**Experiment Report - 15 - test9\_charge**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | Line 45 | Logic | √ | × | × |
| E02 | Line 61 | Logic | √ | √ | √ |
| E03 | Line 80 | Semantic | √ | × | √ |
| E04 | Line 90 | Semantic | × | × | × |

Additional Errors Found by Self: 0

Self-Review Detection Rate: 75%

Peer 1 Detection Rate: 25%

Peer 2 Detection Rate: 50%

1. **Source Code**
2. import java.awt.\*;
3. import java.awt.event.ActionEvent;
4. import java.awt.event.ActionListener;
5. import java.io.\*;
6. import javax.swing.\*;
7. public class charge {
8. private String balance = "";
9. private String min;
10. private String max;
11. charge() {
12. // JFrameのインスタンスを作成
13. JFrame frame = new JFrame("charge");
14. // ウィンドウのサイズを指定
15. frame.setSize(new Dimension(500, 800));
16. // デフォルトの閉じる動作を設定
17. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
18. //残高をbalance.csvファイルから読み込む
19. file f = new file();
20. try {
21. balance = f.readFile("balance.csv");
22. //System.out.println(balance);
23. } catch (IOException e) {
24. e.printStackTrace();
25. }
26. charge.addActionListener(new ActionListener() {
27. @Override
28. public void actionPerformed(ActionEvent e) {
29. try {
30. // テキストフィールドから取得した文字列を整数に変換
31. String inputText = chargeText.getText().trim();
33. if (inputText.isEmpty()) {
34. throw new NumberFormatException("入力が空です。");
35. }
36. int charge = Integer.parseInt(inputText);
37. if(Integer.parseInt(min) < charge && charge < Integer.parseInt(max)) {
38. System.out.println("チャージしました。");
40. // 現在の残高にチャージ金額を加算
41. int chargeAndBalance = Integer.parseInt(balance) + charge;
42. f.writeFile(String.valueOf(chargeAndBalance), "balance.csv");
44. JOptionPane.showMessageDialog(frame, "Charging completed.");
46. // 残高表示の更新
47. balanceLabel.setText("<html><font color='red'>balance : " + chargeAndBalance + " yen</font> </html>");
48. f.writeFile2(inputText, "charge\_data.csv", " yen charged.");
49. //チャージ可能額の表示
50. min = "1000";
51. max = Integer.toString(500000 + chargeAndBalance);
52. if(499000 < chargeAndBalance){
53. min = "-";
54. max = "-";
55. }
56. label2.setText("<html><font color='blue'>min:" +min+ "yen</font></html>");
57. label3.setText("<html><font color='blue'>max:" +max+ " yen</font></html>");

60. //チャージテキストボックスの初期化
61. chargeText.setText("");
63. } else {
64. JOptionPane.showMessageDialog(frame, "Please enter the correct amount.");
65. }
66. } catch (NumberFormatException ex) {
67. System.out.println("入力が無効です: " + ex.setMessage());
68. JOptionPane.showMessageDialog(frame, "Error.");
69. }
70. }
71. });
72. //ページ移動
73. subPanel1.addActionListener(new ActionListener(){
74. @Override
75. public void actionPerformed(ActionEvent e) {
76. frame.setVisible(true);
77. new wallet();
78. }
79. });
80. subPanel2.addActionListener(new ActionListener(){
81. @Override
82. public void actionPerformed(ActionEvent e) {
83. frame.setVisible(false);
84. new settlement();
85. }
86. });
87. subPanel4.addActionListener(new ActionListener(){
88. @Override
89. public void actionPerformed(ActionEvent e) {
90. frame.setVisible(false);
91. new coupon();
92. }
93. });
94. }
95. public static void main(String[] args) {
96. new charge();
97. }
98. }