Mini Project 1 - Battle Ships

This project will help you get more familiar with arrays. You will be recreating the game of battleships. A player will place 5 of their ships on a 10 by 10 grid. The computer and the player will deploy five ships on the same grid. Once the game starts the player and computer take turns, trying to sink each other's ships by guessing the coordinates to "attack". The game ends when either the player or computer has no ships left.

Step 1 – Create the ocean map

The ocean map is represented by a 10 by 10 grid of different characters. The grid is managed by a two-dimensional array. You will use this 2D array to save where the user and computer decide to place their ships, as well as when someone tries to attack a location and misses. At the start of the game the array will be empty and as the game is played you will change what is stored at each index of the array accordingly.

Once you create your 2D array you need a way to display it to the user so they can choose coordinates. You should display your array surrounded by indexes like below:

```
**** Welcome to Battle Ships game ****

Right now, the sea is empty.

   0123456789
0 |          | 0
1 |          | 1
2 |          | 2
3 |          | 3
4 |          | 4
5 |          | 5
6 |          | 6
7 |          | 7
8 |          | 8
9 |          | 9
   0123456789
```

To do this you should create a method that prints out the above by looping over the array and adding the indexes and pipe characters before and after each row. You will want to use a nested for loop to print this two-dimensional image.

Step 2 – Deploy player's ships

Once you have your ocean map, you'll need to ask the user where they would like to place their ships. The player should deploy 5 ships. A ship will be stored in a single index of the array as a special

character. To place the player's ships, they need to tell you the coordinates of where the ship should be placed, and you need to update the ocean map to reflect their choices. Remember you'll need to use a Scanner to allow the user to enter in input.

```
import java.util.Scanner; // you must import Scanner to use it



public class BattleShipsGame {

   public static void main(String[] args) {

      Scanner input = new Scanner(System.in); //This line creates a Scanner for you to use



      // ...



      /*

      Here is some sample code where you ask the user to enter in the coordinates for where to place a ship

      */

      System.out.print("Enter X coordinate for your ship: ");

      int x = input.nextInt();

      System.out.print("Enter Y coordinate for your ship: ");

      int y = input.nextInt();

   }

}
```

As the user is telling you where to place their ships you need to check if that is an appropriate location:

- you can NOT place two or more ships on the same location

- you can’t place ships outside the 10 by 10 grid

If the player is trying to put the ship somewhere it can't be, re-prompt them until they choose legal coordinates for the ship.

You should store the player's ships within the OceanMap as '1'. That way you know within your own code which indexes store the player's ships. However, when you are printing out the map from the method you created in Step 1 you should hide where the ships are by printing an '@' symbol instead. You can accomplish this with an extra if check within the for loop that prints your ocean map.

Once the user is finished placing a ship you should print out the map so they can see the current state of their ships.

```
Deploy your ships:
Enter X coordinate for your 1. ship: 2
Enter Y coordinate for your 1. ship: 2
Enter X coordinate for your 2. ship: 5
Enter Y coordinate for your 2. ship: 5
Enter X coordinate for your 3. ship: 8
Enter Y coordinate for your 3. ship: 3
Enter X coordinate for your 4. ship: 3
Enter Y coordinate for your 4. ship: 7
Enter X coordinate for your 5. ship: 8
Enter Y coordinate for your 5. ship: 8


   0123456789
0 |          | 0
1 |          | 1
2 |  @       | 2
3 |        @ | 3
4 |          | 4
5 |     @    | 5
6 |          | 6
7 |   @      | 7
8 |        @ | 8
9 |          | 9
   0123456789
```

Step 3 – Deploy computer’s ships

The computer will deploy 5 ships by randomly picking X and Y coordinates. Your code is responsible for generating these locations, checking if they are valid, and if so placing the ships accordingly.

Keep in mind:

- you cannot place the ship on a location that is already taken by another ship (player's or computer's)
- you can't place ships outside the 10 by 10 grid

If the computer tries to place the ship somewhere it can't be, regenerate random coordinates until all ships are placed appropriately.

You should store the computer's ships within the OceanMap as '2' and they should be invisible on the ocean map. This is because the game is to have the player guess where the ships are, so you don't want to give the game away by showing the location of the computer's ships. Thus, as the ships are being deployed you want to give the user some feedback about what is happening like so:

```
Computer is deploying ships
1. ship DEPLOYED
2. ship DEPLOYED
3. ship DEPLOYED
4. ship DEPLOYED
5. ship DEPLOYED
----------------------------------------
```

You should generate each line of output each time you successfully place a ship based on random coordinates.

Step 4 – Battle

Player's Turn

Once the player and computer have placed their ships it's time to start the battle! During the battle, the player and computer will take turns guessing X and Y coordinates of the opponent's ships. Every coordinate guessed should be marked so they players know not to guess there again.

When the player enters X and Y coordinates you should check if those coordinates are valid within the Ocean Map and haven't been guessed by the user yet, keep re-prompting until the user enters a valid guess. Once the guess is valid your program needs to evaluate the result of the move.

```
YOUR TURN
Enter X coordinate: 4
Enter Y coordinate: 1
You missed
```

There are three possible results of a valid guess:

- Player correctly guessed coordinates of computer's ship (computer loses ship).

- You should tell the user "Boom! You sunk the ship!"
- You should mark this as a hit when printing the map as a "!". You can choose how to store this result within your own code.
- Player entered coordinates of his/her own ship (player loses ship).
- You should tell the user "Oh no, you sunk your own ship :("
- You should mark this as an "x" when printing the map, replacing the "@"
- Player missed. No ship on the entered coordinates. "Sorry, you missed"
- You should mark this as an "-" when printing the map.

In all of these cases you should mark the coordinates on the ocean map, so the player knows how to guess better next time.

Computer's Turn

After the player guesses a coordinate it's the computer's turn to guess. The computer's attack should be two randomly generated coordinates. You will need to keep generating random numbers until you get a valid guess, meaning a location that is within the bounds of the board and the computer hasn't already guessed. Once the computer makes a valid guess, you want to print a little update to the user:

```
COMPUTER'S TURN
Computer missed
```

When the computer produces a valid guess there are three possible outcomes:

- Computer guessed coordinates of the player's ship (player loses ship).
- You should inform the user "The Computer sunk one of your ships!"
- You should mark this as an "x" when printing the map
- Computer guessed coordinates of its own ship (computer loses ship).
- You should inform the user "The Computer sunk one of its own ships"
- You should mark this as a "!" when printing the map
- Computer missed. No ship on guessed coordinates.
- You should inform the user "Computer missed".

- You do not need to mark the map with the missed computer guesses, however you will want to decide a way to store this information in your map so the computer doesn't duplicate guesses later.

This is how the screen will look after couple turns

```
YOUR TURN
Enter X coordinate: 1
Enter Y coordinate: 6
BOOOM! You sunk the ship!
COMPUTER'S TURN
Computer missed

   0123456789
0 |          | 0
1 |    x     | 1
2 |  @     x | 2
3 |x      x@ | 3
4 |  x       | 4
5 |     @    | 5
6 | x      xx| 6
7 |   @ x    | 7
8 |x       @ | 8
9 |          | 9
   0123456789

Your ships: 5 | Computer ships: 3
----------------------------------------
```

The battle will continue to run until one of the players is out of ships.

Step 5 - Game Over

When the user and computer are done guessing, display the current state of the ocean map and score.

```
   0123456789
0 |          | 0
1 |    x     | 1
2 |  @       | 2
3 |        @ | 3
4 |          | 4
5 |     @    | 5
6 |          | 6
7 |   @      | 7
8 |x       @ | 8
9 |          | 9
   0123456789

Your ships: 5 | Computer ships: 5
```

The game is over when one player or computer has no ship left.

```
Your ships: 5 | Computer ships: 0
Hooray! You win the battle :)
```

Project Help Link:

Project Walk Through