```java
import java.util.*;
import java.io.*;

public class Model {

    public String displayList(int maxCapacity, String facility) {
        // List of students read from file
        List<String> students = readStudentsFromFile("students.txt");

        // Constraints map read from file
        Map<String, Set<String>> constraints = readConstraintsFromFile("constraints.txt");

        // Calculate number of rooms needed
        int numStudents = students.size();
        int numRooms = (int) Math.ceil((double) numStudents / maxCapacity);

        // Allocate students to rooms respecting constraints
        List<List<String>> rooms = new ArrayList<>();
        List<String> remainingStudents = new ArrayList<>(students); // Copy of students list
        for (int i = 0; i < numRooms; i++) {
            List<String> room = new ArrayList<>();
            Iterator<String> iter = remainingStudents.iterator();
            while (iter.hasNext() && room.size() < maxCapacity) {
                String student = iter.next();
                if (!isInRoomWithConstraints(student, room, constraints)) {
                    room.add(student);
                    iter.remove(); // Remove from remaining students
                }
            }
            rooms.add(room);
        }

        // Build the result string
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < rooms.size(); i++) {
            result.append(""+facility+" ").append(i + 1).append(": ").append(rooms.get(i)).append("\n\n");
        }

        return result.toString();
    }

```

page1

```
42      // Read students from file
43      private static List<String> readStudentsFromFile(String fileName) {
44          List<String> students = new ArrayList<>();
45          try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
46              String line;
47              while ((line = reader.readLine()) != null) {
48                  line = line.trim();
49                  if (!line.isEmpty()) {
50                      students.add(line);
51                  }
52              }
53          } catch (IOException e) {
54              System.err.println("Error reading file: " + e.getMessage());
55          }
56          return students;
57      }
58
59      // Read constraints from file
60      private static Map<String, Set<String>> readConstraintsFromFile(String fileName) {
61          Map<String, Set<String>> constraints = new HashMap<>();
62          try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
63              String line;
64              while ((line = reader.readLine()) != null) {
65                  line = line.trim();
66                  if (!line.isEmpty()) {
67                      String[] studentsArray = line.split(",");
68                      if (studentsArray.length > 1) {
69                          String student1 = studentsArray[0].trim();
70                          String student2 = studentsArray[1].trim();
71                          constraints.computeIfAbsent(student1, k -> new HashSet<>()).add(student2);
72                          constraints.computeIfAbsent(student2, k -> new HashSet<>()).add(student1);
73                      }
74                  }
75              }
76          } catch (IOException e) {
77              System.err.println("Error reading file: " + e.getMessage());
78          }
79          return constraints;
80      }
81
82      // Check if adding student to room violates constraints
```

page2

```
83     private static boolean isInRoomWithConstraints(String student, List<String> room, Map<String, Set<String>> constraints) {
84         for (String s : room) {
85             if (constraints.containsKey(student) && constraints.get(student).contains(s)) {
86                 return true;
87             }
88         }
89         return false;
90     }
91 }
```

page3

```
 1 import javax.swing.*;
 2 import java.awt.*;
 3 import java.awt.event.*;
 4 import java.io.*;
 5 import java.util.*;
 6
 7 public class MainFrame extends JFrame {
 8     private JTextArea studentsTextArea;
 9     private JTextArea constraintsTextArea;
10     private JButton updateStudentsButton;
11     private JButton updateConstraintsButton;
12
13     // Components for Transportation tab
14     private JComboBox<Integer> transportationCapacityComboBox;
15     private JTextArea transportationListTextArea;
16     private JButton generateTransportListButton;
17     private JButton printTransportListButton;
18
19     // Components for Accommodation tab
20     private JComboBox<Integer> roomCapacityComboBox;
21     private JTextArea roomListTextArea;
22     private JButton generateRoomListButton;
23     private JButton printRoomListButton;
24
25     // Components for Search tab
26     private JTextField searchTextField;
27     private JButton searchButton;
28     private JTextArea searchResultsTextArea;
29
30     public MainFrame() {
31         setTitle("Student Management System");
32         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
33         setSize(600, 400);
34
35         // Create a tabbed pane
36         JTabbedPane tabbedPane = new JTabbedPane();
37
38         // Students tab
39         JPanel studentsPanel = new JPanel(new BorderLayout());
40
41         // Text areas for students and constraints
```

page4

```
42          studentsTextArea = new JTextArea(10, 30);
43          constraintsTextArea = new JTextArea(10, 30);
44
45          // Buttons to update text files
46          updateStudentsButton = new JButton("Update Students");
47          updateConstraintsButton = new JButton("Update Constraints");
48
49          // Read initial content from files
50          readFromFile("students.txt", studentsTextArea);
51          readFromFile("constraints.txt", constraintsTextArea);
52
53          // Add text areas to the panel
54          JPanel studentsTextPanel = new JPanel(new BorderLayout());
55          studentsTextPanel.add(new JScrollPane(studentsTextArea), BorderLayout.CENTER);
56          studentsTextPanel.add(updateStudentsButton, BorderLayout.SOUTH);
57
58          JPanel constraintsTextPanel = new JPanel(new BorderLayout());
59          constraintsTextPanel.add(new JScrollPane(constraintsTextArea), BorderLayout.CENTER);
60          constraintsTextPanel.add(updateConstraintsButton, BorderLayout.SOUTH);
61
62          studentsPanel.add(studentsTextPanel, BorderLayout.WEST);
63          studentsPanel.add(constraintsTextPanel, BorderLayout.EAST);
64
65          // Add the Students tab to the tabbed pane
66          tabbedPane.addTab("Students", studentsPanel);
67
68          // Transportation tab
69          JPanel transportationPanel = new JPanel(new BorderLayout());
70
71          // JComboBox for transportation capacity
72          JPanel transportationComboPanel = new JPanel();
73          JLabel transportationCapacityLabel = new JLabel("Transportation Capacity:");
74          transportationCapacityComboBox = new JComboBox<>();
75          // Add numbers from 10 to 40 to the combo box
76          for (int i = 10; i <= 40; i++) {
77              transportationCapacityComboBox.addItem(i);
78          }
79          transportationComboPanel.add(transportationCapacityLabel);
80          transportationComboPanel.add(transportationCapacityComboBox);
81
82          // JTextArea for transportation list
```

page5

```
        transportationListTextArea = new JTextArea(10, 30);
        JScrollPane transportationListScrollPane = new JScrollPane(transportationListTextArea);

        // Buttons for generating and printing transportation list
        JPanel transportationButtonPanel = new JPanel();
        generateTransportListButton = new JButton("Generate List");
        printTransportListButton = new JButton("Print List");
        transportationButtonPanel.add(generateTransportListButton);
        transportationButtonPanel.add(printTransportListButton);

        // Add components to transportation panel
        transportationPanel.add(transportationComboPanel, BorderLayout.NORTH);
        transportationPanel.add(transportationListScrollPane, BorderLayout.CENTER);
        transportationPanel.add(transportationButtonPanel, BorderLayout.SOUTH);

        // Add the Transportation tab to the tabbed pane
        tabbedPane.addTab("Transportation", transportationPanel);

        // Accommodation tab
        JPanel accommodationPanel = new JPanel(new BorderLayout());

        // JComboBox for room capacity
        JPanel roomComboPanel = new JPanel();
        JLabel roomCapacityLabel = new JLabel("Room Capacity:");
        roomCapacityComboBox = new JComboBox<>();
        // Add numbers from 2 to 6 to the combo box
        for (int i = 2; i <= 6; i++) {
            roomCapacityComboBox.addItem(i);
        }
        roomComboPanel.add(roomCapacityLabel);
        roomComboPanel.add(roomCapacityComboBox);

        // JTextArea for room list
        roomListTextArea = new JTextArea(10, 30);
        JScrollPane roomListScrollPane = new JScrollPane(roomListTextArea);

        // Buttons for generating and printing room list
        JPanel roomButtonPanel = new JPanel();
        generateRoomListButton = new JButton("Generate Rooms");
        printRoomListButton = new JButton("Print Rooms");
        roomButtonPanel.add(generateRoomListButton);
```

page6

```
124         roomButtonPanel.add(printRoomListButton);
125
126         // Add components to accommodation panel
127         accommodationPanel.add(roomComboPanel, BorderLayout.NORTH);
128         accommodationPanel.add(roomListScrollPane, BorderLayout.CENTER);
129         accommodationPanel.add(roomButtonPanel, BorderLayout.SOUTH);
130
131         // Add the Accommodation tab to the tabbed pane
132         tabbedPane.addTab("Accommodation", accommodationPanel);
133
134         // Search tab
135         JPanel searchPanel = new JPanel(new BorderLayout());
136
137         // Search field and button
138         JPanel searchInputPanel = new JPanel();
139         JLabel searchLabel = new JLabel("Search:");
140         searchTextField = new JTextField(20);
141         searchButton = new JButton("Search");
142         searchInputPanel.add(searchLabel);
143         searchInputPanel.add(searchTextField);
144         searchInputPanel.add(searchButton);
145
146         // Search results area
147         searchResultsTextArea = new JTextArea(10, 30);
148         JScrollPane searchResultsScrollPane = new JScrollPane(searchResultsTextArea);
149
150         // Add components to search panel
151         searchPanel.add(searchInputPanel, BorderLayout.NORTH);
152         searchPanel.add(searchResultsScrollPane, BorderLayout.CENTER);
153
154         // Add the Search tab to the tabbed pane
155         tabbedPane.addTab("Search", searchPanel);
156
157         // Add the tabbed pane to the main frame
158         add(tabbedPane, BorderLayout.CENTER);
159
160         // Add action listeners
161         updateStudentsButton.addActionListener(e -> updateFile("students.txt", studentsTextArea));
162         updateConstraintsButton.addActionListener(e -> updateFile("constraints.txt", constraintsTextArea));
163         generateTransportListButton.addActionListener(e -> generateTransportList());
164         printTransportListButton.addActionListener(e -> printTransportList());
```

page7

```
        generateRoomListButton.addActionListener(e -> generateRoomList());
        printRoomListButton.addActionListener(e -> printRoomList());
        searchButton.addActionListener(e -> performSearch());
    }

    private void readFromFile(String fileName, JTextArea textArea) {
        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            textArea.read(reader, null);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void updateFile(String fileName, JTextArea textArea) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
            textArea.write(writer);
            JOptionPane.showMessageDialog(this, "File updated successfully.");
        } catch (IOException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error updating file.");
        }
    }

    private void generateTransportList() {
        Model myModel = new Model();
        int capacity = (int) transportationCapacityComboBox.getSelectedItem();
        String list = myModel.displayList(capacity, "Bus");

        transportationListTextArea.setLineWrap(true);  // Enable line wrapping
        transportationListTextArea.setWrapStyleWord(true);

        transportationListTextArea.setText(list);
    }

    private void printTransportList() {
        // Print the transportation list
        String listToPrint = transportationListTextArea.getText();

        generateAndDisplayHTML(transportationListTextArea);
    }

```

page8

```
206     private void generateRoomList() {
207         Model myModel = new Model();
208         int capacity = (int) roomCapacityComboBox.getSelectedItem();
209         String list = myModel.displayList(capacity, "Room");
210 
211         roomListTextArea.setLineWrap(true);  // Enable line wrapping
212         roomListTextArea.setWrapStyleWord(true);
213 
214         roomListTextArea.setText(list);
215     }
216 
217     private void printRoomList() {
218         // Print the room list
219         String listToPrint = roomListTextArea.getText();
220 
221         generateAndDisplayHTML(roomListTextArea);
222     }
223 
224     private void generateAndDisplayHTML(JTextArea comp) {
225         String[] lines = comp.getText().split("\\n");
226         StringBuilder contentBuilder = new StringBuilder();
227         contentBuilder.append("<html><head><title>Student List</title><style>")
228                 .append("body { margin: 30px 50px; font-size: 20px; }") // Increased margins and font size
229                 .append("</style></head><body><h1>Student List</h1><div>");
230 
231         for (String line : lines) {
232             contentBuilder.append(line).append("<br/>");
233         }
234 
235         contentBuilder.append("</div></body></html>");
236 
237         String htmlContent = contentBuilder.toString();
238 
239         try {
240             File tempFile = File.createTempFile("student_list", ".html");
241             FileWriter writer = new FileWriter(tempFile);
242             writer.write(htmlContent);
243             writer.close();
244 
245             Desktop.getDesktop().browse(tempFile.toURI());
246         } catch (IOException e) {
```

page9

```
247             e.printStackTrace();
248         }
249     }
250 
251     private void performSearch() {
252         String query = searchTextField.getText().trim().toLowerCase();
253         if (query.isEmpty()) {
254             JOptionPane.showMessageDialog(this, "Please enter a search query.");
255             return;
256         }
257 
258         String transportationContent = transportationListTextArea.getText().toLowerCase();
259         String accommodationContent = roomListTextArea.getText().toLowerCase();
260 
261         String[] transportationLines = transportationContent.split("\\n");
262         String[] accommodationLines = accommodationContent.split("\\n");
263 
264         StringBuilder results = new StringBuilder();
265 
266         boolean foundInTransport = false;
267         boolean foundInAccommodation = false;
268 
269         // Search in Transportation tab
270         for (String line : transportationLines) {
271             if (line.contains(query)) {
272                 results.append("Bus Assignment: ").append(line).append("\n");
273                 foundInTransport = true;
274                 break;
275             }
276         }
277 
278         // Search in Accommodation tab
279         for (String line : accommodationLines) {
280             if (line.contains(query)) {
281                 results.append("Room Assignment: ").append(line).append("\n");
282                 foundInAccommodation = true;
283                 break;
284             }
285         }
286 
287         if (!foundInTransport && !foundInAccommodation) {
```

page10

```
288             results.append("No results found for: ").append(query);
289         }
290
291         searchResultsTextArea.setText(results.toString());
292     }
293
294     public static void main(String[] args) {
295         SwingUtilities.invokeLater(new Runnable() {
296             @Override
297             public void run() {
298                 MainFrame frame = new MainFrame();
299                 frame.setVisible(true);
300             }
301         });
302     }
303 }
```

page11