

Project 2 - Chess Pieces

Due Feb 19, 2024 by 11:59pm

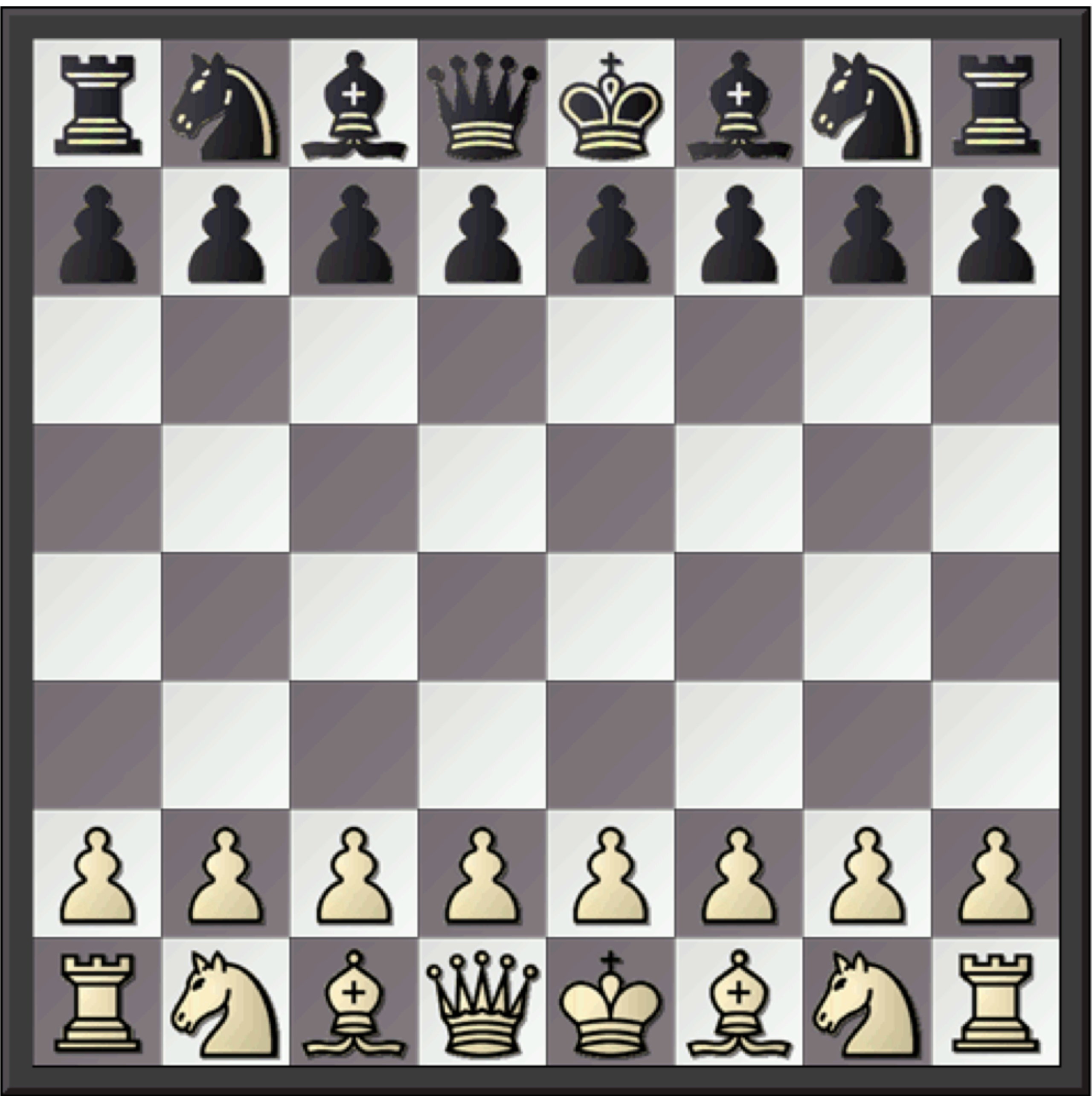
Points 100

Submitting an external tool

Project 2: Chess Pieces

Introduction

A game of chess has several kinds of pieces: pawns, knights, bishops, rooks, queens and kings. These pieces are arranged on a chess board as shown in the figure below.



(Image taken from http://www.chess-game-strategies.com/images/kqa_chessboard_large-picture_2d.gif)

A cell on the board is specified by a (row, column) tuple: rows increasing from bottom to top and columns increasing from left to right. Traditionally, the black pieces are arranged in the top two rows as shown.

Each chess piece can move in a specific way. In addition to moving, each chess piece can also kill a chess piece of the opposite color if it moves to its place. The rules for each chess piece are as follows:

- **Bishop:** A bishop can only move diagonally, and kill any opponent's piece if it can move to its place.
- **Knight:** A knight can move only in an L pattern: two cells horizontally and one vertically or vice versa. It can kill any opponent's piece if it can move to its place.
- **Rook:** A rook can move horizontally or vertically. It can kill any opponent's piece if it can move to its place.
- **Queen:** A queen can move horizontally, vertically and diagonally. It can kill any opponent's piece if it can move to its place.
- **King:** A king can move one square in any direction (horizontally, vertically, or diagonally). It can kill any opponent's piece if it can move to its place.
- **Pawn:** A pawn is interesting: it can move only "ahead," not backwards towards where its color started. It can move only one place forward in its own column (except the first time it moves: it can move one or two places forward). However, to kill it must move one place forward diagonally (it cannot kill by moving straight).

More information on the movement of chess pieces [here](#) and [here](#).

What to do

You must design and implement classes that represent the above chess pieces. These classes should be named exactly as their names above, and all of them should implement the [ChessPiece interface](#). The package must be named `chess`.

Each chess piece should be able to:

- return its current position on the chess board, as methods `getRow()` and `getColumn()`
- return its color as an `enum Color`, using `getColor()`
- determine if it can move to a given cell, as a method `canMove(int row, int col)`
- determine if it can kill a provided piece starting from where it currently is, as a method `canKill(ChessPiece piece)`

In addition, each chess piece should have a constructor that takes in an initial position as a row and column, and a color as an `enum Color` with values `BLACK` and `WHITE`. All rows and columns begin with 0. Rows decrease in number from top to bottom in the above chessboard.

Beyond these classes, enum, interface and methods, you are free to design any other interfaces and classes as you see fit. However, they must have good justification, be designed appropriately, and be used in the appropriate place.

Pay attention to abstraction. Try to minimize reusing code (adding additional classes/abstract if necessary).

You must write tests for each chess piece that test these operations thoroughly. Your methods and constructors should throw an appropriate exception in cases where it is reasonable.

Note: Before you write any code, create a class diagram that describes the structure of your code.



What you may use a language model for:



- You may use a language model to write `getRow()`, `getColumn()`, or `getRow()` for up to 3 chess pieces
- You may use a language model to write up to 3 tests


Remember: You are not required to use a language model! If you feel unsure about the syntax or need the practice, it is especially recommended to write it by hand.

How to submit

1. In addition to your `src` and `test` folders, create a folder named `res` and include your class diagram there.
2. Create a zip file that contains directly your `res`, `src` and `test` folders. When you unzip the file, you should see only these three folders.
3. Upload the zip file on Gradescope.







Autograder Results

ResultsCode

Test knights can move to correct locations (5/5)

Test that pawns can and can't kill (5/5)

Test pawns can move to correct locations (5/5)

Get row (1/1)

Test queens can move to correct locations (5/5)

Project 2 - Chess Pieces

Graded

Select each question to review feedback and grading details.

Student Yunyu Guo

Total Points 88.5 / 100 pts

Autograder Score 50.0 / 50.0

Passed Tests Test knights can move to correct locations (5/5) Test that pawns can and can't kill (5/5)

Select a question. ^ More Request Regrade Download Submission Next Question