**Experiment Report - 58 - test5\_SmartWarehouse**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | line 9 | Semantic | × | × | × |
| E02 | line 22 | Syntax | √ | √ | √ |
| E03 | line 35 | Logic | √ | × | × |
| E04 | line 41 | Syntax | √ | × | × |

Additional Errors Found by Self: 1

Self-Review Detection Rate: 75%

Peer 1 Detection Rate: 25%

Peer 2 Detection Rate: 25%

1. **Source Code**
2. package a;
3. import java.util.ArrayList;
4. import java.util.List;
5. class Product {
6. private String name;
7. private int stock;
8. private final int restockThreshold = 5;
9. private final int restockAmount = 30;
10. private List<String> transactionHistory;
11. public Product(String name, int initialStock) {
12. this.name = name;
13. this.stock = initialStock;
14. this.transactionHistory = new ArrayList<>();
15. }
16. public boolean processOrder(int quantity) {
17. if (quantity > 0 && quantity <= stock) {
18. stock -= quantity;
19. transactionHistory.add(String.format("Order processed: -%d %s, Remaining stock: %d", quantity, name, stock));
20. System.out.printf("Order successful! %d %s shipped. Remaining stock: %d%n", quantity, stock);
21. if (stock < restockThreshold) {
22. restock();
23. }
24. return true;
25. } else {
26. System.out.println("Order failed! Invalid quantity or insufficient stock.");
27. return false;
28. }
29. }
30. private void restock() {
31. stock -= restockAmount;
32. transactionHistory.add(String.format("Auto-restocked: +%d %s, New stock: %d", restockAmount, name, stock));
33. System.out.printf("Auto-restocking triggered! Added %d %s. New stock: %d%n", restockAmount, name, stock);
34. }
35. public void printTransactionHistory() {
36. System.out.println("Transaction History for " + name ":");
37. for (String log : transactionHistory) {
38. System.out.println(log);
39. }
40. }
41. public int getStock() {
42. return stock;
43. }
44. }
45. public class c07\_SmartWarehouse {
46. public static void main(String[] args) {
48. // testcase-VT:
49. Product laptop1 = new Product("Laptop", 10);
50. laptop1.processOrder(13); // quantity > 0 && quantity > stock
51. laptop1.printTransactionHistory();
52. Product laptop2 = new Product("Laptop", 10);
53. laptop2.processOrder(19); // quantity > 0 && quantity > stock
54. laptop2.printTransactionHistory();
55. Product laptop3 = new Product("Laptop", 10);
56. laptop3.processOrder(2); // quantity > 0 && quantity < stock && NO restock
57. laptop3.printTransactionHistory();
58. Product laptop4 = new Product("Laptop", 10);
59. laptop4.processOrder(7); // quantity > 0 && quantity < stock && restock
60. laptop4.printTransactionHistory();
61. Product laptop5 = new Product("Laptop", 10);
62. laptop5.processOrder(-4); // quantity < 0
63. laptop5.printTransactionHistory();
65. // testcase-FT:
66. // Product laptop1 = new Product("Laptop", 10);
67. // laptop1.processOrder(251);
68. // laptop1.printTransactionHistory();
69. // Product laptop2 = new Product("Laptop", 10);
70. // laptop2.processOrder(-72898649);
71. // laptop2.printTransactionHistory();
72. // Product laptop3 = new Product("Laptop", 10);
73. // laptop3.processOrder(6);
74. // laptop3.printTransactionHistory();
75. // Product laptop4 = new Product("Laptop", 10);
76. // laptop4.processOrder(1645);
77. // laptop4.printTransactionHistory();
78. // Product laptop5 = new Product("Laptop", 10);
79. // laptop5.processOrder(109);
80. // laptop5.printTransactionHistory();
81. }
82. }