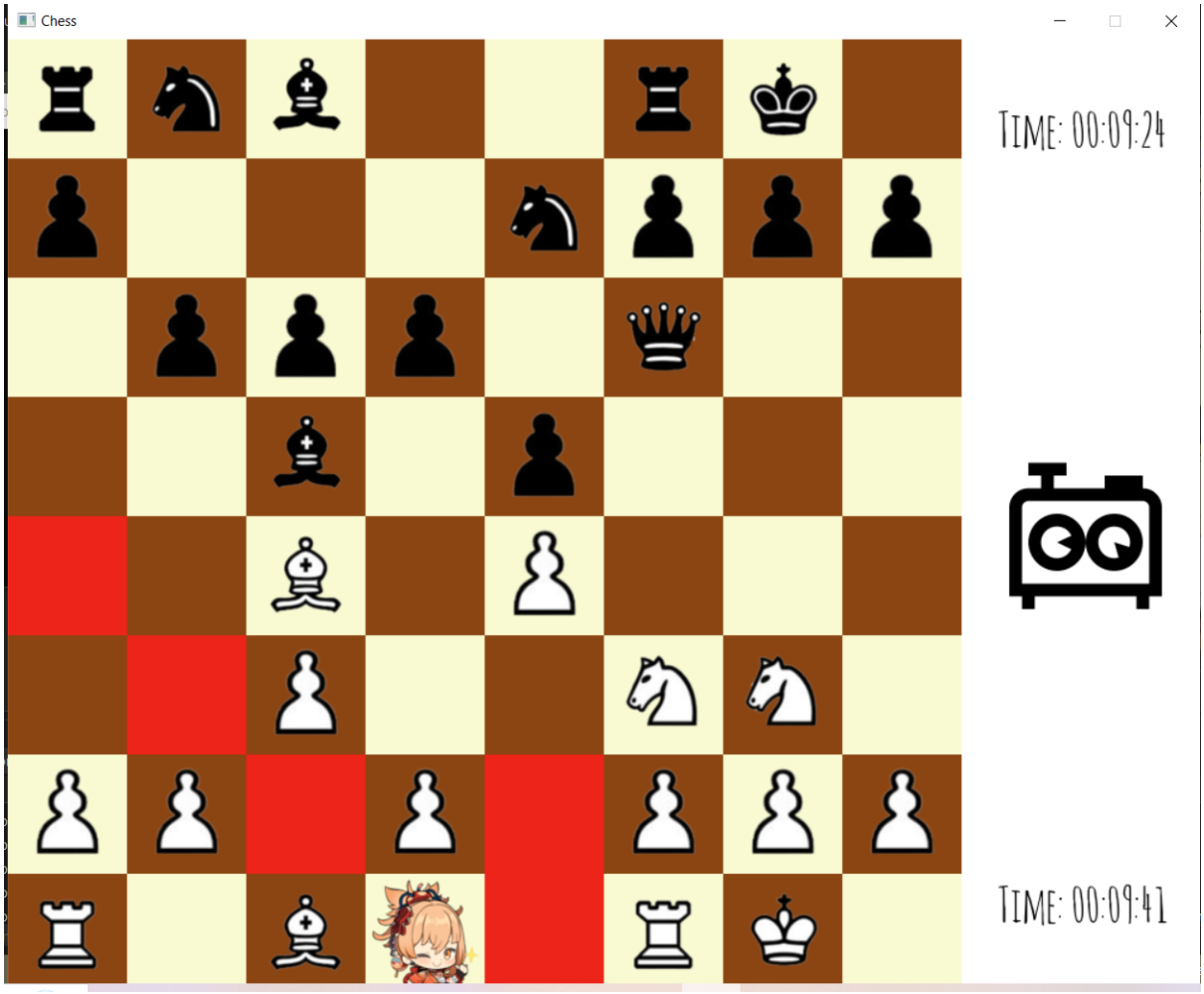
Design Overview for Epic Chess

Name: Le Minh Kha
Student ID: 104179506

Summary of Program

The project I have been working on is called `Epic C#hess` game provides support for playing a traditional chess game, where players can take turns making moves. Legal moves are highlighted on the board and the game prevents players from making illegal moves, especially if their king is already in check. In addition, the game also supports special moves such as castling, as well as detecting when a player's king is in checkmate.
The current project in D level only support Player vs Player mode, Player vs AI is set to be available in the HD custom program.



Required Roles

Describe each of the classes, interfaces, and any enumerations you will create. Use a different table to describe each role you will have, using the following table templates.

Table 1: <<role name>> details – duplicate

Role name: Board

Responsibility	Type Details	Notes
	Field type, parameter and return types	
Board	public class Board	Represents the chess board
Piece	public abstract class Piece	Represents a chess piece
Piece.Color	public enum Color { White, Black }	Represents the color of a piece
Current_Turn	Piece.Color Current_Turn { get; private set; }	Holds the current turn public
LegalMoves	public List<(int, int)> LegalMoves { get { return legalMoves; } }	Legal moves for the selected piece
Coordinate	public Piece Coordinate(int row, int col)	Accesses piece at coordinates, avoid obtaining piece outside of the array.
Draw	public void Draw()	Draws pieces on the board using a nested loop
Select_Piece	public bool Select_Piece(int mouseX, int mouseY, int squareSize)	Handles piece selection
Piece_Chosen	public void Piece_Chosen()	Calculating legal moves for the selected piece
Deselect_Piece	public void Deselect_Piece()	Deselects the selected piece

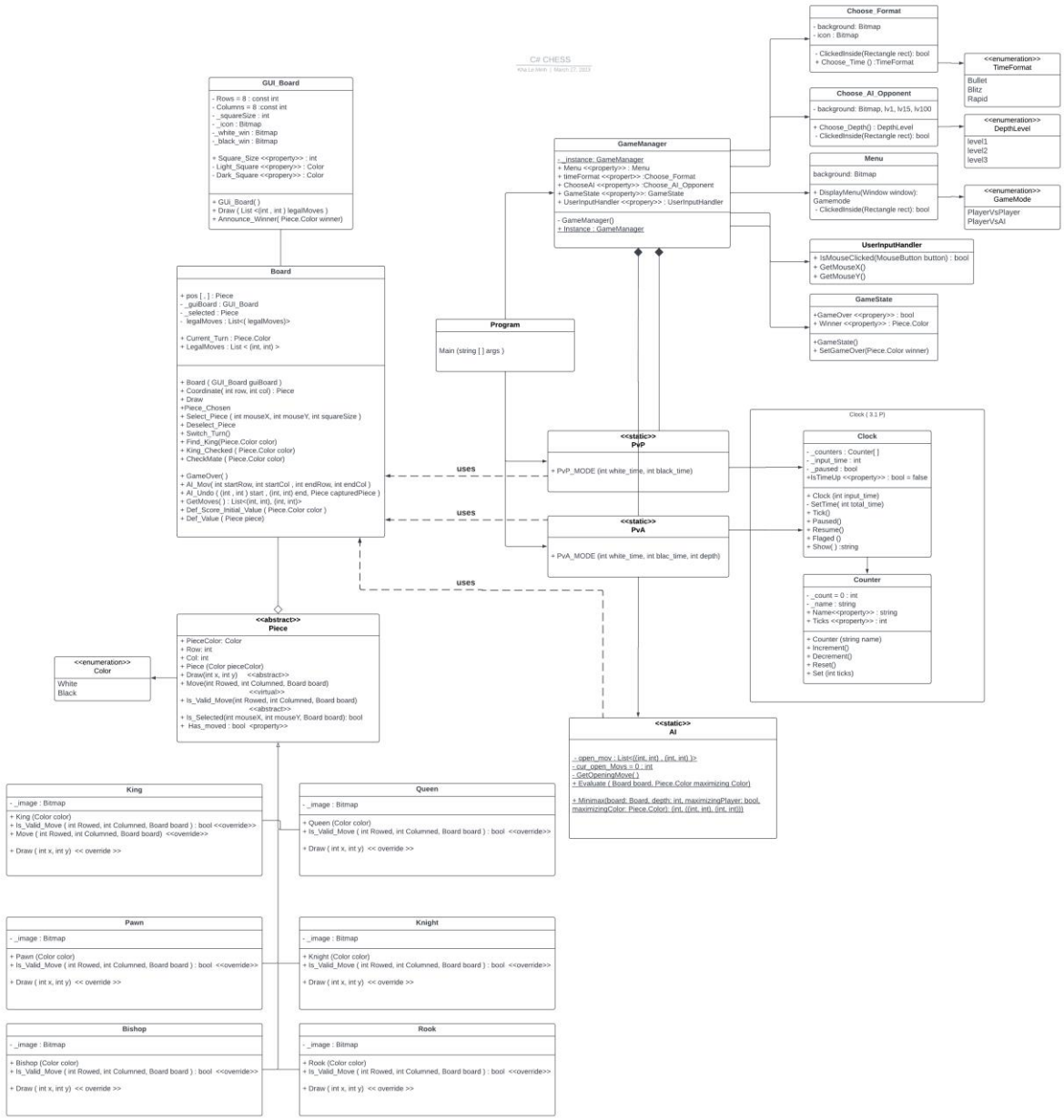
Role name: Piece

Responsibility	Type Details	Notes
	Field type, parameter and return types	
Piece	public abstract class Piece	Represents a chess piece
PieceColor	public Color PieceColor { get; set; }	Holds the piece color
Row	public int Row { get; set; }	Holds the row position of the piece
Col	public int Col { get; set; }	Holds the column position of the piece
LegalMoves	public List<(int, int)> LegalMoves { get { return legalMoves; } }	Legal moves for the selected piece
Coordinate	public Piece Coordinate(int row, int col)	Accesses piece at coordinates, avoid obtaining piece outside of the array.
Draw	public void Draw()	Draws pieces on the board using a nested loop
Move	public virtual void Move(int Rowed, int Columned, Board board)	Moves the piece on the board
Is_Valid_Move	public abstract bool Is_Valid_Move(int Rowed, int Columned, Board board)	Determines if a move is valid, then Move() can update coordinate accordingly.
Is_Selected	public bool Is_Selected(int mouseX, int mouseY, int square_Size)	Checks if a piece is selected
Has_Moved	public bool Has_Moved { get; set;}	Indicates if the piece has moved before

Table 2: <<enumeration name>> details

Value	Notes
TimeFormat	Bullet Blitz Rapid
DepthLevel	level1 level2 level3
GameMode	PlayerVsPlayer PlayerVsAI
Color	White Black

Class Diagram



Note: this class diagram is the finished one which include even some of the classes that belong to HD design (i.e. Game Manager and its support class, the AI mode). I will not be elaborating it, yet!

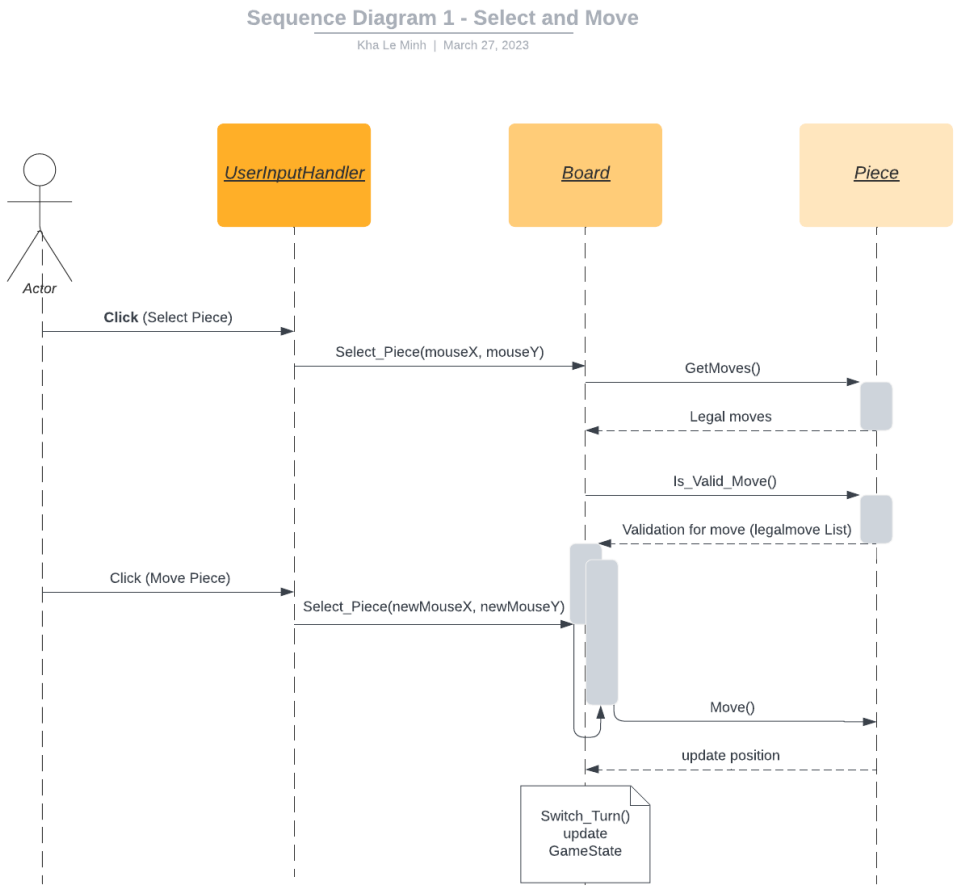
If the image happens to crack or broke upon zooming, here is the alternative link to the software I used to design the chart: https://lucid.app/lucidchart/f325cc8b-05c1-4fd1-9dd0-

42a1ffe13559/edit?viewport_loc=-158%2C1608%2C2560%2C1116%2CT7JIHdY0eP.Q&invitationId=inv_4cc24a8c-2289-4018-97de-722d316dc97d

Sequence Diagram

Sequence diagram of player click on the piece and move:

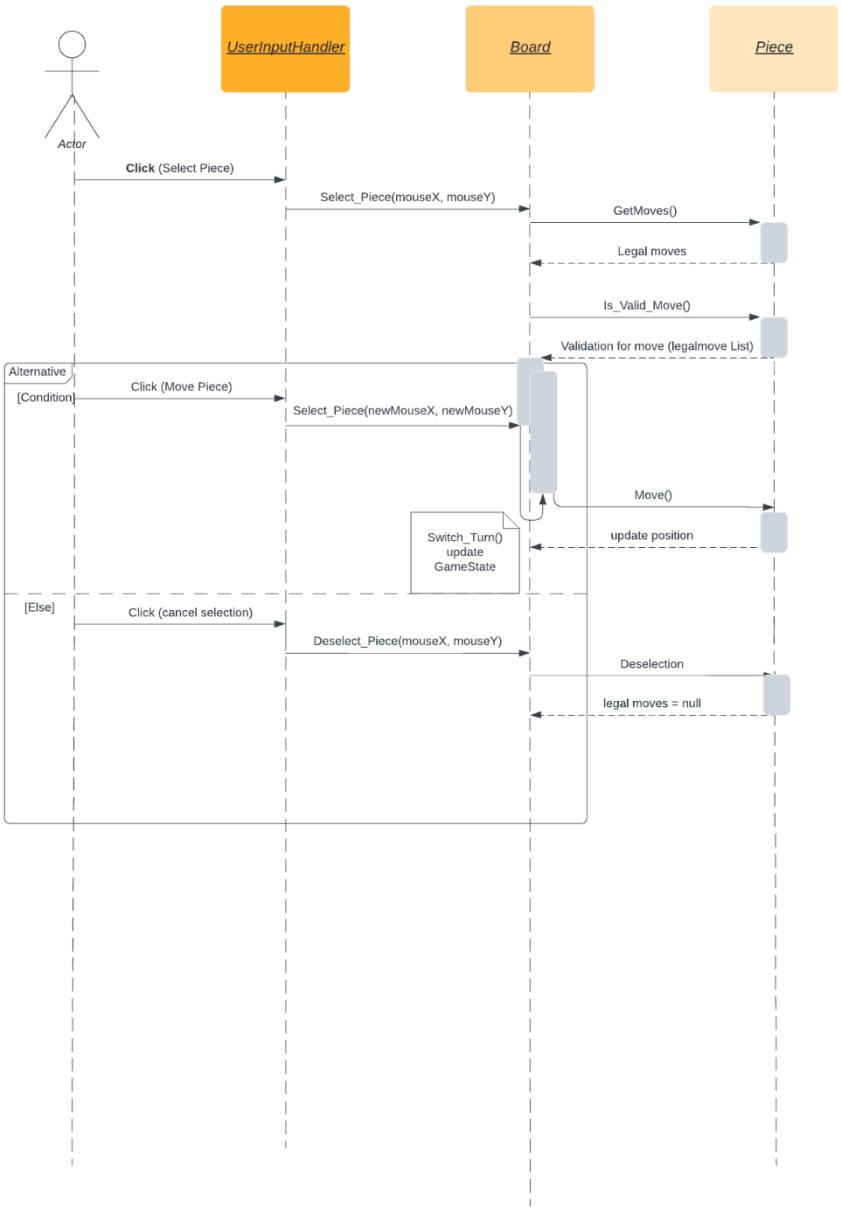
Link for better inspection: https://lucid.app/lucidchart/64012d45-7fd6-49d9-b28a-a9a23ca2e8ed/edit?viewport_loc=-27%2C-23%2C2139%2C1029%2C0_0&invitationId=inv_87f12063-ddaf-4dd9-bd78-0ac52c8469e0



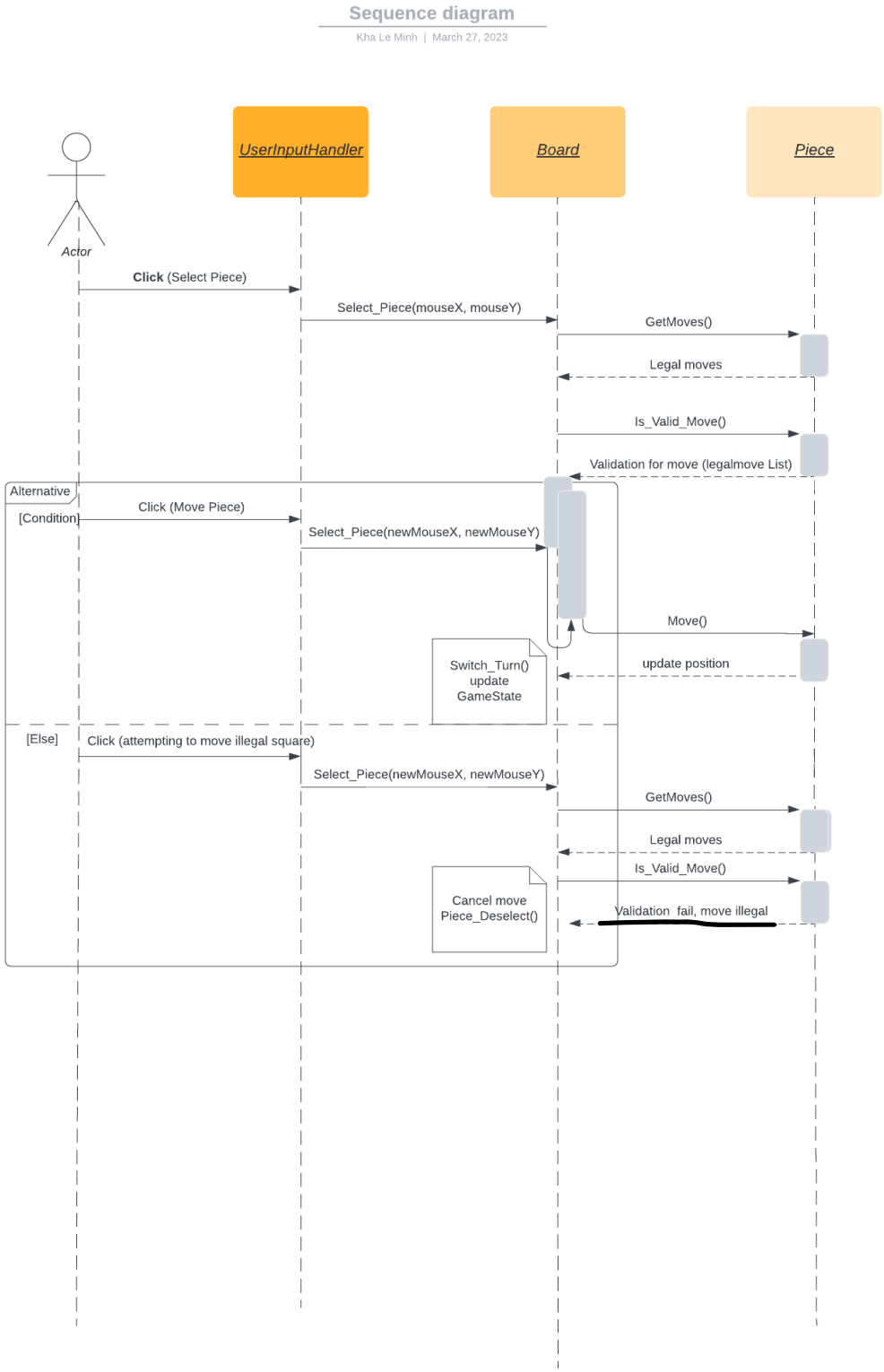
Sequence diagram including deselecting a piece:

Sequence Diagram 1 - Select and Move

Kha Le Minh | March 27, 2023



Sequence diagram for attempting to move to an illegal square:



Question 2: Provide a summary of your program — What does it do? What are some of the key features

`Epic C#hess` uses Splash Kit is a graphical chess application that allows users to play chess against another player (Player vs Player) or against an AI (Player vs AI). The key features of the program include:

Game Modes:
Player vs Player: Two players can play against each other in a local multiplayer setup.
Player vs AI: Will be available in HD project design.

A visually appealing chessboard with clear and distinct chess piece icons, making it easy for players to recognize and interact with the game pieces.
A menu screen that allows users to choose between different game modes.
In-game timer for both players, displayed on the screen, to keep track of the time each player has spent on their moves.

Chess mechanics:
The program supports some standardized chess rules, such as valid piece movements, capturing, and checkmate detection.
User Input:
The game handles mouse input for selecting and moving chess pieces.
The players can interact with the chessboard using simple click actions for a very smooth gameplay experience.

==========

Question3: Describe the main roles: enumerations, classes & interfaces.

Enumerations:

Piece.Color: An enumeration representing the colour of a chess piece (White or Black).
TimeFormat: An enumeration representing the time format for each side (Bullet, Blitz, Rapid).
DepthLevel: An enumeration representing the AI depth level (level1, level2, level3).
GameMode: An enumeration representing the game mode (PlayerVsPlayer, PlayerVsAI).
Classes:

Board: The class that manage all of the mechanic of a conventional chess game, it represents the chessboard and its game state, including the positions of the pieces and the current turn. It also contains methods to handle piece movements, turn switching, and checkmate detection.

Piece: An abstract class representing a generic chess piece, with properties such as color, position, and methods to handle piece-specific movements.
King, Queen, Rook, Bishop, Knight, Pawn: These classes inherit from the parent Piece class, each representing a specific type of chess piece and their unique movement rules.

Clock: Reference from Clock 3.1 P with some additional methods, it represents the in-game timer for each player, with methods to handle ticking, pausing, and displaying the remaining time.

GUI_Board: A class responsible for rendering the graphical elements of the chessboard, including the squares, pieces, and any additional visual elements like legal moves and the winner announcement.

PvA and PvP: These static classes define the main game loop and handle the logic for each game mode (Player vs AI and Player vs Player, respectively).

Some of the notable methods that might need some extra explanations are:

Draw () Method:
The Draw () method is responsible for drawing all the pieces on the chessboard. This method uses a for loop in a for loop to iterate over all the cells on the board. It checks if a piece is

present at the cell, and then calls the Draw () method of that piece to draw it on the GUI board. The x and y coordinates of the piece are calculated by multiplying the cell row and column by the size of a single square on the GUI board.

Select_Piece () Method:

The Select_Piece () method is responsible for handling piece selection and deselection. This method takes the mouse x and y coordinates and the size of a single square on the GUI board as input parameters. The method first checks if the clicked cell is valid and then gets the clicked piece using the Coordinate () method. If a piece is already selected, the method checks if the clicked cell is a legal move for the selected piece. If it is, then the Move () method of the selected piece is called to move the piece to the new cell, and the turn is switched to the other player using the Switch_Turn() method. If the clicked cell is not deemed a legal move, the method will then checks if another piece is selected and if it is, then deselects it using the Deselect_Piece () method. If no piece is currently selected, the method selects the clicked piece if it belongs to the current player.

Piece_Chosen () Method:

The Piece_Chosen () method is called when a piece is selected and is responsible for generating a list of legal moves for the selected piece. This method for loop in a for loop to iterate over all the cells on the board and checks if the selected piece can move to that cell using the Is_Valid_Move () method of the piece. If the move is legal, the cell coordinates are added to the LegalMoves list.

CheckMate () Method:

The CheckMate () method is responsible for detecting if the current player is mated or not. The method first checks if the king of the current player is under attack using the King_Checked () method. If the king is not being checked, then the method returns false. Else, the method iterates over all the pieces of the current player and checks if any of them can move to a cell where the king is not under attack. If that move is found, the method returns false, and the game continues. Else, the method returns true, indicating that the current player is in checkmate, and the game is over.

Question4: Describe the main responsibilities for the classes and interfaces.

Classes and its responsibility:

Board:

Manages the chessboard state, including pieces' positions and the current turn.

Handle piece movement and updates

Switch turns.

Check for checkmate conditions.

GUI_Board:

Renders the GUI of the chessboard.

Draw squares, pieces, and other visual elements on the board.

Highlight legal moves and announce the winner.

Piece (abstract):

Represents a generic chess piece with shared properties and methods.

Store piece's color, position, and movement status

Define methods for movement validation and updating positions

Provide a foundation for derived classes to implement specific behaviour

King, Queen, Rook, Bishop, Knight, Pawn: Inherits from Piece and implements specific movement rules and visuals for each chess piece.

Override movement validation methods to implement piece-specific rules

Override drawing methods to display the correct piece visuals

Clock:

Manages the in-game timer for each player.

Count down the remaining time for each player.

Pause and resume the timer based on the current turn.

Display the remaining time for both players.

UserInputHandler: Handles user input and provides utility methods.

Detect and process mouse events.

Provide helper methods for mouse interaction within the game context.

Question 5 : Show your plans to your tutor, lecturer, help desk staffers, and/or friends to get some feedback.

Reference for the chess pieces' image as my tutor requested:

Brown, A. (n.d.). *Chess piece queen king game chess pieces transparent - 2D chess piece PNG, PNG Download - Kindpng*. KindPNG.com. Retrieved March 28, 2023, from https://www.kindpng.com/imgv/ToiTwm_chess-piece-queen-king-game-chess-pieces-transparent/