicated
- Better to completely avoid GOD objects(big ball of mud)
- Promotes use of objects
- Reusability

Thank You!