Refactoring Documentation for Project “Hangman”

1. Redesigned the project structure:

* Renamed the project to **HangmanGame**
* Renamed the main class **besenka** to **HangmanMain**
* Extracted/Created the following classes in separate files:
  + **ConsoleRenderer.cs**
  + **LetterHandler.cs**
  + **RandomWordGenerator.cs**
  + **ScoreManager.cs**
  + **CommandParser.cs**
  + **Game.cs**
  + **Player.cs**
* Created enumeration **LetterStatus.cs**

1. Reformatted the source code:

* removed unnecessary comments, e.g. “hahaha, izpih edno kilo rakiya vcera i poznavam veche vsichki dumi4ki ot pyrvi puyt, muahahahahahahaaaaaaaa”
* 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. Renamed methods:

* In class **???**: **???** 🡪 **???**.

1. Renamed variables:
   * In class **RandomWordGenerator: rand**🡪 **randomizer**
   * In class **RandomWordGenerator: r**🡪 **wordIndex**
   * In class **RandomWordGenerator: arr**🡪 **words**
2. Introduced constant **MaxNumberOFPlayers**
3. Extracted methods **ParseCommand()** and **ExecuteCommand()** from the **Main()** methodand moved them to the **CommandManager** class.
4. Removed variable **m**, responsible for the health state of the player, and replaced it with variable **mistakes** in **Game** class, since the implementation requires this functionality.
5. In the original code user-friendly messages were missing and also some required in the project documentation. So a class **ConsoleRenderer** was created. It is responsible for all messages printed on the console, i.e. **PrintWelcomeMessage()**, **PrintExitMessage().**
6. Added method **PrintRepeatingLetterMessage(char guessedLetter) -** to print a message if the entered letter has already been used. This functionality is not considered in the project documentation**.**
7. Unit testing:

* Class **ConsoleRenderer** contains 15 public methods . Created 17 test methods. Achieved 100% code coverage. Used **StringWriter** to record the console output and to compare it with the expected output.
* Class **RandomWordGenerator** contains 3 public methods . Created 1 test method. Achieved 100% code coverage.
* Class **ScoreManager** contains 4 public methods . Created 5 test methods. Achieved 100% code coverage.
* Class **Game** contains 10 methods. Created 6 test methods. Achieved 34% code coverage.
* Class **Score Manager** has 6 methods . Created 6 test methods. Achieved 100% code coverage.
* Class **LetterHandler** has 5 mtethods. Created 10 test methods. Achieved 63% code coverage.
* Class **CommandParser** has 1 method. Created 3 test methods. Achieved 63% code coverage.