Team Bulls and Cows – 4

*project refactoring documentation*

1. Redesigned the project structure:
   1. Split the main class **bikove\_i\_kravi.cs** into:
      1. **BullsAndCows.cs** – The entry point of the program
      2. **GameEngine.cs** – Here is the main logic of the program
      3. **GameNumberProvider.cs** – A class to provide functionality concerning the numbers used to guess in the game.
      4. **Helper.cs** – A class used to provide help to the player if he opts to cheat.
      5. **InputOutput.cs** – A façade class used for reading data from the player and releasing data to the player.
      6. **MessageDispatcher.cs** – A class used to get messages for the player.
   2. Introduced interfaces:
      1. **ICalculateBullsAndCowsStrategy** – Different types of calculation strategies implement this interface.
      2. **IGameEngine** – Implementations of this interface are used to run the core game logic.
      3. **IMessageDispatcher** – Implementations of this interface are used to produce messages for the player.
      4. **IPrinter** – Implementations of this interface are used to output messages to the player.
      5. **IReader** - Implementations of this interface are used to read input from the player.
   3. Introduced classes:
      1. **Command.cs** – A base class for all the commands.
      2. **CommandCreator.cs** – A factory class to provide new instances of all the command classes.
      3. **CommandExit.cs** – The command to be executed when the player enters ‘exit’, it stops the game and displays the ‘goodbye’ message.
      4. **CommandHelp.cs** – The command to be executed when the player enters ‘help’, it will display a hint of how the actual number looks like.
      5. **CommandOther.cs** - The command to be executed when the player enters an invalid command string, it will display a warning message --> the user.
      6. **CommandRestart.cs** - The command to be executed when the player enters ‘restart’, it will restart the game.
      7. **CommandTop.cs** - The command to be executed when the player enters ‘top’, it will display the leader board.
      8. **ConsolePrinter.cs** – A class used to print messages on the console.
      9. **ConsoleReader.cs** – A class used to retrieve user input from the console.
      10. **FactoryMethod.cs** – A base class for the CommandCreator class. Provides an abstract definition of the Create method.
      11. **GameEngine.cs** – Contains all of the main logic of the game.
      12. **GameNumberProvider.cs** – Provides random four digit numbers for the player to guess. Also provides functionality to check if the player entered a valid guess number.
      13. **Helper.cs** - A class used to provide help to the player if he opts to cheat.
      14. **InputOutput.cs** - A façade class used for reading data from the player and releasing data to the player.
      15. **LeaderBoard.cs** – Used to store information about previously best performing players.
      16. **MessageDispatcher.cs** – Used to return messages for the player.
      17. **NormalCalculateBullsAndCowsStrategy.cs** – A strategy class, providing method for calculating the number of bulls and cows for normal difficulty.
      18. **GuessResult.cs** – Contains information about how many bulls and how many cows are in a given guess.
2. Reformatted the source code:
   1. Removed all unneeded empty lines.
   2. Inserted empty lines between the methods and all blocks, where necessary.
   3. Changed all “” with string.Empty.
   4. Changed all System.String with “string”, System.Int32 with “int” and so on, for better readability.
   5. Character casing:
      1. Variables and fields made camelCase.
      2. Types, methods and constants made PascalCase.
   6. Formatted all other elements of the source code according to the best practices for high quality programming code.
3. Refactored methods:
   1. **CalculateBullsAndCowsCount()** – Changed it from void to a return value method to avoid the bad practice of using **out** parameters. Method now returns and object containing the previously **out** parameters.
   2. **PlayerInputToPlayerCommand()** – Method is now removed and replaced by the **CommandCreator** class.
   3. **generateNumber()** – Method is now moved to GameNumberProvider class.
   4. **Start()** – Refactored it to avoid the deep nesting of do-while loops and if statements.
4. Renamed variables:
   1. **WELCOME\_MESSAGE** --> **WelcomeMessage**
   2. **WRONG\_COMMAND\_MESSAGE** --> **WrongCommandMessage**
   3. **Klasirane** --> **leaderboard**
   4. **maxCountOfStoredDate** --> **DefaultNumberOfItemsInLeaderBoard**
5. Renamed methods:
   1. **generateNumber()** --> **Generate()**
   2. **PokajiHelp()** --> **GetHelp()**
   3. **generateHelpPattern()** --> **GenerateHelpPattern()**
6. Renamed classes:
   1. **klasirane.cs** --> **LeaderBoard.cs**
   2. **bikove\_i\_kravi.cs** --> **BullsAndCows.cs**
7. Introduced constants:
   1. **DefaultNumberLength** in **GameEngine.cs**
   2. **DefaultNumberOfItems** in **LeaderBoard.cs**
   3. **GoodByeMessage** in **MessageDispatcher.cs**
   4. **InvalidCommandMessage** in **MessageDispatcher.cs**
   5. **NoCheatersMessage** in **MessageDispatcher.cs**
   6. **EnterNameMessage** in **MessageDispatcher.cs**
   7. **EnterCommandMessage** in **MessageDispatcher.cs**
   8. **WelcomeMessage** in **MessageDispatcher.cs**
8. Added validations to all properties to ensure correct state of the objects.
9. Implemented design patterns:
   1. Factory Method – **FactoryMethod.cs**
   2. Command pattern – **Command.cs**
   3. Strategy pattern – **IcalculateBullsAndCowsStrategy.cs**
   4. Bridge pattern – From **IGameEngine.cs** to **IMessageDispatcher.cs**
   5. Singleton pattern – Property **Instance** in **GameEngine.cs**
   6. Facade pattern – **InputOutput.cs**