# EECS 581: SOFTWARE ENGINEERING II

*PROJECT 1*

## CODE DOCUMENTATION

## I. Project Introduction

This project simulated a Battleship game for two players where each player can place their ships on a 10x10 grid and then guess where their opponent's ships are located by taking turns. Ships of different sizes (from 1x1 to 1x5) are placed by players either vertically or horizontally, and then they start firing each other's grids, and get announcements about hits and misses. The first player to sink all of the ships from the opponent wins. The program is organized into modular functions that handle tasks such as board setup, ships placement , gameplay loop and win condition.

## II. Key Functions

1. **create_board():** Sets up a playing field for ships placement by create a 10x10 grid by using list comprehension
2. **display(board)**: Print out the game board in order to help players visualize the game
3. **get_coordinates(pos):** Converts readable input into the internal coordinate system if the game so that moves and ships placement can be processed
4. **place_ship(board, size, orientation, direction, start)**: Places a ship of a specified size on the board based on its orientation and direction from a starting point, and updates the board. Provides a list of the ship's coordinates for tracking or verification.
5. **is_valid _position(pos)**: Verify that the given position string is formatted correctly in order to confirms if the given position (e.g. "A1" or "B10") is inside the board's acceptable game
6. **valid_ship_placement(board, size, orientation, direction, start):** Determine whether a ship can be positioned on the board without overlapping other ships or beyond its bounds.
7. **fire(board, pos, ships):** Convert the shot location into coordinates, and analyze a shot fired at a certain spot on the board to check whether the shot is a hit, miss or a ship was sunk
8. **place_ships(board, ship_list):** Assist users in inputting ship dimensions, positions, orientations and directions so that a list of ships can be placed onto the board. The ship is registered and its details are documented once it has been verified. The process is repeated until every ship is placed.

9. **get_num_ships():** Prompts the user to enter the amount of ships they wish to ass to the board, ensuring the input is in the allowed range
10. **battleship_game():** Manages the whole game. Every turn, the vision is updated based on if the shot's result was successful or not, determines if a win condition has been met then switches turns between two players. The game goes on until one player sinks every opponent's ships

# III. Instruction

1. Preparation
    - Have Python 3.x installed.
    - Save the `battleship.py` file to your working directory
2. Running program
    - Open a terminal and navigate to the directory containing `battleship.py` using the command `cd`.
    - Run the program by typing `python battleship.py`.
3. Setup game
    - Enter the number of ships (from 1 to 5) you wish to have in this game
    - Place ships by specifying a starting location and orientation. The input can be in lowercase or uppercase (e.g., a1 or A1).
4. Playing the game
    - Start firing the opponent's ships by taking turns. Enter the target position you want to fire at (e.g. A1)
    - The system will announce and update with "Hit!", "Miss!", "Hit! You sunk a ship of size (`sunk_ship_size`)" or "You're already fired at that position. Choose a different spot."
5. Exiting
    - When one player's ships are all sunk, the game will end. And the system will announce the winner
    - In case user want to exit early, use Ctrl+Q

Make sure to follow the input prompts so that there will be no errors.

# IV. Conclusion

With the help of this turn-based, interactive Battleship game implementation, two players can place ships and alternately fire at one another's boards. Hit/miss monitoring, ship placement confirmation, and unambiguous player feedback are some of the elements of the game. The directions are simple to follow, so players can quickly set up and start playing. Additionally, this project is a great way to get started with basic programming concepts like conditional logic, loops, and list manipulation. All in all, it's an enjoyable and instructive approach to learn problem-solving and coding.