

Coursework

Software Development 1

Matriculation no: 40451337

Name: Kacper Cichosz

Project: Connect Four game

**Abstract**

We were asked to create Connect Four game, where:

*“...the players first choose a colour and then take turns dropping coloured discs from the top into a seven-column, six-row grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to connect four of one's own discs of the same colour next to each other vertically or horizontally before your opponent.”*

**Design**

The game consists of four classes and one main class to run program. These four objects are called: User, Grid, Check and Game. I decided to create game in 2D displayed array.

User Class

This object class is responsible for creating two players. Each of the players has two attributes: name and displayColour.

Attribute displayColour represents String which is displayed on grid for both players. It is “Y” for player choosing yellow colour and “R” for player choosing red colour.

Grid Class

This class holds all information about the grid. It is where we start the grid, display the whole grid, update the grid based on user entry and set size of the grid.

Check Class

In this object all necessary calculations take place. We check if one of the players collected four discs in a row and in four different ways: horizontally, vertically, diagonally and opposite diagonally. There is an extra checkResult method, which verifies if conditions for one of the aforementioned ways are met.

Also, we check if the column selected by the user is not already full. Another thing in this class is to verify if there is a draw.

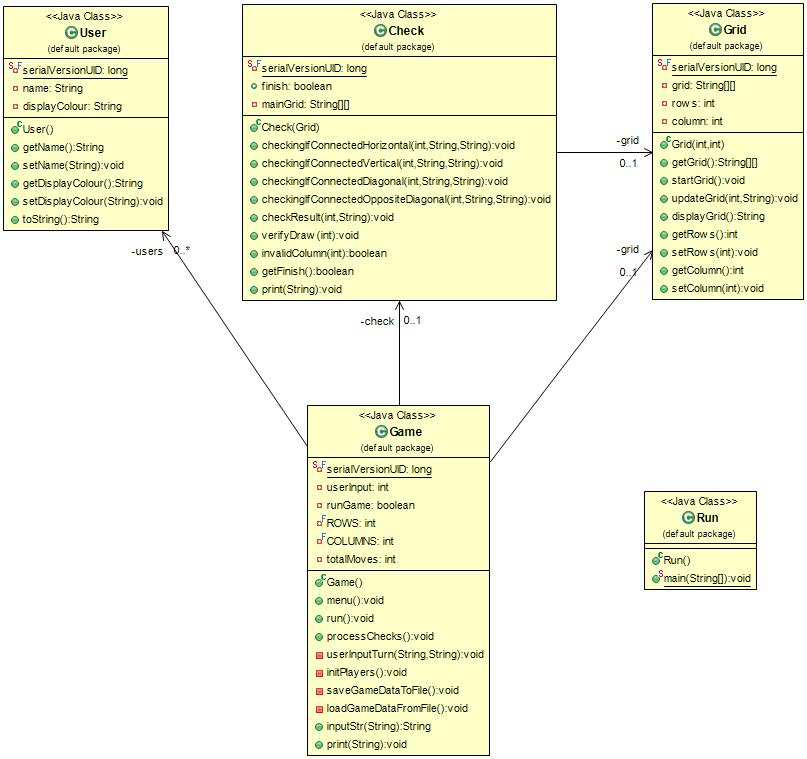
Game Class

This class will collect and process all other objects into one functioning class.

The whole program focuses on the menu method. Depending on the user’s selection, there will be three options:

1. Play game – in this scenario, we choose the run() method, which is responsible for taking all the needed methods, found in this class, to play the game. They are: initialize players, start grid, display grid, place disc in column selected by the player and verify if there is draw.
2. Load game – this option will load previously saved game.
3. Exit game – in this scenario the program will finish.

Diagram



**The Grid**

The grid is displayed as a String in 2D Array. It is done using two for loops, with one of them being nested for loop. First loop iterates row’s number. Second loop is nested inside each row and iterates the number of columns. For each position in this.grid[rows][columns] a square with ANSI code is placed.

In order to place the disc inside the grid on a position selected by the user, we need to iterate all rows as for loop with a decreasing counting. Each row in the column selected by user needs to be checked: the first row containing the ANSI square, will be changed with the user’s displayColour.

**Discs Check**

Horizontally and Vertically

Iteration of all rows is needed first in order to check four discs in a row. For each check the variable count starts at 0. This variable holds a number of player’s discs found in a row. Once it reaches the number 4, the game will finish.

The user’s column input and displayColour are used as parameters to check the grid horizontally. Firstly, a check is performed on the right side of the grid from the place where the user submitted his disc. Once it finds another disc of the same colour, it adds +1 to the variable count. The same is repeated for the left side. In the end variable count is verified by the checkResult method. If the method finds count equals 4, it returns victory for the player being validated.

The same process applies to the vertical check. When the player selects a column to place the disc on the grid, method checks if there are next 3 rows in this column, which have the same colour of discs. If so, then the variable count will equal 4 and checkResult will return victory for the player being validated.

Diagonally

For diagonal check a place where the player submitted the disc had to be found and then with each iteration:

1. Check left from user’s submission: increase row by 1 (go down on grid display) and decrease column by 1.
2. Check right from user’s submission: decrease row by 1 (go up on grid display) and increase column by 1.

In between switching from checking left side to right side, the position has been moved one number higher horizontally and vertically than the one already checked.

Similar scheme applies for opposite diagonally checking. With each iteration:

1. Check right side from user’s input: increase row by 1 (go down on grid display) and increase column by 1.
2. Check left side from user’s input: decrease row by 1 (go up on grid display) and decrease column by 1.

As previously, in between switching from checking right side to left side, the position has been moved one number lower horizontally and vertically than the one already checked.

**Disc Adding Prevention**

In order to prevent the player from adding disc to a full column an invalidColumn method was added with parameter as player’s column input. Next, local variable count was created and iterated by the number of columns. If in the column selected by the player in each row result was different than the default ANSI square, then the count variable was increased by 1. In the end if count variable was equal to the total number of rows, the method returned Boolean true, otherwise it kept returning variable false.

**Save and Load game**

In order to save game, two new methods inside Game class had to be created. First one was saveGameDataToFile(). New file savegame.txt is opened, this file will hold all necessary data about save status. Afterwards Output Stream Object is created with name: fileStream. In order to save game, fileStream writes data about Grid object in txt file and once finished, close it.

Extra button inside game is added, if player decides to save game progress.

Second method is loadGameDataFromFile(). Over here we have to read data saved in previously saved object. To do that new Input Stream Object is created called file. Next, data from grid are read from savegame.txt.

Each of classes is implemented with Serializable and to each of them is added generated serialVersionUID Key.