# CS 101 - Algorithms & Programming I

Fall 2025 - Lab 4

**Due: Week of October 20**

*Remember the **honor code** for your programming assignments.*
*For all labs, your solutions must conform to the CS101 style **guidelines**!*
*All data and results should be stored in variables (or constants where appropriate) with meaningful names.*

The objective of this lab is to learn how to use **for** and **while** loops to implement automated repetition. Remember that analyzing your problems and designing them on a piece of paper *before* starting implementation/coding is always a best practice.

## 0. Setup Workspace

Start VSC and open the previously created workspace named `labs_ws`. Now, under the `labs` folder, create a new folder named `lab4`.

In this lab, you are to have four Java classes/files (under `labs/lab4` folder) as described below. A fourth and fifth Java file containing the revisions should go under this folder as well. We expect you to submit a total of 5 files, including the revisions, **without compressing** them. Do *not* upload other/previous lab solutions in your submission. The user inputs in the sample runs are shown in blue.

For all parts of this assignment, you may assume that the user enters valid types and numbers of values unless stated otherwise (e.g., a positive integer when asked for a positive integer).

## 1. Triangle Shape

Create a new/empty file of your own under the `lab4` folder named `Lab04_Q1.java` with a class with the same name. In this program, you are expected to print an "Equilateral triangle" with stars. To decide the height of the "Triangle" letters, you will get an input **integer** from the user. You are supposed to check the input and proceed if it is a positive number and less than 30. You should ask for input until a valid number is given. Your task is to print the "Triangle" letters using stars at a height provided by the user (see sample run below), using a **while** loop.

***Sample run***

```
Enter a valid height: 0
Invalid input, try again.
Enter a valid height: -4
Invalid input, try again.
Enter a valid height: 30
Invalid input, try again.
Enter a valid height: 8
       *
      ***
     *****
    *******
   *********
  ***********
 *************
***************
```

## 2. Palindrome Check

Create a new file under the `lab4` folder named `Lab04_Q2.java`, containing a class with the same name. Your task is to write a Java program that repeatedly asks the user to input a positive integer. The program should use a for/while loop to calculate whether the user input is a palindrome or not. The program should stop when the user enters zero.

***Note:*** Palindrome - noun, a word, phrase, or sequence that reads the same backwards as forwards

***Sample run***

```
Enter a positive integer (0 to quit): -14
Only positive integers are allowed.
Enter a positive integer (0 to quit): 21
21 is not a palindrome.
Enter a positive integer (0 to quit): 215
215 is not a palindrome.
Enter a positive integer (0 to quit): 212
212 is a palindrome.
Enter a positive integer (0 to quit): 1000001
1000001 is a palindrome.
Enter a positive integer (0 to quit): 13454321
13454321 is not a palindrome.
Enter a positive integer (0 to quit): 0
Program ended.
```

## 3. Battleship[1] (Board Game)

Create a new file under the `lab4` folder named `Lab04_Q3.java`, containing a class with the same name. Your task is to implement a simplified version of the popular game "Battleship". Even if you are familiar with the original game, please carefully read the instructions, as there are several simplifications and modifications. A sample game run is provided at the end of the explanation—*make sure your output aligns exactly with the given examples.*

**Remark:** You are **not** allowed to use any data types or classes that we have not yet covered (e.g., classes from the Collection framework such as `List`, `Array`, and `ArrayList`).

### Game Overview

Our version of **Battleship** is a **3-player guessing game** involving **Player1**, **Player2**, and **Player3**. The game is played on a **5×5 grid** containing **5 hidden ships**, placed randomly by the computer. The players take turns guessing the location of the ships by specifying a row and column coordinate. The computer acts as the game manager, placing ships randomly and checking guesses. Players do not choose moves automatically; instead, each player takes their turn in sequence until all ships are found.

All players' scores are tracked based on the number of ships they successfully find.

### Objective

The objective of the game is to be the player who locates the most ships. Players take turns guessing coordinates on the grid.

- If a player correctly guesses a ship's location, it counts as a **hit**, and the player earns a point.
- If a guess misses, the grid is updated accordingly, and the next player takes their turn.

[1] "Battleship (game)", Wikipedia. Available at: https://en.wikipedia.org/wiki/Battleship_(game)

The game ends when **all 5 ships have been found**. The player with the highest number of hits is declared the winner. If multiple players have the same number of hits, the game ends in a tie.

### Game Flow

The game proceeds in the following steps:

**1. Grid and Ship Setup**

- The computer generates a **5×5 grid** and initializes it with water (~).
- Five ships are placed randomly in distinct grid positions.
- Ship locations are hidden from players.

**2. Player Turns**

- In each round, every player takes one turn.
- A turn consists of guessing a location by specifying a row and a column.
- The computer checks the guess:
    - **If the guess is correct** (a ship is located at that position):
        - The cell is marked with an X.
        - The ship is considered found and removed from play.
        - The player earns a point.
    - **If the guess is incorrect**:
        - The cell is marked with an 0.
        - No points are awarded.
- If the guessed location has already been guessed, the player is informed and loses that turn.

**3. Ending the Game**

- The game continues in rounds until all **five ships** are found.
- Once all ships are found:
    - Scores are displayed for each player.
    - The player with the highest score wins the game.
    - If two or more players have the same highest score, the game ends in a tie.

## A. Grid Creation & Initial Ship Placement

First, construct the game grid. The grid is a 5×5 board, represented as a string containing 25 characters. Each character initially represents water and is set to `"~"`. Next, randomly place 5 ships on the grid. Ships are represented by `"1"` in a separate string of the same length as the grid. Initially, all positions contain `"0"`, representing no ship.

Using a loop, randomly select grid positions until 5 *unique* positions are filled with `"1"`. These positions represent the ships and are hidden from the players during the game. After ship placement, print a welcome message and show the initial empty grid to all players.

**Tip:** To randomize, you can use methods from the Java `Random` class.

```
Welcome to Battleship!
The grid is 5×5. There are 5 hidden ships.

Initial Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

## B. Game Start & Player Turns

The game proceeds in rounds. Each round consists of a turn for each player in sequence: Player1 → Player2 → Player3.

During a player's turn:

1. The program prints the previous grid showing all prior hits (`X`) and misses (`O`).
2. The player is asked to input a row and column guess.
3. The program checks the guess:
    - Hit: If a ship is at the guessed location and hasn't been hit before:
        - Mark the grid cell as `"X"`
        - Remove the ship from the hidden ship list.
        - Award the player a point.
        - Print `"Hit!"`
    - Miss: If no ship is at the guessed location:
        - Mark the grid cell as `"O"`
        - Print `"Miss!"`
    - Already guessed location: If the cell was guessed before:
        - Print `"Already guessed here!"`
    - Invalid guess: If the coordinates are outside the grid bounds:
        - Print `"Invalid guess. Try again."`

After each player's turn, print the updated grid showing all hits and misses.

```
###### Round #1
## Player1 ##
Previous Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 0
Guess column: 0
Miss!
Current Grid:
O ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

## C. Game Continuation & End Condition

The game continues with players taking turns in sequence until all 5 ships are found. Once all ships are found:

- Print `"Game finished!"`
- Show the total number of rounds played.
- Show the total number of guesses.
- Show each player's score (number of ships found).
- Announce the winner. If there is a tie, announce that the game ended in a tie.

```
Game finished!
Total rounds played: 5
Total guesses: 15
Player1 score: 1 ships found
Player2 score: 1 ships found
Player3 score: 3 ships found
Winner: Player3
```

## Sample run for full game:

### *Round 1*

```
Welcome to Battleship!
The grid is 5x5. There are 5 hidden ships.

###### Round #1
## Player1 ##
Previous Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 0
Guess column: 1
Hit!
Current Grid:
~ X ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

```
## Player2 ##
Previous Grid:
~ X ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 0
Guess column: 2
Miss!
Current Grid:
~ X O ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

```
## Player3 ##
Previous Grid:
~ X O ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 3
Guess column: 1
Miss!
Current Grid:
~ X O ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ O ~ ~ ~
~ ~ ~ ~ ~
```

### *Round 2*

```
###### Round #2
## Player1 ##
Previous Grid:
~ X O ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ O ~ ~ ~
~ ~ ~ ~ ~
Guess row: 4
Guess column: 0
Miss!
Current Grid:
~ X O ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ O ~ ~ ~
O ~ ~ ~ ~
```

```
## Player2 ##
Previous Grid:
~ X O ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ O ~ ~ ~
O ~ ~ ~ ~
Guess row: 2
Guess column: 2
Miss!
Current Grid:
~ X O ~ ~
~ ~ ~ ~ ~
~ ~ O ~ ~
~ O ~ ~ ~
O ~ ~ ~ ~
```

```
## Player3 ##
Previous Grid:
~ X O ~ ~
~ ~ ~ ~ ~
~ ~ O ~ ~
~ O ~ ~ ~
O ~ ~ ~ ~
Guess row: 1
Guess column: 4
Miss!
Current Grid:
~ X O ~ ~
~ ~ ~ ~ O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ ~ ~ ~
```

### *Round 3*

```
###### Round #3
## Player1 ##
Previous Grid:
~ X O ~ ~
~ ~ ~ ~ O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ ~ ~ ~
Guess row: 1
Guess column: 3
Miss!
Current Grid:
~ X O ~ ~
~ ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ ~ ~ ~
```

```
## Player2 ##
Previous Grid:
~ X O ~ ~
~ ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ ~ ~ ~
Guess row: 4
Guess column: 3
Miss!
Current Grid:
~ X O ~ ~
~ ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ ~ O ~
```

```
## Player3 ##
Previous Grid:
~ X O ~ ~
~ ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ ~ O ~
Guess row: 4
Guess column: 2
Hit!
Current Grid:
~ X O ~ ~
~ ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ X O ~
```

### *Round 4*

```
###### Round #4
## Player1 ##
Previous Grid:
~ X O ~ ~
~ ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ X O ~
Guess row: 1
Guess column: 0
Hit!
Current Grid:
~ X O ~ ~
X ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ X O ~
```

```
## Player2 ##
Previous Grid:
~ X O ~ ~
X ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ ~
O ~ X O ~
Guess row: 3
Guess column: 4
Miss!
Current Grid:
~ X O ~ ~
X ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ O
O ~ X O ~
```

```
## Player3 ##
Previous Grid:
~ X O ~ ~
X ~ ~ O O
~ ~ O ~ ~
~ O ~ ~ O
O ~ X O ~
Guess row: 3
Guess column: 3
Miss!
Current Grid:
~ X O ~ ~
X ~ ~ O O
~ ~ O ~ ~
~ O ~ O O
O ~ X O ~
```

### *Round 5*

```
###### Round #5
## Player1 ##
Previous Grid:
~ X O ~ ~
X ~ ~ O O
~ ~ O ~ ~
~ O ~ O O
O ~ X O ~
Guess row: 2
Guess column: 4
Miss!
Current Grid:
~ X O ~ ~
X ~ ~ O O
~ ~ O ~ O
~ O ~ O O
O ~ X O ~
```

```
## Player2 ##
Previous Grid:
~ X O ~ ~
X ~ ~ O O
~ ~ O ~ O
~ O ~ O O
O ~ X O ~
Guess row: 0
Guess column: 4
Miss!
Current Grid:
~ X O ~ O
X ~ ~ O O
~ ~ O ~ O
~ O ~ O O
O ~ X O ~
```

```
## Player3 ##
Previous Grid:
~ X O ~ O
X ~ ~ O O
~ ~ O ~ O
~ O ~ O O
O ~ X O ~
Guess row: 4
Guess column: 4
Miss!
Current Grid:
~ X O ~ O
X ~ ~ O O
~ ~ O ~ O
~ O ~ O O
O ~ X O O
```

### *Round 6*

```
###### Round #6
## Player1 ##
Previous Grid:
~ X O ~ O
X ~ ~ O O
~ ~ O ~ O
~ O ~ O O
O ~ X O O
Guess row: 3
Guess column: 0
Miss!
Current Grid:
~ X O ~ O
X ~ ~ O O
~ ~ O ~ O
O O ~ O O
O ~ X O O
```

```
## Player2 ##
Previous Grid:
~ X O ~ O
X ~ ~ O O
~ ~ O ~ O
O O ~ O O
O ~ X O O
Guess row: 0
Guess column: 0
Miss!
Current Grid:
O X O ~ O
X ~ ~ O O
~ ~ O ~ O
O O ~ O O
O ~ X O O
```

```
## Player3 ##
Previous Grid:
O X O ~ O
X ~ ~ O O
~ ~ O ~ O
O O ~ O O
O ~ X O O
Guess row: 2
Guess column: 0
Miss!
Current Grid:
O X O ~ O
X ~ ~ O O
O ~ O ~ O
O O ~ O O
O ~ X O O
```

### *Round 7*

```
###### Round #7
## Player1 ##
Previous Grid:
O X O ~ O
X ~ ~ O O
O ~ O ~ O
O O ~ O O
O ~ X O O
Guess row: 1
Guess column: 1
Miss!
Current Grid:
O X O ~ O
X O ~ O O
O ~ O ~ O
O O ~ O O
O ~ X O O
```

```
## Player2 ##
Previous Grid:
O X O ~ O
X O ~ O O
O ~ O ~ O
O O ~ O O
O ~ X O O
Guess row: 0
Guess column: 3
Miss!
Current Grid:
O X O O O
X O ~ O O
O ~ O ~ O
O O ~ O O
O ~ X O O
```

```
## Player3 ##
Previous Grid:
O X O O O
X O ~ O O
O ~ O ~ O
O O ~ O O
O ~ X O O
Guess row: 1
Guess column: 2
Miss!
Current Grid:
O X O O O
X O O O O
O ~ O ~ O
O O ~ O O
O ~ X O O
```

### *Round 8*

```
###### Round #8
## Player1 ##
Previous Grid:
O X O O O
X O O O O
O ~ O ~ O
O O ~ O O
O ~ X O O
Guess row: 2
Guess column: 1
Hit!
Current Grid:
O X O O O
X O O O O
O X O ~ O
O O ~ O O
O ~ X O O
```

```
## Player2 ##
Previous Grid:
O X O O O
X O O O O
O X O ~ O
O O ~ O O
O ~ X O O
Guess row: 2
Guess column: 3
Miss!
Current Grid:
O X O O O
X O O O O
O X O O O
O O ~ O O
O ~ X O O
```

```
## Player3 ##
Previous Grid:
O X O O O
X O O O O
O X O O O
O O ~ O O
O ~ X O O
Guess row: 3
Guess column: 2
Hit!
Current Grid:
O X O O O
X O O O O
O X O O O
O O X O O
O ~ X O O

**********
Game finished!
Total rounds played: 8
Total guesses: 24
Player1 score: 3 ships found
Player2 score: 0 ships found
Player3 score: 2 ships found
Winner: Player1
```

## Other cases:

### Case 1 — Guessing the same location multiple times (not necessarily in a sequence)

```
###### Round #1
## Player1 ##
Previous Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 2
Guess column: 3
Miss!
```

```
Current Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ O ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

```
## Player2 ##
Previous Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ O ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 2
Guess column: 3
Already guessed here!
```

```
Current Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ O ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

### Case 2 — Guess out of bounds

```
## Player1 ##
Previous Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 5
Guess column: 1
Invalid guess. Try again.
```

```
Current Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

### Case 3 — Hitting the same ship twice (multiple times)

```
## Player1 ##
Previous Grid:
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 1
Guess column: 1
Hit!
```

```
Current Grid:
~ ~ ~ ~ ~
~ X ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

```
## Player2 ##
Previous Grid:
~ ~ ~ ~ ~
~ X ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
Guess row: 1
Guess column: 1
Already hit this ship!
```

```
Current Grid:
~ ~ ~ ~ ~
~ X ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
~ ~ ~ ~ ~
```

### Case 4 — Tie at the end

```
**********
Game finished!
Total rounds played: 6
Total guesses: 18
Player1 score: 2 ships found
Player2 score: 2 ships found
Player3 score: 1 ships found
It's a tie!
```