

# Comprehensive . Exercise Connect4

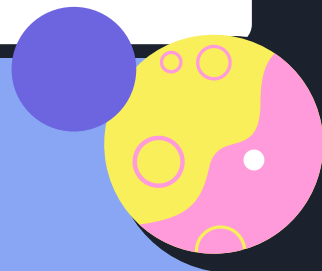


Aman, Benjamin, Noah, Yamini





## Group Members



Aman Anas



Benjamin Uy



Noah Lehner

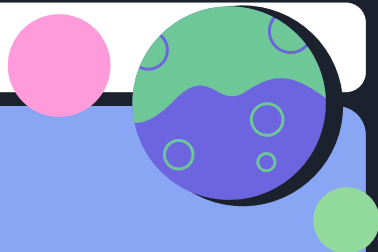


Yamini Ramadurai



# Connect4 Game

Our project is Connect4, an interactive game where the player must get four in a row either vertically, horizontally, or diagonally, in order to win. For our game, the user can create a grid ranging from an 8x8 to a 20x20 (4 connected to 10 connected, inclusive).



# The software should be able to...

## Place the piece

...place the piece in the bottom-most available row of their desired column

## Change grid size

...allow players to have the ability to change the size of their grid

## Show the pieces

...show how many pieces the client has placed

## Show connected pieces

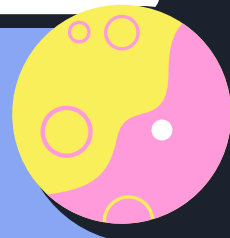
...show the max number of connected pieces until the game ends

## Determine a win

...determine when a player wins (based on # of connected pieces)

## Stop when the game is over

...stop running when a player wins and show who won



### Connect4Interface

- NUM\_PIECES\_PROMPT: String  
- PLAYER\_ONE\_CHAR: char  
- PLAYER\_TWO\_CHAR: char  
- MIN\_CONNECT: int  
- MAX\_CONNECT: int

+ main(args: String[]): void

### Piece

- type: char

+ Piece (type: char)  
+ getType(): char  
+ equals (o: Object): boolean  
+ toString(): String

game  
(created  
within  
main  
method)

0..1

0..1

(-) pieces

### Connect4Game

- rows: int  
- cols: int  
- maxToConnect: int  
- numOfMoves: int  
- gameFinished: boolean  
- draw: boolean  
- MIN\_CONNECT: int  
- MAX\_CONNECT: int

+ connect4Game(maxToConnect: int)  
+ getRows(): int  
+ getCols(): int  
+ getMaxConnectedType(): int  
+ getNumPlacedType(): int  
+ getGameFinished(): boolean  
+ getGameDraw(): boolean  
+ updateGameState(): void  
+ findConnectedDir(startCol: int, startRow: int,  
colDirection: int, rowDirection: int): int  
+ getMaxConnectedToPiece(col: int, row: int): int  
+ pieceExists(col: int, row: int): boolean  
+ getPieceAtPos(rows: int, cols: int): Piece  
+ placePiece(col: int): boolean  
+ piecesToString(): String

# UML Diagram




# Pros/Cons of our Designs

## Pros

- Design 1
  - Less classes to write
  - Potentially less unit testing involved
- Design 2
  - Each class might be simpler

## Cons

- Design 1
    - With less classes comes less compartmentalization
  - Design 2
    - Additional classes to write and document
    - More unit testing involved
- 

# Testing

## System Tests

We tested many different test cases with our system tests

## Unit and Integration Testing

We had at least 1 unit and integration test for each method of each object class

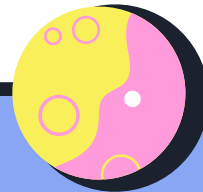


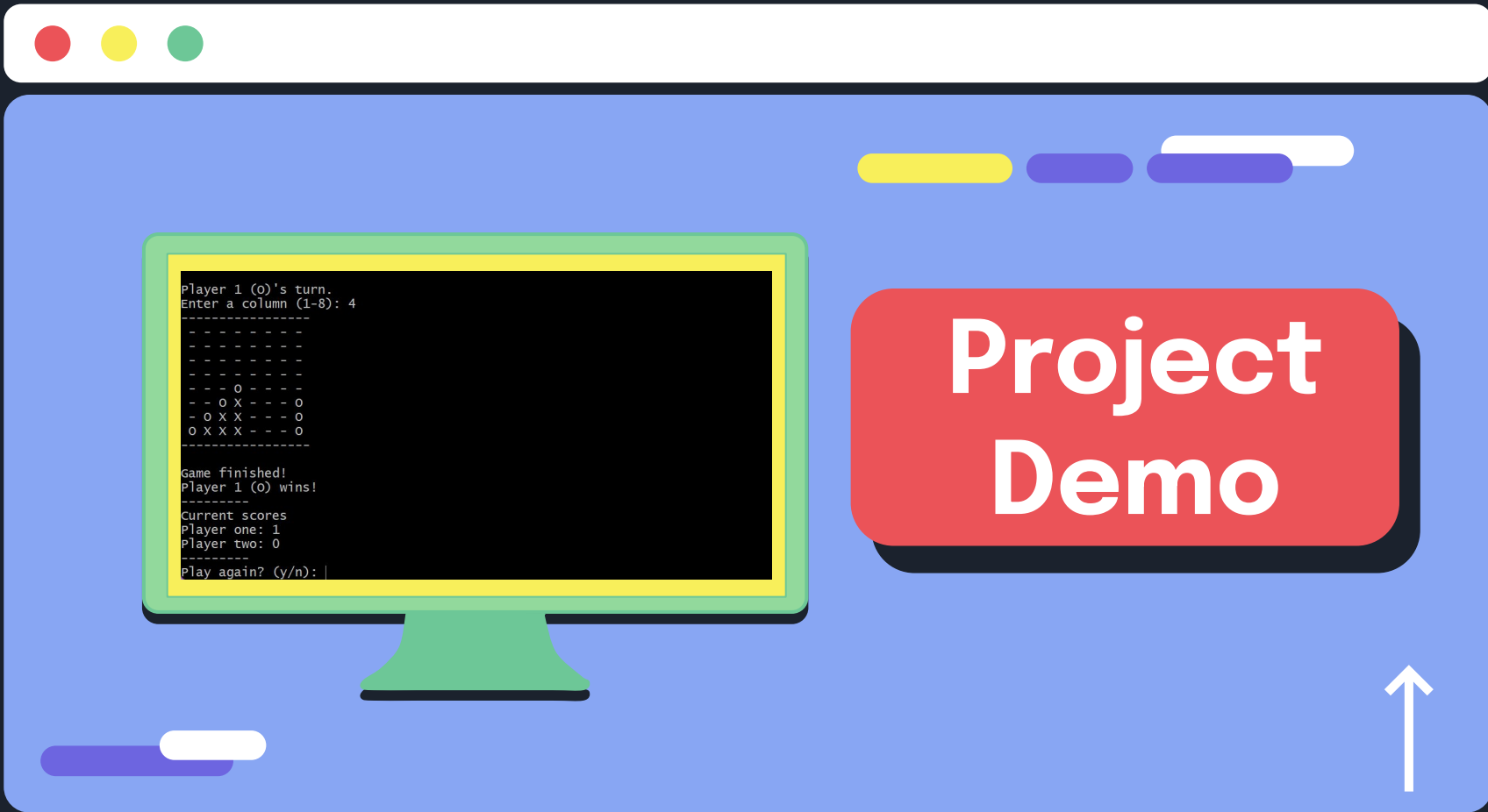
## Equivalence Classes

Inputting value from 1 to (max number of pieces connected \* 2) and possible size of grid (e.g. 8 max for 4x4 grid)

## Boundary Values

(For the column number)  
Inputting not an integer, a value less than 1, a value greater than max grid size, and (for grid size) value within 4 to 10 (inclusive)





# Project Demo







# Lessons learned


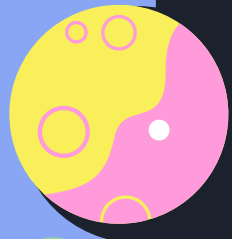
## Lesson 1

Adding inline comments and documentation helps organize and understand the code as we write it, which can be helpful during the debugging process and for other programmers looking back on the same code

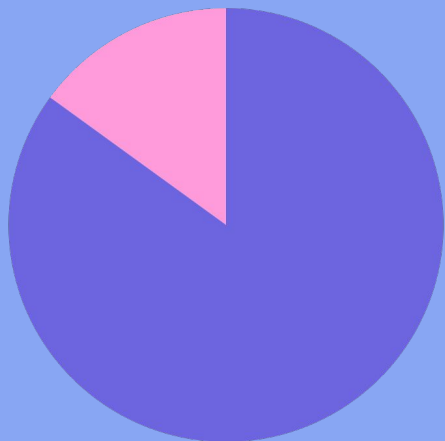


## Lesson 2

Software design is a work in progress; often not is it a fixed path.



# Project Progress



**90%**

**Complete**

**10%**

**Improvements**



## Possible Improvements

- Single player mode (player against the computer AI)

- 2+ players interacting with the game (three-way Connect4)



# Thanks!

# Questions?

**CREDITS:** This presentation template was created by  
Slidesgo, including icons by Flaticon, and infographics &  
images by Freepik

Please keep this slide as attribution

