**Game-Fifteen-5**

**Refactoring Documentation**

# Redesigned project structure

* Created **Game.Common** project
* Created **Game.Core** project
* Created **Game.UI** project
* Extracted all classes in separate files in the appropriate project
* Classes arranged in separate folders in the appropriate project
* All constants are separate in different classes
* Interfaces arranged in separate folders
* Unit test project added to the solution

# Reformatted source code

* Removed all unneeded empty lines
* Separate methods with an empty line
* Empty line added after each closing curly bracket to separate logic
* Split lines containing long statements
* Long if conditions split into separate bool values in order to debug easily
* 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 made camelCase 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://telerikacademy.com/Courses/Courses/Details/174)”.

# Renamed variables and identifiers

* Variables renamed appropriate to their use
* Methods renamed appropriate to their use
* Classes renamed appropriate to their use

# Constants

* Every magic number or string from is put at region Constants in their classes
* All fields that are not changed in properties are made read-only

# Class refactoring

* Each class is glued to the Single responsibility principle.
* Abstract class Command introduced as parent of all commands
* Access modifiers introduced to all classes

# Interfaces introduced

* Every object is used by its interface.

# Methods refactoring

* Single responsibility principle
* Long methods shortened to e screen scroll
* Method logic not appropriate to the method name extracted into separate method
* Access modifiers introduced to all methods

# Design patterns introduced

### Creational:

* Singleton
* Prototype

### Structural

* Flyweight
* Decorator
* Bridge

### Behavior

* Command
* Strategy
* Iterator
* Observer
* Template Method
* Dependency Injection

# Other features

* SOLID, DRY, YAGNI principles