**TestNotes/pset2/Chessboard.java**

```java
// Chess.java

/**
 * PSET #6
 * This program simulates a chessboard with different pieces (Bishop, Knight, and King).
 * It allows the user to select a piece type and displays the chessboard with valid moves for that piece.
 * It is an extension for the Chessboard program from Unit 5.
 *
 * For the purpose of testing King.java
 *
 * @author Kulijt Takhar
 * @version October 8, 2023
 */

import java.util.Scanner;

// Abstract class representing a chess piece

abstract class Piece {
    // Instance variables presenting the piece's row and column
    protected int pieceRow;
    protected int pieceColumn;

    // Constructor
    public Piece() {
        // Initialize the piece's row and column
    }

    // Abstract method to check if the piece can attack a location
    abstract boolean attackingThisLocation(int row, int col);

    // Method to check if a position is on the chessboard
    public boolean positionOnBoard(int row, int col) {
        return row >= 1 && row <= 8 && col >= 1 && col <= 8;
    }

    // Getter method for row
    public int getRow() {
        return pieceRow;
    }

    // Getter method for column
    public int getCol() {
        return pieceColumn;
    }

    // Method to determine position on the chessboard
    public void placeOnChessBoard() {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Type the ROW where your chess piece is located: ");
        pieceRow = keyboard.nextInt();
        System.out.print("Type the COLUMN where your chess piece is located: ");
```

```
53          pieceColumn = keyboard.nextInt();
54          if (pieceRow < 1 || pieceRow > 8 || pieceColumn < 1 || pieceColumn > 8) {
55              System.out.print("Invalid input, but I'll try anyway");
56          }
57          keyboard.nextLine();
58      }
59  }
60
61  // Bishop class extends Piece
62  class Bishop extends Piece {
63      boolean attackingThisLocation(int indexRow, int indexColumn) {
64          int columnDiff = pieceColumn - indexColumn;
65          int rowDiff = pieceRow - indexRow;
66
67          return (columnDiff + rowDiff == 0) || (columnDiff == rowDiff);
68      }
69  }
70
71  // Knight class extends Piece
72  class Knight extends Piece {
73      boolean attackingThisLocation(int indexRow, int indexColumn) {
74          int columnDiff = pieceColumn - indexColumn;
75          int rowDiff = pieceRow - indexRow;
76
77          return (columnDiff * columnDiff + rowDiff * rowDiff == 5) || (columnDiff == 0
    && rowDiff == 0);
78      }
79  }
80
81  // Chess class to test the Piece class and its subclasses, specifically the King
    class.
82  public class Chessboard {
83      public static void main(String[] args) {
84          Piece p;
85          Scanner keyboard = new Scanner(System.in);
86          System.out.print("Would you like to play with a Bishop, Knight, or King? ");
87          String answer = keyboard.nextLine();
88
89            if (answer.equalsIgnoreCase("Bishop")) {
90              p = new Bishop();
91          } else if (answer.equalsIgnoreCase("Knight") || answer.equalsIgnoreCase("N"))
    {
92              p = new Knight();
93          } else if (answer.equalsIgnoreCase("King") || answer.equalsIgnoreCase("K")) {
94              p = new King();
95          } else {
96              System.out.println("Invalid choice. Exiting.");
97              return;
98          }
99
100         p.placeOnChessBoard(); // Place the selected piece on the chessboard
101
102         System.out.println("\n  1 2 3 4 5 6 7 8"); // Number the columns
103
104         for (int indexRow = 1; indexRow <= 8; indexRow++) {
105             System.out.print(indexRow); // Number the rows
106             for (int indexColumn = 1; indexColumn <= 8; indexColumn++) {
```

```
107                 if (p.attackingThisLocation(indexRow, indexColumn)) {
108                     System.out.print(" *"); // Mark valid moves with an asterisk
109                 } else if ((indexColumn + indexRow) % 2 == 0) {
110                     System.out.print(" b"); // Display black squares
111                 } else {
112                     System.out.print(" w"); // Display white squares
113                 }
114             }
115             System.out.println();
116         }
117         System.out.println();
118     }
119 }
120
```