|  |
| --- |
| Refactoring Documentation for Project “Battle-Field-4” Team “Indium”   1. Redesigned the project structure:    * Renamed the project to **Mines**.    * Renamed the main class **Program** to **Battlefield**.    * Extracted each method in a separate class with a good name    * **PrintField**->**PringGameBoard**    * **GrymOtQsnoNebe**->**MinesExplosion**    * **TimeToPlay**->**GameInput**    * **InitiateGame**->**GameBoardGenerator** (moved only creating the game field part)    * **InitiateGame** (second part from Chinese symbol)->**GameEngine**    * Moved the 5 types of arrays(explosions) with the switch statement in **Initiategame** method to **TypesOfExplosionChoice**    * All **classes** except **Battlefield** are static 2. Reformatted the source code:    * Removed all unneeded empty lines    * Inserted empty lines between the methods.    * Split the lines containing several statements into several simple lines    * 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/)”. 3. Class changes **GameBoardGenerator**    * Created property **GameField** that can only be set by the method but can be seen from everyone and throws exception if field is empty    * Created property **MineNumber** that can only be seen from others but changed from the **GameBoardGenerator** class only    * Created the one public method only **GetBoardSize()** that asks for user input.Untill a correct number is selected(1<=number<=10) it will keep asking.Then the private method **CreateGameBoard(int boardSize)** is called with parameter the number that the user entered    * Extracted the random number generator in different method **RandomGenerator(int startRange, int endRange)** //originality++    * Changed the formula of the MinesNumber creation,so it will start from 1 and there will always be mines    * **CreateGameBoard(int boardSize)** calls the **RandomGenerator(int startRange, int endRange)** with the parameters that are 2 formulas that can be seen((15\*n\*n/100)+1) for start and the same for end but with 30 instead of 15.    * The field is generated by placing number of mines randomly on the field.We use the **RandomGenerator(int startRange, int endRange)** to find random positions 4. Class changes **PrintGameField**    * One public method only **PrintField(int[,] matrix)** that takes the **matrix** parameter and calls the private method **GenerateField(int[,] matrix, int matrixSize)** and prints it on the console in the game format(with numbers for rows and columns and X and – for blown or empty fields)    * The private method **GetCharacter(int[,] matrix, int row, int col)** decides what type of character to print if the field is blown    * We use **Stringbuilder** for the storage of the field until printed 5. Class changes **TypesOfExplosionChoice**    * One public method only **GetExplosion(int[,] matrix, int row, int col)** that takes the element in the **matrix[row,col]** and returns two-dimensional array representing the explosion depending of the number in the **matrix[row,col].**We check that with switch statement 6. Class changes **MinesExplosion**    * One public method only **CheckForExplosion(int[,] gameField, int xCoordinate, int yCoordinate).**The method than calls **TypesOfExplosionChoice. GetExplosion** with the parameters that are provided and returns 2-dimensional array representing the explosion.Than it uses two nested loops to check the elements arround the bomb for other bombs.If found a counter is incremented.Finally it returns the counter. 7. Class changes **GameInput**    * Created property **RowCoordinate** that can only be set by the method but can be seen from everyone to remove some coupling.    * Created property **ColCoordinate** that can only be seen from others but changed from the class only to remove some coupling.    * Created the one public method only **ManageUserInput(int[,] gameField)** that asks for user input.Untill a correct coordinates are[Example:2 5] it will keep asking.We use while and bool **isSelectingNextCoordinates**  that is changed to false when we like the coordinates. 8. Class changes **GameEngine**    * Created the one public method only **InitiateGame().**It controls the behaviour of the game at its start.First it’s the **GameBoardGenerator.GetBoardSize()** call to get the Game field.It prints on the console using **PrintGameBoard.PrintField(GameBoardGenerator.GameField).**Finally it calls the private method    * The **CheckForVictory(int totalMinesNumber, int[,] gameField, int fieldSize)** method will ask for coordinates from **GameInput** and substract the number of blown mines during that turn from the whole **GameBoardGenerator.MinesNumber.**Each turn increments a counter and shows the number of blown mines this turn**.**When it reaches zero,the game is won and the number of turns needed to win is shown. 9. Class changes **Battlefield**    * Main method calls **GameEngine.InitateGame()** 10. **Variables**     * Added better names.Almost in every class the game field is called gameField and not arr     * We tried to aim for self documenting code |