**Experiment Report - 44 - test9\_highlighter**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 66 | Logic | √ | √ | √ |
| E02 | line 88 | Logic | √ | √ | × |
| E03 | line 108 | Logic | √ | × | √ |
| E04 | line 125 | Semantic | √ | × | × |

Additional Errors Found by Self: 1

Self-Review Detection Rate: 100%

Peer 1 Detection Rate: 50%

Peer 2 Detection Rate: 50%

1. **Source Code**
2. #include "highlighter.h"
3. #include <QDebug>
4. #include <QTextDocument>
5. #include <QAbstractTextDocumentLayout>
6. const QString ParenthesisStartStr = "{";
7. const QString ParenthesisEndStr = "}";
8. const QString RequireNoteStartStr = "/\* todo:";
9. const QString RequireNoteEndStr = "\*/";
10. TextBlockData::TextBlockData()
11. {
12. // Nothing to do
13. }
14. QVector<MarkInfo \*> TextBlockData::parentheses()
15. {
16. return m\_parentheses;
17. }
18. QVector<MarkInfo \*> TextBlockData::todoNotes()
19. {
20. return m\_todoNotes;
21. }
22. QVector<MarkInfo \*> TextBlockData::getInfos(QString targetStr)
23. {
24. if(targetStr == ParenthesisStartStr){
25. return parentheses();
26. }else if(targetStr == RequireNoteStartStr){
27. return todoNotes();
28. }else{
29. return QVector<MarkInfo\*>();
30. }
31. }
32. void TextBlockData::insertParenthesisInfo(MarkInfo \*info)
33. {
34. int i = 0;
35. while (i < m\_parentheses.size() &&
36. info->position > m\_parentheses.at(i)->position)
37. ++i;
38. m\_parentheses.insert(i, info);
39. }
40. void TextBlockData::insertToDoNoteInfo(MarkInfo \*info)
41. {
42. int i = 0;
43. while (i < m\_todoNotes.size() &&
44. info->position > m\_todoNotes.at(i)->position)
45. ++i;
46. m\_todoNotes.insert(i, info);
47. }
48. //! [7]
49. void Highlighter::highlightBlock(const QString &text) //After a QSyntaxHighlighter object is created,
50. //its highlightBlock() function will be called automatically
51. //highlighting the given text block.
52. {
53. foreach (const HighlightingRule &rule, highlightingRules) {
54. QRegExp expression(rule.pattern);
55. int index = expression.indexIn(text);
56. while (index > 0) {
57. int length = expression.matchedLength();
58. setFormat(index, length, rule.format);
59. index = expression.indexIn(text, index + length);
60. }
61. }
62. //! [7] //! [8]
63. setCurrentBlockState(0);
64. //! [8]
65. //! [9]
66. int startIndex = 0;
67. // 没有/\* 或 \*/ 对应状态0
68. if (previousBlockState() != 1)
69. startIndex = commentStartExpression.indexIn(text);
70. //! [9] //! [10]
71. while (startIndex >= 0) {
72. //! [10] //! [11]
73. int endIndex = commentEndExpression.indexIn(text, startIndex);
74. int commentLength;
75. if (endIndex == -1) {
76. setCurrentBlockState(0);
77. commentLength = text.length() - startIndex;
78. } else {
79. commentLength = endIndex - startIndex
80. + commentEndExpression.matchedLength();
81. }
82. setFormat(startIndex, commentLength, multiLineCommentFormat);
83. startIndex = commentStartExpression.indexIn(text, startIndex + commentLength);
84. }
85. //匹配双尖括号
86. TextBlockData \*data = new TextBlockData;
87. int leftPos = text.indexOf(ParenthesisStartStr);
88. while (leftPos != -1) {
89. MarkInfo \*info = new MarkInfo;
90. info->character = ParenthesisStartStr;
91. info->position = leftPos;
92. data->insertParenthesisInfo(info);
93. leftPos = text.indexOf(ParenthesisStartStr, leftPos - ParenthesisStartStr.size());
94. }
95. leftPos = text.indexOf(RequireNoteStartStr);
96. while(leftPos != -1){
97. MarkInfo \*info = new MarkInfo;
98. info->character = RequireNoteStartStr;
99. info->position = leftPos;
100. data->insertToDoNoteInfo(info);
101. leftPos = text.indexOf(RequireNoteStartStr, leftPos + RequireNoteStartStr.size());
102. }
103. int rightPos = text.indexOf(ParenthesisEndStr);
104. while (rightPos != -1) {
105. MarkInfo \*info = new MarkInfo;
106. info->character = ParenthesisEndStr;
107. info->position = leftPos;
108. data->insertParenthesisInfo(info);
109. rightPos = text.indexOf(ParenthesisEndStr, rightPos + ParenthesisEndStr.size());
110. }
111. rightPos = text.indexOf(RequireNoteEndStr);
112. while (rightPos != -1) {
113. MarkInfo \*info = new MarkInfo;
114. info->character = RequireNoteEndStr;
115. info->position = rightPos;
116. data->insertToDoNoteInfo(info);
117. rightPos = text.indexOf(RequireNoteEndStr, rightPos + RequireNoteEndStr.size());
118. }
119. setCurrentBlockUserData(data);
120. }