# **Sample Refactoring Documentation for Project “Minesweeper-4”**

1. Redesigned the project structure: Team “Minesweeper-4”

* Extracted class in a separate file: **ScoreRecord.cs**.
* Renamed namespace from **mini4ki** to **Minesweeper**.
* Created classes with appropriate names to make the code testable: **ScoreBoard.cs**, **GameField.cs** and **Posision.cs**.
* Renamed main class from **MinesweeperMain** to **Minesweeper**.

1. Reformatted the source code:

* Removed unnecessary using statements.
* Removed all unneeded empty lines.
* Inserted empty lines between the methods.
* 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. Extracted class **ScoreRecord** from the **Minesweeper** class and moved all related functionality to it.

* Specified its fields as private.
* Removed empty constructor.
* Rearranged the code to match the best practices introduced in the course e.g. first fields, than constructors, than properties, than methods, etc.
* Added “**this.**” when accessing the fields from the properties.
* Made all access to the fields through the properties.
* Introduced **ToString()** method.

1. Introduced class **ScoreBoard** and moved all related functionality in it.
2. Renamed variables:

* In class **Minesweeper: counter -> score**.
* In class **Minesweeper: bombed -> isGameOver**.
* In class **Minesweeper: flag -> isGameWon**.
* In class **Minesweeper: welcomeFlag -> isNewGame**.
* In class **Minesweeper: rowIndex -> row**.
* In class **Minesweeper: columnIndex -> col**.
* In class **Minesweeper: selectedCommand -> command**.
* In class **Minesweeper:** **MaxRevealedCells -> ScoreToWin**.
* In class **ScoreRecord**: **personName ->** **playerName**.
* In class **ScoreRecord**: **scorePoints ->** **playerScore**.
* In class **ScoreBoard**: **champions ->** **highScores**.

1. Introduced constants in the **GameField** class:

* **FieldRows** = 5.
* **FieldColumns** = 10.
* **BombsCount** = 15.

1. Moved **PrintScoreBoard(…)**’s logic from the main class to a **ToString()** override in the **ScoreBoard** class.
2. Moved **PrintBoard(…)**’s logic from the main class to a **ToString()** override in the **GameField** class.
3. Moved **CreateWhiteBoard()** and **CreateBombBoard()**’s logic from the main class to a **GenerateGameField()** method in the **GameField** class.
4. Moved **CalculateBombBoard(…)**’s logic from the main class to a **RevealField()** method in the **GameField** class.
5. Moved **CalculateHowManyBombs (…)**’s logic from the main class to a **RevealPosision(…)** method in the **GameField** class.
6. Removed **MakeAMove(…)** method from the main class and reassigned its job to **RevealPosision(…)** and **RevealField()**.
7. Extracted all game logic from the **Main()** method to a new class **Engine**. Encapsulated the data and made the new class a **Singleton**.
8. Made the fields **gameField** and **scoreBoard** static. Implemented lazy instantiation.
9. !!! EXAMPLE !!! Extracted the method **GenerateRandomGame()** from the method **Main()**.