Lab 39
Caelan Bratland, Havin Lim
12/05/22
Sources: none
We confirm that the above list of sources is complete AND that we have not talked to anyone else (e.g., CSC 207 students) about the solution to this problem.

checkBraille

```
  /**
   * recursive method for finding the node in the path defined by braille_string.
   * @param cur instance of Node class to begin our search.
   * @param brailleString a string of 6 0's and 1's representing braille cells.
   * @return TreeTraverser, containing:
   *     TreeTraverser.current: the last visited node
   *     TreeTraverser.brailleString: a string of the remaining characters in the string
   */
  private TreeTraverser checkBraille(Node cur, String brailleString) {
        if (brailleString.length() == 0) return new TreeTraverser(cur, brailleString);

        char c = brailleString.charAt(0);
        switch (c) {
        case '0': // go left
                if (cur.left == null) break;
                return checkBraille(cur.left, brailleString.substring(1));
        case '1': // go right
                if (cur.right == null) break;
                return checkBraille(cur.right, brailleString.substring(1));
        }

        return new TreeTraverser(cur, brailleString);
  }
```

addValue

```
  private void addValue(Node cur, String brailleString, char val) {
        if (brailleString.length() == 0) {
                cur.data = val;
                return;
        }
        char c = brailleString.charAt(0);
        brailleString = brailleString.substring(1);

        switch (c) {
        case '0': // go left
            if (cur.left == null) {
                cur.left = new Node('\0');
            }
            addValue(cur.left, brailleString, val);
            break;
        case '1': // go right
            if (cur.right == null) {
                cur.right = new Node('\0');
            }
            addValue(cur.right, brailleString, val);
            break;
        }
    }
```

writeValues

```
    /**
     * Writes all characters and their braille representation to file named filename.
     * @param filePath string; full path of file to write to.
     * @throws IOException
     */
    public void writeValues(String filePath) throws IOException {
        File outputFile = new File(filePath);
        FileWriter fw = new FileWriter(outputFile);
        String start = "";
        writeValues(root, fw, start);
        fw.close();
    }
    private void writeValues(Node cur, FileWriter fw, String path) throws IOException {
        if (path.length() == 6) {
            fw.write(path + " " + cur.data + "\n");
        } else {
            if (cur.left != null) writeValues(cur.left, fw, path + "0");
            if (cur.right != null) writeValues(cur.right, fw, path + "1");
        }
    }
```

interactiveTranslator

```
   public static void interactiveTranslator(BrailleTree brailleTree, String debug) throws IOException {
		while(true) {
			System.out.println("Give a braille sequence of 0's and 1's to translate.");
			String input = getInput();

			boolean validInput = true;
			if (input.length() % 6 != 0) {
				validInput = false;
			} else {
				for (char c : input.toCharArray()) {
					if (c != '1' && c != '0') {
						validInput = false;
						break;
					}
				}
			}

			if (validInput) {
				String[] characters = input.split("(?<=\\G.{6})");
				String result = "";
				for (String c : characters) {
					result += brailleTree.checkBraille(c).current.data;
				}
				System.out.println("Braille Translation: " + result);
			} else {
				System.out.println("Invalid input.");
			}

			if (isYes(debug)) {
				brailleTree.print();
			}

			//check if continuing
			System.out.print("Enter another sequence? ");
			String repeat = getInput();

			if (!isYes(repeat)) {
```

```
                break;
            }
        }
    }
```

traversal.txt

```
001111 #
010100 i
010110 j
010111 w
011100 s
011110 t
100000 a
100010 e
100100 c
100110 d
101000 k
101001 u
101010 o
101011 z
101100 m
101101 x
101110 n
101111 y
110000 b
110010 h
110100 f
110110 g
111000 l
111001 v
111010 r
111100 p
111110 q
```

Output

```
Print tree as you go (yes/no)?
no
Give a braille sequence of 0's and 1's to translate.
100000
Braille Translation: a
```

```
Enter another sequence? yes
Give a braille sequence of 0's and 1's to translate.
110010100010111000111000101010
Braille Translation: hello
Enter another sequence? yes
Give a braille sequence of 0's and 1's to translate.
11001010001011100011100010101a
Invalid input.
Enter another sequence? yes
Give a braille sequence of 0's and 1's to translate.
a
Invalid input.
Enter another sequence? no
```