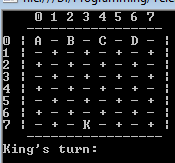
# Refactoring Documentation for Project “King Survival”

Author: Todor Mihailov, TU-Sofia, f.n. 121208040

1. Fix wrong output:

* Fix game board appearance – Print(char[,] matrix) method
  + Get board size from the matrix array instead of static size property

 -> 

* Fix illegal move appearance – removed “\*\*Press any key to continue\*\*” message – it is not included in the requirements

1. Gameplay changes:
   * Pawn move over the king - Pawn should not be able to “catch” the king and win. When Pawn tries to move over king move is Invalid!
   * FIXED king move over pawn- When king moved over pawn move was valid and pawn disappeared. King move over pawn should be invalid
   * King lose when has no valid moves – introduced method IsKingLose() that checks if king has no valid moves
   * FIXED - When king reach the top of the board he doesn’t “win” immediately but after next pawn move
   * Introduced KingMovesCount, PawnsMovesCount moves counters to show the number of moves
   * Figures are shifted left with 1 position to meet initial requirements
2. Merge Pawns move methods PawnAMove, PawnBMove, PawnCMove into one method PawnMove(PawnSymbol,x,y, matrix):
   * Introduce PawnSymbol parameter to specify the pawn that moves
   * Replace PawnAMove(x,y,board) with PawnMove(‘A’,x,y,board)
3. Method IsCellValidForMove checks if cell of next move is in range and empty
4. Method IsCellValidAndEmpty(int row, int column) that checks if cell location is out of board range and is empty
5. Main body extracted to Start method
6. Change project structure
   * Solution renamed to KingSurvival
   * ProjectRenamed to KingSurvival
7. Game class is made non static – static methods are changed into non-static
8. Variable matrix is extracted as a class member from Start() method
9. Rename variables
   * Matrix -> \_gameBoard -> introduced property GameBoard
10. IExtracted method SetInitializeState() from the Start method
11. Rename variables, methods and classes
    * Car -> King
    * Peshka -> Pawn
    * Field x -> \_x
    * Field y -> \_y
    * Game -> KingSurvivalGame
    * Input method parameter :matrix -> gameBoard
    * Print -> PringGameBoard
    * Field car -> introduced property King
    * Field peshkaA -> introduced property PawnA
    * Field peshkaB-> introduced property PawnB
    * Field peshkaC-> introduced property PawnC
    * Field peshkaD-> introduced property PawnD
12. Format the code
    * Change varibles and methods positions in classes
    * Removed empty spaces
    * Removed unnecessary comments
13. Introduce class Figure that replaces the identical King and Pawn
    * Replace variables of type King and Pawn with Figure type
    * Added new property DisplaySymbol to class Figure
    * Added constructor Figure(string displaySymbol)
    * public Figure(int x, int y) -> public Figure(char displaySymbol, int x, int y)
14. **Write UnitTest over console output**
15. Rename field and properties in class Figure
    * \_x to \_column
    * X to Column
    * \_y to \_row
    * Y to Row
16. Refactor Method void KingMove(int dirX, int dirY, char[,] matrica) changed to

* dirX renamed to columnDirection
* dirY renamed to rowDirection
* char[,] matrica is removed from input parameter and replaced with GameBoard as moving a Figure in that way has no sense if other gameBoard is given
* Multiple Ifs combined into one with OR clause
* Extract GameBoard columns and rows - int gameBoardRows = GameBoard.GetLength(0);

             int gameBoardColumns = GameBoard.GetLength(1);

* Extract max column index and max row index:
  + int gameBoardMaxColumnIndex = gameBoardColumns - 1;
  + int gameBoardMaxRowIndex = gameBoardRows - 1;
* Changmethod type- bool

bool KingMove(int columnDirection, int rowDirection)

1. Declare properties that checks if it is KingsTurn or Pawns Turn
2. Change method void PawnMove(char pawnSymbol, int dirX, int dirY, char[,] matrix)

* dirX renamed to columnDirection
* dirY renamed to rowDirection
* char[,] matrica is removed from input parameter and replaced with GameBoard as moving a Figure in that way has no sense if other gameBoard is given
* Multiple Ifs combined into one with OR clause
* Extract GameBoard columns and rows - int gameBoardRows = GameBoard.GetLength(0);
  + - int gameBoardColumns = GameBoard.GetLength(1);
* Extract max column index and max row index:
  + int gameBoardMaxColumnIndex = gameBoardColumns - 1;
  + int gameBoardMaxRowIndex = gameBoardRows - 1;
* Changmethod type- bool
  + - bool KingMove(int columnDirection, int rowDirection)

1. Extract method FigureMove
2. Introduce Figures list that holds all figures in the game
   * Remove figure variables
3. Change project structure
   * Project renamed to KingSurvival.Console
   * Added project KingSurvival.Core
   * KingSurvivalGame class moved to KingSurvival.Core
   * Created new Class KingSurvivalGameConsole that inherits derives from KingSurvivalGame
     1. KingSurvivalGame console functionality is moved to KingSurvivalGame console
4. Enum FigureType defined
   * Added property Type FigureType
   * Type is initialized in Figure constructor
5. Refactoring KingSurvivalGame class
   * Introduced list of valid king commands
   * Introduced list of valid pawn commands
6. Refactoring Start method
   * Introduced if statement that checks what turn is now and runs king turn commands or pawns turn commands
   * Extracted KingTurn method that performs king turns operations
   * Extracted PawnsTurn method that performs pawns turns operations
   * Extracted methods printing game state
     1. KingLoses
     2. KingWinds