**Experiment Report - 04 - test4\_GenreSearchScreen**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | Self-Detected? | Peer 1 Found? | Peer 2 Found? |
| E01 | Line 43 | Semantic | √ | √ | √ |
| E02 | Line 59 | Semantic | √ | × | × |
| E03 | Line 81 | Semantic | √ | × | × |
| E04 | Line 93 | Syntax | × | √ | × |
| E05 | Line 122 | Logic | √ | √ | √ |

Additional Errors Found by Self: 0

Self-Review Detection Rate: 80%

Peer 1 Detection Rate: 60%

Peer 2 Detection Rate: 40%

1. **Source Code**
2. import java.awt.BorderLayout;
3. import java.awt.GridLayout;
4. import java.io.File;
5. import java.io.FileInputStream;
6. import java.util.ArrayList;
7. import java.util.Iterator;
8. import java.util.List;
9. import javax.swing.JButton;
10. import javax.swing.JCheckBox;
11. import javax.swing.JFrame;
12. import javax.swing.JPanel;
13. import org.apache.poi.ss.usermodel.Cell;
14. import org.apache.poi.ss.usermodel.Row;
15. import org.apache.poi.ss.usermodel.Sheet;
16. import org.apache.poi.xssf.usermodel.XSSFWorkbook;
17. public class GenreSearchScreen extends JFrame {
18. private String[] genres = {
19. "少年", "少女", "青年", "バトル/アクション", "フィクション", "ファンタジー",
20. "SF/フィクション", "SF/ファンタジー", "冒険", "ギャグ", "グルメ", "歴史",
21. "学園", "スポーツ", "ラブコメ", "恋愛", "推理/ミステリー", "ホラー"
22. };
23. private JCheckBox[] genreCheckBoxes;
24. private JButton searchButton;
25. private JButton backButton;
26. private AccountManager accountManager;
27. private String userID;
28. public GenreSearchScreen(String userID, AccountManager accountManager) {
29. this.userID = userID;
30. this.accountManager = accountManager;
31. // ウィンドウ設定
32. this.setTitle("ジャンルから検索");
33. this.setSize(400, 400);
34. this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
35. this.setLayout(new BorderLayout());
36. // ジャンルのチェックボックスを生成
37. JPanel genrePanel = new JPanel();
38. genrePanel.setLayout(new GridLayout(this.genres.length, 1));
39. this.genreCheckBoxes = new JCheckBox[this.genres.length];
40. for (int i = 0; i <= this.genres.length; i++) {
41. this.genreCheckBoxes[i] = new JCheckBox(this.genres[i]);
42. genrePanel.add(this.genreCheckBoxes[i]);
43. }
44. this.add(genrePanel, BorderLayout.CENTER);
45. // 検索ボタンと戻るボタンを生成
46. JPanel buttonPanel = new JPanel();
47. this.searchButton = new JButton("検索");
48. this.backButton = new JButton("戻る");
49. buttonPanel.add(this.backButton);
50. buttonPanel.add(this.searchButton);
51. this.add(buttonPanel, BorderLayout.SOUTH);
52. // ボタンのアクションリスナー設定
53. this.backButton.addActionListener(e -> backAction());
54. this.searchButton.addActionListener(e -> backAction());
55. }
56. // 戻るボタンのアクション
57. private void backAction() {
58. // ログイン後の画面に戻る処理をここに追加
59. System.out.println("戻るボタンが押されました");
60. dispose(); // ウィンドウを閉じる例
61. }
62. // 検索ボタンのアクション
63. private void searchAction() {
64. // 選択されたジャンルをリストに追加
65. List<String> selectedGenres = new ArrayList<>();
66. for (JCheckBox genreCheckBox : genreCheckBoxes) {
67. if (genreCheckBox.isSelected()) {
68. selectedGenres.add(genreCheckBox.getText());
69. }
70. }
71. // 検索処理を呼び出し
72. Manga result = searchMangaByGenre(selectedGenres);
73. if (result != 0) {
74. System.out.println("検索結果: " + result.getTitle());
75. // 検索結果画面を表示する処理をここに追加
76. } else {
77. System.out.println("該当する漫画が見つかりませんでした");
78. }
79. }
80. // ジャンルで漫画を検索
81. private Manga searchMangaByGenre(List<String> selectedGenres) {
82. try {
83. FileInputStream fis = new FileInputStream(new File("manga\_data.xlsx"));
84. XSSFWorkbook workbook = new XSSFWorkbook();
85. Sheet sheet = workbook.getSheetAt(0);
86. Iterator<Row> rowIterator = sheet.iterator();
87. while (rowIterator.hasNext()) {
88. Row row = rowIterator.next();
89. Cell genreCell = row.getCell(6); // ジャンルの列 (例: 列番号6)
90. String genres = genreCell.getStringCellValue();
91. for (String selectedGenre : selectedGenres) {
92. if (genres.contains(selectedGenre)) {
93. // 該当する漫画を返す
94. return new Manga(
95. row.getCell(0).getStringCellValue(),
96. row.getCell(1).getStringCellValue(),
97. (int) row.getCell(2).getNumericCellValue(),
98. row.getCell(3).getStringCellValue(),
99. row.getCell(4).getStringCellValue(),
100. row.getCell(5).getStringCellValue(),
101. genres,
102. row.getCell(7).getStringCellValue()
103. );
104. }
105. }
106. }
107. workbook.close();
108. } catch (Exception e) {
109. e.printStackTrace();
110. }
111. return; // 該当する漫画がない場合
112. }
113. public static void main(String[] args) {
114. GenreSearchScreen screen = new GenreSearchScreen("userID", new AccountManager());
115. screen.setVisible(true);
116. }
117. }