Hacettepe University Department of Computer Engineering

BBM 103 Assignment 4 Report

Ahmet Yavuz Mutlu -2210356014

03.01.2023

Analysis

The problem this code is trying to solve is to create a simulation of the classic board game Battleship. The game consists of two players, each with a board where they place their ships. The players then take turns shooting at the other player's board, trying to sink all of their opponent's ships. The code reads in input files containing the positions of the ships and the shots taken by each player, and then simulates the game, printing out the state of the boards after each round.

Design

- `createPlayerGrid():`

  This function takes in a player's ship positions file and returns a dictionary representing the player's board and a dictionary containing the positions of each of the player's ships. The function contains 2 important function;

  - `gridFunc():` This function first initializes the player's board as a dictionary of empty strings, with the keys being the positions on the board (e.g. "A1", "D4", etc.). It then reads in the ship positions from the input file and stores them in a dictionary where the keys are the ship types (e.g. "C", "B", etc.) and the values are lists of positions.

  - `seperateShipsFunc():` The function iterates through the ship positions, creating a new dictionary for each ship and storing it in the player's ships dictionary. It also updates the player's board dictionary to include the ship at each position.

- `shipSituations():`

  This function will take in a player's ship positions and return a dictionary indicating the sink status of each ship type and a boolean indicating if all of the player's ships have been sunk.

- `winnerFunc():`

  This function will take in the game over status of both players and return a string indicating the winner of the game or None if the game is not yet over.

- `display():`

  The display function has several subfunctions that are used to display different aspects of the game. The displayNormalHeader function displays a header for the current round of the game, including the round number and the grid size. The displayFinalHeader function displays a header for the final round of the game, including the winner of the game. The displayBattleGrid function displays the current state of each player's grid, including which positions have been attacked and whether they contain a ship. The displayShipSituations function displays the sink status of each player's ships. The winnerFunc function determines the winner of the game based on whether one of the players has sunk all of their opponent's ships.

The display function uses these subfunctions to display the current state of the game to the user. If there is no a winner of the game, it will call the displayNormalHeader and displayBattleGrid functions. If it is the final round of the game, it will call the displayFinalHeader, displayBattleGrid, and displayShipSituations functions. In either case, it will also call the winnerFunc function to determine the winner of the game.

- ## attack():

The function "attack" accepts three arguments: playerDatas, position, and errorExist. playerDatas is a dictionary containing the grid and ships of a player. The position argument is a string representing the coordinates of the attack in the format "rowcolumn" (e.g. "5E"). The errorExist argument is a boolean that indicates whether an error occurred during the input of the position.

The function first removes the comma from the position string if it exists. Then, it checks if the errorExist argument is False. If it is, the function proceeds to access the grid and ships information in the playerDatas dictionary. It checks if the position on the grid has already been attacked by checking the "hitStatus" value. If it has not been attacked, the function sets the "hitStatus" value to True and updates the "displaySymbol" value based on whether the position contained a ship or not. If it contained a ship, the function also updates the "hitStatus" value of that ship in the ships information as True . If the position has already been attacked, it raises an error. If the errorExist argument is True, the function sets attackSuccess to None and does nothing else.

Finally, the function returns a tuple containing the updated playerDatas dictionary and the attackSuccess value.

- ## allShotsFunc():

This function accepts as arguments player1 and player2 attacks. This function contains 3 functions:

- playerShotsFunc(): This function reads the player's attack position file and returns a list that contains of player's attack positions.
- positionError(): This function will take in a shot position and check for any formatting errors in the position. If an error is found, it creates an specific error message and then it returns this message an a boolean variable that shows an error exist or not.
- zipFunc(): This function takes as argument player1's and player2's attack positions zipped them as a list round by round.

This function calls this three function and finally returns zipped players attacks and positions that has an error.

- ## battle():

"battle" function just calls the other functions above up in a sequance up for simulate the all game from round 1 until the last round.

Programmer's Catalogue

Analyzing: 2 hours Designing: 4 hours Implementing: 8 hours
Testing: 6 hours Reporting: 1 hour

- outputFunc() :

This function is used to output text to the output file and to the console. It takes a string as an input and does not return anything.

```
outputFunc(text: str) -> None:
    """ Outputs the given text to the output file and to the console.

    Args:
    text (str): The text to be outputted.

    Returns:
    None      """
```

- createPlayerGrid() :

This function takes a file object containing information about a player's ships and returns a player's grid and ship positions as a tuple of dictionaries.

```
createPlayerGrid(playerShipFile: TextIOWrapper) -> Tuple[Dict[str,
Dict[str, Any]], Dict[str, List[str]]]:
    """Creates a player grid and a dictionary of ships for a given
      player.

    Args:
    playerShipFile (TextIOWrapper): The file containing the positions
of a player's ships.

    Returns:
    Tuple[Dict[str, Dict[str, Any]], Dict[str, List[str]]]: A player
grid and a dictionary of ships.
    """
```

- seperateShipsFunc() :

This function takes a dictionary with keys representing ship types and values containing lists of positions and returns a tuple containing a dictionary with keys representing ship types and values containing lists of dictionaries representing each ship, and a dictionary representing the player's grid. The ship dictionaries have keys representing positions on the grid (e.g. "5E") and values containing a dictionary with the following key:

hitStatus: A boolean indicating whether the position has been hit.

```
seperateShipsFunc(playerShipPositions: Dict[str, List[str]]) ->
Tuple[Dict[str, List[Dict[str, Any]]], Dict[str, Dict[str, Any]]]:
    """Separates a player's ships into individual dictionaries.

    Args:
    playerShipPositions (Dict[str, List[str]]): A dictionary with keys
representing ship types and values containing lists of positions.

    Returns:
    Tuple[Dict[str, List[Dict[str, Any]]], Dict[str, Dict[str, Any]]]:
A dictionary with keys representing ship types and values containing
lists of dictionaries representing each ship, and a dictionary
representing the player's grid.
    """
```

- gridFunc() :

This function returns a player's grid with default values as a dictionary. The keys represent positions on the grid (e.g. "5E") and the values contain a dictionary with the following keys:

positionStatus: A string indicating whether the position is a ship ("S") or water ("-"). The default value is "-".

hitStatus: A boolean indicating whether the position has been hit. The default value is False.

displaySymbol: A string representing how the position should be displayed (either "O" for a miss or "X" for a hit). The default value is None.

ship: A string representing the ship at the position (e.g. "C1" for the first cruiser). If the position is water, this value will be None. The default value is None.

```
gridFunc() -> Dict[str, Dict[str, Any]]:
    """Creates a player's grid with default values.

    Returns:
    Dict[str, Dict[str, Any]]: A player's grid with default values.
    """
```

- shipSituationsFunc():

```
shipSituationsFunc(playerShips: Dict[str, List[Dict[str, Any]]]) ->
Dict[str, bool]:
    """Checks the sink status of a player's ships.

    Args:
    playerShips (Dict[str, List[Dict[str, Any]]]): A dictionary with
keys representing ship types and values containing lists of
dictionaries representing each ship.

    Returns:
    Dict[str, bool], Bool: A dictionary with keys representing ship
types  and values indicating whether the ship has been sunk (`X`) or
not (`-`), a boolean that shows player all ships sinked or not.
    """
```

- winnerFunc():

```
winnerFunc(player1Ships: Dict[str, bool], player2Ships: Dict[str,
bool]) -> Union[str, None]:
    """Determines the winner of the game.

    Args:
    player1Ships (Dict[str, bool]): A dictionary with keys
representing ship types and values indicating whether the ship has
been sunk (`True`) or not (`False`) for player 1.
    player2Ships (Dict[str, bool]): A dictionary with keys
representing ship types and values indicating whether the ship has
been sunk (`True`) or not (`False`) for player 2.

    Returns:
    Union[str, None]: The name of the winning player (either "player1"
or "player2") or `None` if the game is not yet over.
    """
```

- display():

The display function is responsible for displaying the current state of the game in both the terminal and *"Battleship.out"* . This includes printing the current round, the grids for each player, the ship situation for each player, and the attack position for the current round. If the winner parameter is different from *"None"*, the function will display the final state of the game including the winner.

```
display(player1Datas: Dict[str, Any], player2Datas: Dict[str, Any],
playerMove: Optional[str] = None, roundNum: Optional[int] = None,
attackPosition: Optional[str] = None, winner: Optional[str] = None)
-> None:
    """Displays the current state of the game on the screen.

    Args:
    player1Datas (Dict[str, Any]): A dictionary containing player
1's grid and ship information.
    player2Datas (Dict[str, Any]): A dictionary containing player
2's grid and ship information.
    playerMove (Optional[str], optional): The current player's name.
Defaults to `None`.
    roundNum (Optional[int], optional): The current round number.
Defaults to `None`.
    attackPosition (Optional[str], optional): The attack position.
Defaults to `None`.
    winner (Optional[str], optional): The name of the winning
player. Defaults to `None`.

    Returns:
    None
    """
```

- allShotsFunc():

  This function contains 3 important functions:

  - playerShotsFunc():

```
playerShotsFunc(playerShotsFile: TextIO) -> List[str]:
    """Gets a list of positions for a player's shots.

    Args:
    playerShotsFile (TextIO): A file object representing a player's
shots.

    Returns:
    List[str]: A list of positions for a player's shots.
    """
```

  - positionErrror():

```
positionError(position: str) -> Optional[str]:
    """Checks for errors in the format of a shot position.

    Args:
    position (str): The shot position to be checked.

    Returns:
    Optional[str]: An error message if an error is found, otherwise None.
    """
```

- zipShots()

```
zipShots(player1Shots: List[str], player2Shots: List[str]) ->
List[Tuple[str, str]]:
    """Combines the shots taken by each player into pairs.

    Args:
    player1Shots (List[str]): A list of positions for player 1's shots.
    player2Shots (List[str]): A list of positions for player 2's shots.

    Returns:
    List[Tuple[str, str]]: A list of tuples containing the shots taken by
each player.
    """
```

- attack():

```
attack(playerDatas: Dict[str, Any], position: str, errorExist:
Optional[bool] = False) -> Tuple[Dict[str, Any], Union[bool, None]]:
    """Simulates an attack on a player's grid.

    Args:
    playerDatas (Dict[str, Any]): A dictionary containing a player's
grid and ship information.
    position (str): The position to be attacked.
    errorExist (Optional[bool], optional): A boolean indicating
whether there is an error in the input position. Defaults to `False`.

    Returns:
    Tuple[Dict[str, Any], Union[bool, None]]: A tuple containing
updated player data and a boolean indicating whether the attack was
successful (`True`) or not (`False`). If there is an error in the
input position, the boolean value will be `None`.
    """
```

- battle():

```
def battle(player1Shots: List[str], player2Shots: List[str],
player1Datas: Dict[str, Any], player2Datas: Dict[str, Any]) ->
Tuple[Dict[str, Any], Dict[str, Any]]:
    """It is the main function of this program. It calls the other
functions and simulates the all battle between two players.

    Args:
    player1Shots (List[str]): A list of strings representing attack
positions for player 1.
    player2Shots (List[str]): A list of strings representing attack
positions for player 2.
    player1Datas (Dict[str, Any]): A dictionary containing player 1's
grid and ship information.
    player2Datas (Dict[str, Any]): A dictionary containing player 2's
grid and ship information.

    Returns:
    None """
```

Grading

Evaluation	Points	Evaluate Yourself / Guess Grading
Readable Codes and Meaningful Naming	5	5
Using Explanatory Comments	5	5
Efficiency (avoiding unnecessary actions)	5	5
Function Usage	15	15
Correctness, File I/O	30	30
Exceptions	20	20
Report	20	15
There are several negative evaluations	…	… Total : 95