Comprehensive Exercise Report

Team 007
Mehmet Sahin Uces 211ADB056
Parviz Atakishiyev 211ADB059

Requirements/Analysis

Week 2

Journal

The following prompts are meant to aid your thought process as you complete the requirements/analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
    - Develop a digital Connect Four game featuring classic two-player gameplay
        - Classic 7 column 6 row connect four board.
        - Turn based game between 2 players.
        - Players can drop pieces into the chosen column.
        - If a player makes horizontal, vertical, or diagonal lines of four connected pieces of the same color, the player wins the game.
        - If the board is full and no winning line is formed, then the game reaches a tie state.
- After reading the client's brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.
    - Question: What color pieces do you want to be applied in the game?
    - Answer: Red and blue.
      Question: Do you want the game to be 2D or 3D?
      Answer: 2D please.
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - Connect Four: https://en.wikipedia.org/wiki/Connect_Four
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - Children in education age
- Describe how each user would interact with the software
    - Users will be interacting with the game through input devices, specifically mouse.
    - With the input device, users can drop pieces to the columns by clicking.
- What features must the software have? What should the users be able to do?
    - Turn-based logic
    - Piece placement mechanics
    - Game result condition detection
    - UI

Software Requirements

Description

The aim of this project is to create a digital version of the traditional game Connect Four. The game should be designed to be user-friendly, allowing for two-player, turn-based matches with classic gameplay mechanics.

Project overview

The main goal of this project is to develop a digital version of Connect Four that accurately simulates the mechanics of the physical game. The key objectives are:

- Offer intuitive gameplay: The interface and gameplay should be easy for players of all ages to understand and interact with.
- Implementation of the rules: The turn-based system, piece placement, and win/tie conditions will strictly comply with the classic Connect Four rules.

Functional requirements

- Game board: A 7x6 grid representing the Connect Four game board.
- Player pieces: Pieces will be visualized with two different colors (red and blue) to distinguish two players.
- Turn mechanism: The game requires a turn-base system allowing players to play one by one.
- Input: Players will be able to click on columns of the board to place their pieces.
- Game conditions: Conditions of win and tie should be checked by the software constantly.
- Visual feedback: The game will be providing visuals for players turn, piece placement and final game outcome (win and tie).
- Game restart: After a game condition is reached, software should display a button to restart the game.

Non-functional requirements

- User-friendly UI: The user interface should be designed to be simple and easy to use, especially for young children in education.
- Clarity: The visuals of the game board, pieces, the color of the pieces and most importantly the state of the game should be clearly visible.

Black-Box Testing

Instructions: Week 4

Journal

Remember: Black box tests should only be based on your requirements and should work independent of design.

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

- What does input for the software look like (e.g., what type of data, how many pieces of data)?
    - Mouse clicks
        - Column number that is between 1 and 7, according to where the player decides to drop the piece.
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - Visual or text elements
        - Updated board visuals after piece placement.
        - Indication for current player.
        - Win or Tie message.
- What equivalence classes can the input be broken into?
    - Valid inputs
        - Click events on designated button areas. Button for each column and if required for other interactions.
    - Invalid inputs
        - Clicks outside of the button areas.
        - Clicks on the button of a column that is full.
- What boundary values exist for the input?
        - All valid inputs must be within the designated button areas.
        - All other inputs are considered invalid.
- Are there other cases that must be tested to test all requirements?
        - Testing the resizing of the game window is necessary as the game should be flexible and adapt to any changes.

Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

Test ID	**Description**	**Expected Results**	**Actual Results**
Test-1	Button click on a non-full column.	Current player's piece falls into the column and the player-turn updates to the next player.	Expected results have been achieved, current players' piece drops into the column and the next player takes the turn.
Test-2	Button click on the full column.	No action is taken and the player-turn is not changed. A visual cue might be displayed to indicate a full column.	Expected results have been achieved, when clicked on a full column, no action is taken.
Test-3	Click outside the button areas.	System does not respond, no action occurs.	Expected results have been achieved, no action.
Test-4	When a winning condition occurs.	System displays winning and losing messages accordingly, alongside with the winning line.	Expected results have been obtained, the system displays the winning player/color and the winning line (marking it green).
Test-5	All columns are filled with pieces but there is no winning condition.	Game ends, a message saying the game ended in a tie state is displayed.	Expected results have been achieved, when the board is full and there is no winning line, the system displays a tie message.
Test-6	Button click on the Restart button.	Game board should reset to the beginning and the game should restart.	Same with the expected results, after the game ends, there is a restart option.
Test-7	When the game is launched.	Game board should be constructed and the game should start.	Expected results have been achieved.

Design

Instructions: Week 6

Journal

Remember: You still will not be writing code at this point in the process.

The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- List the nouns from your requirements/analysis documentation.
    - Row, column, pieces, board, lines, color, mouse, click, winning, tie, restart, visuals,turn.
- Which nouns potentially may represent a class in your design?
    - Pieces, Board
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - Piece
        - Color
        - Row
        - Column
        - visuals
    - Board
        - Pieces
        - Mouse
        - Click
        - Wining
        - Visuals
        - Turn
    - Main
        - Game loop
        - Visuals
        - restart
- Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.

  Design 1
    - Pieces
        - Color
    - Board
        - pieces
        - Row
        - Column
    - Game
        - Board

        - Current player
    Pros
        Simplicity
        Clear separation of classes
    Cons
        Limited functionality and scalability

Design 2

- Pieces
    - Color
    - row,column
    - drop
- Board
    - Pieces
    - Winning
    - Tie
- Player
    - Turn
    - Mouse
- Main
    - Board
    - Players
    - Game loop

    Pros
        Versatility
        Extensibility
    Cons
        Complexity

- Which design do you plan to use? Explain why you have chosen this design.
    - We are planning to use our Design 1 as it is simple and clear. Having game, board, and pieces (circles in our case) classes provides good structure to start the implementation. As connect 4 is not a big game, we are trying to keep the implementation process as simple as possible.
- List the verbs from your requirements/analysis documentation.
    - Drop, visualize, restarting, winning.
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - Drop:
        - Board
    - Visualize:
        - Board
        - Game
        - Circle
    - Winning:
        - Board
    - Restarting:
        - Game

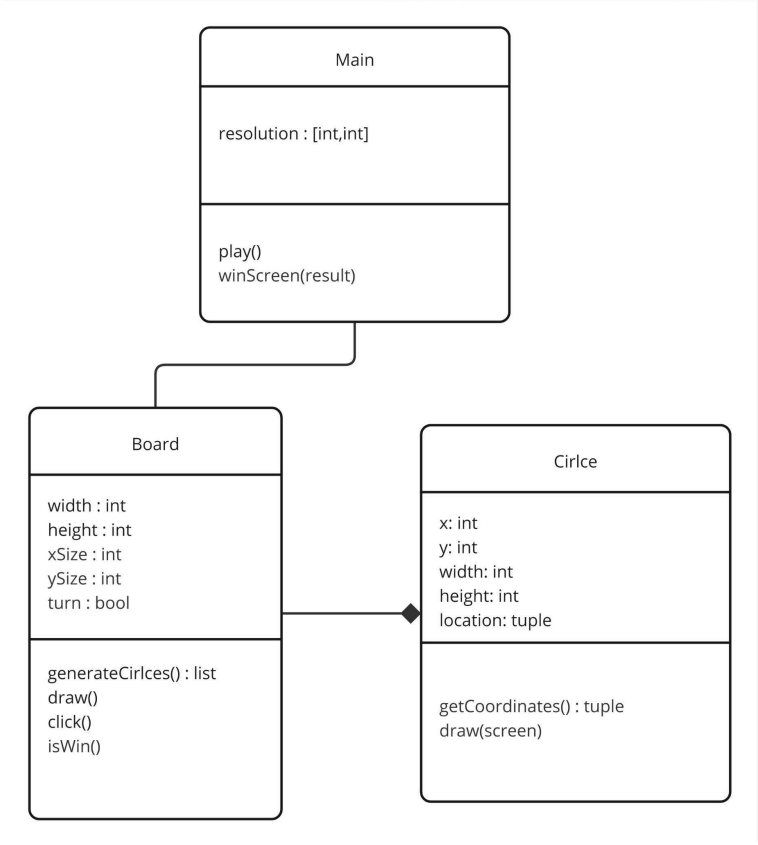
Software Design

Use your notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will be used in the software. Your design should include UML class diagrams along with method headers. ***Prior to starting the formal documentation, you should show your answers to the above prompts to your instructor.***

Main class:
Main class will mainly include the game loop. It will also have a function to show who won the game. As the main (game) class interacts with board and pieces lots of changes will be made here.

Board class:
Board class will be handling the game board. Generating the initial position and creating Circle objects will be by the function generateCirlces().
draw() function in the board class will initiate the drawing of the board and Pieces
To know where the player is clicking a click() function that will “drop” the piece depending on the mouse position will be implemented in click()
And a function to check the winning conditions or tie cases will be checked in isWin().

Pieces (Circle) class:
Circle class objects will be the representatives of each piece in the game.
Circle objects will be created by the Board class.
Circle objects will be responsible for holding the information about the specific piece in the game such as location and color.



Implementation

Instructions: Week 8

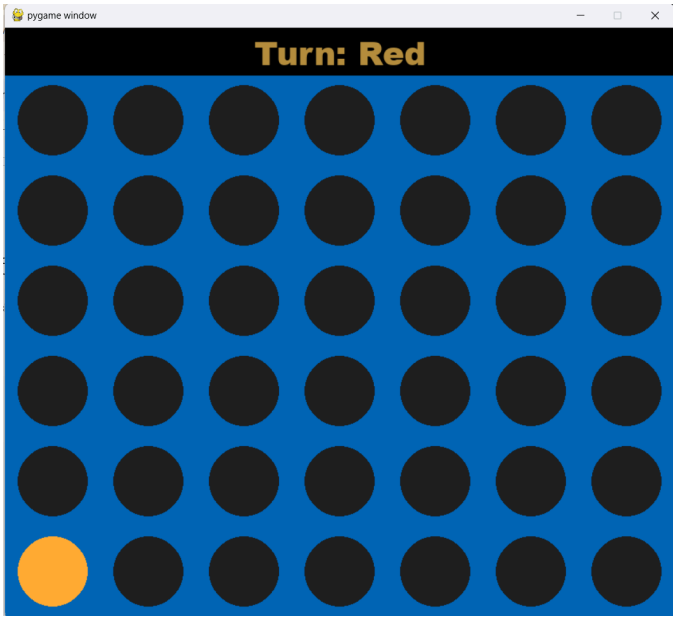
Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

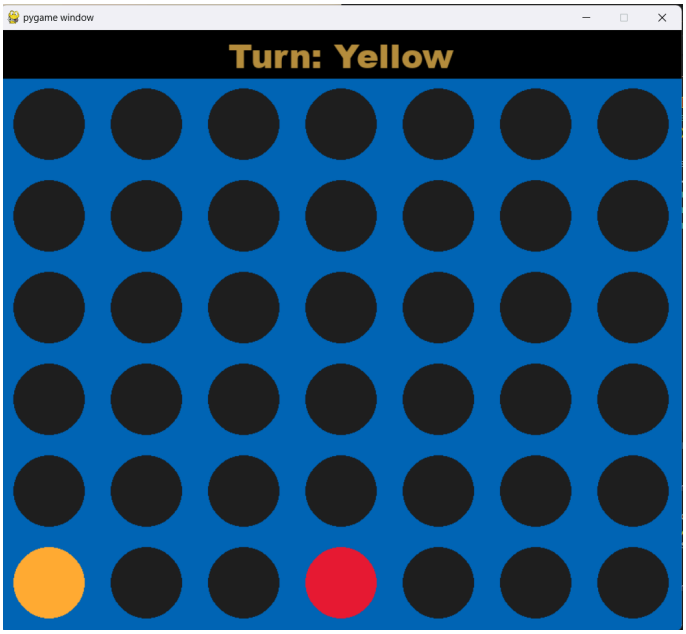
- What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.
    - Object-Oriented Programming (OOP)
        - Classes like Board and Circle will be created to represent the game elements, encapsulating their behavior and attributes.
    - Waterfall Development Method
        - As a small team, the waterfall method is the easiest and quickest development method we can use. We will follow the steps from requirements all the way to deployment.
    - UML Diagram
        - UML diagrams will guide the design and implementation process. The class diagrams provided earlier show the structure of the software, including classes, attributes, and methods, as well as the relationships between them.
    - Software Development Life Cycle (SDLC)
        - The SDLC provides a framework for planning, creating, testing, and deploying software applications.
        - Each phase of the SDLC will be followed, starting from requirements gathering and analysis, through design, implementation, testing and deployment.

Implementation Details

After the user enters the game, he/she has to decide a column to drop a piece. The game starts when the player one drops the yellow piece.

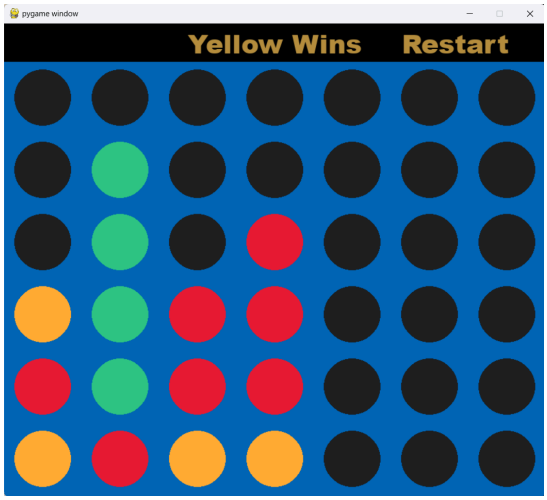


The next turn is for Player2 which has to drop the Red piece.

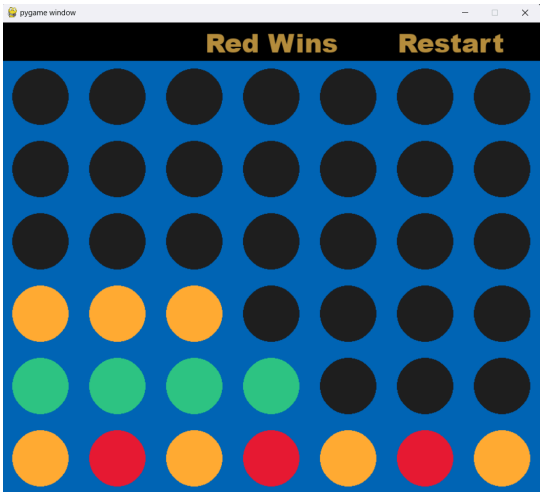


The game will continue in this manner until one of the predetermined win conditions or a tie state is reached. Once a winning line has been established, the game will display the line in green to indicate the winner.

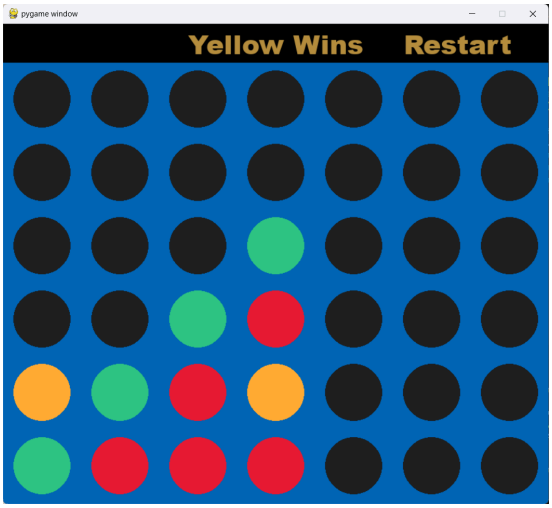
Win condition (column line)



Win condition (row line)

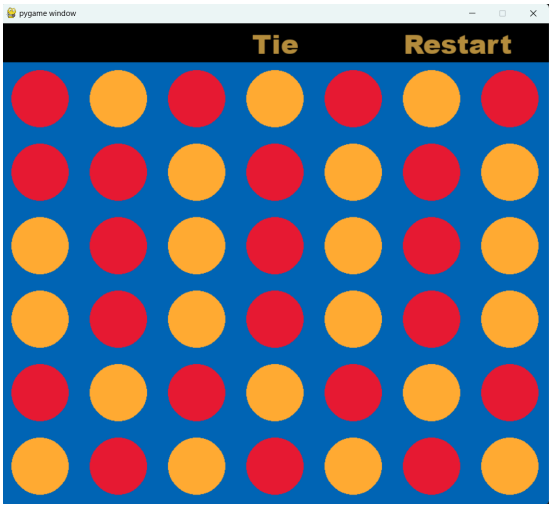


Win condition (diagonal line)



The game will reach tie state if the board is full and there is no winning line found. The state of the game will be displayed in the topbar above.

Tie state



Upon the fulfillment of one of the conditions, the player can initiate a new game session by pressing restart.

Testing

Instructions: Week 10

Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
    - There is no change in requirement selection. Game is constructed according to them.
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
    - Board
        - Empty Board initialization
        - Testing board sizes
        - Click on valid area
        - Checking win condition
        - Handling outside clicks
    - Circle
        - Drawing circles through Board

    - Test board creation
    - Test circle creation
    - Test click on valid column
    - Test win condition in all directions
    - Test tie
    - Test restarting
    - Test outside click handling
    - Test turn change

Testing Details

Test_board_creation

This test initiates the board and checks the sizes of it.

Test_circle_creation

This test creates a dummy circle and checks the location and center of it to assess.

Test_turn_change

This test checks whether or not after a click on the board the turn is change to the next person.

Test_placement

This test adds a new circle based on click to the board and checks the existence of it.

Test_outside_click

This test checks if the handling for outside click are correct.

Test_isWIn

This test checks the winning conditions by creating horizontal vertical and both diagonal directions win states.

Test_board_tie

This test creates a tie condition and checks if the tie state is achieved.

Test_restart_button

This test, after the game ends, checks if the restart button restarts the game.

Presentation

Instructions:Week 12

Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

- Give a brief description of your final project
    - The goal is to create a digital version of the traditional game Connect Four. The game should be allowing for two-player and turn-based matches.

- Describe your requirement assumptions/additions.
    - We had two types of requirements in our project which were functional and non-functional. Functional requirements include board, visuals, piece, turn mechanism , input and so on. While non functional requirements consisted of clarity and the user friendly interface.
- Describe your design options and decisions. How did you weigh the pros and cons of the different designs to make your decision?
    - We have compared modular and monolithic design.
    - Modular design was overcomplicating our simple game.
    - That is why we have chosen to continue with monolithic design.
    - Monolithic design provides:
        - Simplicity
        - Faster development
        - Easy to develop with static requirements such as connect 4 game
- How did the extension affect your design?
    - No extensions are added.
- Describe your tests (e.g., what you tested, equivalence classes).
    - Main components that make the game playable and crucial to the usability of the application has been tested.
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - OOP
    - Waterfall development
    - SDLC
- What functionalities are you going to demo?
    - Finished connect 4 game application with full playability.
- Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
    - Parviz starts the presentation, and has to present from the first page to the 6th page.
    - Mehmet takes from the 6th and continues till the end
- Other notes:
    - <<Insert notes>>