Project Planning Worksheet - Unlocked Work

Below is how GWC Alum Robin Acosta thought through the case study.

Instructions: Design a plan for how you would approach this take home project.

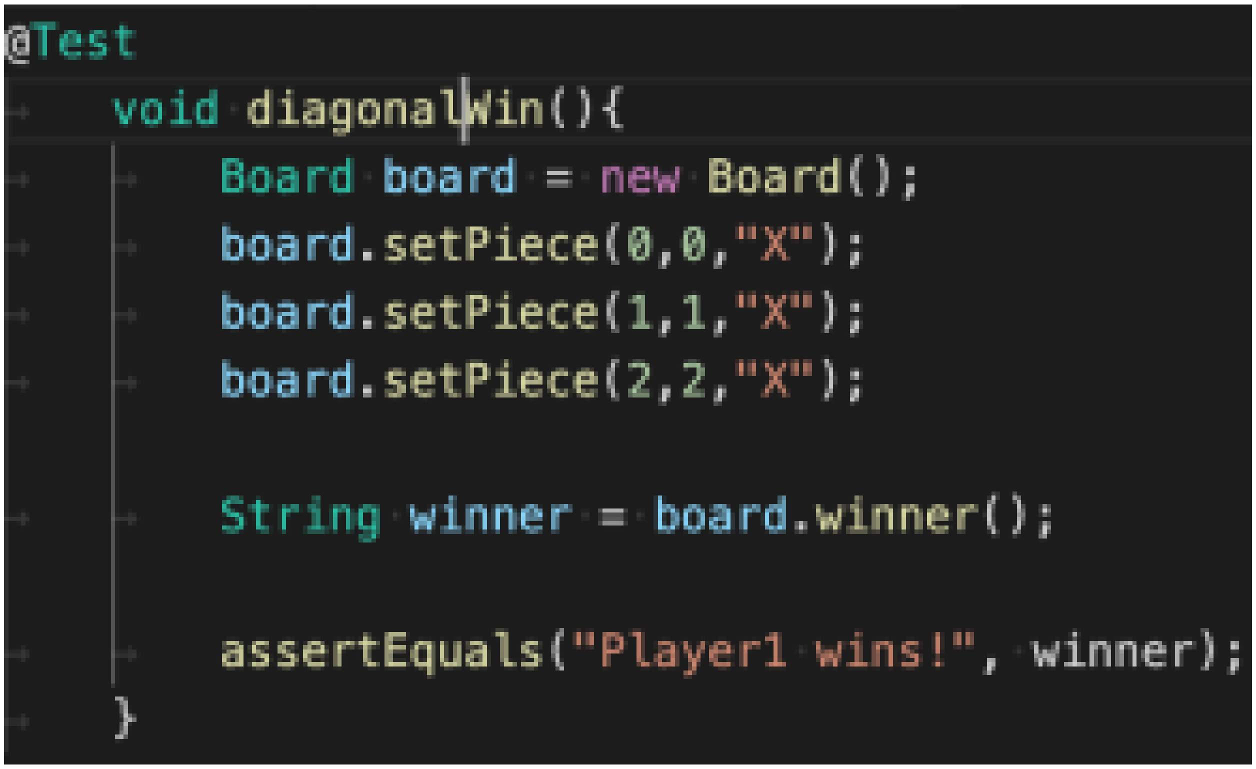
Start by considering the fundamentals of Tic-Tac-Toe:

- ➔ Goal: Be the first to get three in a row
- ➔ Challenge: You don't know where your opponent will place their symbol
- ➔ Core Mechanics: Blocking and writing
- ➔ Components: 9 square grid, players, Xs and Os
- ➔ Rules: There are 2 players. Each player alternates turns placing their symbols until the grid is full or one player gets three in a row, vertically, horizontally, or diagonally.
- ➔ Space: 3x3 grid

Requirements *(What requirements do you have for the game?)*	● 3x3 grid ● 2 players: one computer, one human ● Alternating turns: Xs and Os ● Game ends when grid is full or one player gets three in a row, vertically, horizontally, or diagonally
Clarifying Questions *(What clarifying questions would you ask the hiring manager before you begin?)*	● What happens when the game is over? Should there be the option to play again? ● When there is no possible way for a winner (the game is going to end in a tie) should the game end? Or wait until the game board is full? ● Who goes first? Is it always the human player?
Technical Decisions *1. Consider which programming language you want to use and why?* *2. What user interface will you use?* *3. How will the user provide inputs and see the outputs?*	1. Technical Decision #1: Which programming language to use a. I will use Java. b. **Rationale**: I am familiar with it and I feel confident using this language to make my game. 2. Technical Decision #2: What user interface will I use? a. I will print to the console. b. **Rationale:** It is simple and will be faster than making a GUI. I will make it readable, but I want to focus on functionality before I make things pretty. 3. Technical Decision #3: How will the user provide inputs and see the outputs? a. The user will type the row and column number where they would like to place their piece. The updated board will be printed for the player to see b. **Rationale:** The player's input is simple and easy to read in and will work nicely with a 2d array. I can easily create a method that formats and prints the board and I can just call that when it is updated.

Core Technical Components *(Think about how the program will need to work, what are the rules of the game? How will you know if someone won or lost? What edge cases will you consider? What will you ignore?)*	• 3x3 grid (game board) ○ Status of each space: X, O, or empty • X type and O types ○ One will be associated with the human player, the other will be associated with the computer player • Read input from human player ○ Row and column number will be provided and correspond to the grid. ○ Check if legal move ■ Check human input is within bounds ■ Check space does not already have an X or O • Computer player logic ○ Computer player needs to check for legal moves ○ Make the best legal move. • Alternating turns Xs and Os ○ Once human player goes, trigger the computer turn and vice versa. • Check for end of game ○ Check for no empty spaces (tie) ○ Check for three in a row of the same type, vertically, horizontally, or diagonally (winner)
Testing Plan *(Consider the scope of your testing and the types of cases or scenarios you would test?)*	• Test user input • Test for legal moves: ○ In bounds ○ Empty space • Test all the winning cases: ○ vertically, horizontally, or diagonally for human player and computer player ○ Test what happens when there is a tie.
Task Breakdown *(What steps will you need to take to complete this take home project? Be specific and list in order of priority.)*	1. Create 3x3 grid (game board) with status of each space: X, O, or empty (can't do anything without the board) 2. Be able to print out a representation of the board. This will make it easier to use when the user input comes in. 3. Make a move. Change status of space on board from empty to X or O. 4. Check for three in a row in any direction (winner) 5. Check if there are any empty spaces left on the board 6. Have the game loop so that multiple moves can be made until there is an end game condition 7. Read in user input and make sure user input is within the bounds of the board

	8. Check if space is empty (legal move) 9. Alternate player turn after move is made. 10. Find a legal move on board 11. Have computer payer make move
Write a Unit Test *(Write the code for a simple unit test of one of your game components in the your language of choice.)*	**What will your unit test check for? Why did you decide to write this unit test?** My unit test will check if a diagonal win is correctly found for player1.
	Paste a link to your unit test code or paste a screenshot of your unit test code below.



```
@Test
    void diagonalWin(){
        Board board = new Board();
        board.setPiece(0,0,"X");
        board.setPiece(1,1,"X");
        board.setPiece(2,2,"X");

        String winner = board.winner();

        assertEquals("Player1 wins!", winner);
    }
```