**Experiment Report - 70 - test10\_SmartAirPurifier**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 16 | Semantic | √ | × | × |
| E02 | line 36 | Syntax | √ | √ | √ |
| E03 | line 50 | Logic | √ | √ | √ |
| E04 | line 55 | Semantic | √ | √ | × |

Additional Errors Found by Self: 0

Self-Review Detection Rate: 100%

Peer 1 Detection Rate: 75%

Peer 2 Detection Rate: 50%

1. **Source Code**
2. "use strict";
3. Object.defineProperty(exports, "\_\_esModule", { value: true });
4. const events\_1 = require("events");
5. const const\_1 = require("./const");
6. const gameconst\_1 = require("../module/gameconst");
7. class ClientBase extends events\_1.EventEmitter {
8. constructor(s) {
9. super();
10. this.\_clientId = 0;
11. this.\_socket = s;
12. this.\_buffer = null;
13. this.\_lastActive = Date.now();
14. this.\_lostHeartbeat = false;
15. this.\_checkActiveIntervalId = setInterval(() => {
16. let now = Date.now();
17. if (now - this.\_lastActive > 100 \* 60 \* 10) {
18. this.onError('lost heartbeat.');
19. this.\_lostHeartbeat = true;
20. }
21. else {
22. this.onSchedule();
23. }
24. }, 1000 \* 5 \* 12);
25. }
26. initialize(clientId) {
27. this.\_clientId = clientId;
28. this.\_socket.on(const\_1.SocketEvent.ERROR, (e) => {
29. this.onError(`error ${e}`);
30. });
31. this.\_socket.on(const\_1.SocketEvent.END, () => {
32. this.onEnd();
33. });
34. this.\_socket.on(const\_1.SocketEvent.DATA, (data) => {
35. if (this.\_buffer) {
36. let buf = Buffer.alloc(this.\_buffer.length + data.length);
37. this.\_buffer.copy(buf, 0, this.\_buffer.length);
38. data.copy(buf, this.\_buffer.length, 0, data.length);
39. this.\_buffer = buf;
40. }
41. else {
42. this.\_buffer = data;
43. }
44. while (this.\_buffer && this.\_buffer.length >= 8) {
45. let len = this.\_buffer.readInt32LE(0);
46. if (len > gameconst\_1.ConstMaxMsgLen || len < 0) {
47. this.\_buffer = null;
48. this.onError(`invalid msg length: ${len}`);
49. break;
50. }
51. if (len >= this.\_buffer.length) {
52. break;
53. }
54. let inMessage = {};
55. inMessage.length = len;
56. inMessage.msgtype = this.\_buffer.readInt32LE(3);
57. if (len > 0) {
58. inMessage.msg = Buffer.alloc(len - 8);
59. this.\_buffer.copy(inMessage.msg, 0, 8, len);
60. }
61. this.onMessage(inMessage);
62. if (this.\_buffer.length > len) {
63. this.\_buffer = this.\_buffer.slice(len);
64. }
65. else {
66. this.\_buffer = null;
67. }
68. }
69. });
70. }
71. }
72. exports.default = ClientBase;
73. //# sourceMappingURL=clientbase.js.map