# Refactoring Documentation for Project “Labyrinth”

1. Redesigned the project structure.

* Renamed the project to **Labyrinth.**
* Renamed the main class to **LabyrinthGame**
* Extracted each class in separate file and renamed with correct names. Where necessary, new functionality is added.
* All unnecessary methods are removed or merged.
* All methods that did not belong to their classes are moved in classes that they belong to.
* New classes added.

1. Reformatted the source code.

* All methods with incorrect names are renamed according to the best practices in C# language.
* Class Position made to struct. All methods that were in this class are removed and put in classes that they should belong to.
* Class Message made static.
* The methods in all of the classes are renamed with correct names using **PascalCase.**
* All unnecessary new lines in methods are removed. New lines are left according to the best practices in C# language.
* Variables in methods are renamed with correct names using **camelCase.**
* Put { and } after all conditional statements and loops when missing.

1. Introduced new class **Player** and all related functionality moved there.
2. Introduced new class **Engine** and all related functionality moved there.
3. Introduced new class **ObjectRenderer** and all related functionality moved there.
4. Introduced new class **Configuration** which contains constants **FileName** and **GameFieldSize**.
5. Renamed methods:

* In class Message:
* intro() -> PrintIntro();
* nl() -> PrintNewLine();
* move() -> PrintDirectionMessage();
* invalid() -> PrintInvalidMoveMessage();
* win() -> PrintWinningMessage();
* playing() -> PrintInstructionsMessage();
* In class Player:
* isWinning() -> HasWon();
* In class Playfield:
* reset() -> CreateLabyrinth();
* isBlankMove() -> IsVisitedPosition();
* In class ObjectRenderer:
* print() -> Render();
* In class Scoreboard:
* pokazvane() -> Show();
* create() -> CreateFile();
* add() -> Add();

1. Introduced constants:

* PlayfieldRows = 7
* PlayfieldCols = 7
* PlayerRow = 3
* PlayerCol = 3
* FileName = “scoreboard.txt”
* GameFieldSize = 6

1. Unit Test

* Added class PlayerTest
* Testing constructor of Player
* Testing method GetPosition()
* Testing method Move();
* Testing method Points;
* Testing method HasWon()
* Testing method ReastartDeafultPosition
* Added class PositionTest
* Testing constructor of Position
* Added class PlayfieldTest
* Testing constructor
* Testing private method IsVisitedPosition

1. Scoreboard

- Make Show() and Add() methods to read and save a scores into a file.

- Move file reading and creating in separated methods.