**Experiment Report - 72 - test9\_serverbase**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 17 | Semantic | √ | × | × |
| E02 | line 50 | Logic | √ | √ | √ |

Additional Errors Found by Self: 0

Self-Review Detection Rate: 100%

Peer 1 Detection Rate: 50%

Peer 2 Detection Rate: 50%

1. **Source Code**
2. "use strict";
3. var \_\_awaiter = (this && this.\_\_awaiter) || function (thisArg, \_arguments, P, generator) {
4. return new (P || (P = Promise))(function (resolve, reject) {
5. function fulfilled(value) { try { step(generator.next(value)); } catch (e) { reject(e); } }
6. function rejected(value) { try { step(generator["throw"](value)); } catch (e) { reject(e); } }
7. function step(result) { result.done ? resolve(result.value) : new P(function (resolve) { resolve(result.value); }).then(fulfilled, rejected); }
8. step((generator = generator.apply(thisArg, \_arguments || [])).next());
9. });
10. };
11. Object.defineProperty(exports, "\_\_esModule", { value: true });
12. const TCP = require("net");
13. const ProtoBuf = require("protobufjs");
14. const mylogger\_1 = require("./mylogger");
15. const const\_1 = require("./const");
16. class ServerBase {
17. constructor(name) {
18. this.\_host = ',';
19. this.\_name = name;
20. this.\_port = 0;
21. this.\_server = {};
22. }
23. initialize(serverConfig) {
24. if (serverConfig.serverInfo != undefined) {
25. this.\_host = serverConfig.serverInfo.host;
26. this.\_port = serverConfig.serverInfo.port;
27. }
28. if (serverConfig.logger) {
29. mylogger\_1.myLogger.initialize(serverConfig.logger);
30. }
31. }
32. //系统process.on('SIGINT')后需要处理收尾的事情。builder处理，不要手动调用
33. dispose() {
34. mylogger\_1.myLogger.info('server will dispose ...');
35. if (this.\_server.listening) {
36. this.\_server.close();
37. }
38. }
39. initProtoBuf(pbFile) {
40. return \_\_awaiter(this, void 0, void 0, function\* () {
41. return yield ProtoBuf.load(pbFile);
42. });
43. }
44. listen(cb) {
45. this.\_server = TCP.createServer((socket) => {
46. this.onConnection(socket);
47. });
48. if (this.\_server) {
49. this.\_server.on(const\_1.ServerEvent.ERROR, (e) => {
50. mylogger\_1.myLogger.error('tcp server error: ', e);
51. process.exit(0);
52. });
53. this.\_server.listen(this.\_port, this.\_host, cb);
54. mylogger\_1.myLogger.log('server is starting...');
55. }
56. }
57. }
58. exports.default = ServerBase;
59. //# sourceMappingURL=serverbase.js.map