# Chess Game C++

Explanation: To create the game of chess using object-oriented programming in C++. This is a personal project that I am excited to code as it will test my knowledge of the C++ language and my O.O.P. principles. This documentation is created to preplan my design of the game, and to provide insight into my thought process/design decisions. The ability to think and plan before coding is vital for creating well structured code, saving time and therefore cost, and solving complex problems before they arise during production.

Design: The design goal is to follow best practices when implementing solutions using object-oriented programming. This means using the four principles of Encapsulation, Data Abstraction, Polymorphism and Inheritance. I am thinking of making this as just a console application, and then possibly adding a ui to the game once coded, potentially porting the code to c# and using wpf as I have experience with it. Another expansion of this game could be to add a connection via sockets allowing for multiplayer! I will create documentation for this as the architecture could be peer to peer or I created a server and have 2 clients connect to it.

Pre-Coding Solution: Classes for each board piece must be created. Each piece can inherit from a base class called Chest Piece. My logic is that all chess pieces share similar rules. I will make a list of things to consider for the base chess piece class:

* Things such as position.
* Being able to move.
* The ability to take other pieces and the ability to be taken (Move to a position an opponent piece is occupying/ be removed).
* The constraint they cannot leave the chess board.
* They cannot occupy another space if the same color chess piece occupies it.
* Cannot go through other pieces unless knight!
* What color they are.

The following pieces need to have their own classes king, rook, bishop, queen, knight, and pawn. Each with their own set of rules.

Pawns:

* Can move 1 space or 2 spaces (only if first move).
* Can transform into another piece if it reaches the last tile of the opponents side.
* Can take pieces diagonal, left or right, but not in-front.

Knight:

* Moves in L shapes, 2 spaces vertical/horizontal and then one space left/right.
* Can Jump pieces.

Bishop:

* Can move diagonally any number of spaces until the end of the board.

Castle:

* Can move horizontally/vertically any number of spaces until the end of the board.

Queen:

* Can move horizontally, vertically, diagonally any number of spaces until the end of the board.

King:

* Can move horizontally, vertically, diagonally one space.
* If taken the game ends.
* Special case can “Castle” with a castle piece once per game if requirements are met.
* Can’t move into a space where another piece could take it.

Another classes I would like to create is the chess board class. I want this board to hold all the pieces for the game. It should be 8 x 8 tiles. At the start of the game, it will populate each player’s side with pieces based on the correct starting position chess follows. Each time a player moves they need to know if they can move a piece, how far, and if they can take a piece. Another possible solution would allow each piece to store its position, but I think this is much messier. Instead, I will create a board that will be a 2D vector that is 8 by 8 which will hold the pieces. Maybe have a tile class, in which each tile has a piece on it or not.

Chess Board:

* Keeps track of pieces
* 8x8 vector for tiles
* Has out of bounds (cant’ move outside vector)

Game:

* Has 2 opponents (white and black)
* Keep track of who’s turn it is
* Has a chessboard with pieces

Players:

* keep track of game state; playing, win, loss.
* Color of player (black or white).
* Possible player time if specified
* Name of player

These are just basic concepts for each class that can be expanded upon to fit requirements or readjusted in case of oversights.

Detailed Design:

This will be a more low-level design of the classes and program. Things like functions and their return types, variables and Inheritance.

Class name: ChessPiece (abstract class)

Variables:

|  |  |
| --- | --- |
| String Color | (black or white) |
|  |  |

Methods:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Class name: ChessBoard

Variables:

|  |  |
| --- | --- |
| Vector<Objects, Objects> tiles? | This would represent the board tiles, |
|  |  |
|  |  |

Methods:

|  |  |
| --- | --- |
| Bool MovePiece() | Function to move the piece. Class on the pieces move function which specifies who to move. Returns true if it can move or false if it cant. |
|  |  |

Class name: Game

Variables:

|  |  |
| --- | --- |
| String StateOfGame | “Win, Lose, Inprogress” |
| String PlayerNames[2] | “Contains player 1 and 2’s name” |
| Int PlayerTurn | 1 or 2 depending on whos turn |

Methods:

Class name: Player

Variables:

|  |  |
| --- | --- |
| String name | “Entered Name” |
| String Color | “The player color, if black or white” |
| Int NumberTurns | “how many turns they have moved” |

Methods:

This design is still rough as there are interacts between objects that I am still trying to plan out. Such as which object should be able to move the piece and how should that be communicated. Currently I think the game should communicate whos turn it is to the player. This allows that player to move, but to move they should interact with the chessboard object as this object holds all the pieces. Player1 who, who is white for example, should only be able to move white pieces. Therefore, the chessboard needs to know which pieces are white and which can move. Only pieces that can move should be available. When a movement is performed many things need to be considered.

Did a piece get taken? If so, it needs to be removed.

Did a player lose? Game should end.

It is now the opponents turn to move.

Possible time needs to be stopped/started when that feature is implemented.

I will begin coding now using this documentation as an outline for constructing my game. I will continue to modify and update this documentation as development continues. This is my first construction of the game and therefore optimizations and new ideas should be generated during the development of this project and also need to be integrated.