Problem and design report – 3rd Lab Assignment

As a team, do the following.

**Problem statement**

Write in your own words a description of the system’s purpose.

**Design constraints:**

* You must use inheritance and polymorphism.
* You must use a hierarchy of chess pieces that reuse behavior
* Must override a method such as isValidMove
* Each class should have attributes and methods and be defined in a separate file.

**Design**

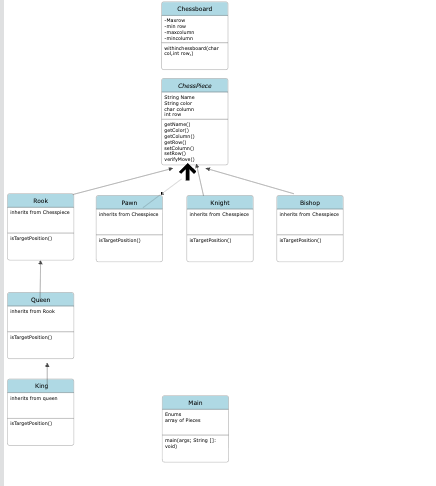
List the elements/parts/modules/ functions that you need to solve the problem and the purpose of each part.

***Write the pseudocode for each part FIRST. You shall NOT write any code until you finish the pseudocode as a TEAM.***

| Part name | Purpose | Pseudocode |
| --- | --- | --- |
| e.g. Chessboard | Define the size of the board and determine if a chess piece’s position is within the board. | - check if the chess piece (posX is between columns a and h) and (posY is between 1 and 8) |
| Pawn | Hold the color of the piece and its position, and determine if a target position is a valid position | - isValidPosition: if the color is white, the target position has a posY value greater than 1 for the current position and the same X value; if the color is black, the target position has a posY value less than 8 for the current position and the same X value.  - We’ll extend ChessPiece using Super |
| ChessPiece | The parent class of all pieces which holds information for the pieces to inherit from this class. | -String Name  String Color  Char Column  Int row  Empty constructor  Constructor with parameters  Getter  Setter  verifyMove - which will be overridden by the isvalidposition |
| Knight | Hold the color of the piece and its position, and determine if a target position is a valid position | - isValidPosition: if the color is white, the target position has a posY value greater than 1 moving two positions vertically and one position horizontally from the current position or two positions horizontally and one position vertically, relative to the current position; If the color is black, the target position has a posY value less than 8. From the current position the knight can move two positions vertically and one position horizontally or two positions horizontally and one position vertically, relative to the current position.  - We’ll extend ChessPiece using Super |
| Rook | Hold the color of the piece and its position, and determine if a target position is a valid position | - isValidPosition: if the color is white, the target position has a posY value greater than 1 for the current position if at starting position and less than or greater than 1 if the position is at any y value greater than 1, if the y position moves any number of squares, the x position remains the same. The same logic we applied for the y deltas we would apply for x if in any case we are moving the x direction, if the color is black, the target position has a posY value less than 8 for current position if at starting position and less than or greater than 1 if the position is at any y value greater than 1, if the y position moves any number of squares, the x position remains the same. The same logic we applied for the y deltas we would apply for x if in any case we are moving the x direction.  - We’ll extend ChessPiece using Super |
| Bishop | Hold the color of the piece and its position, and determine if a target position is a valid position | - isValidPosition: if the color is white, the target position has a posY value greater than 1 for the current position. The bishop can move any number of spaces diagonally, so the change in x and y values from the current position stay equal; if the color is black, the target position has a posY value less than 8. The bishop can move any number of spaces diagonally, so the change in x and y values from the current position stay equal.  - We’ll extend ChessPiece using Super |
| Queen | Hold the color of the piece and its position, and determine if a target position is a valid position | - isValidPosition: if the color is white, the target position has a posY value greater than 1 for the current position with the ability to move any number of squares in the y column, x column or horizontally assuming no other movement is done; if the color is black, the target position has a posY value less than 8 but greater than 0 for the current position isValidPosition: if the color is white, the target position has a posY value greater than 1 for the current position; if the color is black, the target position has a posY value less than 8 but greater than 0 for the current position and the same X value.  - We’ll extend ChessPiece using Super |
| King | Hold the color of the piece and its position, and determine if a target position is a valid position | - isValidPosition: if the color is white, the target position has a posY value greater than 1 for the current position. The King can move only one space horizontally, vertically, and diagonally from its current position; if the color is black, the target position has a posY value less than 8 for the current position. The King can move only one space horizontally, vertically, and diagonally from its current position.  - We’ll extend ChessPiece using Super |
| Main loop | Run the main code of the game and user interaction, this is where we will be getting all inputs and verifying positions | -Create array to hold pieces  -Scanner  -Populate array with pieces  - enums for color,piece,columns  - for checking if a piece has already been created, use a boolean array with a length of 6, each boolean matches an entry in the PieceType enum. After prompting the user for a piece type we check if the boolean which corresponds to that type is true or not.  - create piece objects using switch case  -get target positions (char, column)  -run through a loop to check them all and print our the results |
|  |  |  |

**Design Model**

<<Create your class diagram for LAB 3 and include attributes, methods, access levels, associations, and multiplicity. Describe your model.>>

1. **Model description  
   This UML Diagram represents the system we are creating. In this UML Diagram it illustrates the hierarchy of classes, where our abstract class ChessPiece is extended by Pawn, Rook, Knight and Bishop. The Queen class inherits from Rook, and the King class inherits from the Queen.**
2. **<< insert diagram >>**
3. ****

**Design Limitations or Disadvantages.**

Write any disadvantages of this solution (Lab 3) compared to Lab 1 and Lab 2.

There aren’t very many disadvantages to this paradigm over the first 2 labs, because the labs are very small. Though the amount of code required quickly grows with the amount of getters, setters, and constructors that were introduced in lab 2. If the project became much larger the amount of boiler plate code would be an issue.

**Design benefits.**

Write any benefits of this solution (Lab 3) compared to Lab 1 and Lab 2.

Heavily simplifies the code from lab2, removing an extra switch statement.  
Allows for more flexibility in using the pieces