The goal of this project is to show how a video game can be created using introductory Python concepts. The program fulfilled this by creating a fully-functioning text-based version of the classic game, Battleship, using basic concepts of the Python programming language. The program needs to process user input, evaluate logic, and output accurately while functioning smoothly and handling errors.

The program fulfills the project requirements by featuring several classes, methods, dictionaries, arrays, conditionals and loops. The program has Game, Ship, and Board classes along with a User and a Computer Opponent class that inherit from the parent Player class. The Ship class takes in name and size attributes, while each ship object has a list of coordinates to keep track of its location on the board and a variable that updates the number of hits sustained. Ships are arranged randomly on the board while avoiding overlap by calling on a function to check each cell before inserting the ships. If the ships end up overlapping, then the board is erased and the ships are rearranged, and the process repeats until there is no overlapping. The Board class manages each board instance for each player and has methods that display the board, place ships, and update squares to indicate a hit or a miss. The player class manages each player's ships and boards while also having methods to allow the player to hit the opponent by taking in input as a coordinate point for the user and randomly generating a coordinate point for the computer opponent. A dictionary is used to translate the letter representing the vertical coordinate to an integer that can be used as an array index. Arrays are used to represent the board and what occupies each index. A while loop is used to run the main body until the player loses and the loop breaks. Conditionals are used for checking ship overlap, whether a shot hit or miss, check ship, evaluate logic from input, and check endgame.

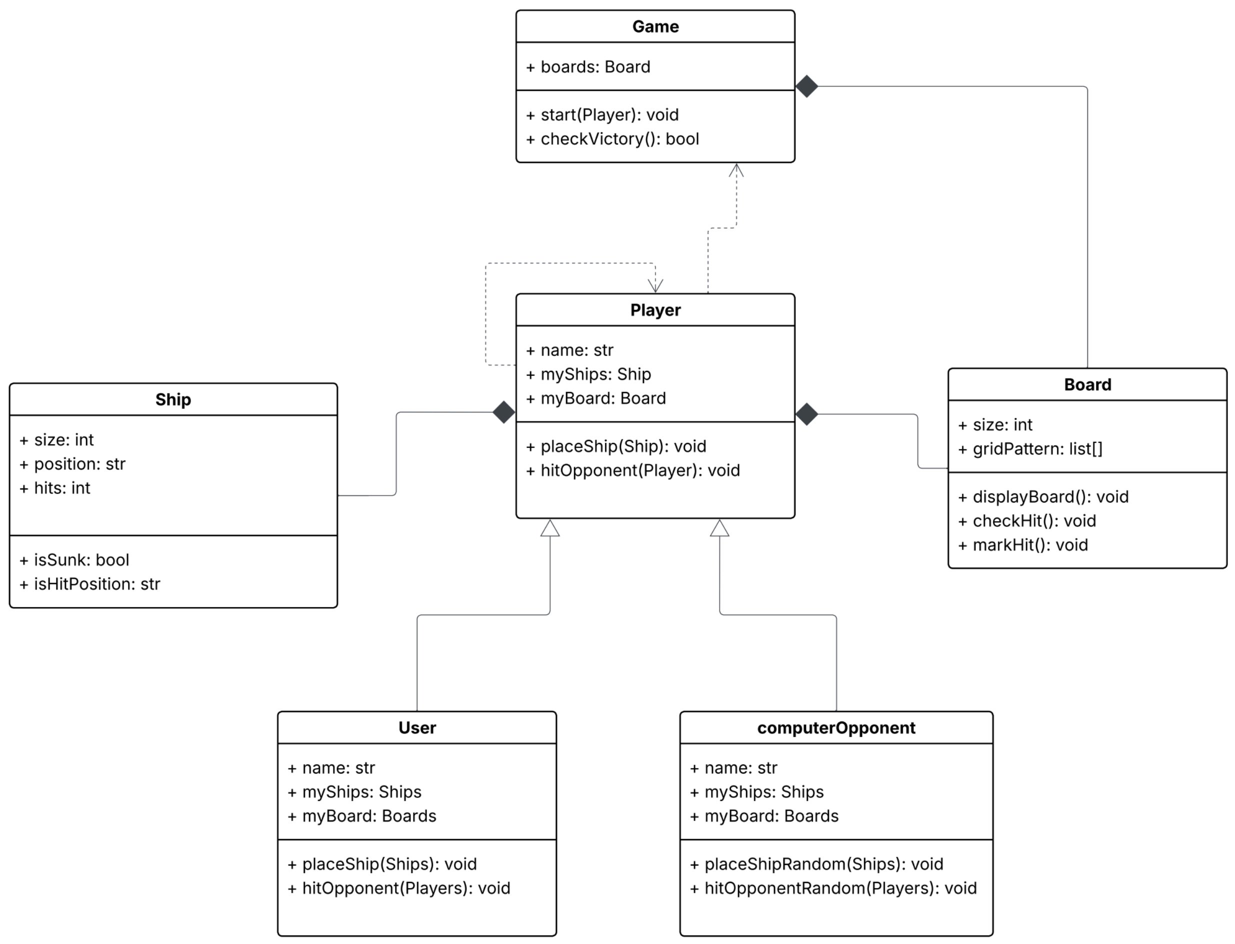
The project is object-oriented by using several classes, instances of these classes with appropriate attributes, and methods inside of classes. Classes in general make the program much more organized, clean, and efficient. Objects act as instances of classes with different attributes to define itself differently from other objects. Class methods defined inside the class make the overall code more organized. These methods are also easier to call through an object and easily interact with the class's attributes and functionalities.

Several difficulties came up while developing this project. One of them was preventing overlap between ships during placement. This required a separate function to check each square that would be filled by a ship and return true if all squares were empty, and false if any one of them were filled, which would cause the ship arrangement function to call on itself to rearrange the ships until there is no overlap. Tracking the game to check if a player lost was surprisingly difficult as it required the program to first check and record if a ship was destroyed, then count for every ship destroyed until that number equals the number of ships. The ships were recorded as destroyed when they sustained the same number of hits as the number of spaces it occupied. A variable in the Player class starts at five and decreases by one every time a ship is recorded as destroyed. When that value becomes zero, the player loses.

If fully released, this project would demonstrate a fully-working video game programmed using the Python language. It would show how concepts such as variables, loops, methods, lists, dictionaries, classes, error handling, and recursion can all be used together for one application. This project could inspire others to see how just learning the basics of Python can be enough for a beginner to create a fully functioning program with real world applications. The game could be improved by using graphics to illustrate the game rather than being text-based. The game could also have a smarter computer opponent that would make more accurate guesses after landing a hit.

This project successfully demonstrates a simple version of the timeless Battleship video game programmed in the Python language and involving several key coding concepts. The game successfully implemented ship placement, hit detection, turn management, and win conditions. It fulfilled all the project’s required concepts to be implemented: classes, methods, conditionals, loops, arrays, dictionaries, variables, operators, and error handling. This project also highlights the importance of approaching software creation with awareness of how design choices affect different users. Working with a game that processes input, evaluates logic, and responds to user actions reinforces how essential it is for programs to behave predictably, handle errors responsibly, and avoid assumptions that exclude or disadvantage certain groups. Considering how different players might interact with the program helps expose potential blind spots, such as barriers for people who rely on screen readers or those who benefit from clearer feedback cues. Reflecting on these issues supports the practice of writing code that is more accessible, respectful, and considerate of communities that have historically faced obstacles in digital spaces.

UML class diagram used to outline the program's classes:



- AI was used to assist in the writing of this report