SYSC3110 – Scrabble Project

Team Members

Gabriel Bugarija 101273397

Kemal Sogut 101280677

Mark Osbourne 101312763

Random Hygaard 101273397

Project Overview

This project is a simplified version of the classic board game **Scrabble**, built as part of the SYSC3110 course at Carleton University. The main goal was to recreate the game logic, board system, and player interaction in Java, following an object-oriented design approach.

For this milestone, the focus was on the **Model** portion of the MVC architecture — meaning all the core game logic is implemented, while graphical elements (GUI) will be developed in future iterations.

Players can play the game directly from the **console**, draw letter tiles, form words, and see their scores updated based on valid moves.

Features Implemented

- **Board setup** with cells and multipliers
- **Tile bag** that randomly distributes letter tiles
- **Player system** supporting multiple players with individual racks and scores
- **Word validation** using a dictionary file
- **Scoring system** that calculates and updates player scores
- **Turn-based gameplay** allowing players to place words or skip turns
- **Console output** displaying the game board and player information

Classes

Here’s a quick rundown of the main classes and their roles:

- **Tile** – Represents an individual letter tile with its character and point value.
- **Cell** – Represents each square on the Scrabble board, holding a tile and a multiplier.
- **Board** – Manages the grid of cells, word placement, and overall game layout.
- **TileBag** – Stores all available tiles and handles random drawing.
- **Player** – Manages player name, rack, and score.
- **Dictionary** – Validates words against a predefined list of valid entries.
- **Move** – Handles the logic behind placing words and validating moves.
- **ScoreCalc** – Calculates the score for each word placement.
- **Game** – Ties all components together, managing turns and game flow.
- **Main** – Entry point that initializes and runs the game in the console.

Data Structures Justification

We used **ArrayLists** to store variable-sized collections like player racks and the tile bag because of their dynamic resizing and indexing efficiency.
A **2D array** was chosen for the board since it provides straightforward access using row and column indices.
For the **dictionary**, a **HashSet** is ideal because of its fast lookup times when checking whether a word is valid.

Known Issues

- Some edge cases in word validation may not be fully handled (e.g., crosswords formed from existing letters).
- The game currently runs in console mode only — GUI will be added in Milestone 2.
- Premium squares (like double word or triple letter) are not yet implemented.

Next Steps

- Implement the **graphical interface (GUI)** using Java Swing.
- Add **premium tiles** and **blank tiles** for bonus scoring.
- Include **AI opponents** that can play automatically.
- Add **undo/redo** and **save/load** features in later milestones.

How to Run

Compile all .java files in the src directory.

- javac *.java

Run the Main class:

- java Main
- Follow on-screen prompts to play the game via the console.