# Refactoring Documentation for Project “ Balloons Pops”

1. Redesigned the project structure:

* Renamed the project and the solution to BalloonsPopsGame
* Renamed the namespace to BalloonsPopsGame
* Removed useless empty destructors
* Renamed GameState class to Game
* Renamed ballonsState class to BalloonsContainer
* Renamed the Program class to BalloonsPopsGameMain.
* Added new BalloonsPopsGame.Balloons namespace where is all the creational, storage and popping logic for the balloons is stored.
* Added new BalloonsPopsGame.Score namespace for IScoreboard and Scoreboard.
* Added new namespace for BalloonsPopsGame.UserInterface for all UI related classes.
* Added new namespace BalloonsPopsGame.Exceptions for all application exceptions.
* Added new namespace BalloonsPopsGame.RandomProvider for all random generation logic.

1. Reformatted the source code:

* 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).
* Inserted the usings into the namespace
* Removed unused usings.
* Character casing: variables and fields made camelCase; types and methods made PascalCase.

1. Renamed variables:

* In class BalloonsContainer: poleto -> ballons.
* In class BalloonsContainer:cnt -> numberOfTurn.
* In class Game: \_st -> ballons

1. Renamed methods:

* In class BalloonsContainer: kraj -> IsEmpty.
* In class BalloonsContainer: printArray -> DispayBalloons.
* In class BalloonsContainer: pr -> ConvertBallonToChar.
* In class BalloonsContainer: PopBallon -> PopBallons.
* In class Game: sendCommand -> PopBallons.

1. Introduced constants:

* In class BalloonsContainer: NumberOfRows.
* In class BalloonsContainer: NumberOfColumns.
* In class BalloonsContainer: NumberOfBalloonColors.

1. Introduced Properties:

* In class BalloonsContainer: NumberOfTurn.
* In class BalloonsContainer: Balloons
* In class Game: Scoreboard.
* In class Game: Balloons.
* In class Game: UIGenerator.

1. Moved the property NumberOfTurn into the Game class.
2. Removed unnecessary comments:
3. Added access modifiers to all methods:
4. Changed methods:

* In class BallonsContainer -> PopBallons: Changed the return type of the method to void. Removed the logic for checking if the container is empty from him. Added logic for checking if the balloon position is valid. Also extracted the logic for finding the popping ranges in separate methods
* In class Game -> PopBalloons: Removed the logic for validating the balloon position. Made the method to update the scoreboard only after the end of the game.
* In class Game -> ExecuteCommand: Fixed the parsing of the input when entering balloon position.
* In class Game -> ExecuteCommand: Added trimming and toLower to the input.

1. Extracted methods:

* In class BalloonsContainer placed the logic for filling the container into method Fill().
* In class BalloonsContainer placed the logic for checking if it is empty in IsEmpty() method.
* In class placed the logic for finding popping balloons ranges in BalloonsContainer FindTopPoppingRange(), FindBottomPoppingRange(), FindLeftPoppingRange() and FindRightPoppingRange() methods.
* In calss BalloonsContainer made two overloads for PopBalloons() method in order to make the class more testable.

1. Made new methods:

* In class Game -> Start: moved the endless loop and command execution logic in it.

1. Introduced new Balloon class:

* Made the class override == , != operators and Equals() and GetHashCode() methods.
* Added a property Type of type BallonType to store information for the balloons.

1. Introduced new class ScoreBoard and moved all related functionality in it.

* Made the delegate for adding a player a private method AddPlayer().
* Change the sorting logic to selection sort.

1. Introduced new interface UIHandler and new class ConsoleUIHandler and moved the logic for creating UI into them:

* Moved the DisplayBalloons method from BalloonsContainer class.
* Created DispayMessage method for writing messages to the user.
* Moved the logic for displaying scoreboard into UIHandler.
* Moved the logic for working with the Console into ConsoleUIHandler.

1. Introduced new struct UIMessages and added all ui messages as constants in it.
2. Introduced new interface IRandomNumberProvider and new class StandardRandomNumberProvider and moved the functionality for creating random numbers into them.
3. Introduced new InvalidRowOrColumnExceptionand MissingBalloonException:

* Made PopBallons method in BallonsContainer to throw these exceptions instead of writing on the console.

1. Made the drawing of the balloons via abstract BalloonDrawingManager class and its’ implementations GreenBalloonDrawingManager, YellowBalloonDrawingManager, BlueBalloonDrawingManager and RedBalloonDrawingManager.
2. Added Poor man’s Inversion of control principle where possible.
3. Implemented the Singleton design pattern:

* Made the StandardRandomNumbersProvider class a singleton one.
* Added Instance property for returning the static instance of StandardRandomNumbersProvider class.

1. Implemented the Prototype design pattern:

* Made Balloon class implement IClonable.
* Balloons are cloned when are passes by the Enumerator. Implemented the Flyweight design pattern:

1. Made balloon storage with Flyweight pattern.

* Create new IBalloonFactory interface, BalloonFactory class and BalloonType enumeration.

1. Implemented the Iterator pattern:

* Made IBalloonsContainer to implement IEnumerable<Balloon>.

1. Implemented the Observer pattern:

* Added ContainerChanged event to the BalloonsContainer and protected void OnContainerChanged method.
* UIHandler subscribes to the ContainerChanged event.

1. Implemented Simple Factory:

* Made BalloonDrawingManagerFactory class for giving concrete BalloonDrawingManager by given BalloonType.

1. Implemented Façade pattern:

* Made the Game class hide it’s logic under the Start and Resrt methods.

1. Added new TestBalloonsPopsGame project for Unit testing:

* Made uint tests for all implementational classes.
* Used mocking with JustMock for easier testing of Game, BalloonsContainer and ConsoleUIHandler classes.

1. Made StyleCop tests pass on all classes.