

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green. They are positioned diagonally, with the blue one partially covering the green one.

# Scrabble Project Presentation & Demo

CA314 - OO Analysis & Design - Group 15



## Part One : Analysis

In Part One we focused on the initial design of our scrabble game and began the brainstorming process as a group.

We narrowed down the framework of the game to give a clear and concise path to the next stage of our agile development process.

We decided we would have to evenly distribute work to be done, this included defining requirements, creating the classes and developing CRC cards.

We created the following CRC card based on the player classification.

## Example: CRC Card of a Player

Class Name: Player	ID: 1	Type: Domain		
Description: A person who is using the game		Associated Use Cases: 4		
<table border="1"><tr><td><p>Responsibilities:</p><ul style="list-style-type: none"><li>• Place Tile</li><li>• Refill Tiles</li></ul></td><td><p>Collaborators:</p><ul style="list-style-type: none"><li>• Board</li><li>• Tile Rack</li><li>• Tile Bag</li></ul></td></tr></table>			<p>Responsibilities:</p> <ul style="list-style-type: none"><li>• Place Tile</li><li>• Refill Tiles</li></ul>	<p>Collaborators:</p> <ul style="list-style-type: none"><li>• Board</li><li>• Tile Rack</li><li>• Tile Bag</li></ul>
<p>Responsibilities:</p> <ul style="list-style-type: none"><li>• Place Tile</li><li>• Refill Tiles</li></ul>	<p>Collaborators:</p> <ul style="list-style-type: none"><li>• Board</li><li>• Tile Rack</li><li>• Tile Bag</li></ul>			



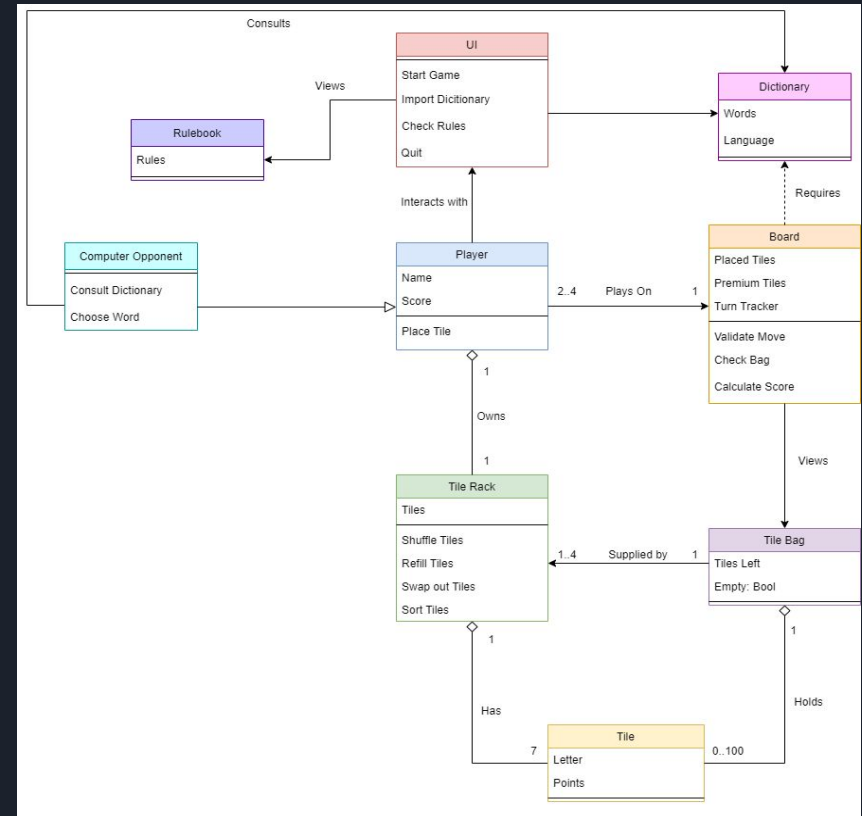
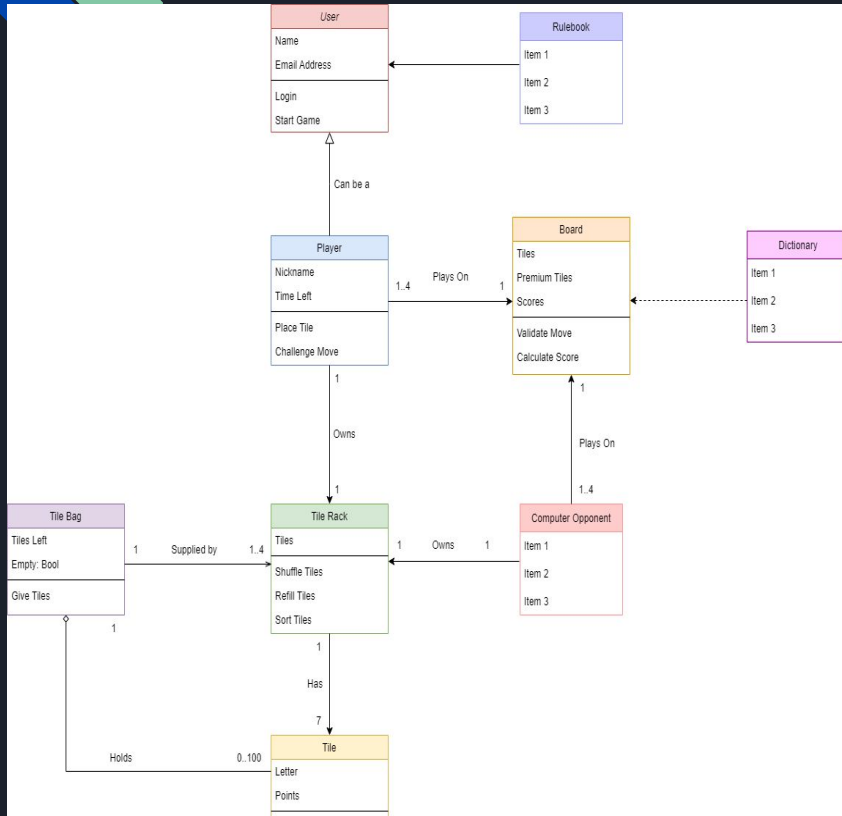
# Lessons Learned in Part 1

While working on Part 1 of this assignment, we learned how to work using an Agile Development Style. This was emphasised in our weekly meetings.

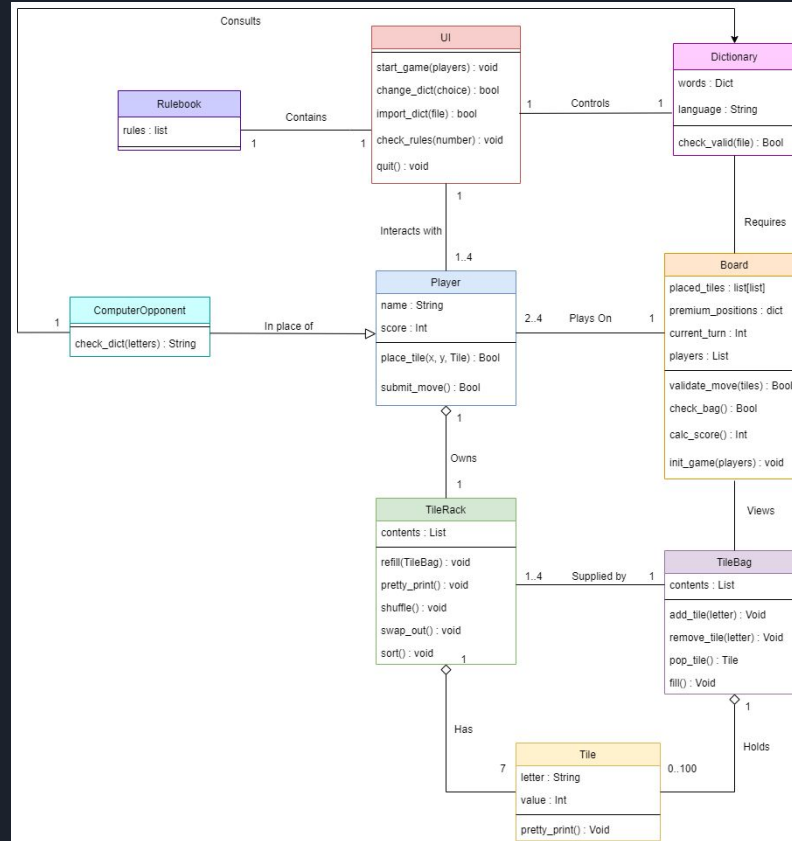
Further understood the importance of iterative development as when we refined one model, we'd have to review others too. (refined requirements - scenarios - diagrams - code)

Abstract skeleton (class diagram) Each step we've taken so far was guiding how our code would be written.

# Our First Class Diagrams



# Our Iterated Class Diagram





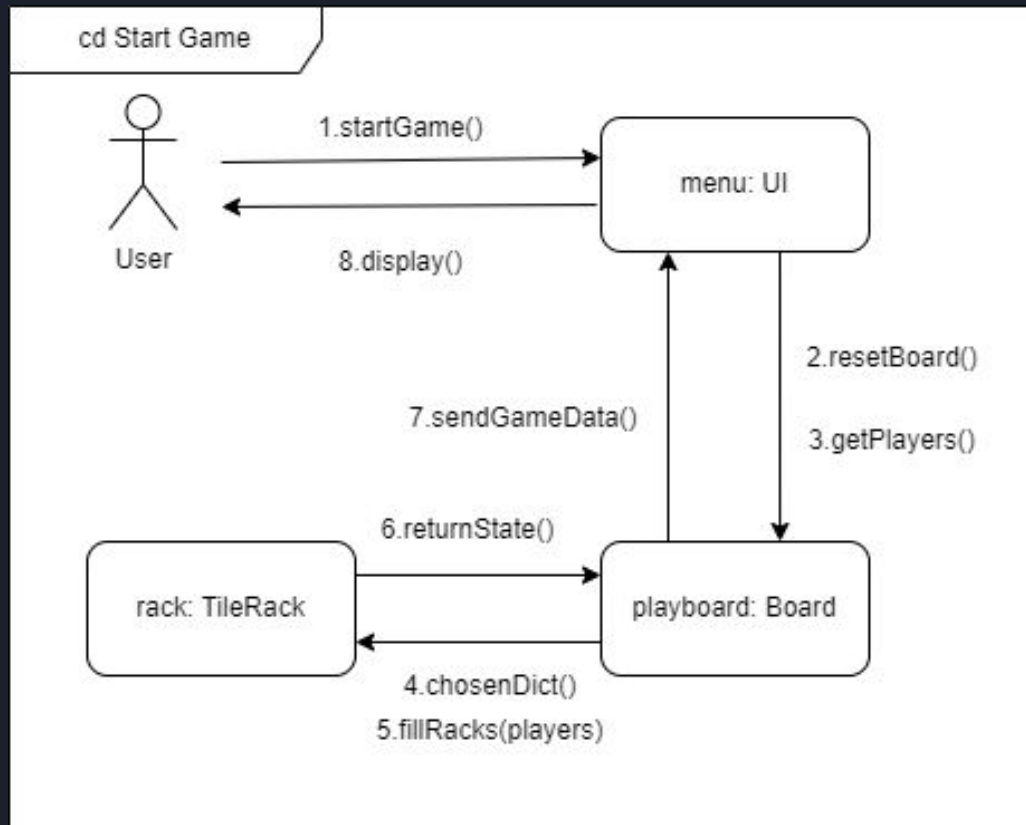
## SECOND SUBMISSION INTRO

For the second submission we focused on the implementation of the game itself, product and class design.

We used diagrams to show how the program would interact with the users and also how the functions within the game would interact with each other.

Diagrams such as communication, sequence and object diagrams were created to show how this would work, We created this communication diagram to show how the game would start:

# EXAMPLE OF A COMMUNICATION DIAGRAM







# Lessons Learned in Part 2

Goal was to refine the work we did in part 1. We also focused in on the UI mockup and Networking

Refinements - Altered the methods in the Class Diagram as they were redundant. Makes the classes more straightforward.

UI Mockup - designed using principles learned in the UI module, simple layout. We ultimately chose to go with a command line version due to time constraints.

Networking - allow for online gaming using a matchmaker as well as a LAN version to play against nearby friends. Complexity prevented implementation of matchmaker but LAN version is achievable had we more time.



# Showcase: Scrabble

- We got our backend and basic classes working in a terminal version of Scrabble
- Our design shifted slightly from class diagram - UI was more prominent
- Short main.py file to power the game - uses functions of all the classes



# Starting the game

```
.-----.  
| Welcome to Scrabble! |  
'-----'  
Please enter nickname for Player 1: Gareth  
Please enter nickname for Player 2: Eimear  
Want to add another player? (Y/N): Y  
Please enter nickname for Player 3: Detutu  
Want to add another player? (Y/N):
```

The welcome screen takes in players names. When the game starts it will set up a player object and tile rack for each name.



# Starting the game

## Players get option to see rules before they start

Alright everything is set up, would you like to know the rules first? (Y/N): Y

Okay, here they are!

- 
1. The game can be played by 2-4 players
  2. The game starts with each player filling their rack with 7 letter tiles
  3. The first player combines two or more of his or her letters to form a word and places it on the board to read either across or down with one letter on the center square. Diagonal words are not allowed.
  4. Complete your turn by submitting, the word score for that turn will be added to your score.  
Then refill your rack will be refilled to 7 unless there are not enough letters in the tile bag.
  5. Play passes to the next player. The next player, and then each in turn, adds one or more letters to those already played to form new words. All letters played on a turn must be placed in one row across or down the board, to form at least one complete word. If, at the same time, they touch others letters in adjacent rows, those must also form complete words, crossword fashion, with all such letters.  
The player gets full credit for all words formed or modified on their turn.
  6. New words may be formed by:
    - Adding one or more letters to a word or letters already on the board.
    - Placing a word at right angles to a word already on the board. The new word must use one of the letters already on the board or must add a letter to it.
    - Placing a complete word parallel to a word already played so that adjacent letters also form complete words.
  7. No tile may be shifted or replaced after it has been played and scored.
  8. Blanks: The two blank tiles may be used as any letters. When playing a blank, you must state which letter it represents. It remains that letter for the rest of the game.
  9. You may use a turn to exchange all of your letters, your letters will be added back to the bag, then you draw 7 letters. This ends your turn.
  10. The game ends when all letters have been drawn and one player uses his or her last letter; or when all possible plays have been made.

-----  
Press Enter to continue...\_



```
for player in game.players:
    game.take_turn(player)
```

```
def sort(self):
def shuffle(self):
def swap_out(self, bag):
```

```

+--3w +--+ +--+ 21--+ +--+ +--+ +--+ +--+ 21--+ +--+ +--+ 3w
| | | | | | | | | | | | | | | | | | | | | |
+--+ +--+ +--+ +--+ +--+ +--+ +--+ +--+ +--+ +--+ +--+
-----
Gareth it is your turn! Your Score is: 0
-----
Your Rack:
++ + + + + + +
|I| |L| |Q| |H| |M| |E| |Z|
+-1 +-1 +-10 +-4 +-3 +-1 +-10
-----
What would you like to do Gareth ?
-----
1. Sort Tiles
2. Shuffle Tiles
3. Swap-Out Tiles (Uses Turn)
4. Place Word
-----
Choice (1-4):
-----
+--+ +--+ 3w
| | | | | | | | | | | | | | | | | | | | | |
+--+ +--+ +--+
-----
Eimear it is your turn! Your Score is: 0
-----
Your Rack:
++ + + + + + +
|E| |O| |R| |E| |T| |G| |H|
+-1 +-1 +-1 +-1 +-1 +-2 +-4
-----
What would you like to do Eimear ?
-----
1. Sort Tiles
2. Shuffle Tiles
3. Swap-Out Tiles (Uses Turn)
4. Place Word
-----
Choice (1-4):
-----
+--+ +--+ 3w
| | | | | | | | | | | | | | | | | | | | | |
+--+ +--+ +--+ +--+ +--+ +--+ +--+ +--+ +--+ +--+ +--+

```



```

+-----+ +-----+ 21--+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ 21
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
Eimear it is your turn! Your Score is: 12
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
w +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
31--+ +-----+ +-----+ +-----+ 31--+ +-----+ +-----+ +-----+ 31--+ +-
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
+-----+ 21--+ +-----+ 21--+ +-----+ +-----+ +-----+ 21--+ +-----+ +-
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
21--+ +-----+ +-----+ +-----+ 21--+ +-----+ +-----+ +-----+ 21--+ +-----+ +-
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
+-----+ 21--+ +-----+ 21--+ +-----+ +-----+ +-----+ 21--+ +-----+ +-
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-
Choice (1-4): 4
What word do you want to place: Deal

```


Play continues until bag is empty, the board filling up, each players score increasing

We used an API to check if the words were from the dictionary

```

def check_word(self, word) -> bool:
    URL = "https://www.dictionaryapi.com/api/v3/references/collegiate/json/" + word + "?key=" + self.API_KEY
    response = requests.get(URL)

```



```
-----  
GAME OVER, BAG WAS EMPTY  
-----
```

```
Gareth your score was: 0  
-----
```

```
Eimear your score was: 4  
-----
```

```
Do you want to play again? (Y/N): n
```

When the game is over, an option to replay is presented,  
this is easily achieved using our OO model.

There are still some unfinished parts  
and features, but we are very happy  
to be able to see how our classes and  
methods interact and work together  
through this simple terminal game

```
choice = input("Do you want to play again? (Y/N): ")  
if choice.lower() == "y":  
    os.system("clear")  
    game.reset()  
    setup()  
    main()  
    replay()  
else:  
    exit
```



Thanks for Listening  
Any Questions?