* Redesigned structure:
* Separated the labyrinth generation (InitializeLabyrinth()) and visualization (ShowLabyrinth()) in a separate class: LabyrinthBoard
* Separated the Top Scores logic (method ShowTopScores and string[] topScores) in a class TopScores
* Renamed the class Labyrinth to Engine as only the main logic of the moving remains there
* Removed some not necessary comments.
* Work on the LabyrinthBoard class:
* Added method bool IsPossibleCell(int row, int col) which checks if the cell on the given row and column is free to move on it and if it is within range.
* Added method bool isPieceOnEdge(int row, int col) which checks if the cell where the Piece is located at the moment, is on an edge
* Introduced constant int LabyrinthSize = 7 for the number of row and consts in the Labyrinth
* Introduced fields that keep the position of the moving piece (int piecePositionRow and int piecePositionCol) plus respective properties.
* Introduced methods that move the piece left, right up and down if possible. This is to take away this functionality from the engine.
* Due to the above two changes, modified the ShowLabyrinth method, so that it can print \* where the piece is;
* Reconfigured the ShowLabyrinth method into a ToString() method, so that the LabyrinthBoard class be decoupled from the actual means of rendering the board.
* Set capital letters in constant LabyrinthSize
* Created new constant PIECEPOSITION for piece of position for initial row and column equal to 3.
* Removed the key work “this” in properties PiecePositionRow and PiecePositionCol
* Changed the name of ll to labyrinth.
* Removed some whitespaces.
* Added ‘this.’ Prefix to indicate a members of the class.
* Work on the TopScores class:
* Removed whitespace.
* Added ‘this.’ Prefix to indicate a members of the class.
* Added a constant for the number of TopScores, int NumberOfTopScores
* Replaced the string array with a KeyValuePair<string, int> list, so that the scores can be sortable and easily searched.
* Removed the check for null values in the ShowTopScores method, as it is no longer necessary with a list
* Removed the long if condition in the end of ShowTopScores, replaced with check if the list count is 0.
* Added a new method EnterTopScore, as there was no such functionality and this made the class useless
* The class is changed to a public class
* Work on the Engine class:
* Made the engine class use the functionality of the LabyrinthBoard class:
* The engine doesn’t make the checks if a cell is free or not and if it possible to move there or not, instead it asks the LabyrinthBoard class
* The engine doesn’t keep the information where the piece is anymore, it just asks the LabyrinthBoard class where it is and to move it, if possible. Removed the fields m and n that used to keep this.
* Replaced the topScores functionality with the functionality from the TopScores class.
* Made a method “Start” which contains the logic from the Program method.
* Renamed the variable \_continue with correct name flagContinue.
* Renamed the variable ll with correct name labyrinth.
* Removed the rest of the occurrences of private i=0, j=0,m=3,n=3.
* Changed the if (input.Length > 1 || input.Length == 0) with short boolean expression input.Length == 1.
* Created two new methods ProcessInputDirection(input) and ProcessInputCommand(input) for process input command or direction from the user.
* Created new method WalkInLabirinth() – moved here all doubled logic for checking of the new cell or exit.
* Crated method Restart () with initialization of the flagContinue for every time when this method is called. Used when needed ‘restart’.
* Removed some whitespaces.
* Added ‘this.’ Prefix to indicate a members of the class.
* Made the Engine class use the functionality of the TopScores class
* Added properties for the Labyrinth and TopScore fields
* Added user interface field and property
* Added default constructor for the Engine class
* Added optional constructor for the Engine class - accepting three parameters - a labyrinth board, top scores and user interface
* Changed Console.ReadLine ( ) to userInterface.GetInput ( ) in methods Move( ) and WalkInLabirinth(ref int steps)
* Work on the UserInput class:
* Added method GetInput( ) - which returns the result of a Console.ReadLine( )