Table of Contents:

Student and Computing Information

- Nicholas Phillips and Samantha Duffy Student ID: A031344011 and A01345195

- Advance Programming Principles and Assignment 5.

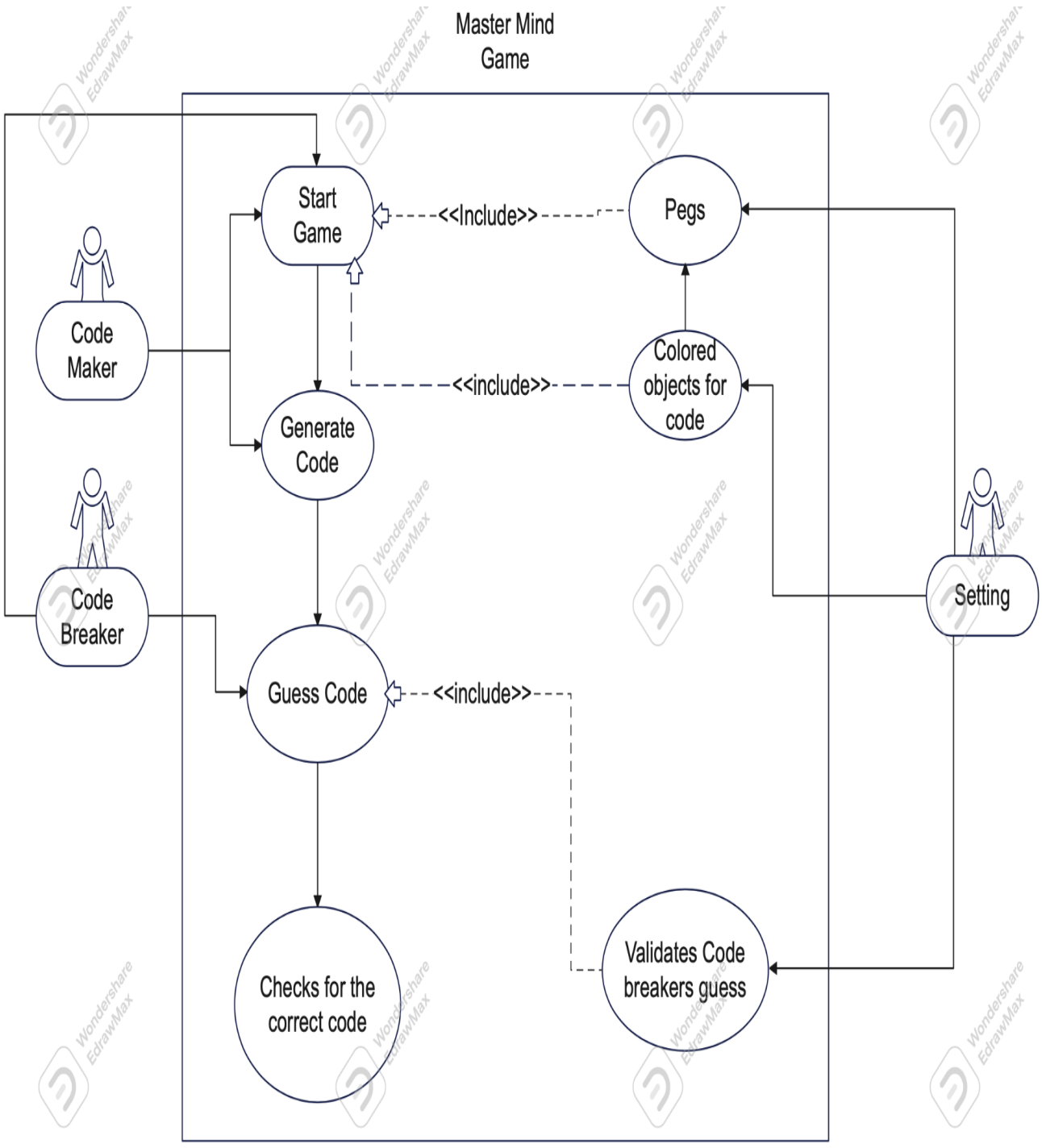
- Completed on a MacBook Pro running macOS Monterey version 12.1. Completed the use case diagram, class diagram, and sequence diagram using EdrawMax.

- Purpose: The purpose of this assignment is for us to design and develop OOP for the game called “Mastermind.” This will allow us to help develop and understand the progress behind building larger-scale applications. In this assignment, we will be creating a scenario, class diagram, sequence diagram, and pseudo code. Each of these processes will help us design and implement any future code that is needed to build the game so-called “Mastermind.”

(2) Scenario:

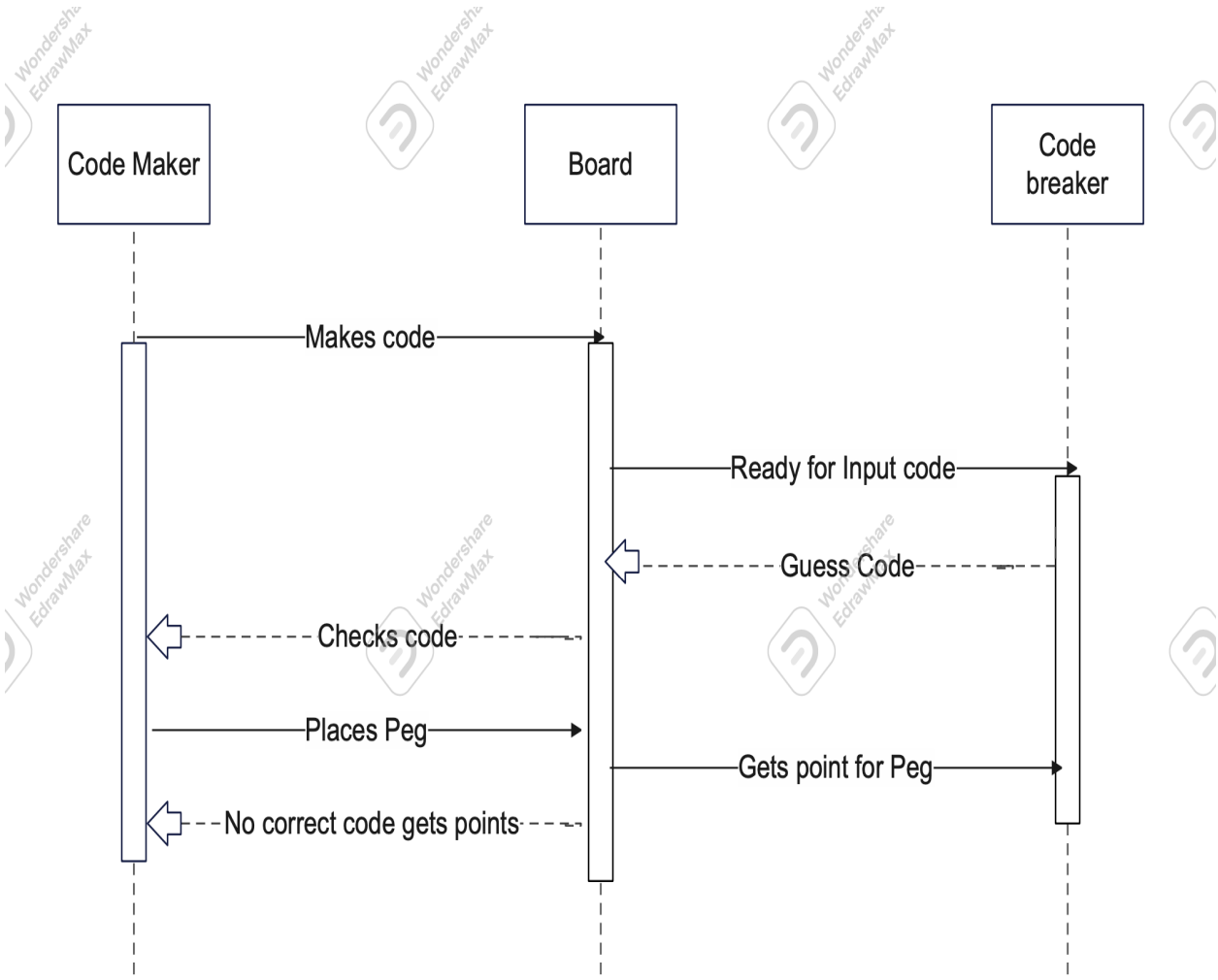
1. Decide who is the code breaker and who is the code maker.
2. Both players should position themselves on the correct side of the board; code makers should have the “secret code area” on their side of the board and should use the plastic cover to hide their secret code using the colored pegs.
3. Then, the code maker will set the code using any of the four colors or duplicates of the same color.
4. The code breaker can now place any combinations of colored pegs into 4 or more peg holes.
5. The code maker will now indicate if the code breaker’s combinations are correct using a white and red peg, white means correct color but wrong position, and red means correct position and color.
6. If neither is correct, no peg shall be placed.
7. Code maker will be granted one point for each row of peg played by the code breaker and 11 points if all 10 slots are used and the code breaker doesn't get it.
8. The code maker should place the pegs in any order arranged in the square pattern next to the line of peg holes holding the guess pegs being answered.
9. Repeat steps 4 and 5 until the game reaches an end or the code breaker gets the correct code.
10. Once the game has come to an end after the ten moves or the correct code is found, switch sides.
11. The player who discovers the code in the least amount of tries wins the round, if no one discovers the code the game ends in a tie.
12. Repeat step 10 until you finish playing.

(3) Use Case Diagram:



(4) Class Diagram:

(5) Sequence Diagram:



(6) Pseudo Code:

1. Welcome the user to the program and read off instructions
    cout<<"Welcome to mastermind game, this is how you play…"
2. Generate a random code sequence using library
    srand(time(0));
    Variable = (rand () % some number) + 1;
3. Set up an array of numbers [0-3], this will allow and check if anything is correct or wrong.
    Array = [0,1,2,3] //the numbers are the options for the computer to use
    0 = red 1 = green 2 = yellow 3 = blue //Logic
4. Have a program to generate code ranging from 0-4 using srand and rand.
    srand(time(0));
    Variable = (rand () % some number) + 1;
    For (variable is = to 0 ; Make variable less than array ; variable increase by 1)
        Loop 4 times though
    End Loop
5. Ask for user input for input 1, input 2, input 3, and input 4. Input can only be numbers 0-3.
    Cin function to ask and store the user input
    Cin >>input 1 … cin>>input 4
6. If statement to check if input # is equal to computer generated input
    For (conditions)
    If input 1 == computer generated input 1
        Set up red peg (make variable)
    End IF
    End for loop
7. If the user guess is equal to the computer number use another if statement to check for the next input.
    For (variable is = to 0 ; Make variable less than array ; variable increase by 1)
    If input 1 == computer generated input 1
        Set up red peg (make variable)
    If Else input 2 == computer generated input 2
        Set up red peg (make variable)
    End IF
    End for loop
8. If the user guess is not equal, the program should check if the input is equal to another number of the computer code.
    For (variable is = to 0 ; Make variable less than array ; variable increase by 1)
    If input 1 != computer generated input 1
        Check if input 1 is equal to computer generated input 2-3
            If it is equal to computer generated input 2-3
                Set up white peg
        End If
    End for loop
9. If no input is equal to the computer's number, ask for user for another input
10. Exit loop once all user inputs are equal to the computer's code or number of tries ends
11. Congrats the user if he/she got the code or sorry if they didn't get it
    If user guess == computer generated input
    You win
    If else user guess != computer generated input
        You lost
    End If
12. Option if the user wants to play again
13. If they don't want to play end loop