|  |  |  |  |
| --- | --- | --- | --- |
| Refactoring Documentation for Project “Labyrinth-7” Team “Hydrogen”   1. Redesigned the project structure:    * Extracted the main class **Program**.    * Renamed the main class **Program** to **LabyrinthDemo.**    * Extracted each class in a separate file with a good name: **ScoreBoard.cs**, **Score.cs**, **Cell.cs**, **Message.cs**, **PlayerPosition.cs** etc. 2. Reformated the source code:    * Removed all unneeded empty lines.    * Inserted empty lines between methods.    * Renamed all variables that needed **flag4** to **IsGameWon**, **flag1** to **IsWonWithEscape**    * Split the lines containing several statements into several simple lines  |  |  |  | | --- | --- | --- | | if (scores.Count == 0)) { Console.WriteLine("The scoreboard is empty! "); } | **🡪** | if(scores.Count == 0))  {  Console.WriteLine("The scoreboard is empty! ");  } |  * + Formatted the curly braces **{** and **}** according to the best practices for the C# language.   + Character casing: variables and fields made **camelCase**; types and methods made **PascalCase**.   + Formatted all other elements of the source code according to the best practices introduced in the course “[High-Quality Programming Code](http://codecourse.telerik.com/)”.  1. Changes during work:    * Changed name of class **Table** to **Score**.    * Implemented **CompareTo** in **Score** class for easier **Sorting**.    * Implemented class **ScoreBoard** and moved all the functionality that belongs to it.    * Override **ToString()** method in **ScoreBoard** class.    * Changed **Getter** and **Setter** in every property on one line.    * Removed unnecessary printing of a new line in **Run** method, in the **top** case.    * Removed **Console.WriteLine(IsRunning)** from **Run** method **d** case, no need of it.    * changed the magic number **6** -> **LabyrinthSize - 1** in **Run** method in **Labyrinth** class.    * Removing the magic number **7** and switching it with the **number of columns** in the **labyrinth**.    * renamed **isAbleToMove** -> **isBlocked**.    * Removed **Environment.Exit(0);** in **Run()** method and made it exit in another way.    * Removed method **Print(labyrinth)** from Labyrinth and implemented **ToString()** instead.    * Removed method **FindNewCellCoordinates**(), no need of it.    * Moved **isAbleToMove = false**; **isFinished = true**; outside, no need to be inside the loop. **SolutionChecker**() methd in the Labyrinth class.    * Removed **SolutionChecker()** method from **Labyrinth** class. Added **ExitPathAvailable()** method which fixes the **bug** of never finding a way out.    * Extracted all directions manipulation from **Run()** method in separate methods for each direction.    * Changed the return type of the method **Generate()** for easier testing.    * Removed **isGameFinished** references from methods and use the field property **IsRunning** insted.    * Added a **field** in the **Labyrinth** class for better **random** number.    * Removed **List<Score> Scores** from Game class.    * Put {} in the **Run** method in every case for the **switch** for better reading.    * Changed the **access modifiers** to **private** where not necessary to be public.    * Inserted **this** where necessary for more clear reading.    * Changed Null exceptions from **ArgumentException** to **ArgumentNullException** to be more accurate.    * Added **Position\_NullValue() test**. 2. Extracted the method **StartGame()** from the method **Main()**. 3. Introduced class **ScoreBoard** and moved all related functionality in it. 4. Introduced class **Score** and moved all related functionality in it. 5. Introduced class **Cell** and moved all related functionality in it. 6. Introduced class **Message** and moved all related functionality in it. 7. Introduced class **Player Position** and moved all related functionality in it. |