Battleship

Description

- The program shall randomize the positions of the battleships or if the player chooses to randomize a file from which the program reads the positions from.
- The user shall be able to fire at a coordinate by pressing a button in a grid and the program shall print out if the shot hit the ship or not and store the information.
- The user shall also be able to choose between actions in a menu, where these are to shoot at a coordinate, to cheat or to quit the program. If the user chooses to cheat all information is shown to player (positions of battleships and markings of shots). The user shall also be able to use a main menu where the user has the options of playing the game, looking at the leaderboard or quitting.
- The program shall print out the menu options and positions of the hit ships.
- The user shall be able to shoot at coordinates by typing in the coordinates in the program.

Algorithm

1. The program randomizes the positions of the battleships.
2. The following shall be repeated until the user wants to exit the program:
    a. Show the UI with menu options and grid
    b. Execute the choice the player makes
    c. Update the dictionary and UI
3. When the player has finished the game, the player will be brought out to the main menu. If the player makes it in to the top ten high scores the player will be asked for a first name and last name. For a high score to be added to the list the player must hit all the battleships and have a better high score than the 10th player on the leaderboard.

Data structures

Every layout of the game is represented by a Game-object. All the layouts are stored in a dictionary where every key is a coordinate and the value being the status of the coordinate (if there is a ship on the coordinate or not and if the player already shot at it). While the program is running the information of where the ships are, and which have been hit is stored in a dictionary. The high scores are stored in a dictionary.

Classes and methods

```
class Game:
    """
    Attributes:
    layout_dict: A dictionary where the key is the coordinate and the value
is the string that says if the
   coordinate has been shot at before and if it was a hit or not.
   shot: A string containing information of which coordinate to fire at.
   hits: The amount of shots the player has hit on a ship.
   shots: The amount of shots the player has taken.
    """
```

```
    def __init__(self, layout_dict):
        """Creates variables for the object

        :param layout_dict: A dictionary where the key is the coordinate
and the value is the string that says if the
        coordinate has been shot at before and if it was a hit or not.
        """


    def __str__(self):
        """Creates a grid of type string by reading information from a
dictionary

        :return: returns a string that shows the grid
        """


    def shoot(self, shot):
        """Fires at a coordinate and updates the dictonary containing the
information of the grid accordingly

        :param shot: The coordinate the player wants to fire at
        :return: returns a string confirming the outcome
        """

    def cheat(self):
        """Shows all the battleships and marks of the shots

        :return: A string declairing that you are cheating
        """


    def high_score(self):
        """Takes out the hit percentage the player has by dividing hits
with shots and multiplying with 100.

        :return: returns the percentage score the player achieved
        """


def read_layout():
    """Reads the grid layout from a randomly choosen json file and stores
it in a python dictionary.

    :return: returns a dictionary with information about the positions of
the battleships
    """




def empty_grid():
    """Creates an dictionary with coordinates as keys and empty strings as
```

```
values.

    return: returns a dictionary with coordinates as keys that are assigned
to empty values
    """


def grid_check(grid_dict, origin, lenght, height):
    """Checks if the ship can be placed on the origin by checking if the
coordinates it's going to be occupying are
    empty and inside the grid. Checks in the direction given by the length
and height variables from the origin.

    :param grid_dict: A dictionary containing information about the
battleship positions
    :param origin: The start point of the ship, where it starts to build
out from
    :param lenght: A list with 2 values that describes how many coordinates
to check along the x-axis
    :param height: A list with 2 values that describes how many coordinates
to check along the y-axis
    :return: returns a boolean
    """



def create_ship(origin, direction, ship_size, grid_dict):
    """Creates a ship starting from the origin coordinate and expanding in
the given direction until the
    ship size is reached.

    :param origin: The first coordinate of the ship
    :param direction: The direction in which the ship shall expand from the
origin
    :param ship_size: The size of the ship
    :param grid_dict: A dictionary containing information about the grid
and the battle ships on it
    :return: returns an updated dictionary
    """


def randomize_grid(grid_dict):
    """Randomizes the positions of the battleships by changing the values
in the dictionary
    which are connected to keys that represent the coordinates in the grid.
The while loop
    randomizes new positions for the battleships until it finds a position
that is empty.

    :param grid_dict: An dictionary that displays an empty grid
    :return: returns a dictionary containing information of the positions
of the battle ships
    """
```

```
def start_game(grid):
    """Creates a new object in the class Game and prints out the grid

    :param grid: A dictionary containing information about the battleships
positions
    :return:
    """


def read_file(file_name):
    """
    Reads the information of the file and stores it in a dictionary
    :param file_name: The name of the text file
    :return: returns a sorted dictionary where the key is the name and the
value is the score
    """


def update_file(file_name, high_score, score_dict, name):
    """Updates the high score dictionary that is stored in a json file.
Only 10 items are allowed in the dictionary.

    :param file_name: The name of the file to update
    :param high_score: The score the player achieved
    :param score_dict: The dictionary with all the high scores
    :param name: The name the player has choosen
    :return:
    """


def get_int_input(prompt_string):
    """Asks the user the prompt and expects an string. The program tries if
the it is a string and if it
    is able to convert into an int. If it doesn't work it will rais an
exception and print out the problem and let the
    user try again.

    :param prompt_string: A string that is printed to ask the user what
integer should be
    :return: returns an int
    """


def menu():
    """Prints the options you can choose from in the program

    :return: Null
    """


def menu_choice():
```

```
    """Asks which option the user wants to run with get_int_input and
stores it as a variable.

    :return: returns the int of the choice made
    """


def execute(choice):
    """The program checks what int the choice is and executes the action
connected to it. If choice == 3 the program
    will shut down. If choice isn't in the range of options the program
will print an error message and ask the user
    to enter the choice again.

    :param choice: An integer
    :return: Null
    """



def game_menu():
    """A while loop that loops the program until you choose the exit to
menu option or manage to hit all ships.

    :return: Null
    """


def menu_options():
    """Prints the options you can choose from in the program

    :return: Null
    """


def execute_main(choice):
    """The program checks what int the choice is and executes the action
connected to it. If choice == 3 the program
    will shut down. If choice isn't in the range of options the program
will print an error message and ask the user
    to enter the choice again.

    :param choice: An integer that matches 1 of the choices
    :return: Null
    """


def start_menu():
    """A while loop that loops the program until you choose the exit
option.

    :return: Null
    "
```