# Refactoring Documentation for Project “Balloon-Pop-5”

Team”Lutetium”

Refactored the code and made it easier to read and go around.

Renamed the project to BallonsPop5Game  
**Renamed the given classes and created new classes:**

1. BalloonsMain.cs
2. Command.cs
3. ConsoleCommunicator.cs
4. ConsoleCommunicator.cs
5. CurrentAction.cs
6. Engine.cs
7. ICommandParse.cs
8. IPlayField.cs
9. IUICommunicator.cs
10. Player.cs
11. PlayField.cs
12. ScoreBoard.cs
13. Used to start the game and create the field that we will be playing.
14. The commands that the engine can do
15. Checks the player inputted command from the console and sends the command to the engine to tell it what to do. Checks if the command entered is a coordinate or a command to show top etc.
16. Used to call all things that require something to be printed to the console (field, winner board, instructions, player names etc.)
17. The current action of the engine that it is in.
18. Used to coordinate all the commands and actions and send it across the code to do its purpose. Checks if the game is still running or if the player has cleared the field.
19. Interface determine operations over a user command
20. Represents the interface for the game field
21. Interface determine methods that should be implemented by a class that makes communication between the game and user
22. Class that holds the information about the player – his name and moves. Also can compare two players.
23. The field that we use that is generated using the engine. The class has the possibility to check if the player command that matches the required needs for coordinates in the field, has coordinates of a cell inside the field and “pop” the balloon in that cell.
24. Class that holds the scoreboard that is a list of players. The scoreboard can have more than 5 players but it will only show the top 5. It sorts the players and places them in ascending order (lower is better). Has the possibility to check if a player’ score is good enough to get to the top 5.

**Created unit tests that check the classes. (Code coverage 96.27%)**

1. EngineUnitTest.cs
2. PlayerUnitTest.cs
3. PlayFieldUnitTest.cs
4. ScoreboardUnitTest.cs
5. TestCommandParser.cs
6. Tests the engine and possible scenarios of the game and commands. (restart, exit etc.)
7. Tests for the players, creating adding and editing them.
8. Tests for creating, accessing, “poping” a balloon at a cell and other.
9. Creating and managing a scoreboard. Adding players sorting.
10. Tests the possible commands that the player can give from the console.

Overall we have made the game more OOP oriented and easier to use. Refactored the code so it is easy to navigate in it. We have added a documentation to each method and class so it is easier to understand what they do. We have made the console clear itself each time the field is changed so the information on the console does not pile up.

More information about our work in our github.com repository.