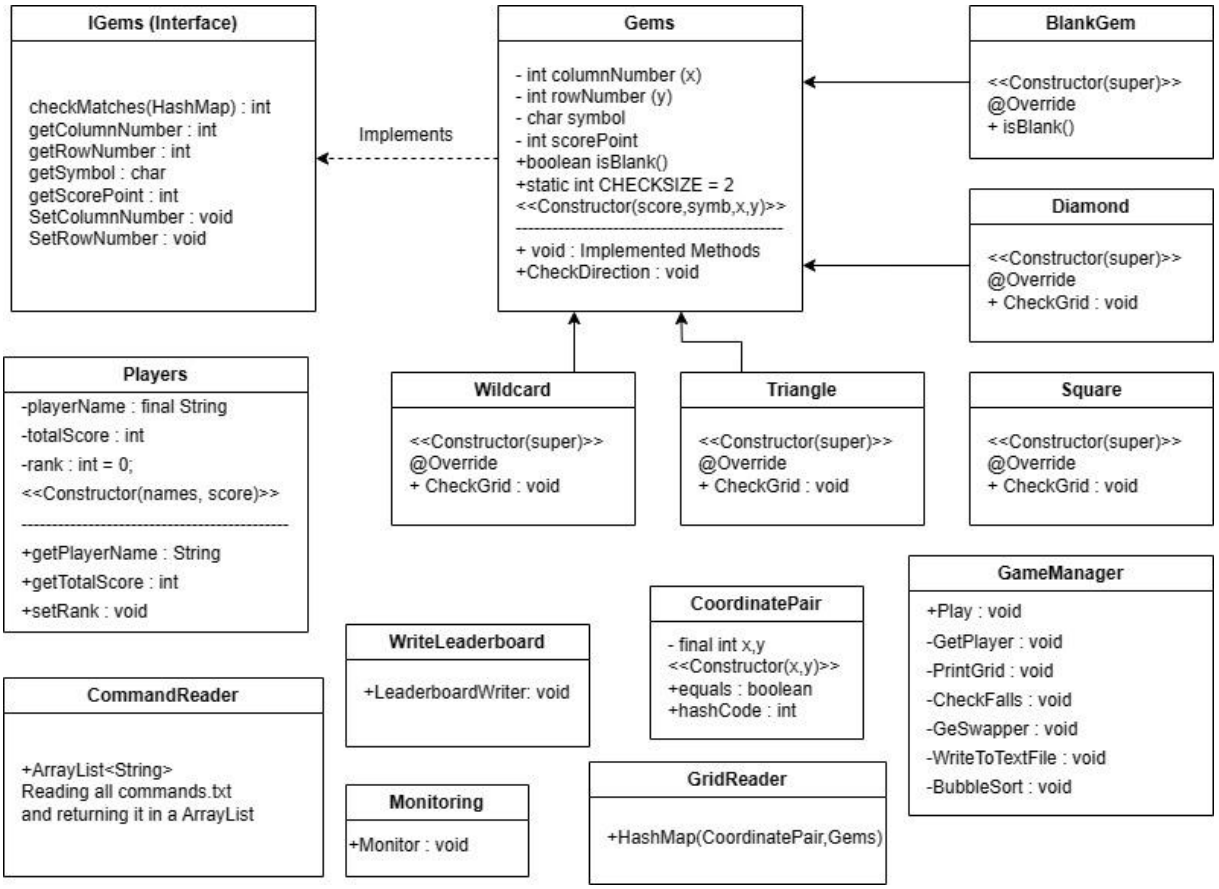
Advanced Programming

Project Assignment 2 - Report

Nuri Yalcin - 210313011



UML Diagram of the project.

Classes and Interfaces

GridReader Class

It reads gameGrid.txt file and adds all gems by their symbol type and gives them unique coordinate keys in a HashMap (starting from 0,0 to 9,9 in format of x,y).

IGems Interface

“IGems” interface has CheckMatches and getter-setter functions in it.

Gems Class

Gems class implements IGems interface. This class has common variables for all other gems that extend this super class. There is a static variable named “CHECKSIZE” and it represents the checking range for gem matches for this project checker looks for triplets so “CHECKSIZE” needs to be 2 (selected gem + two other matching gems).

Class has all getter-setter functions and CheckMatches that are implemented from interface IGems.

CheckMatches function is used for selected gems and returns the total point that has been gained from an action that was made by the player (Selecting a coordinate, checking matches and destroying gems if there is a triple match).

There is a CheckDirection function that returns an int array. The int array contains 2 elements.

First is the checker part. It can be 0, 1 or 2. And the second is points.

```
public int[] CheckDirection(HashMap<CoordinatePair, Gems> hashmap, int xOffset, int yOffset, char targetSymbol) {
    int x = this.getColumnNumber();
    int y = this.getRowNumber();
    int n = (int) (Math.sqrt(hashmap.size()) - 1);
    int[] matchChecker = {0, this.getScorePoint()};//0 is checker 1 is adding points

    for (int i = 1; i <= CHECKSIZE; i++) {
        int newX = x + (i * xOffset);
        int newY = y + (i * yOffset);

        if (newX >= 0 && newX <= n && newY >= 0 && newY <= n) {
            Gems nextGem = hashmap.get(new CoordinatePair(newX, newY));
            if (nextGem.getSymbol() == this.getSymbol() || nextGem.getSymbol() == targetSymbol) {
                matchChecker[0]++;
                matchChecker[1] += (nextGem.getSymbol() == this.getSymbol()) ? this.getScorePoint() : 10;
            }
        }
    }
    if (matchChecker[0] == CHECKSIZE) {
        for (int i = 0; i <= CHECKSIZE; i++) {
            int newX = x + i * xOffset;
            int newY = y + i * yOffset;
            hashmap.put(new CoordinatePair(newX, newY), new BlankGem(newX, newY));
        }
        return matchChecker;
    } else {
        matchChecker[0] = 0;
        matchChecker[1] = this.getScorePoint();
    }
    return matchChecker;
```

HashMap is the grid that has been read from the gameGrid.txt file. xOffSet and yOffSet can be -1, 0 and +1 (like scale factor). They show the way the program will check. Program checks the direction and if there is a match it increases checker and points.
After the check part finishes and if the checker equals CHECKSIZE program replaces checked gems with BlankGems in HashMap.

Diamond,Square,Triangle,Wildcard and BlankGem class.

These gems extend “Gems” super class. They only have different symbols, scorePoints and different Overridden CheckMatches functions which have different xOffSet and yOffset for diagonal, vertical and horizontal search.

GameManager Class

The GameManager class has gameplay functions.
-Play is for playing the game with the read inputs from commands.txt and printing all events in the gameplay session.
-GetPlayer adds a new player to the leaderboard list.
-PrintGrid prints the whole grid after checking whether the gems will fall or not.
-CheckFalls looks at all gems in HashMap. And swaps the gems whose bottoms are empty with their bottoms.
-WriteToTextFile writes all leaderboard to leaderboard.txt file whenever leaderboard is updated.
-This project uses BubbleSort for sorting players by their scores.

CommandReader Class

Read all the inputs in commands.txt file and put them in a String ArrayList.

Monitoring Class

***(I got help from ChatGPT and BingAI with printStream and FileOutputStream parts.)
Monitor function plays the game and prints all events in the game in the monitoring.txt file.

Commands.txt File Format

commands.txt file format needs to be like :

1. coordinate selections -> “(x y)” Integers.
    a. You can play as many moves as you want.
2. E to end the game -> “E”
3. Player’s name
4. 0 or 1 to replay as a new player.
5. If the answer is 1 same as the beginning.
6. If 0 it’s the last line of commands.txt.

If the order is not like this game can be not working well.

Adding New Type of Gems in Game

You can add new Gem types like mathematical functions (/, \, +, |) in 4 steps :

1. Update the game grid in the gameGrid.txt file.
2. Create the new gem class that extends Gems Class and has the functions like others.
3. Change CheckMatches function’s xOffset and yOffset by your new rules wish.
    For diagonal search change x and y need to be -1 or 1.
    For horizontal search x needs to be -1 or 1 and y is 0.
    For vertical search y needs to be -1 or 1 and x is 0.
4. Add gem type in GridReader class’s switch statement.

```
switch (lett) {
    case 'D' -> gemx = new Diamond(columnNumber, rowNumber);
    case 'S' -> gemx = new Square(columnNumber, rowNumber);
    case 'T' -> gemx = new Triangle(columnNumber, rowNumber);
    case 'W' -> gemx = new Wildcard(columnNumber, rowNumber);
}
abc.put(new CoordinatePair(gemx.getColumnNumber(), gemx.getRowNumber()), gemx);
```