**Refactoring Documentation for Project “Game 15”**

1. **Redesigned the project structure: Team “Game-Fifteen-4”**

* Renamed the project to **GameFifteen**.
* Renamed the main class Program to **GameFifteenEngine**.
* Extracted each class in a separate file with a good name:
* GameFifteenEngine.cs
* Board.cs
* Game.cs
* State.cs
* PersonalScore.cs
* TopScore.cs
* ConsoleWriter.cs

1. **Reformatted the source code:**

* Removed all unneeded empty lines, e.g. in the struct DvoikaImeRezultat, refactored and renamed to **PersonalScore**.
* Inserted empty lines between the methods.
* Reformated statings taking several lines to a single line, e.g.

bool isEmptyCellInPlace =

emptyCellRow == MatrixSizeRows - 1 &&

emptyCellColumn == MatrixSizeColumns - 1;

bool isEmptyCellInPlace = (this.EmptyCellRow == MATRIX\_SIZE\_ROWS - 1) && (this.EmptyCellColumn == MATRIX\_SIZE\_COLUMNS - 1);

* 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”.

1. **Renamed variables and methods:**

* Method “**proverka”**, initially in class Program, renamed to “**CheckIfCellIsValid”**, class Board.
* Method “**proverka2”**, initially in class Program, renamed to “**IsMatrixOrdered”**, class Board.
* Struct “**DvoikaImeRezultat”**: renamed to “**PersonalScore”**.
* Initially in struct DvoikaImeRezultat: “**name**” renamed to “**userName**”, struct PersonalScore**.**
* Initially in struct DvoikaImeRezultat: “**score**” renamed to “**userScore**”, struct PersonalScore**.**

1. **Introduced constants:**

* **In class Board:**
  + MATRIX\_SIZE\_ROWS = 4
  + MATRIX\_SIZE\_COLUMNS = 4
  + MATRIX\_SIZE = MATRIX\_SIZE\_ROWS \* MATRIX\_SIZE\_COLUMNS
  + EMPTY\_CELL\_VALUE = " "
* **In class TopScores:**
  + TOP\_SCORES\_AMOUNT = 5
  + TOP\_SCORES\_FILENAME = "Top.txt"
  + TOP\_SCORES\_PERSON\_PATTERN = @"^\d+\. (.+) --> (\d+) moves?$"

1. **Introduced** **ICommand interface.**
2. **Introduced new classes.**

* Add class ConsoleWriter to manage all messages.
* Add class PersonalScore to manage functionality related to an individual score.
* Add class TopScore to manage the top scores.
* Add class Game to manage functionality related to instantiating the current game.
* Add class Board to manage functionality related to the game board.
* Add class State to define information needed to save game state.
* Refactored game logic to call the methods with their corresponding classes.

1. **Changed access modifiers.**

* In class TopScore: change access modifier of UpgradeTopScore from private to public.
* In class ConsoleWriter: change the methods from private to internal.
* In class Score: change constants from private to public.
* In class Board: change access modifier of the CheckIfCellIsValid method to public.

1. **Introduced unit tests for classes Board, Score and PersonalScore.**
2. **Implemented data validation in PersonalScore’s properties.**
3. **Added design patterns.**

* Class Game: Singleton design pattern and Observer design pattern.
* Class GameController: Command design pattern, introducing methods for each command.
  + Make next move- Move.cs
  + Exit game- Exit.cs
  + Save game state- Save.cs
  + Restore game state- Restore.cs
  + Restart game– Restart.cs
  + Classes Save and Restore: Memento design pattern to save the current state in a Memento object and replace the current board with the saved one, when needed.

1. **Solved StyleCop warnings by reordering fields and methods.**