Homework 3 (due Tuesday Feb 21, 11 AM)

In this assignment, you will write classes to create a chess game with basic rules. The chess game consists of two teams, team 1 and team 2. It should not be necessary to know the rules of chess very well, but if you are not familiar with them you may want to look at http://en.wikipedia.org/wiki/Chess. Subsequent homework assignments will continue to build upon what was accomplished the week before.

BASIC RULES:

1. The 8×8 board is numbered from the bottom left.
2. Team 1 starts on the bottom side of the board and team 2 starts on the top side.
3. Team 1 moves first.
4. A team's piece is captured when an opposing piece moves to its position (taking over the piece's position).
5. The game is over when a team's King is captured.

CLASSES:

There are three main classes:

The first class is **`ChessGame`** and is given to you completed. **`ChessGame`** contains the methods for playing the game. The two methods you will use are **`play`** and **`replay`**. You can view the help for the methods to find out how to use them.

The second class is **`ChessBoard`**. This represents the state of the game: it contains all of the active pieces in a property called **`ActiveList,`** which is a cell array that is empty on initialization. The class definition m-file is given to you. You must determine the necessary properties for this class and fill out the following three methods:

The first method in **`ChessBoard`** is **`addPiece`**. This method will place a given piece on the chess board at a specific location. This method takes in a **`ChessPiece`** and adds it to the **`ChessBoard`**'s list of active pieces. It should check if the position (a property of the **`ChessPiece`**) is already occupied (using the **`checkPosition`** method, see below). If the position is occupied, return an error.

The second method in **`ChessBoard`** is **`removePiece`**. This method will remove a given piece off the chess board, and will be used when a piece has been captured. This method takes in a **`ChessPiece`** and removes it from the **`ChessBoard`**'s list of active pieces. If the **`ChessPiece`** is not in the list of active pieces, return an error.

The third method in **`ChessBoard`** is **`checkPosition`**. This method will check to see if a specific position on the chess board is occupied. This method takes in a position (a 1×2 double array) and checks the **`ChessBoard`**'s list of active pieces to see if the position is occupied. If it finds a piece with the given position, it returns true for **`occupied`** and the corresponding **`ChessPiece`** as the output **`piece`**. If it does not find any pieces in that position, it should return false for **`occupied`** and an empty array for **`piece`**.

The third class is **`ChessPiece`**. You will create this class from scratch. This is an abstract handle class (which will hold methods and properties common to each of the pieces in chess). This class will end up having three properties, one constructor, three concrete methods (one of them written for you), and two abstract methods. This week you will only add the following:

Properties:

1. **`Position`**, which is a 1×2 double array of the x and y coordinates.
2. **`Team`**, which should be either 1 or 2 for which team the piece belongs to.

Constructor:

The constructor should take three input arguments: a position (again as a 1×2 double array), a ChessBoard object, and a team number. After setting the properties accordingly, the constructor must call **`addPiece`** (for the given **`ChessPiece`**) to add itself to the ChessBoard object.

Methods:

The first abstract method is **`getSymbol`**. This method has one input argument (a **`ChessPiece`**) and one output argument: the symbol for that piece. This is how the method should be implemented in the subclass: the symbol should be a single uppercase character. For the Knight piece this will be 'N'. For all other pieces it is the first letter of the piece.

You must also create 6 child classes for **`ChessPiece`**: **`King`**, **`Rook`**, **`Knight`**, **`Pawn`**, **`Queen`**, and **`Bishop`**. The king needs a new constructor, which takes a fourth input: a **`ChessGame`**. Nothing needs to be done with this fourth input yet. All other classes do not have any additional properties or methods other than those inherited from **`ChessPiece`** (at least yet).

Submit the homework on bCourses. You should create a folder named **`lastname_firstname_hw3`**. Place all of your m-files in this folder and zip it. Please upload this single zip file.

Note: You can test what you have done so far by constructing a chess game (which will initialize the chess board).

```
>> game = ChessGame
```

For this assignment you do not need to have a working play or replay function. These parts will be implemented in later assignments.

Things we will check:

1. create ChessGame
2. create ChessBoard
3. verify ActiveList property of ChessBoard
4. verify addPiece method of Chessboard
5. verify checkPosition method of ChessBoard
6. verify removePiece method of Chessboard
7. verify ChessPiece as abstract
8. verify ChessPiece subclasses are handles
9. verify Position property of ChessPiece
10. verify Team property of ChessPiece
11. create Rook, Knight, Bishop, King, Queen Pawn
12. verify each piece is a ChessPiece
13. verify getSymbol method for each piece