# High-Quality Code Construction – Team Projects – April 2013

## Overview

You are given a **C# software project** designed to implement some of the following **console-based games**:

* Balloon Pops
* Battle Field
* Bulls and Cows
* Game 15
* Hangman
* King Survival
* Labyrinth
* Minesweeper

The project consists of one or more **source code files** and sometimes contains other files and is provided as ZIP archive. You need to refactor the project in order to **improve its quality** following the best practices learned in the course “[High-Quality Programming Code](http://telerikacademy.com/Courses/Courses/Details/40)” and to **implement unit tests** that ensure that the code has correct behavior.

## Detailed Assignment Description

In order to ensure the high quality of the assigned project you need to fulfill the following tasks:

1. **Perform refactoring of the entire project** (its directory structure, project files, source code, classes, interfaces, methods, properties, fields and other class members and program members and its programming logic) in order to **make the code “high quality”** according to the best practices introduced in the course “[High-Quality Programming Code](http://telerikacademy.com/Courses/Courses/Details/40)”. The obtained **refactored code** should conform to the following characteristics:
   * **Easy to read, understand and maintain** – the code should be well structured; should be easy to read and understand, easy to modify and maintain; should follow the concept of self-documenting code; should use good names for classes, methods, variables, and other identifiers; should be consistently formatted following the best formatting practices; should have strong cohesion at all levels (modules, classes, methods, etc.); should have loose coupling between modules, classes, methods, etc.; should follow the best practices of organizing programming logic at all levels (classes, methods, loops, conditional statements and other statements); should follow the best practices for working with variables, data, expressions, constants, control structures, exceptions, comments, etc.
   * **Correct behavior** – the project should fulfill correctly the requirements and to behave correctly in all possible use cases. This means that all bugs or other problems in the project (e.g. performance or usability issues) should be fixed and any unfinished or missing functionality should be completed. The code should be very well tested with properly designed unit tests.
2. **Design and implement unit tests** covering the entire project functionality. To ensure the project works correctly according to the requirements and behaves correctly in all possible use cases, design and implement unit tests that cover all use cases and the entire program logic. If needed, first redesign the program logic to **make the code testable**. Test the normal expected behavior (correct data) and possible expected failures (incorrect data). Put special attention to the border cases. The code coverage of the unit tests should be at least 80%. Use unit testing framework of your choice (e.g. Visual Studio Team Test, NUnit, MbUnit or other).
3. **Document the refactorings** you have performed in order to improve the quality of the project. Use English or Bulgarian language and follow the sample (see below).

## Deliverables

1. The **original source code** (project files, .cs files) without executables.
2. The **refactored source code** (project files, .cs files) without executables.
3. The **unit tests** – source code (project files, .cs files) without executables.
4. The **refactoring documentation**.

## Team Work Requirements

* Obligatory use **Git** as source code repository and **GitHub** (<http://github.com>) as project hosting and team collaboration environment. SVN or TFS are **not** allowed for this project.
* **Each team member** should have contributions to the project and **commits in the source control repository in 3 different days**. We acknowledge that this requirement seems a bit unnatural, but we want to track **how the team collaborates over the time** and that the **project is developed incrementally**, not in the “last minute”.

## Other Requirements

* Pack the project deliverables in a **single ZIP archive**. Be sure to avoid including large unused files in the archives (e.g. compilation binaries). Your archive should be up to 8 MB. Each team member should submit the same archive as a homework.
* Be prepared as a team to **defend your project** in front of the course lecturers. You should be able to explain what refactorings have been performed and why. The documentation will definitely help you. Be prepared to **demonstrate how the unit tests cover the project’s functionality**. Preferably bring your own laptop to reduce the effort to setup your development environment and project workspace.
* Be prepared to **show the commit logs** from the source control system to demonstrate how the project development efforts are shared between the team members and over the time.

## Discussion Forum

* You can freely discuss the course projects and ask questions in the official discussion forum of the course: <http://forums.academy.telerik.com/high-quality-code>.

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Refactoring Documentation for Project “Game 15” Team “…”   1. Redesigned the project structure:    * Renamed the project to **Game-15**.    * Renamed the main class **Program** to **GameFifteen**.    * Extracted each class in a separate file with a good name: **GameFifteen.cs**, **Board.cs**, **Point.cs**.    * … 2. Reformatted the source code:    * Removed all unneeded empty lines, e.g. in the method **PlayGame()**.    * Inserted empty lines between the methods.    * Split the lines containing several statements into several simple lines, e.g.:  |  |  |  | | --- | --- | --- | | **if (input[i] != ' ') break;** | **🡪** | **if (input[i] != ' ')**  **{**  **break;**  **}** |  * + 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/)”.   + …  1. Renamed variables:    * In class **Fifteen**: **number** 🡪 **numberOfMoves**.    * In **Main(string[] args)**: **g** 🡪 **gameFifteen**.    * … 2. Introduced constants:    * **GAME\_BOARD\_SIZE = 4**    * **SCORE\_BOARD\_SIZE = 5**.    * … 3. Extracted the method **GenerateRandomGame()** from the method **Main()**. 4. Introduced class **ScoreBoard** and moved all related functionality in it. 5. Moved method **GenerateRandomNumber(int start, int end)** to separate class **RandomUtils**. 6. … |

# “Bulls and Cows” Game

Your task is to write an interactive **console-based implementation of the game “Cows and Bulls”** in which the **player tries to guess a secret number**. At the start of the game the computer generates a random secret number of 4 decimal digits (e.g. 7725). At each step the player makes a guess to the computer (e.g. 2375) and the computer answers how many digits are **correct** (**bulls**) and how many digits are present in the secret number but are **misplaced** (**cows**). For example, if the secret number is 7725 and the guess number is 2375, this means 1 bull (the last digit ‘5’) and 2 cows (the digits ‘2’ and ‘7’). Note that a single digit can either represent a bull, or a cow, but not two bulls, two cows or bull and cow in the same time. For example, if the secret number is 2424 and the guess number is 4144, the result is 1 bull and 1 cow. The game ends when the player’s guess is correct. Otherwise the computer prints the number of bulls and cows of the last guess and invites the player to try another guess. When the game is finished, a new game automatically starts.

If the player is bored, he or she can enter the command **'help'** instead of a guess number and as a response the computer should reveal a bull at a random position (different position every time). The player can request starting a new game by entering the command **'restart'**.

Your program should implement a local top scoreboard to keep the best results and the names of their authors. Initially, at the program start, the top scoreboard is empty. It **keeps the top 5 results** sorted in ascending order by the number of guesses. When a game is finished by correct guess of the secret number and without using the **'help'** command, the player’s result can enter in the top scoreboard if his or her number of guesses is less than some of the other achievements staying in the top scoreboard. When the player’s result enters the scoreboard, the player should enter his or her name or nickname.

The player can request printing the top scoreboard during the game by entering the command **'top'**. The player can request stopping the game and exiting from the program the command **'exit'**.

## Example Game Session

At the start of this game session the computer’s secret random number is 7725. The player’s input is shown in *italic*:

|  |
| --- |
| **Welcome to “Bulls and Cows” game. Please try to guess my secret 4-digit number.**  **Use 'top' to view the top scoreboard, 'restart' to start a new game and 'help' to cheat and 'exit' to quit the game.**  **Enter your guess or command: *2375***  **Wrong number! Bulls: 1, Cows: 2**  Assume the computer’s secret number is 7725.  **Enter your guess or command: *8946***  **Wrong number! Bulls: 0, Cows: 0**  **Enter your guess or command: *top***  **Top scoreboard is empty.**  **Enter your guess or command: *1055***  **Wrong number! Bulls: 1, Cows: 0**  **Enter your guess or command: *2253***  **Wrong number! Bulls: 0, Cows: 2**  **Enter your guess or command: *7375***  **Wrong number! Bulls: 2, Cows: 1**  **Enter your guess or command: *2775***  **Wrong number! Bulls: 2, Cows: 2**  **Enter your guess or command: *7725***  **Congratulations! You guessed the secret number in 7 attempts.**  **Please enter your name for the top scoreboard: *Bay Ivan***  **Scoreboard:**  **1. Bay Ivan --> 7 guesses**  **Welcome to “Bulls and Cows” game. Please try to guess my secret 4-digit number.**  **Use 'top' to view the top scoreboard, 'restart' to start a new game and 'help' to cheat and 'exit' to quit the game.**  **Enter your guess or command: *1234***  **Wrong number! Bulls: 1, Cows: 1**  Assume the computer’s secret number is 8130.  **Enter your guess or command: *4567***  **Wrong number! Bulls: 0, Cows: 0**  **Enter your guess or command: *8901***  **Wrong number! Bulls: 1, Cows: 2**  **Enter your guess or command: *top***  **Scoreboard:**  **1. Bay Ivan --> 7 guesses**  **Enter your guess or command: *restart***  **Welcome to “Bulls and Cows” game. Please try to guess my secret 4-digit number.**  **Use 'top' to view the top scoreboard, 'restart' to start a new game and 'help' to cheat and 'exit' to quit the game.**  **Enter your guess or command: *1234***  **Wrong number! Bulls: 0, Cows: 2**  Assume the computer’s secret number is 0347.  **Enter your guess or command: *5678***  **Wrong number! Bulls: 0, Cows: 1**  **Enter your guess or command: *help***  **The number looks like XX4X.**  **Enter your guess or command: *9043***  **Wrong number! Bulls: 1, Cows: 2**  **Enter your guess or command: *help***  **The number looks like 0X4X.**  **Enter your guess or command: *0642***  **Wrong number! Bulls: 2, Cows: 0**  **Enter your guess or command: *0348***  **Wrong number! Bulls: 8, Cows: 0**  **Enter your guess or command: *0347***  **Congratulations! You guessed the secret number in 6 attempts and 2 cheats.**  **You are not allowed to enter the top scoreboard.**  **Scoreboard:**  **1. Bay Ivan --> 7 guesses**  **Welcome to “Bulls and Cows” game. Please try to guess my secret 4-digit number.**  **Use 'top' to view the top scoreboard, 'restart' to start a new game and 'help' to cheat and 'exit' to quit the game.**  **Enter your guess or command: *1234***  **Wrong number! Bulls: 1, Cows: 2**  Assume the computer’s secret number is 2133.  **Enter your guess or command: *5678***  **Wrong number! Bulls: 0, Cows: 0**  **Enter your guess or command: *9012***  **Wrong number! Bulls: 0, Cows: 2**  **Enter your guess or command: *win***  **Incorrect guess or command!**  **Enter your guess or command: *exit***  **Good bye!** |

Some players could try to cheat by entering illegal moves, so be cautious and **prevent illegal input**.