**C:\Users\admin\Desktop\TU-LOGO1.png**

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

**Програмни системи**

**Курсова работа**

**Изготвил: Росен Димов Одобрил:................**

**ФКСУ, група 50 /подпис/**

**Фак. номер: 121210053**

Описание на използваните класове и методи в

играта „Царици“

* **Класове, свързани с логиката на играта**

**enum QueensTurnOutput –** енумерация, представляваща изхода от даден ход (валиден, невалиден, загубен, спечелена игра).

**class Queensboard** – клас, описващ шахматната дъска, върху която се поставят кралиците на ниво логика.

**Конструктор:**

**public Queensboard(int size)** – приема за параметър размера на дъската.

**Полета:**

**int size** – големина на дъската.

**bool[,] boardPositions** – представлява дъската на логическо ниво, като показва къде може да се играе, къде е играно до момента и т.н.

**bool[,] queensPositions** – съдържа само информация къде има поставени кралици.

**bool[] rowsUnplayable** - съдържа информация на кои редове вече има поставени кралици.

**bool[] colsUnplayable** – на кои колони има поставени кралици.

**int queensPlaced** – брой на поставените до момента краилици.

**Свойства:**

**bool IsCompleted** – показва дали играта е завършила успешно (поставени са N кралици).

**Методи:**

**bool HasOptionsLeft() –** връща true ако има поне един ред (съответно и колона), на който може да се играе, в противен случай връща false (когато всички незаети позиции са маркирани като невъзможни за игра).

**bool IsPlayablePosition(int row, int col)** – проверява дали позиция [row, col] е възможна за игра.

**void OccupyPosition(int row, int col)** – поставя кралица на дадена позиция (вече е гарантирано, че тя не е заета).

**void MarkPositionsAsUnplayable(int row, int col)** – маркира всички позиции (включително и [row, col]), които могат бъдат обрани от [row, col] (хоризонтално, вертикално и по двата диагонала) като невъзможни за игра.

**QueensTurnOutput GetOutputFromTurn(int row, int col)** – единствения публичен метод в класа, който може да се извиква от класа на формата. Връща какъв е резултата от хода – *валиден* (ако в резултат на хода играта продължава), *невалиден* (има кралица на тази позиция – това не означава, че играта приключва), *приключена игра* (всички N на брой кралици са поставени успешно без засичане) и *загубен* (това може да се случи, дори на дадена позиция да няма засичане с други кралици, играта е загубена и ако след изиграването на хода няма други подходящи позиции за игра, т.е. всички останали полета са маркирани като невъзможни за игра).

* **Класове, свързани с изгледа на играта**

**class ChessPanel : Panel –** клас, който наследява стандартния клас Panel, като добавя полета Row и Col, нужни за логиката на играта и представляващи позицията, на която е играно.

**class WelcomeForm : Form** – клас, който представлява началния изглед на играта. Единственият метод в него void confirmBoardButton\_Click(object sender, EventArgs e) валидира избора за размер на полето при натискане на бутона за потвърждение на избора. Ако избора е валиден се отваря формата, където е самата игра.

**class GameForm : Form** - клас, който представлява визуално самата игра – полето за игра и текстово поле в което се получават съобщения от играта (при загуба, направен ход, невалиден ход и т.н.).

**Конструктор:**

**public GameForm(int size)** – приема за параметър размера на дъската, получен и валиден от по-горната форма. Допълнително се инициализира подходящ размер на прозореца, в зависимост от size, създава се текстовото поле за съобщения от играта и се променят още някои визуални свойства.

**Полета:**

**ChessPanel[,] chessTiles** – представлява самата дъска, но на визуално ниво, като всяко квадратче има визуални свойства – размер, цвят (бял или черен), позиция в прозореца и т.н.

**Queensboard board** – описан по-горе.

**Label infoBlock** – показва съобщения от играта.

**Image queenImage** – статично поле, което съдържа изображение на шахматната фигура кралица (използвано и за икона на играта).

**int boardSize** – големина на дъската.

**int tileSize** – статично поле, което съдържа големината на квадратчето от дъската – то е с големината на картинката (в случая 32, тъй като изображението е 32х32).

**Методи:**

**void InitializeFormBoard() –** инициализира визуално дъската за игра с размер NxN, като добавя event listener за клик към всяко квадратче.

**void chessTile\_Click(object sender, EventArgs e) –** тук се осъществява връзката между визията и логиката на играта. Извиква се гореописания метод **GetOutputFromTurn** за дъската. След това прозореца се обновява (най-вече полето за съобщения) в зависимост от върнатия резултат от хода.

**void SetInfoBlockMessage(string message, Color foreColor) –** променя се текста и цвета на полето за съобщения в зависимост от резултата от хода.

**void DisableTiles() –** извиква се след приключване на играта (независимо успешно или не), като се махат event listener-ите за всички квадратчета, така че при допълнително кликане вече да не се случава нищо.