Architektur

Gruppe 45
Mounir Darwish
Ahmad Mohammad
Seyedbehnam Mirhashemi

May 2021

Contents

1 Basic Idea

The idea behind the architecture is that objects, logic and the user interface are separate from each other.

This software design pattern called Model–view–controller (MVC).
The architecture is explained in detail in the following chapters.

2 Systems component

The components are divided as follows:

- Model
- View
- Controller

in order to isolate the model from the user interface (View), with also a third component (Controller) that coordinates both the models and views.
The 'main' method creates a new console view when specifying the –no-gui parameter, and transfers it to a new console controller.
The Attribute class holds all the attributes that the game needs, for example, the color of pieces that the player choose to play with, and their respective methods. It holds also the attributes which specifies the status of the game, for example, if the game has ended in a draw or a win, and when one of the player's King is in danger.

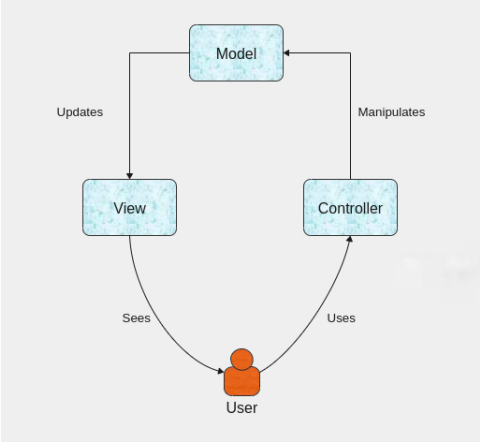

Figure 1: Model–view–controller (MVC)

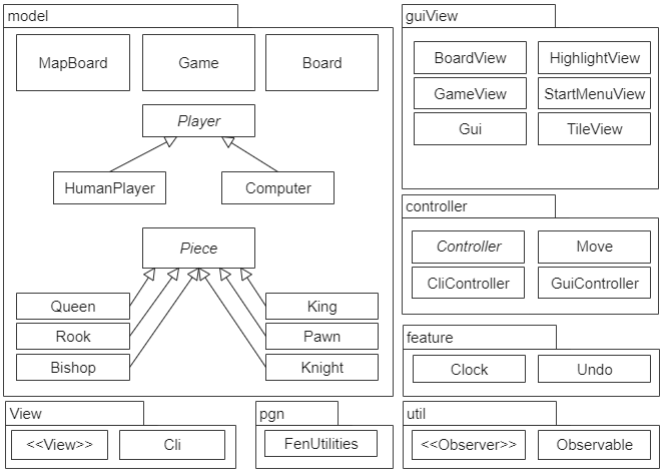

Figure 2: Class diagram. Overview of the classes without methods and communication

2.1 Model

The Model, in this case, the Game, is responsible for managing the data of the program. It receives user input from the controller.
The game has multiple fields, for example, the board of the game and the players, that are playing and interacting with game. The players choose to play with color either the white or the black.
The model has different methods, for example methods that load the player's pieces, add or remove pieces to the beaten list of the player, the game also checks if the player inputted a legal move.
After each round the game checks the status of itself, if the player's king is in danger or the game ended with a win or a draw, it tells the controller, which then the controller tells the view to display the message.
Each piece has it's own class that extends the overall piece class. In each piece class, the respective logic of the legal moves of the piece is implemented based on the current position of the piece and the move offsets of the piece, with regard to some exception that are added to each piece individually.
In the class player, each player has it's own list of pieces and king. The beaten pieces can be called from the respective player. The class has a field 'allowEnPassant' that tells the game if the player can preform an En Passant move or if he missed his chance in doing so.

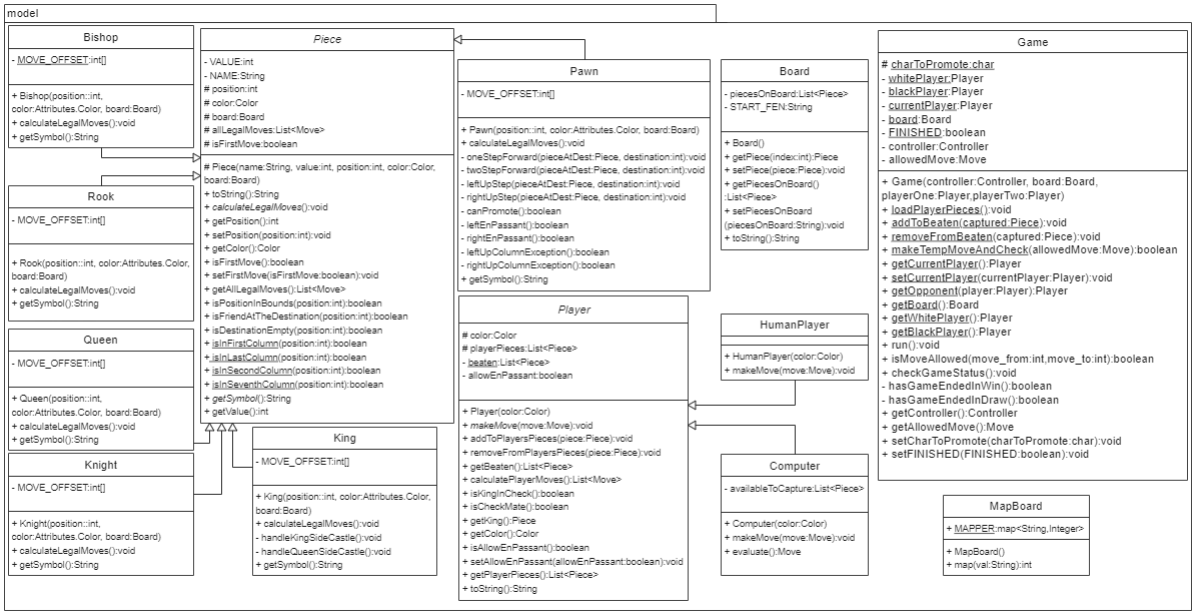

Figure 3: Model

2.2 View

The View means the presentation of the model in a particular format. Here, at the first stage of the program, the view displays the console representation of the program.
The console based interface reads an input from the player using the scanner and passes that input to the controller for validation.
The view receives requests from the controller, which in turns get displayed to the user.

- User interface Console
  This package contains classes for implementing the required console front end. While the class Console only functions for receiving Provides user input and console output, the CliController ensures that the inputs and outputs are made in the correct order and that the required information is sent from the controller to the Console can be forwarded and vice versa. The loop in which the console game runs can be found in the CliController.

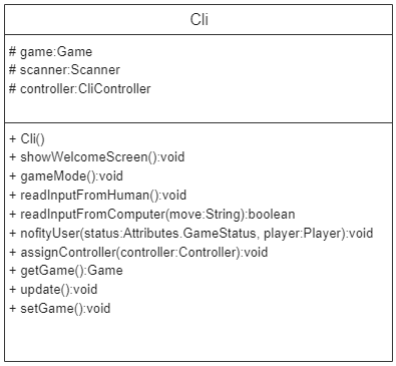

Figure 4: Cli

- User interface Gui
  This package contains classes for implementing the required GUI front end. The Views represent the different windows of the GUI. The controllers are used to process the interactions of the user. In doing so, they process Interactions with the respective view and manage switching between the Views.

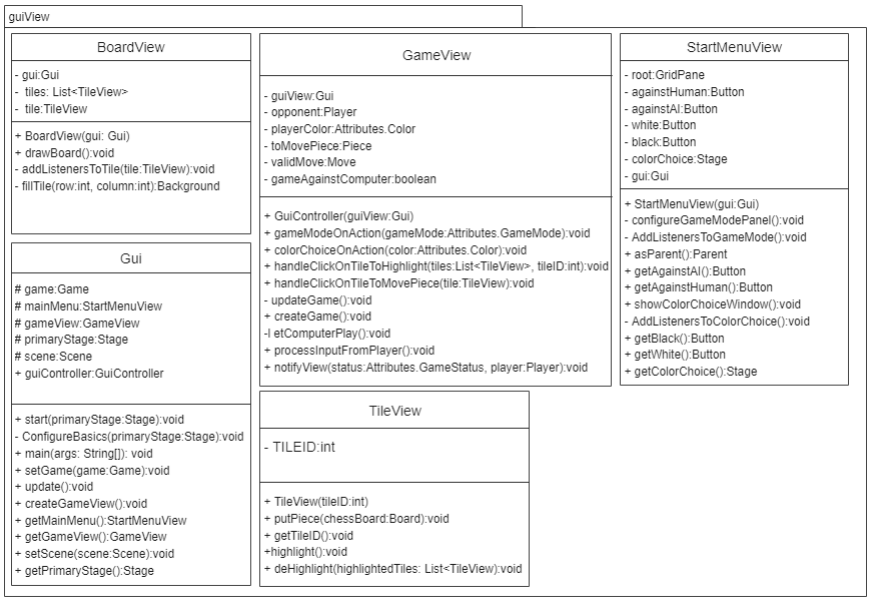

Figure 5: GuiView

2.3 Controller

The Controller creates the game when it gets created itself and responds to the user input from the view and performs interactions on the data model objects. The controller receives the input, validates it and then passes the input to the model.

This package mainly contains the classes Move, GuiController and CliController. While Move is responsible for the organization of the game, the provision of data, the execution of moves and also the provision of pieces that have already been captured, Gui- and CliController behave like a kind of set of rules that can be asked whether a move is possible or not .

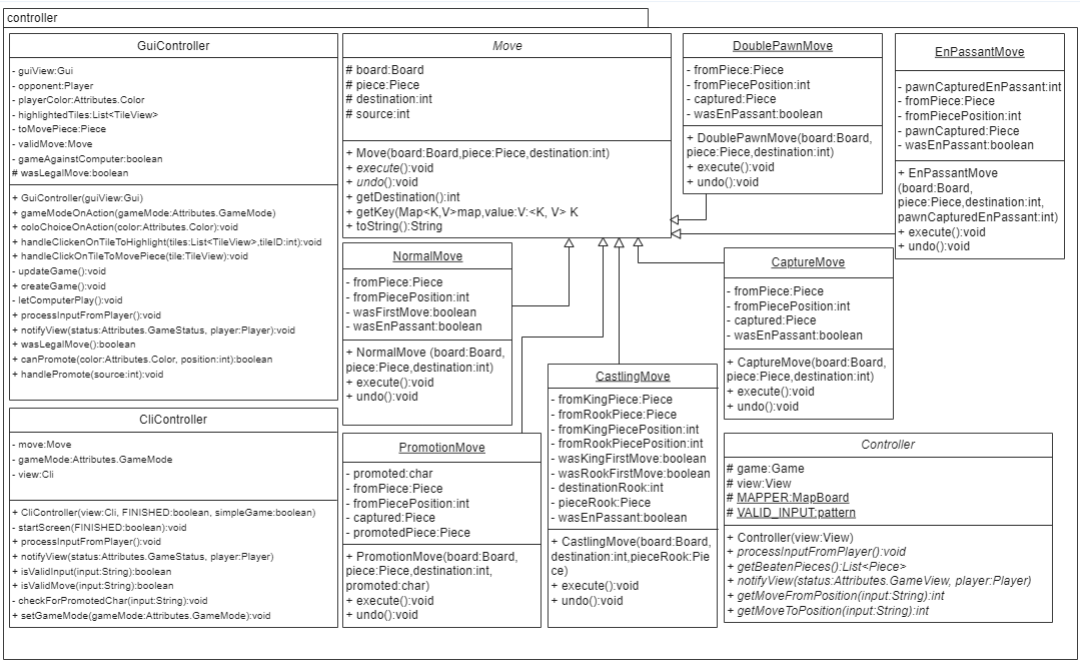

Figure 6: Controller

2.4 Additional features

2.4.1 Chess Clock

2.4.2 Undo