Assignment 04 / Final Part 02

10% of overall course marks, 10/35 of Final
Deadline: September 15, 2021

Your objective is to design and display a random 8x8 (8 row, 8 column) chessboard using unicode with all chess pieces, and write the output to a file. The file output will look like the attached file.

Points to note:

1. Each empty chess position (where there are no chess pieces) is denoted by a X.
2. There is an empty space between two chess positions.
3. Each chess piece is displayed in the output using it's corresponding chess symbol in unicode. (https://en.wikipedia.org/wiki/Chess_symbols_in_Unicode).
4. Rows are denoted from top to bottom, numbers 1 to 8. Columns are denoted from left to right, characters a-f. For example, White king ♔ is located at 5th row from the top, and 3rd column from the left. So its position is denoted by **c5.** Here, c means 3rd column from the left, and 5 means 5th row from the top.
5. 32 chess pieces will occupy 32 unique positions in the chess board. No two chess pieces will occupy the same location. You can keep track of currently empty and occupied positions.
6. Write the output to a .txt file using **file i/o** operations.
7. There are 6 types of chess pieces (King, Queen, Rook, Bishop, Knight, Pawn) in White and Black color.
    a. Each piece should contain a color and a position.
    b. Create a base abstract chess piece class and extend all concrete chess piece classes from that base abstract class.
    c. Each class should have a default, no-arg constructor. You can create as many other constructors as needed.
    d. Each chess object will keep track of the position it is in. Meaning, if you are maintaining a two-dimensional matrix for the board, then each individual chess piece will know where in the board matrix it is located.
    e. There are a total 32 chess pieces when the game is initialized. Look at the rules of chess, and initialize the chess pieces accordingly.
    f. Create and use an Enum for the colors.
    g. Implement a comparable interface for two chess pieces. Two chess pieces will be the same if they are of the same type, have the same ID, color and location. You are free to choose the less than or greater than criteria as you see fit.
    h. Use a static data field to generate a unique ID for each type of piece. Start the ID count from 1. Each chess piece of the same type will have unique IDs (see output file). For example, the 8 pawns will have ids from 1 to 8.
    i. Implement the toString method for each class.

Assignment 04 / Final Part 02

10% of overall course marks, 10/35 of Final
Deadline: September 15, 2021

---

j. Write a method that returns a list/array of possible next positions that a chess piece can move to next. Look at the chess rules. A position will be a String containing two letter (like e7). The first letter signifies te column (a - f, from left to right), and second letter signifies row (1 to 8, from top to bottom).
k. Create your custom exception class as needed and use them.

Submit the .java file, output file and UML diagram. UML diagram must be drawn using Hand.

Mark distribution:

1) **Java Code: 7 Marks (5 marks for general code, 2 marks for method for each class for generating possible next positions).**
2) **UML diagram: 3 marks.**