**Labyrinth game documentation:**

1. Extracted class ScoreBoard in different file and renamed to PlayerScore.

2. Introduced properties Name and Moves for fields name and moves in the PlayerScore class.

3. Introduced ScoreBoard class, which will show (introduce) the top scores to the player.

4. Introduced Maze class, which will hold the state of the labyrinth at any time.

5. Introduced IMaze interface, which the methods to be implemented by Labyrinth class.

6. Introduced abstract class Cell (with ICell interface with all essential methods) and his inheritors MazeCell and Player (IPlayer interface with all essential methods) classes. Lab cell holds the state of any cell and the Player class holds the state of the Player. It can be moved with the Move method taking as a parameter the CommandExecutioner class implemented by its interface.

7. Introduced CommandListener class and ICommander interface whicl will manipulate the player’s input.

8. Removed method HasSolution from the ILabyrinth interface and made HasSolution a private method in the Labyrinth class.

9. Introduced GameEngine class with Start method to be implemented.

10. Renamed kursov-proekt (Program) class and project name to LabyrinthGame.

11. Moved AddScore method from the main class to ScoreBoard class, making it from static to instance.

12. Extracted sorting of the score list into different private SortScore method in ScoreBoard class

13. Replaced List<ScoreBoard> scores with ScoreBoard scores and replaced ShowScoreBoard method with scores.ShowScore() in Labyrinth3Game class.

14. Introduced PlayerScore currentPlayerScore variable.

15. Removed from currentMoves global variable in Labyrinth3Game class and replaced it with currentPlayer.Moves.

16. In AddScore method in the ScoreBoard class the player score name input is kept directly in the name of the current instance of the player.

17. Introduced MAX\_SCORELIST\_SIZE constant in the ScoreBoard class.

18. Renamed variable in ShowScore method in the ScoreBoard class from i to playerPosition.

19. Moved the remaining game logic to the GameEngine class (content of main placed in the Start method) for being reallocated later.

20. Introduced new constants INITIAL \_POSITION in the Labyrinth class to keep the starting position of the player.

21. Moved SolutionChecker method content to the private method HasSolution method in the Labyrinth class.

22. LabyrinthGenerator method move to GenerateMaze method in MazeClass

23. Maze initialized as matrix of type Lab instead of string.

24. GameEngine.Start method cleared from maze methods. Created Player player and Maze labyrinth variables.

25. Classes reallocated to namespaces.

26. Removed PlayerScore instance and assigned property Score of type PlayerScore to Player class.

27. Extracted method PrintHighScores to print the scores if the player is out of the maze

28. Removed reapeated code in TypeCommand method.

29. Introduced enumeration Direction with all directions for the player to move (Up,Down,Left and Right).

30. Introduced message constants:

31. Introduced static class renderer, responsible for rendering (drawing) on the console.

32. Replaced all magic strings and numbers and defined as constants in the corresponding class.

33. Renamed variable s in AddScore method in the ScoreBoard class to currentPlayer and renamed variables s1 and s2 in SortScores method in the ScoreBoard class respectively to currentPlayer and otherPlayer.

34. Removed all the rendering (printing) in the Renderer class. All renderable objects implement IRenderable interface and the Render method. The game object should not now how they will be rendered for better coupling as well as the renderer does not care what will render. The game objects can be rendered with other implementation easily and objects can be added for rendering through the IRenderable interface.

35. Replaced Move method to Player class with the next cell checker altogether.

36. All renderable objects’ render methods receive particular IRenderer implementation as argument. It can be easily replaced with other implementation of the IRenderer. Bridge design pattern implementation.

37. Rendering the Maze renders each cell of it. Maze and the cells have tree-like structure. Same with rendering the score list (each score item is rendered). Composite pattern implementation.

38. Maze solution checker method refactored to recursive solution for easier comprehension.