Comprehensive Exercise Report

Team <<X>> of Section <<000>>

Wilfried RUIZ, ID = 240ADM002
Guy Leonard KAMGA FOTSO WAFO, ID = 240ADM005
Elias CREMNITER, ID = 240AEB020
Kevin RANDRIANIMANANA, ID = 240AEB027

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

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 connect 4 is a game which plays to 2 players in which the winner is the player who put first 4 discs in the grid.>>
        - <<a grid to put the discs in; the grid has a sliding bar in the bottom; red discs and yellow discs; put the disc on top and drop it to let it falling; the players play each their turn; to get first 4 discs successive (row, column or diagonal)>>
- 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.
    - <<How many rounds in this game ? How to play if there are more than 2 players ? What could be the consequences for the loser and the awards for the winner ?>>
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - <<There aren't any topics we are unfamiliar with.>>
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - <<Everybody can play to this game except children who has an age less than 4 years old.>>
- Describe how each user would interact with the software
    - <<On a computer, the players can place their discs by clicking on the desired column; players can write the number of the column to place their disc; players can use controllers to navigate and select the column, then press a button to place their disc>>
- What features must the software have? What should the s be able to do?
- <<A graphical interface to allow to players understanding quickly how to play; the players play in the same computer (local interface)>>
- Other notes:
    - <<Notes/rules/descriptions are available on Ortus>>

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.>>

Project Overview:

The project involves implementing the rules of the game Connect Four in a computer environment. Connect Four is a strategic game for two players, where each player takes turns selecting a column to drop a token of their color (red or yellow) into the grid. The goal is to align four tokens of one's own color horizontally, vertically, or diagonally before the opponent does. The game is played on a 6x7 grid. The game ends when the grid is full or when a player successfully aligns four tokens. The player who aligns four tokens first wins the game.

Project Requirements:

1. The game must allow two players to take turns.
2. Players must be able to place their token in a non-full column of the grid.
3. The game must detect and signal when a player aligns four tokens.
4. The game must end when the grid is full or when a player aligns four tokens.
5. The game must clearly display the current state of the grid after each move.
6. Players must be able to start a new game after the end of the previous one.
7. The game must include a user-friendly interface for players to select columns to place their tokens.
8. The game must handle draw cases when the grid is full without any player aligning four tokens.
9. The program must be developed in an appropriate programming language with suitable data structures to represent the grid and tokens.

User Stories:

1. As a player, I want to be able to select a column to place my token so that I can take my turn.
2. As a player, I want to be notified when I align four tokens to win the game.
3. As a player, I want to see the current state of the grid after each move to understand the game's progress.
4. As a player, I want to be able to start a new game at any time to continue playing.
5. As a player, I want the game to automatically detect a draw when the grid is full without any player aligning four tokens.

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)?
    - The inputs are the click done with the mouse and the key (integer) put on the keyboard who will place the disc.
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - The software will show some evolutions (disc placed) on the screen and text by moment.
- What equivalence classes can the input be broken into?
    - A Disc and player class.
- What boundary values exist for the input?
    - The Column 1 and 7 are the boundaries values for the game board and players can't place a disc outside.
- Are there other cases that must be tested to test all requirements?
    - Yes, we should assure that if a column is full the player can't place a disc. The number of turn should not exceed the number of cases on the gameboard. One disc and change turn for player.
- 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**
1	Try to put a disc on the right column. In column 1 and 8	If column number between 1 and 7 succeed if no fail	
2	The first player place his disc then the second player	If the disc is placed, it announces the end of the turn	
3	Testing place on the column (if full)	If column full and player try putting disc : error message	
4	Test if game board full (draw)	If the number of tokens is 49 the game ends	
6	Test end game	If 4 disc aligned, show the winner screen.	

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.
    - <<Insert answer>>
- Which nouns potentially may represent a class in your design?
    - <<Insert answer>>
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - <<Insert answer>>
- 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.
    - <<List at least two design options with pros and cons of each>>
- Which design do you plan to use? Explain why you have chosen this design.
- List the verbs from your requirements/analysis documentation.
    - <<Insert answer>>
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - <<Insert answer>>
- Other notes:
    - <<Insert notes>>

Software Design

<<Use your notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will 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.***>>

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.
    - <<Insert answer>>
- Other notes:
    - <<Insert notes>>

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.>>

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.
    - <<Insert class>>
        - <<Insert needed tests>>
    - <<Insert class and tests for each class>>
- Other notes:
    - <<Insert notes>>

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.>>

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.>>