Comprehensive Exercise Report

Bakhtiyor Karimov - 201ADB007

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.
    - The game is for 2 players, players select one of two disc colors at the start of the game and try to put 4 discs of the same color in a row, whoever first stacks their own 4 discs vertically, horizontally or diagonally wins the game
      .
        - <<Insert known requirements from client description, add more bullets as needed>>
        - Game should be for 2 players
        - The game should have a grid to place the discs and 21 blue and 21 yellow discs
        - The game must comply with rules of winning - stacking 4 discs in a row horizontally, vertically, diagonally.
- 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.
    -
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - <<Insert answer>>
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - Kids from 6 years old and above
- Describe how each user would interact with the software
    - Users click start the game button and the game starts with a blue disc as for the client's brief. Then they drop the disc over the grid aiming to place 4 of them in a row. Moves are implemented sequentially between players - one after another turn by turn.
- What features must the software have? What should the users be able to do?
    - Game must display the grid and colored discs
    - Players are able to drop their disc from a particular column
    - Game checks for the row of discs to determine the winner
    - Once the game is over, display the winner
- Other notes:
    - Planned to develop on C++, worst case scenario Python
    - For C++, raylib.h library is used, for python turtle module

Software Requirements

<<Use your notes from above to complete this section of the formal documentation by writing a detailed description of the project, including a paragraph overview of the project followed by a list of requirements (see lecture for format of requirements). You may also choose to include user stories.>>

The project is aimed to make an application adaptation of the classic game, Connect 4. The application allows two players to play with each other, by selecting their disc colors and then drop a disc from a particular column of a grid. Moves are performed turn by turn among players. The game is over when one of the players gets 4 discs of their color in a row, horizontally, vertically, or diagonally. The software game follows the same rules as the original game and the game itself must be GUI.

- The application should be two player based - allowing two players to play against each other
- The application should include a grid and colored discs (blue, yellow).
- The application should have a started button to start the game when it is run
- The game adopts the same rules from the original game - including winning rules - 4 discs in a row horizontally, vertically, diagonally.
- The application must allow players to be able to drop their disc from a particular column
- The application checks for the row of discs to determine the winner
- The application should involve the rules for those who are playing for the 1st time.
- The application shows the winner

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)?
    - Start a New Game - Key Press, A single click
    - Select a column (to drop the disc) - Numeric, a single piece of data, column number selected by user
    - Game rules - Text, piece of data - depends on the length of text
    - Start a New Game - Key Press, A single click
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - Visual Layout - Graphical (Pixels), object instances will be 3-5 on the GUI of the game
    - Game Status - Text, A single piece of data - Text
    - Game Result - Text, A single piece of data - Text
    - Game Start Window - Text, A single piece of data - Text
- What equivalence classes can the input be broken into?
    - Column Selection:
        - Valid Input: User selects a column between 1-7
        - Invalid Input: User selects a column already full or selects out of range
    - Start a New Game:
        - Valid Input: User clicks to start a game or exits
        - Invalid Input: User makes no clicks or goes to invalid option

- What boundary values exist for the input?
    - Column selection - number of columns 7, boundary 1-7
- Are there other cases that must be tested to test all requirements?
    - Test when a player gets 4 discs in a row (horizontal or vertical or diagonal)
    - Test player throws a disc in one column, but fills another column
    - Test game over - game exits automatically without a pop-up window
    - Game status - player turns changed, status for turns remain same
- Other notes:
    - <<Insert notes>>

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**
01	Start a game with button	Pop-up window closes, game starts to play	Game is playable after clicking Start game button
02	Game status show colors	Game status shows player disc color	Game status is shown by player disc colors
03	Disc throw in a column for user 1	Player 1 selects a column and puts the disc	Disc is places at the bottom of column
04	Game status update for a player	Game status is updated respectively in turn of playing player	After one turn, next player’s turn is shown by their color
05	User is unable to place a disc to full column	Column remains unchanged	Column and disc color remains unchanged
06	Game Over condition when 4 disc in row	Game status shows winner	Game status shows the winner player
07	Pop-up window when game is over	“Game Start” window pops-up	Again pop up window reappears upon completion of game

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.

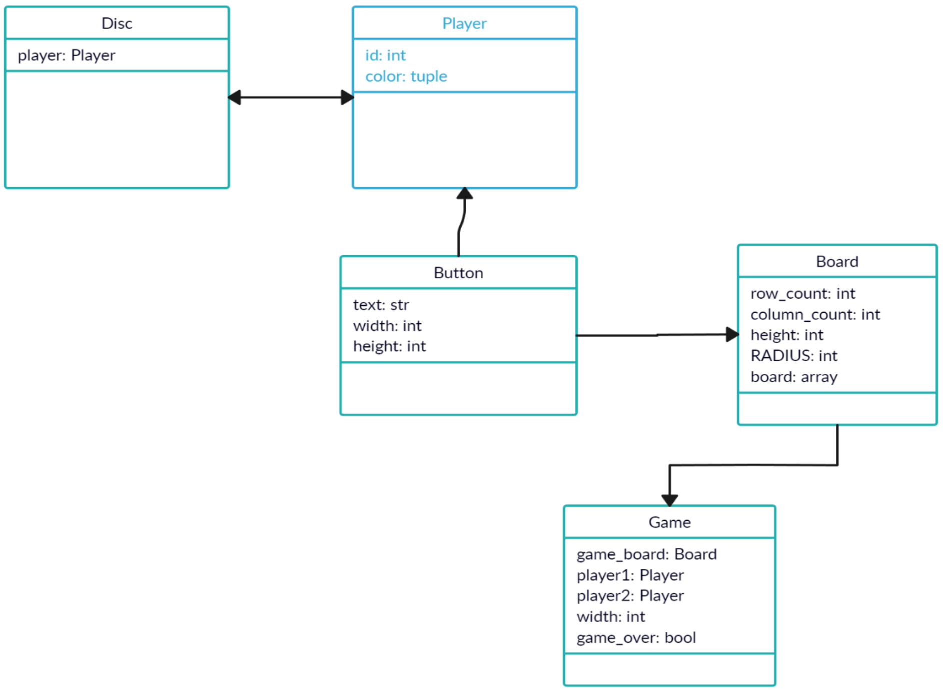
- List the nouns from your requirements/analysis documentation.
    - Player, Discs, Grid, Rules, Button
- Which nouns potentially may represent a class in your design?
    - Player, Disc, Grid
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - Class Player
    - Class Disc
    - Class Grid
- 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.
    - **Classes:**
    - Players
    - Disc
    - Grid/Board
    - Button
    -
    - **Attributes for Players**:
    - Player number, Color
    -
    - **Attributes for Disc:**
    - Color, columns
    -
    - **Attributes for Grid:**
    - Rows, Columns, 2D Row
- **Pros:** Simple design
- East to build
- **Cons:** Limited functionality
- Not possible to create rules, buttons in flexible manner

    - **Classes:**
    - Players
    - Disc

    - Grid
    - Button
    -
    -
    - **Attributes for Players**:
    - Player number, Color
    -
    - **Attributes for Disc:**
    - player
    -
    - **Attributes for Grid:**
    - Rows, Columns, Height, Radius, board
    -
    - **Attributes for Button:**
    - Text, width, height, pos, elevation
    -
- **Pros:** Detailed functionalities in OOP
- Able to handle relatively complex functions
- **Cons:** More functionality
- Difficult to implement

- Which design do you plan to use? Explain why you have chosen this design.
- Going with the 2nd design, taking into account the project is not big and there is a big probability of creating numerous object instances for buttons. Thus more functional design suits better.
- List the verbs from your requirements/analysis documentation.
    - Put, Show, Win, Update, Change
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - Win and Update for Player and Disc classes respectively
- Other notes:
    - <<Insert notes>>

Software Design



1

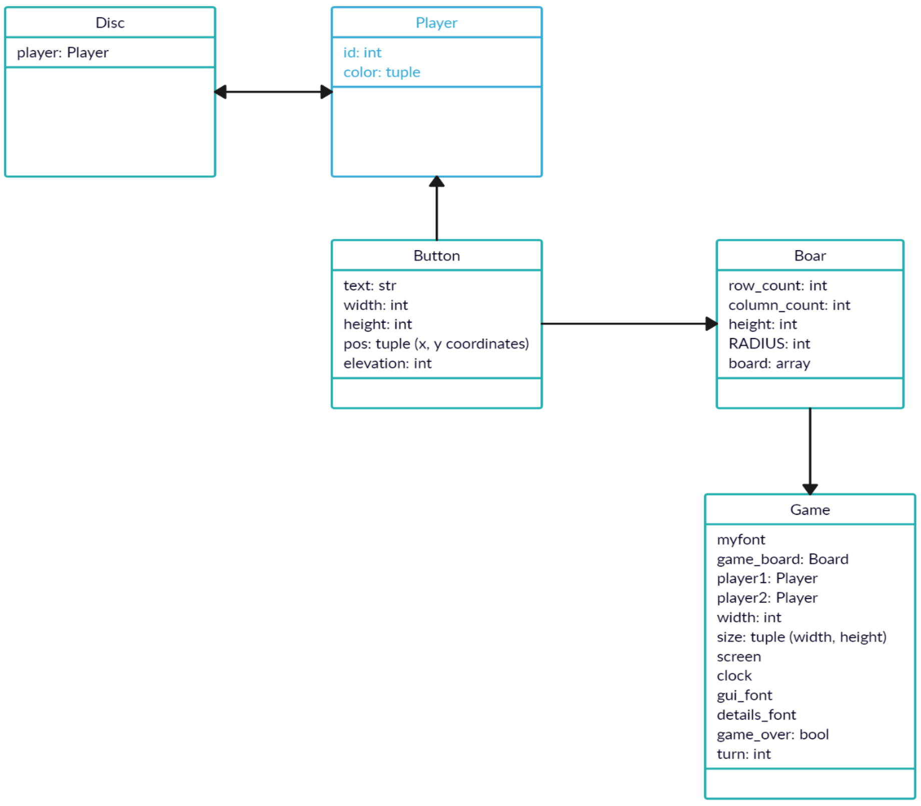
2

3

4

Initial raw design, UML diagram

However, during the implementation there were some additions, below updated UML diagram for documentation



1

2

3

4

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.
    - Designing - Initial design and draft view of the software
    - During and after coding, Debugging
    - Refactoring
    - Documentation - short readme file could be used as a to-the-point documentation, however, current report shows detailed information as well as developer's notes as a full version of software documentation

Implementation Details

<<Use your notes from above to write code and complete this section of the formal documentation with a README for the user that explains how he/she will interact with the system.>>

Current README is added to GITHUB repo as well.

This is a Python-based implementation of the popular game "Connect Four". The game is played between two players, taking turns to drop colored tokens into a seven-column, six-row vertically suspended grid. The objective of the game is to form a line of four tokens of the same color horizontally, vertically, or diagonally.

The game is built using the Pygame library and NumPy for the game logic.

Features:

- Graphical User Interface (GUI) using Pygame.
- Button interaction for starting the game and quitting the game.
- On-screen text for game descriptions and other details.
- Player's turn indication with colored tokens.
- Game over scenario when a player forms a line of four tokens.
- Winning player announcement after each game.

Instructions:

- Run the Python script.
- The game window will open with the game description and two buttons: "Game Start" and "Quit".
- Click "Game Start" to start the game.
- Players take turns to click on the column they want to drop their token in.
- The game ends when a player forms a line of four tokens or when all slots are filled.
- A message will be displayed announcing the winner.
- After the game ends, you can start a new game by clicking "Game Start" again or quit the game by clicking "Quit".

Requirements:

- Python 3.x
- Pygame
- NumPy

Make sure you have the above requirements installed on your system to successfully run the game.

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.
    - <<Insert answer>>
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.

    - **Board Class:**

    - Equivalence Classes:
    - - Valid board dimensions (e.g., 6 rows x 7 columns)
    - - Invalid board dimensions (e.g., 0 rows x 0 columns)
    - Boundary Values:
    - - Minimum and maximum row and column counts

    - Rest of the classes don't possess specific equivalence classes or boundary values as they are not built involving complex functionality in that level. The focus of my testing mainly focused on the behavior and interaction of the game board and game 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.>>

Usability testing:

Game Start and End: Ensure user can easily identify how to start the game and end it

- There are 2 buttons for users to interact, Game Start button and Quit button.

Disc Placement: Ensure users are able to understand how to place the discs, hover over effect

- Game includes rules on the first intro window where users are instructed how to place the discs, additionally, they can understand it after starting the game, mouse hover effect is enabled.

Error handling on columns: Ensure there are no errors when user drops disc on full column

- Game remains same, there are no errors, or odd reactions when user drops a disc on full column

Text Clarity: All texts involved in the game must be readable and understandable for users

- Texts are written in plain english and their color is distinct from background and shadow colors

Board Dimensions: Ensure board dimensions are as the same as game rule requires

- Board is 6x7 and it remains same during 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
    - <<Insert answer>>
- Describe your requirement assumptions/additions.
    - <<Insert answer>>
- Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
    - <<Insert answer>>
- How did the extension affect your design?
    - <<Insert answer>>
- Describe your tests (e.g., what you tested, equivalence classes).
    - <<Insert answer>>
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - <<Insert answer>>
- What functionalities are you going to demo?
    - <<Insert answer>>
- 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.)
    - <<Insert answer>>
- Other notes:
    - <<Insert notes>>

<<Use your notes from above to complete create your slides and plan your presentation and demo.>>