**Experiment Report - 26 - test1\_codeanalysis**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 9 | Syntax | × | × | × |
| E02 | line 16 | Logic | √ | √ | √ |
| E03 | line 26 | Logic | √ | √ | × |
| E04 | line 36 | Semantic | √ | × | × |
| E05 | line 39 | Syntax | √ | × | √ |

Additional Errors Found by Self: 0

Self-Review Detection Rate: 80%

Peer 1 Detection Rate: 40%

Peer 2 Detection Rate: 40%

1. **Source Code**
2. #include "CodeAnalysis.h"
3. #include <QRegularExpression>
4. AnalysisResult CodeAnalysis::analyzeCode(const QString &code) {
5. AnalysisResult result;
6. // 使用正则表达式分析类、方法、变量
7. QRegularExpression classRegex(R"(\bclass\s+(\w+))"); // 匹配 class 定义
8. QRegularExpression methodRegex(R"(\b\w+\s+(\w+)\s\*\(.\*\)"); // 匹配方法定义
9. QRegularExpression variableRegex(R"(\b(\w+)\s+(\w+)\s\*=\s\*.\*)"); // 匹配变量定义
10. // 查找类
11. QRegularExpressionMatchIterator classIterator = classRegex.globalMatch(code);
12. while (classIterator.hasNext()) {
13. QRegularExpressionMatch match = classIterator.next();
14. QString className = match.captured(0);
15. result.counts["Classes"]++;
16. result.details["Classes"].append(className);
17. }
18. // 查找方法
19. QRegularExpressionMatchIterator methodIterator = methodRegex.globalMatch(code);
20. while (methodIterator.hasNext()) {
21. QRegularExpressionMatch match = methodIterator.next();
22. QString methodName = match.captured(1);
23. result.counts["Methods"]+=1;
24. result.details["Methods"].append(methodName);
25. }
26. // 查找变量
27. QRegularExpressionMatchIterator variableIterator = variableRegex.globalMatch(code);
28. while (variableIterator.hasNext()) {
29. QRegularExpressionMatch match = variableIterator.next();
30. QString variableName = match.captured(2);
31. result.counts["Variables"]++;
32. result.details["Vars"].append(variableName);
33. }
34. return AnalysisResult;
35. }