# Refactoring Documentation for Project “Hangman-2”

***Team “Technetium”***

1. Redesigned the project structure:

Renamed the project to form KPK.Proekt1 to Hangman

Renamed the main class Program to MainClass.

Extracted class Player from CommandExecutor class in a separate class: Player.

Extract method GetPlayerName from the method GetHighScoreEntry from the WordInitialzator class.

Renamed the project to form WordInitializator to GameLogic

Renamed the project to form EndOfTheGameInitializatio to ShowResults

Renamed the project to form BegginingOfTheGameInitialization to PlayRound

Renamed the project to form InitializationAfterTheGuess to HandleUserGuess

Renamed the method Restart in CommandExecuter to Start

Rename method RevealTheNextLetter to RevealTheNextLetterByHelp

Rename method Top in CommandExecitor to TopResults

Remove method RevealTheNextLetter from CommandExecuter and move to GuessHandler

Rename class WordGuesser to UserInputHandler

1. Reformatted the source code:
   * Removed all unneeded empty lines, e.g. in the class CommandExecutor
   * Inserted empty lines between the methods.
   * Split the lines containing several statements into several simple lines, e.g.:

|  |  |  |
| --- | --- | --- |
| for (int i = 0; i < 5; i++)  if(scoreboard[i] != null)  Console.WriteLine("{0}. {1} ---> {2}", i+1, scoreboard[i].PlayerName, scoreboard[i].NumberOfMistakes); | 🡪 | for (int i = 0; i < 5; i++)  {  if (Scoreboard[i] != null)  {  Console.WriteLine("{0}.{1} ---> {2}",  i + 1, Scoreboard[i].PlayerName, Scoreboard[i].NumberOfMistakes);  }  }  Console.WriteLine(); |

* + Formatted the curly braces { and } according to the best practices for the C# language.
  + Put { and } after all conditionals and loops (when missing).
  + 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 variables:

In class CommandExecutor: scoreboard 🡪 Scoreboard due to it is public access.

In GameLogic():allGuessedLettersOrderedByPositionInTheWord 🡪 revealedChars

In GameLogic():positionOfTheFirstFreePositionInTheScoereboard 🡪 firstFreePosition

In GameLogic():num2 🡪 MistakesCounter

In GameLogic():flag 🡪 PlayerHasUsedHelp

In GameLogic():num1 🡪 GuessedCharsCounter

In CommandExecuter() : wg 🡪 UserInputHandler

In PlayersScore() : HIGH\_SCORES\_NUMBER🡪 HighScoresNumber

In PlayersScore() : TopResults 🡪 PrintTopResults

In UserInputHandler() : IsExited 🡪 HasExited

1. Introduced constants:

HIGH-SCORE-NUMBER=5;

SpecialCharacter = '&'

readonly words = {

"computer", "programmer", "software", "debugger", "compiler",

"developer", "algorithm", "array", "method", "variable"

};

1. Extracted the methods: (example GenerateRandomGame() from the method Main().)

Extract method InputUserName from EndOfTheGameInitialization.

Extract method GetHighScoreEntry from EndOfTheGameInitialization.

Extract method GetFirstUnrevealedLetter from method RevealTheNextLetterByHelp

Extract method ReadPlayerInput from UserInputHandler

1. Introduced classes: (example ScoreBoard and moved all related functionality in it).

Create new class PlayersScore;

Move there method ScoreSort from WordInitializator class;

Move Scoreboard field from Command Executor class;

Move public static void TopResults() from CommandExecutor;

Create new class GuessHandler from GameLogic class in.

Create method Help in CommandExecutor

1. Moved method to separate class (method GenerateRandomNumber(int start, int end) to RandomUtils.)
2. Encapsulate fields in all classes;

In class GameLogic encapsulate fields are :

playerHasUsedHelp

revealedChars

guessedCharsCounter

mistakesCounter

In class Player encapsulate fields are :

playerName

numberOfMistakes

In class PlayersScore encapsulate fields is :

scoreboard

In class UserInputHandler encapsulate fields are :

word

hasExited