IS 545 Object Oriented Programming and Data Structures

Assignment #2

1. Upload your solutions (as a zip file) to the odtuclass web site for the course.
2. Please add comments into your source codes, in order to make them more readable.
3. Submit UML class diagram for the package **chess.game**

Submissions will include:

- Complete **chess.game** directory
- UML diagram for the classes in this package

Late submissions will incur a -1% penalty per hour after the deadline.

Deadline: December 10th, 2018 23:55

You are expected to work **individually**, NOT in groups. You will also be expected to follow **the academic integrity rules**.

Requirements

- Please refer to https://www.chess.com/learn-how-to-play-chess for information on the chess game and the game rules.
- In this assignment, a tool to play chess game shall be developed.
- Two players shall play chess game using the tool. The program shall get input for the moves from the players and display the chess board after each valid move. (That is, the program will not play chess, it will only get input from the users and output the board after each player move)
- Chess is played on board having 8x8=64 squares. Each square is identified with a pair of letter and a number corresponding to its row and its column in the board. Rows are labeled through '1' to '8' and columns are labeled through 'a' to 'h'. For example, the square at the third column and the fourth row is identified by 'c4'. Square 'a1' is black and the neighboring squares' colors alternate (black and white).
- In the board, black pieces shall be represented by lowercase letters, and white pieces shall be represented by uppercase letters.
    - i, r, n, b, q, and k represent black pawn, rook, knight, bishop, queen, and king, respectively.
    - I, R, N, B, Q, and K represent white pawn, rook, knight, bishop, queen, and king, respectively. -After each turn, the board shall be displayed (sample output are shown in the following sections). While displaying the board,
    - Row and column labels shall be shown on both sides (left-right, top-bottom).
    - Square borders shall be drawn with characters '+', '-', '|'.
    - Empty squares: the white squares shall be left blank, the black squares shall be marked with '.'.
    - Occupied squares: the squares shall be marked with the letter representing the piece in it.

The starting position shall be:

```
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
```

```
 8 | r | n | b | q | k | b | n | r | 8
   +---+---+---+---+---+---+---+---+
 7 | i | i | i | i | i | i | i | i | 7
   +---+---+---+---+---+---+---+---+
 6 |   | . |   | . |   | . |   | . | 6
   +---+---+---+---+---+---+---+---+
 5 | . |   | . |   | . |   | . |   | 5
   +---+---+---+---+---+---+---+---+
 4 |   | . |   | . |   | . |   | . | 4
   +---+---+---+---+---+---+---+---+
 3 | . |   | . |   | . |   | . |   | 3
   +---+---+---+---+---+---+---+---+
 2 | I | I | I | I | I | I | I | I | 2
   +---+---+---+---+---+---+---+---+
 1 | R | N | B | Q | K | B | N | R | 1
   +---+---+---+---+---+---+---+---+
     a   b   c   d   e   f   g   h
```

This multi-line string should be returned by the interface method `ChessBoard.getDisplayString()`.

- White starts play and black plays next. The same is repeated at each round.
- A player enters a move to the system by specifying the current location of the piece to be moved and a target location. For example, for the above board, the input “g1 f3” can be used to move white knight at square g1 to square f3.
- Pawns are always promoted to queen when they reach to 8th rank (when a white pawn reaches to row 8, or a black pawn reaches to row 1)
- Castling moves shall be omitted.
- “En passant” moves shall be omitted.
- Checkmate shall not be checked by the program but the game has to be over when opponent’s king is captured.
- The program will check the following for a move
    - MOVE_PARSE_ERROR: The move input should be well-formatted. (e.g. **e2 e4** with exactly single space between two coordinates)
    - and must not be the same (NO_PIECE_TO_MOVE or CANNOT_CAPTURE_OWN_PIECE).
    - NO_PIECE_TO_MOVE: At there must be a piece of current player.
    - CANNOT_MOVE_OPPONENTS_PIECE: At there must not be a piece of the opponent player.
    - CANNOT_CAPTURE_OWN_PIECE: At there must not be any of the current player’s pieces (i.e., a player cannot capture his/her own piece)
    - must be empty or there must be one of opponent’s pieces (i.e. one of opponent’s piece is captured).
    - MOVE_RULE_VIOLATION: Pieces should move according to the chess rules
    - JUMP_OVER_PIECE_NOT_ALLOWED: Pieces except Knight may not jump over pieces, it should be checked.
- If any error is detected, ChessException should be thrown with correct error message.
- Input `q` shall terminate the program.
- Input `help` displays help message, which is provided for you.

- Input `save` stores current board status to the disk. Save only allowed for white player's turn. You have to implement `ChessBoard.save()`
- Input `load` restores previously saved board status. After loading, move number resets to **1** and it will be White player's turn. Pawns that are not at their original position shall be considered as *moved* (as first move of pawns can be two steps). You have to implement `ChessBoard.load()`

Save/Load Format

It is very important to implement save(..) and load(..) correctly because **your results will be tested programmatically** using these methods.

File format of the saved board is a text file. It contains 8 lines having strings of length 8.

Initial board would be saved as

```
rnbqkbnr
iiiiiiii
........
........
........
........
IIIIIIII
RNBQKBNR
```

Empty board would be saved as

```
........
........
........
........
........
........
........
........
```

A board with single *white queen* at location **c2** is saved as

```
........
........
........
........
........
........
..Q.....
........
```

Design Requirements

- Hold squares in a two-dimensional array in the class representing the chess board.
- Obey naming conventions.
- Use regular expressions to check proper formatting of user move input (5 pts).
- Use abstraction (5 pts)
- Use polymorphism (5 pts)
- Use inheritance (5 pts)
- Throw ChessException with the correct error value at `ChessBoard.move(..)` method (10pts).
- Submit a running application without extra effort required for the instructor (10 pts)
- Submit UML class diagram for package `chess.game`, **do not use** automatically generated UML diagrams, draw your own (10 pts)
- Implement all the application correctly and everything works as described (50 pts)
- You are **not allowed** to change supplied files in `chess.app` and `chess.common` packages.
- You shall declare `chess.game.Board` which implements `chess.common.ChessBoard` class implemented as it is used in the main class.
- You are expected to add all your classes into the `chess.game` package **only**.

Codes and Templates

You have to use the classes described below, do not make any change on them.

Main Class

Do not change this file, use as it is.

```
package chess.app;

import chess.common.*;
import java.io.*;
import java.util.*;

public class ChessApp {

    public static final String FILENAME = "chessboard.txt";

    public static final String PROMPT_FORMAT = "Move %d for  %s>";

    public static final String WELCOME_MESSAGE = "Welcome to the Chess application.";

    public static final String HELP_MESSAGE
            = "Enter your moves as: \n"
            + "     <current location of the piece> <target location of the piece>\n"
            + "For example enter `e2 e4` to move piece located at e2 to e4\n"
            + "Other commands are HELP, SAVE and LOAD\n"
            + "Enter 'q' to quit.";

    public static void main(String[] args) throws FileNotFoundException, IOException {
        ChessBoard board = new chess.game.Board();
        Scanner scanner = new Scanner(System.in);
        System.out.println(WELCOME_MESSAGE);
```

```
        String line = "help";
        boolean gameFinished = false;
        while (!"q".equalsIgnoreCase(line)) {
            line = line.toLowerCase(Locale.ENGLISH);
            switch (line) {
                case "help":
                    System.out.println(HELP_MESSAGE);
                    break;
                case "save":
                    if (board.isWhitesTurn()) {
                        //We use try-with-resources to automatically close the file
after save
                        try (PrintWriter writer = new PrintWriter(FILENAME)) {
                            board.save(writer);
                            System.out.println("Board saved.");
                        }
                    } else {
                        System.out.println("ERROR: Save is allowed only when it is
white's turn.");
                    }
                    break;
                case "load":
                    //We use try-with-resources to automatically close the file after
load
                    try (BufferedReader reader = new BufferedReader(new
FileReader(FILENAME))) {
                        board.load(reader);
                        System.out.println("Board loaded.");
                    }
                    break;
                default: {
                    try {
                        ChessPiece captured = board.move(line);
                        if (captured != null) {
                            System.out.println(String.format("Piece %s was captured",
captured.toChar()));
                            gameFinished = Character.toUpperCase(captured.toChar()) ==
'K';
                        }
                    } catch (ChessException ex) {
                        System.out.println("ERROR: " + ex.getMessage());
                    }
                }
                break;
            }
            if (gameFinished) {
                System.out.println("Game is finished");
                break;
            } else {
                System.out.print(board.getDisplayString());
                System.out.format(PROMPT_FORMAT, board.getMoveCount(),
board.isWhitesTurn() ? "White" : "Black");
                line = scanner.nextLine();
```

```
            }
        }
        System.out.println("Bye...");
    }

}
```

Common Classes and Interfaces

Do not change these files, implement them as they are given. Otherwise tests during your evaluation may fail.

```
package chess.common;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.PrintWriter;

public interface ChessBoard {

    //returns captured piece or null if nothing was captured
    ChessPiece move(String moveText) throws ChessException;

    boolean isWhitesTurn();

    int getMoveCount();

    String getDisplayString();

    void load(BufferedReader reader) throws IOException;

    void save(PrintWriter writer);

}
```

```
package chess.common;

public interface ChessPiece {

    boolean isWhite();

    char toChar();
}
```

```
package chess.common;

public class ChessException extends Exception {

    private final ChessError error;

    public enum ChessError {
```

```
        MOVE_PARSE_ERROR,
        NO_PIECE_TO_MOVE,
        CANNOT_MOVE_OPPONENTS_PIECE,
        CANNOT_CAPTURE_OWN_PIECE,
        JUMP_OVER_PIECE_NOT_ALLOWED,
        MOVE_RULE_VIOLATION
    }

    public ChessException(ChessError error) {
        super();
        this.error = error;
    }

    public ChessError getError() {
        return error;
    }

    @Override
    public String getMessage() {
        switch (error) {
            case MOVE_PARSE_ERROR:
                return "Invalid move.";
            case NO_PIECE_TO_MOVE:
                return "No piece to move.";
            case CANNOT_MOVE_OPPONENTS_PIECE:
                return "You cannot move opponent's piece";
            case CANNOT_CAPTURE_OWN_PIECE:
                return "You cannot capture your own piece.";
            case JUMP_OVER_PIECE_NOT_ALLOWED:
                return "This piece cannot jump over other pieces.";
            case MOVE_RULE_VIOLATION:
                return "This piece cannot move to here.";
            default:
                throw new AssertionError(error.name());

        }
    }

}
```

Your package (chess.game)

This class is required with its exact naming. You are also expected to implement other classes in the same package.

```
package chess.game;


public class Board implements ChessBoard {

    //......
    //......
    //......

}
```

Sample Output

The text after **Move [n] for [Player]>** string are user's input.

```
Welcome to the Chess application.
Enter your moves as:
     <current location of the piece> <target location of the piece>
For example enter `e2 e4` to move piece located at e2 to e4
Other commands are HELP, SAVE and LOAD
Enter 'q' to quit.
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i | i | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . |   | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I | I | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 1 for  White>e2 e4
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i | i | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . |   | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . | I | . |   | . | 4
```

```
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 1 for  Black>d7 d5
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | i | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . | I | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 2 for  White>e4 d5
Piece i was captured
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | I | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 2 for  Black>a8 a5
ERROR: This piece cannot jump over other pieces.
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
```

```
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | I | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 2 for  Black>a6 a7
ERROR: No piece to move.
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | I | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 2 for  Black>a8 b8
ERROR: You cannot capture your own piece.
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | I | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
```

```
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 2 for  Black>a8 c3
ERROR: This piece cannot move to here.
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | I | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 2 for  Black>a6 k9
ERROR: Invalid move.
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | I | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 2 for  Black>a7a6
ERROR: Invalid move.
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n | b | q | k | b | n | r | 8
```

```
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | I | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . |   | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 2 for  Black>c8 h3
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 | r | n |   | q | k | b | n | r | 8
  +---+---+---+---+---+---+---+---+
7 | i | i | i |   | i | i | i | i | 7
  +---+---+---+---+---+---+---+---+
6 |   | . |   | . |   | . |   | . | 6
  +---+---+---+---+---+---+---+---+
5 | . |   | . | I | . |   | . |   | 5
  +---+---+---+---+---+---+---+---+
4 |   | . |   | . |   | . |   | . | 4
  +---+---+---+---+---+---+---+---+
3 | . |   | . |   | . |   | . | b | 3
  +---+---+---+---+---+---+---+---+
2 | I | I | I | I |   | I | I | I | 2
  +---+---+---+---+---+---+---+---+
1 | R | N | B | Q | K | B | N | R | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
Move 3 for  White>
```