**Team “Patrick Star”**

Team members:

* Александър Маринов (profile in TILS: AleksanderMarinov)
* Ивайло Кенов (profile in TILS: ivaylo.kenov)
* Иван Пудев (profile in TILS: ivan.pudev)

Project summary:

Our project is a simple (yet almost complete) command driven console chess game named Just Star Chess. The game is played with two players, no AI bots are implemented. Just Star Chess has full and complete user interface with menu.

Available commands in the menu are:

* **start** (creates new game);
* **load** (loads a saved game from a file “savedgame.chess”);
* **help** (shows available commands in the game, not implemented currently);
* **exit** (quits the game).

Once started the game asks for the two players’ names (first player is always white in the game) and the UI is executed.

The UI has the following:

* **the board** (shows the game board with all available figures);
* **player input** (players take turns inputting commands for figure movements);
* **history** (shows a history of previous moves – 10 for each player);
* **taken figures** (shows a history of all taken figures for each player).

Player input has these available in-game commands:

* **figure movement** (entered as “a4-a5”. This command will move the figure from field A4 to field A5 on the board (standard chess notation is used). Another example: “f7-e6” will move the figure from field F7 to field E6. Movement validation is implemented for each figure according to the chess game rules. If invalid command or invalid movement is entered, the program will show error message and ask you again for command.);
* **save** (saves the current game in a file “savedgame.chess”);
* **load** (loads a saved game from a file “savedgame.chess”);
* **exit** (quits the game).

All commands in the game are case-insensitive. For example: **save** equals **SavE** equals **SAVE** and **a5-e7** equals **A5-e7** equals **A5-E7**.

Game logic implements validation for commands, validation for correct movement depending on the used figure (example: pawns and queen move differently), validation for figure collision (depending on whether the two figures are on the same color – does not allow the move and if they are not – the second figure is taken and removed from the board), validation for chess (if a figure makes chess to the other player’s king).

Source code:

Full source available here:

<http://patrickstar.codeplex.com/SourceControl/changeset/view/23909>

*Interfaces:*

* **IBoard** – implements board printing.
* **IColission** – implements collision of two figures.
* **IMovable** – implements figure movement.
* **IUserInput** – implements user input.

*Enumerations:*

* **Names** – contains figures’ names (None, Pawn, Bishop…). None is empty field.
* **Colors** – contains figures’ and players’ colors (White, Black, None). None is for empty field.

*Struct:*

* **CursorCoordinates** – contains information used for printing the board on the console.
* **FigureProperties** – contains information about a particular figure about name and color.

*Exception:*

* **InvalidInputException** – it is thrown when player enters invalid command.

*Static Classes:*

* **ChessChecker** – validates whether the composition on the board has a check.
* **Menu** – implements the menu – its commands and visualization.

*Abstract Class:*

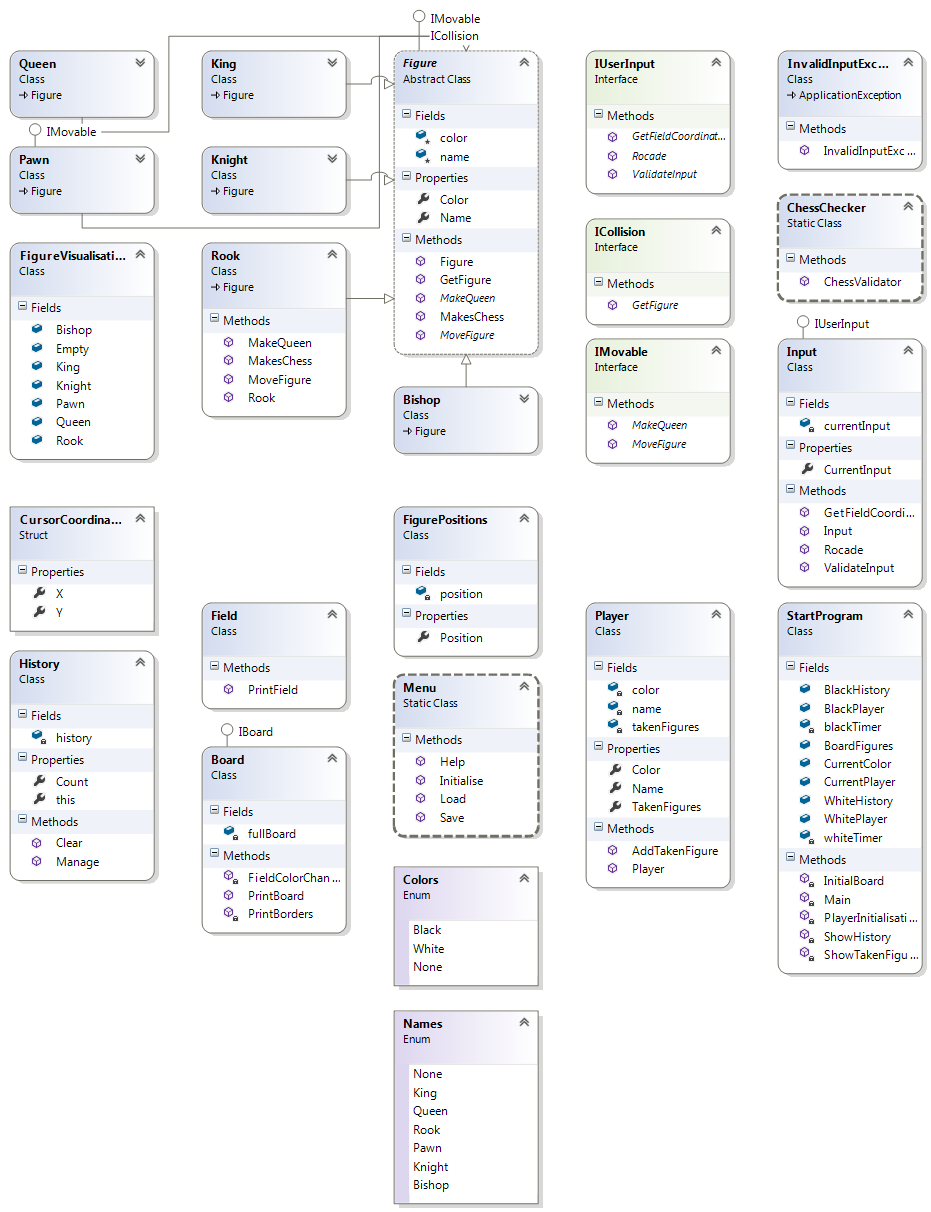
* **Figure** – implements figure properties, abstract method movement and method for taking figures from the board.

*Classes:*

* **Pawn, Rook, Bishop, Knight, King, Queen** – all inherit the abstract class **Figure**. Implements movements for each figure and check to the king.
* **Player** – class for players and taken figures.
* **History** – saves all previous inputs.
* **Board** – inherits **IBoard** interface and prints the board on the console. The full board is a massive 8x8 of **Fields**.
* **Field** – a single field from the board. Can be empty or can contain figure. Prints the field on the screen.
* **FigurePosition** – saves the positions of each figure. Contains a matrix of 8x8 **FigureProperties**.
* **FigureVisualisation** *–* contains char arrays for each figure and how it is shown on the console.
* **Input** – validates the player input and converts it to matrix coordinates.
* **StartProgram** – the game engine. All the magic happens here. Using all other classes implements the game logic.

*Class diagram:*

* Shows all the classes and their inheritance. Available in “Class diagram” folder.

**

Contributions:

Projects history logs are available here:

<http://patrickstar.codeplex.com/SourceControl/list/changesets>

**Александър Маринов (profile in TILS: AleksanderMarinov)**

* Implemented all different figures, their movement, check to the king, the custom exception, enumeration for colors and taken figures methods.

**Ивайло Кенов (profile in TILS: ivaylo.kenov)**

* Implemented the game engine and logic, board and field printing, bug fixing and refactoring, the menu, input and command validation, coordinates converting, history visualization, save and load game from file. Written this documentation.

**Иван Пудев (profile in TILS: ivan.pudev)**

* Implemented visualization of figures and board indicators, all interfaces, player class, enumerations, taken figures logic and visualization.