Comprehensive Exercise Report

Team 3 of Section 001

Hayes Derbyshire hcderbys, Maddy Cole mjcole3, Bella Adriano iradrian, and Nathan Holmes nlholmes

NOTE: You will replace all placeholders that are given in <<>>

Requirements/Analysis

Instructions: http://go.ncsu.edu/csc116-ce-reqt

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.
    - The Connect Four application will simulate playing the actual game against another player, one person winning by connecting four checkers in three possible orientations.
        - 6 rows of checker spaces and 7 columns of checker spaces
        - 2 players
        - 21 checkers for each player
        - Players take turns with one checker placement per turn
        - End game when one player gets four checkers in a row horizontally, vertically, or diagonally
        - End game if board fills completely with no winner
        - Game restarts until user decides to quit
- 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 you hand and ask the client (your instructor) the questions; make sure to document his/her answers.
    - Q: Do we need to use a GUI? A: It would be nice, but it is not necessary.
    - Q: Would a GUI be easy to implement after making the majority of the program? A: Yes. It is easier to implement a GUI afterwards.
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - It uses what we have learned this semester. We may need to research GUIs if we decide to implement one.
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - Small child or older whose goal is entertainment
- Describe how each user would interact with the software
    - The user will select a column. The row will be decided by the program according to the rules.
- What features must the software have? What should the users be able to do?
    - The software must have a feature that updates the board every time a player makes a move. It will then check if there is a winning play made. The score will be displayed as well as pop-ups for winners. The users should be able to see the updated board before making their next move. A button will be implemented above each column where the users can click to make a move.
- Other notes:
    - Increased functionality.
    - Tournament style (e.g. best of 3, best of 10)
        - Playing through all games even if win occurs before all games are finished
    - Instructions menu displayed as a menu option

Software Requirements

The project requires that we create an application that simulates a game of Connect Four between two players (the other player not being the computer). A player can win the game by either getting four of the same checkers in a horizontal row, vertical row, or diagonal line. If a player wins the game, the program will restart and prompt if the players want to play another game. If the players decide not to play another game, the program will quit. The board needs to consist of 6 rows and 7 columns, with 21 checker pieces for each player (possibly of different colors). In order for the game to be fair, each player takes a turn putting their respective checker piece in their desired spot. If the board fills, and there is no winner, the game will end because there are no more possible places for the players to put their checker pieces, which they will have run out of if that situation were to happen.

In order for the program to work, the program must update the board every time a player makes a move, and check if there is a winning move that was made. If a winning move is made, a pop-up will display which player has won, and prompt if they would like to restart. For each game, a score system will be displayed to show how many times each player has won until they decide to quit. The type of player that would be interested in this type of game can range from all ages. Therefore, the application would have to be user-friendly and clear enough for the youngest of users to play it, and the oldest of users to play it. There will be instructions at the beginning of each game in order for the users to understand how to play, if they do not originally know how to. In order for the players to interact with the application, they will select a column and the row will be picked by the program.

Black-Box Testing

Instructions: http://go.ncsu.edu/csc116-ce-bbt

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)?
    - The input will be the row and column position for where the user made their move as an integer array, treating the board like a table/grid.
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - The output will be a boolean value. If four checkers have been connected it returns true.
- What equivalence classes can the input be broken into?
    - Already filled spaces
    - The rows fill up according to how many times each person plays a specific column
- What boundary values exist for the input?
    - Row boundaries are integers from 0-5
    - Column boundaries are integers from 0-6
- Are there other cases that must be tested to test all requirements?
    - Invalid types including double values in boundaries
    - Negative numbers
    - Numbers above boundaries
    - Unreachable spaces (higher vertically than a space in the same column)

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**
testStringType	[PLAYER1] Enter column: one	Invalid input	
testFilledSpace	Requirements: Column has one space left [PLAYER1] Enter column: 1 [PLAYER2] Enter column: 1	Invalid input	
testAboveBoundary	[PLAYER1] Enter column: 7	Invalid input	
testFourInARow	Requirements: Player 1 has checkers in the same row in columns 3, 4, and 6 and lines up with an empty space in column 5 [PLAYER1] Enter column: 5	Player 1 Wins! *Prompts for restart or quit*	
testDraw	Requirements: One space left in column 5 and no possibility of four in a row for Player 2 [PLAYER2] Enter column: 5	Draw! *Prompts for restart or quit*	

testHorizontalWin	**Requirements: Pieces in second row, and in the 0, 1, 2, 3 section of the row.**	Red wins	Red wins
testVerticalWin	Requirements: Pieces in first column, and in 0, 1, 2, 3 column.	Black wins	Black wins
testDiagonalRight	Requirement: Pieces in 0, 1, 2, 3 row and in 0, 1, 2, 3 columns.	Red wins	Red wins

testDiagonalLeft	Requirement: Pieces in 0, 1, 2, 3 row and in 3, 2, 1, 0 columns.	Black wins	Black wins
testTie	Requirement: Board is filled with all pieces, but the board finds no winner	Draw	Draw

Design

Instructions: http://go.ncsu.edu/csc116-ce-design

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.
    - Board, rows, columns, spaces, players, checkers, turns, win, GUI
- Which nouns potentially may represent a class in your design?
    - Board, checker, GUI
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - Empty space (board), filled space (board), color (checker), position (checker)
- 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.
    - One design has three classes ConnectFour, ConnectFourGUI, and Checker. An attribute included in the ConnectFour class could be the players, width and height, and the board 2d array. This design would be nice because it would be better looking because it has a user interface instead of a console UI. However, this could be time consuming and take away from fine-tuning the actual functionality of the program. On the other hand, a design with the classes of ConnectFour and Checker would be another good approach that would use similar fields to the example above. This design approach would be nice for establishing the foundation of the program and understanding how it is woven together. Adding a GUI is an after effect and would be made easier with an added understanding of the program.
- Which design do you plan to use? Explain why you have chosen this design.
    - We plan to use the design that does not include a GUI at first. The end goal is to implement a GUI but we want to inherently start with the barebones of the program and fill it in.
- List the verbs from your requirements/analysis documentation.

    - Place (fill space), end (end game), quit (quit game), alternate (take turns), restart (repeat game), win (checks for four in a row), update (updates board)
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - Place (Checker), win (ConnectFour), update (ConnectFour)

Software Design

ConnectFour
+board: Two-dimensional Array
+main (args: String[]): void +checkVert (board: Checker[]): Checker +checkHoriz (board: Checker[]): Checker +checkDiag (board: Checker[]): Checker +updateBoard (board: Checker[]): Checker[]

Checker
-color: String
+getColor (): String +setColor(c: String): void +toString (): String +equals(o: Object): boolean

ConnectFourGUI
-streakToWin: int -rows: int -cols: int
+actionPerformed(e: ActionEvent): ConnectFour +ConnectFourGUI(): void

ConnectFour uses arrays of Checker Objects and uses the objects' color.The Checker class implements the color of the checkered pieces by using a getter method, a setter method, a toString method, and an equals method. ConnectFour uses a main method, a method to update the board, and three different methods to test if a game has been won: horizontally, diagonally, or vertically. We decided not to implement a GUI at first because it would take a little bit more time, and it's more important to us to get a working program before making it look cool.

UPDATE:

ConnectFourGame
-streakToWin: int -height: int -width: int -board: char[][]
+ConnectFourGame(streakToWin: int): void +getBoard(): char[][] +getStreakToWin(): int +checkHorizontal(): char +checkVertical(): char +checkDiagonalDR(): char +checkDiagonalDL(): char +dropPiece(col: int, color: char): int +toString(): String +equals(o: Object): boolean

ConnectFourGUI
-streakToWin: int -rows: int -cols: int -turn: character -turnCount: int -buttons: JButton[][] -turnLabel: JLabel -numRowsAndColumns: JTextField -game: ConnectFourGame -board: char[][]
+ConnectFourGUI(): void +actionPerformed(e: ActionEvent): ConnectFour +main(): void

ConnectFourGame and ConnectFourGUI use two-dimensional arrays, representing the board, containing characters representing game pieces. The pieces are differentiated by color. ConnectFourGUI uses a two-dimensional array of buttons that lets the users interact with the board. ConnectFourGame constructs a game, updates the board, and checks for a win. ConnectFourGUI uses a main method, responds to user actions, calls methods in ConnectFourGame in order to play, and displays graphical game elements.

Implementation

Instructions: http://go.ncsu.edu/csc116-ce-imp

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.
    - A large part of the Connect Four project will be the navigation of the board with a two dimensional array of [rows][columns].
    - For loops will be widely used to navigate through the two dimensional array.
    - Knowledge of making a working GUI.

Implementation Details

We decided to implement a GUI for easier gameplay. In order for the program to navigate the board, we needed to use a two dimensional array of rows and columns. To navigate those arrays, we needed to use for loops, so that the program can check for any kind of win. If there was no win, then there would be a draw. Arrays are helpful for this kind of game because of the set-up of the game board itself. A GUI would help visualize actual game play, and is able to accurately update the board every time.

Testing

Instructions: http://go.ncsu.edu/csc116-ce-test

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.
    - The only requirements we have changed for the black box test plan would be to now test some of the GUI attributes instead of just the program of the game.
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
    - ConnectFour
        - Testing horizontal win
        - Testing vertical win
        - Testing diagonal win
        - Testing draw
            - Equivalence class: they all depend on how the board is filled up
    - <<Insert class and tests for each class>>

Testing Details

<<Use your notes from above to write your test programs and complete this section of the formal documentation by creating a list of your test programs along with descriptions of what they are testing. You will also complete the black-box test plan by running the program and filling in the Actual Results column.>>

1. Testing a Horizontal Win
    a. Our test case uses a height of 4 spaces, a width of 4 spaces, and a win streak of 4 as well.
    b. We have to create a dummy board in order to test against the actual board that is being presented by the program.
        i. A win by the red player.
2. Testing a Vertical Win
    a. Our test case uses a height of four spaces, a width of four spaces, and a win streak of 4 spaces.
    b. We created a board to test against the board the program we wrote was showing.
        i. A win by the black player.
3. Testing a Diagonal Win to the right
    a. Our test case uses a height of four spaces, a width of four spaces, and a win streak of 4 spaces.
    b. The board had to be going from bottom (rows) to top (columns and then from bottom (rows) to top (columns) again.
        i. A win by the black player.
4. Testing a Diagonal Win to the left
    a. Our test case uses a height of four spaces, a width of four spaces, and a win streak of 4 spaces.
    b. The board had to be going from bottom (rows) to top (columns) and then from top (rows) to bottom (columns).

Presentation

Instructions: http://go.ncsu.edu/csc116-ce-presentation

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
    - We are supposed to simulate a Connect Four game by letting the players choose the number of rows and columns they would like on their board.
- Describe your requirement assumptions/additions.
    - The assumptions we made about our requirements are that we have to figure out three ways to recognize three different kinds of wins on the board: horizontal row, vertical row, and diagonal line. We also have to figure out a way to update the board every single time, and determine if there is a winner or a tie.
- Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
    - The two design options we had were to either implement a GUI or have the game be played directly on the console. Both would have worked, but the GUI would have looked nicer and might be easier to play, visually. The only con when weighing one design over the other, is how to efficiently and correctly use a GUI for the game because the GUI has to reflect the internals of the game.
- How did the extension affect your design?
    - We were eventually able to use a GUI for our Connect Four game.
- Describe your tests (e.g., what you tested, equivalence classes).
    -
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - How to efficiently interact classes and implements GUIs
    - How to deal with “customers” and the broad descriptions they give.
- What functionalities are you going to demo?
    - Being able to play the game according to the rules
    - The game working as efficiently as possible
- 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.)
    - Maddy: Requirements and analysis
    - Bella: Design and testing
    - Hayes: GUI and demo
    - Nathan: everything else

Grading Rubric

http://go.ncsu.edu/csc116-ce-grading