Chess Classes

Introduction

In this assignment you will practice

- writing classes,
- using enums, and
- writing polymorphic classes.

Problem Description

Your mean professor gave you a super hard homework and you need a break. In a future homework assgignment you'll re-write the PGN reader you wrote in HW1 but for now you'll write a few simple classes and an enum that you may find useful when you re-write your PGN reader.

Solution Description

Write the following classes and enums:

- `Color` – an enum with two values, `WHITE` and `BLACK` which represent the colors of the chess pieces.
- `Square` – a class to represent squares on a chess board. `Square` should be instantiable and have the following constructors and methods:
    - a public constructor `Square(char file, char rank)` which uses a `file` name such as `'a'` and `rank` name such as `'1'` to initialize instance variables that store the file and rank (as `char`s), and optionally the `String` name of the square that would be returned by `toString()` (see below). Ideally, this constructor should delegate to the other constructor, described below.
    - a public constructor `Square(String name)` which uses a square name such as `"a1"` to initialize the instance variables described in the other constructor above.
    - a public instance method `toString()` which returns a `String` representation of the square name, e.g., `"a1"`.
    - a properly written `equals` method that overrides the `equals` method from `java.lang.Object` and returns `true` for `Square` objects that have the same file and rank values, `false` otherwise.
- `Piece` – a class to represent chess pieces (Some people distinguish between pawns and pieces, we'll call pawns pieces as well.) `Piece` should be abstract and have the following constructors and methods:
    - a public constructor that takes a `Color` parameter and stores its value in an instance variable
    - a public `getColor()` instance method that returns the `Color` of the piece
    - a public abstract instance method `algebraicName()` which returns a `String` containing the algebraic name of the piece, e.g., `""` for pawns, or one of `"K", "Q", "B", "N", "R"`.
    - a public abstract instance method `fenName()` which returns a `String` containing the FEN name for the piece.
    - a public abstract instance method `movesFrom(Square square)` which returns a `Square[]` containg all the squares the piece could move to from `square` on a chess board containing only the piece.
- A subclass of `Piece` named `King` which overrides `Piece`'s abstract methods appropriately
- A subclass of `Piece` named `Queen` which overrides `Piece`'s abstract methods appropriately
- A subclass of `Piece` named `Bishop` which overrides `Piece`'s abstract methods appropriately
- A subclass of `Piece` named `Knight` which overrides `Piece`'s abstract methods appropriately
- A subclass of `Piece` named `Rook` which overrides `Piece`'s abstract methods appropriately
- A subclass of `Piece` named `Pawn` which overrides `Piece`'s abstract methods appropriately

For each class include Javadoc comments as described in the CS 1331 style guide.
We will test your classes by simply using them, for example:

```
Piece knight = new Knight(Color.BLACK);
assert knight.algebraicName().equals("N");
assert knight.fenName().equals("n");
Square[] attackedSquares = knight.movesFrom(new Square("f6"));
// test that attackedSquares contains e8, g8, etc.
Square a1 = new Square("a1");
Square otherA1 = new Square('a', '1');
Square h8 = new Square("h8");
assert a1.equals(otherA1);
assert !a1.equals(h8);
```

Grading

There are many bonus points available in this assignment.

- 5 points `Color` enum
- 5 points constructor `Square(char file, char rank)`
- 5 points constructor `Square(String name)`
- 5 points `Square`'s `toString()` method
- 5 points `Square`'s `equals` method
- 5 points `Piece`'s proper declaration and constructor
- 5 points `Piece`'s `getColor()` method
- 5 points `Piece`'s `algebraicName()` method
- 5 points `Piece`'s `fenName()` method
- 5 points `Piece`'s `movesFrom(Square square)` method
- 15 points for each subclass of `Piece` being instantiable and having correct implementations of the abstract methods from 'Piece`

Checkstyle deduction will be capped at 20 points for this homework.

Tips

- `Color` and 'Square` are easy and you need to get these right in order to get credit for other things.
- While most of `Piece` and the subclasses depend on `Color` and `Square`, we will try to grade most of `Piece` and its subclasses in isolation. Stub methods so that all classes compile and return values. That way if you, say, don't get any `movesFrom` methods working, you can still earn 110 points.