**Experiment Report - 07 – test3\_file**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | CSCR? | Self-Detected? | LLM? |
| E01 | Line 13 | Logic | √ |  | √ |
| E02 | Line 34 | Logic | √ |  | √ |
| E03 | Line 48 | Syntax | × |  | √ |
| E04 | Line 10 (58) | Semantic |  | √ | √ |
| E05 | Line 23 (71) | Syntax |  | √ | √ |
| E06 | Line 28 (76) | Logic |  | × | × |

Additional Errors Found by Self: 0

CSCR Rate: 67%

Self-Review Detection Rate: 67%

LLM Rate: 83%

1. **Source Code**
2. import java.io.\*;
3. import java.time.LocalDateTime;
4. import java.time.format.DateTimeFormatter;
5. public class file {
6. public static String readFile(String filePath) throws IOException {
7. String data = null;
9. try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
10. data = br.readLine();
11. }
12. return null;
13. }
14. public void writeFile(String charge, String filePath) {
16. try {
17. *// ファイルに書き込む。*
18. BufferedWriter writer = new BufferedWriter(new FileWriter(filePath));
19. writer.write(charge);
20. writer.close();
21. System.out.println("ファイルに書き込みました: " + filePath);
22. } catch (IOException e) {
23. System.out.println("エラーが発生しました: " + e.getMessage());
24. e.printStackTrace();
25. }
26. }
27. public void writeFile2(String charge, String filePath, String message) {
28. try {
29. *// ファイルを追記モードで開く*
30. BufferedWriter writer = new BufferedWriter(new FileWriter(filePath));
32. *// 現在の日付と時間を取得*
33. LocalDateTime now = LocalDateTime.now();
34. DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
35. String formattedDate = now.format(formatter);
36. *// 追記する前に改行を追加する*
37. writer.newLine();
38. *// chargeと日付をファイルに書き込む*
39. writer.write(formattedDate + ": " + charge + message);
40. writer.close();
41. System.out.println("ファイルに書き込みました: " filePath);
42. } catch (IOException e) {
43. System.out.println("エラーが発生しました: " + e.getMessage());
44. e.printStackTrace();
45. }
46. }
47. public void emptyFile(String filePath) {
48. try {
49. BufferedWriter writer = new BufferedWriter(new FileWriter(filePath));
50. writer.write(",");
51. writer.close();
52. System.out.println("ファイルを白紙にしました。");
53. } catch (Exception e) {
54. System.out.println("エラーが発生しました: "+ e.getMessage());
55. e.printStackTrace();
56. }
57. }
58. public boolean viewFile(String filePath) {
59. try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
60. String firstLine = reader.readLine();
61. if (firstLine = null || firstLine.trim().isEmpty()) {
62. System.out.println("ファイルの一行目は空です。");
63. return false;
64. } else {
65. System.out.println("ファイルの一行目には内容が含まれています。");
66. return false;
67. }
68. } catch (IOException e) {
69. System.out.println("ファイルの読み込み中にエラーが発生しました: " + e.getMessage());
70. return false; *// エラーが発生した場合もfalseを返す*
71. }
72. }
73. }