# Refactoring Documentation for Project King-Survival-6

|  |  |  |
| --- | --- | --- |
| Name | Telerik ID | Github ID |
| Ivaylo Georgiev | IvayloGeorgiev | IvayloGeorgiev |
| Stoyan Penchev | stoianpp | stoianpp |
| Liubomir Svilenov | LSvilenov | LSvilenov |
| Hinkah Hineva | hinkah | hinkah |

GitHub repo - <https://github.com/IvayloGeorgiev/King-Survival-6>

1. Initial redesign:
   1. Goal – Clean up the original KingSurvival code by removing repetitions, useless if/else statements, and renaming variables to follow proper quality code standards. Ideally, it should improve the team’s understanding of the code to facilitate further refactoring into classes and patterns.
   2. Execution:
      * Changed method names to follow proper naming conventions – all now employ PascalCase and names such as PokajiDyskata and Proverka2 were changed to a more descriptive DrawBoard and CheckCommand.
      * Removed many pointless repetition loops, most often and reworked if/else statements to avoid initializing common variables two times or more.
      * Split method logic into separate private methods – for example, in the method Movement (previously ProverkaProcess), MovePawn and MoveKing were reassigned to a new private method.
      * Fixed internal spacing errors where necessary by removing multiple line breaks between methods and variables.
      * Moved some hard coded variables to separate constant fields.
   3. Result – Overall Knowledge of the project has been greatly improved. Further discussion will be necessary to determine optimal class refactoring as currently the entire code base is inside a single inheritance hierarchy.
2. Secondary redesign and patterns:
   1. Goal – after the initial redesign, we realized that the project will need to be split into separate Classes and that, as per the task guidelines, patterns will have to be used. Additionally, unit tests will have to be implemented.
   2. Execution:
      * Created a Figure abstract class to represent Pawn/King figures, containing a char symbol, int[2] position and bool CheckCommand and int[] Move(offset[]) methods.
      * Valid figure commands are initialized in the King and Pawn inheritors. They will implement the IMovable interface, allowing a figure to change its position when given an int[2] offset.
      * Added a simple factory pattern to facilitate pawn and king creation.
      * Moved the Move(offset) method to the Figure class as it was the same for both king and pawn.
      * Created a Board singleton class to serve as a drawing interface. DrawBoard method was moved to it. Added methods to display a message and get user input.
      * Added method to draw a collection of figures on the board when given a specific position.
      * Added constants to represent Board restrictions.
      * Added Engine class to serve as a main initialization and start of the game. It will employ the façade pattern and it will serve as a connection between the drawing and gamelogic interfaces.
      * Added abstract class Turn utilizing the State Pattern with two specific implementations, PawnTurn and KingTurn. CurrentTurn property is held by the Engine class.
      * MovePawn and MoveKing from the original implementation were incorporated within the PawnTurn and KingTurn methods. CheckKingExit was moved to the KingTurn class. TurnCounter increment is in the turn class and is invoked only by the kingTurn (it is a protected method).
      * Further methods checking whether a board position is free or a figure is alive were incorporated from the legacy code and reworked to operate with figures.
      * Offset determination moved to protected method within the Turn class to be used by both PawnTurn and KingTurn. They will get their valid commands from the figures related to each implementation.
      * Added GetAllFigures method to return a list with all figures related to both turns for drawing by the board.
      * Added FigureFactory abstract class to create pawn and king figures. The Simple Factory class was deleted.
      * Two implementations were added to FigureFactory for creating pawns and kings respectively.
      * Command possibilities for each figured moved to the specific PawnCreator and KingCreator classes. Commands list for pawns and kings moved as a static readonly array within the respective creator that passes it to on to the figure.
      * Renderer class added. DrawFigure and ShowMessage methods were moved from the Board to the Renderer. Board remains a singleton with a single method, DrawBoard.
      * Added string[] shapes to each figure to represent the shape to be displayed by the renderer class when drawing the figure.
      * Changed Pawn and King Creators to pass a dictionary with key string command and value command offset to the Pawn and King figures instead of a simple string array. Command to offset method removed from the Turn class.
      * The essentially meaningless methods with the old implementation were moved to a separate folder and unloaded from the project.
      * Added ShowError and ShowInfo methods to the Renderer. ShowInfo takes a string[] and prints each string on a new line.
      * Added an adapter class to translate from the IDisplay that use a list of figures to IDrawingDisplay that uses a list of IDrawings.
      * Logic for Figure to shape translation moved to the concrete implementation of the adapter. Shape variables removed from the figures.
      * Added code comments for all classes.
      * Added unit tests focusing on the figure and turn classes. The display class is not tested as it can be more properly observed visually.
   3. Result – Most metrics for the project were covered. Currently, we have 2 Creational Patterns, Singleton and Factory, 2 Structural, Façade and Adapter, and 1 Behavioural, State. Further addition of a Command or Memento patterns were discussed by we were unable to implement them due to time constraints.

Warning! If figures appear warped or have symbols that don’t display properly, you will need to add a console font that supports all symbol fonts. The program was tested internally using the DejaVu Sans Mono font. Information on how to install new console fonts can be found here - <http://www.techrepublic.com/blog/windows-and-office/quick-tip-add-fonts-to-the-command-prompt/>