CS 449 Homework Overview

All homework assignments are based on this semester-long project. It aims to help you understand the key software engineering activities by developing a board game of ***product quality*** through an incremental process.

1. Problem Description

Your customer asks you to develop the software that allows a blue player to play the SOS game against a red player. Either player can be human or computer.

The game board is a grid of n×n (**n>2**) squares. The two players take turns to add either an "S" or an "O" to an **unoccupied** square, with no requirement to use the same letter each turn. Each player attempts to create the straight sequence S-O-S among connected squares (diagonally, horizontally, or vertically). To keep track of who made which SOSs, a line in the player's color (i.e., blue or red) is drawn for each SOS sequence, as shown in the following figure.

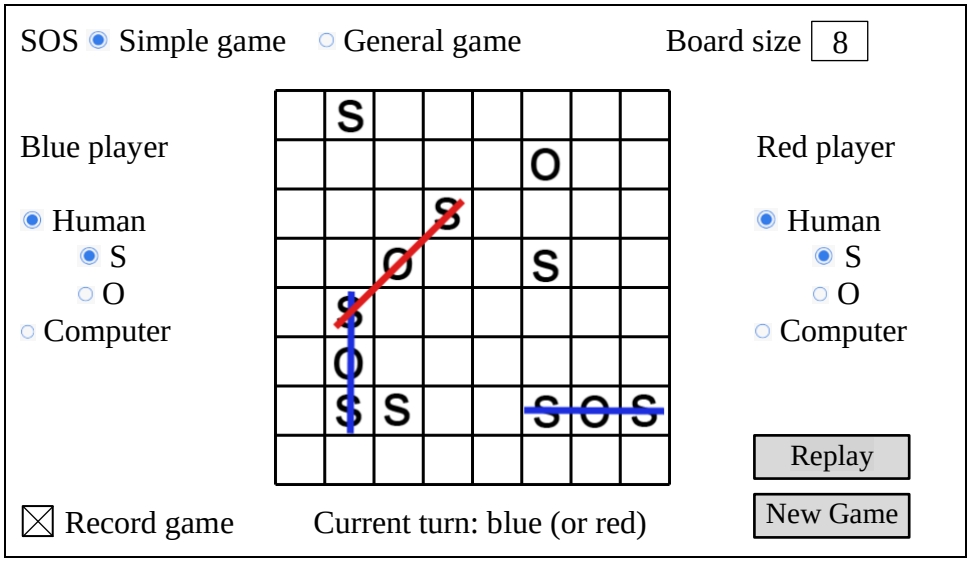

Figure 1. Sample GUI layout of the final product

The instructions of each assignment may include a sample graphical user interface (GUI). **As GUI is a topic in CS 101**, you are strongly encouraged to implement a GUI for the SOS game. The use of an interactive console interface is strongly discouraged. The TicTacToe case study to be introduced in class is an excellent example to follow.

The SOS game can be played in one of the following modes:

(a) **Simple game**: The player who creates the first SOS is the winner. If no SOS is created when the board has been filled up, then the game is a draw. Turns alternate between players after each move.

(b) **General game**: The game continues until the board has been filled up. The winner is the player who made the most SOSs. If both players made the same number of SOSs, then the game is a draw. When a player succeeds in creating an SOS, that player immediately

takes another turn and continues to do so until no SOS is created on their turn. Otherwise, turns alternate between players after each move.

You may determine the functional and quality requirements according to the above description, future enhancement for other similar games, and your imagination (e.g., undo and redo).

You may implement the software as a standalone program, a web application, or a mobile app. Although Java is strongly recommended, you may choose C++, Python, or C#.

- Your development environment must include a unit test framework and a GUI library.
- You should maximize the practices of object-oriented programming (OOP), which is an important topic in CS 101 and CS 201. If you are not familiar with OOP, you should quickly review the essential concepts such as **class, inheritance**, **method overriding**, **polymorphism**, and **dynamic binding**.
- **You must use GitHub as your code version control system. You will get no points if your submission does not have a link to your project repository.**
- **Feel free to use the free version. If you would like to explore the PRO version, you can do so for free while you are enrolled in the university: https://github.com/education/students**

2. Deliverables, Weights, and Deadlines

The project consists of an initial planning and five iterations (sprints). The deliverables, weights, and deadlines are given in the following table.

Iteration	***Deliverable***	***Weight***	***Deadline***
Sprint 0	Planning and self-study on GUI & unit testing Make decisions on the programming language, GUI library, IDE, unit test framework, programming style, and project hosting site (e.g., at github.com)	10%	Week 4
Sprint 1	User stories and acceptance criteria for a human player to play a simple or general SOS game against another human player	15%	Week 6
Sprint 2	Game initialization and S/O placement for human players Coding requirements: separation of user interface and game logic; unit testing	15%	Week 8
Sprint 3	Complete simple/general games Coding requirements: separation of user interface and game logic; unit testing Design: use class hierarchy to deal with simple/general game	20%	Week 10
Sprint 4	Computer opponent, including user stories/acceptance criteria and implementation Coding requirements: separation of user interface and game logic; unit testing Design and refactoring: class diagram, inheritance	24%	Week 13
	Record and replay (including user stories, acceptance criteria, and implementation), design review	16%	Week 16

Sprint 5	Coding requirements: unit testing Design and refactoring: class diagram, review, pre/postconditions		

3. Extra Credit

Up to **5%** may be added to your ***final grade*** for quality or other enhancements, for exceptionally well-written reports, and for the overall impression of the project, which the instructor deems to be deserving of special recognition. Projects without a GUI are not eligible for extra credit.