```java
import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

// javna klasa FoodTrackerApp koja nasljeđuje JFrame
public class FoodTrackerApp extends JFrame {
    private DefaultTableModel tableModel;
    private JTable foodTable;
    private JTextField nameField, caloriesField, goalCaloriesField;
    private JComboBox<String> categoryBox, filterBox;
    private JLabel totalCaloriesLabel, goalLabel;
    private JProgressBar progressBar;
    private int goalCalories = 0;

    public FoodTrackerApp() {
        setTitle("Food Tracker");
        setSize(650, 500);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        // Input panel za ime, kalorije, ciljne kalorije i kategoriju
        JPanel inputPanel = new JPanel(new GridLayout(3, 4, 10, 5));
        nameField = new JTextField();
        caloriesField = new JTextField();
        goalCaloriesField = new JTextField();
        categoryBox = new JComboBox<>(new String[]{
                "Voće",
                "Povrće",
                "Slatkiši",
                "Meso",
                "Riba",
                "Tjestenina",
                "Mliječni proizvodi",
                "Piće",
                "Zamrznuto",
```

```
39                  "Grickalice",
40                  "Kruh i pekarski proizvodi",
41                  "Žitarice i pahuljice",
42                  "Konzervirana hrana",
43                  "Začini i umaci",
44                  "Ulje i ocat",
45                  "Dodaci prehrani",
46                  "Dječja hrana",
47                  "Ostalo",
48          });
49
50          // sučelje za unos
51          JButton addButton = new JButton("Dodaj");
52          JButton setGoalButton = new JButton("Postavi
    cilj");
53
54          inputPanel.add(new JLabel("Hrana:"));
55          inputPanel.add(nameField);
56          inputPanel.add(new JLabel("Kalorije:"));
57          inputPanel.add(caloriesField);
58          inputPanel.add(new JLabel("Kategorija:"));
59          inputPanel.add(categoryBox);
60          inputPanel.add(new JLabel(""));
61          inputPanel.add(addButton);
62
63          inputPanel.add(new JLabel("Ciljane kalorije:"
    ));
64          inputPanel.add(goalCaloriesField);
65          inputPanel.add(new JLabel(""));
66          inputPanel.add(setGoalButton);
67
68          add(inputPanel, BorderLayout.NORTH);
69
70          // Tablica za prikaz podataka
71          tableModel = new DefaultTableModel(new String
    []{"Hrana", "Kalorije", "Kategorija"}, 0);
72          foodTable = new JTable(tableModel);
73          foodTable.getColumnModel().getColumn(1).
    setCellEditor(new DefaultCellEditor(new JTextField
    ()));
74          add(new JScrollPane(foodTable), BorderLayout.
```

```
 74 CENTER);
 75
 76         // Donji panel
 77         JPanel bottomPanel = new JPanel(new
    GridLayout(3, 1));
 78
 79         // tipka za prikaz grafikona
 80         JButton showChartButton = new JButton("
    Prikaži tortni graf");
 81         showChartButton.addActionListener(e ->
    showPieChartDialog());
 82         bottomPanel.add(showChartButton);
 83
 84
 85         // tipke i form za filtriranje
 86         JPanel filterPanel = new JPanel(new
    FlowLayout(FlowLayout.LEFT));
 87         filterBox = new JComboBox<>(new String[]{"
    Sve", "Voće", "Povrće", "Slatkiši", "Meso", "Ostalo"
    });
 88         JButton filterButton = new JButton("
    Filtriraj");
 89         JButton resetFilterButton = new JButton("
    Resetiraj");
 90         JButton deleteButton = new JButton("Obriši
    odabrano");
 91         totalCaloriesLabel = new JLabel("Ukupno
    kalorija: 0");
 92
 93         filterPanel.add(new JLabel("Filter:"));
 94         filterPanel.add(filterBox);
 95         filterPanel.add(filterButton);
 96         filterPanel.add(resetFilterButton);
 97         filterPanel.add(deleteButton);
 98         filterPanel.add(Box.createHorizontalStrut(20
    ));
 99         filterPanel.add(totalCaloriesLabel);
100
101         // Traka za praćenje napretka
102         progressBar = new JProgressBar(0, 100);
103         progressBar.setStringPainted(true);
```

```
104         goalLabel = new JLabel("Cilj: 0 kalorija");
105
106         JPanel progressPanel = new JPanel(new
    BorderLayout());
107         progressPanel.add(goalLabel, BorderLayout.
    WEST);
108         progressPanel.add(progressBar, BorderLayout.
    CENTER);
109
110         bottomPanel.add(filterPanel);
111         bottomPanel.add(progressPanel);
112
113         add(bottomPanel, BorderLayout.SOUTH);
114
115         // Event listeners
116         addButton.addActionListener(e -> addFood());
117         deleteButton.addActionListener(e ->
    deleteSelectedRow());
118         filterButton.addActionListener(e ->
    applyFilter());
119         resetFilterButton.addActionListener(e ->
    resetFilter());
120         setGoalButton.addActionListener(e ->
    setGoalCalories());
121
122         foodTable.getModel().addTableModelListener(e
     -> updateTotalCalories());
123     }
124
125     // privatna void metoda addFood koja prima ime,
    kalorije i kategoriju putem korisničkog unosa/
    dropdown formulara
126     private void addFood() {
127         String name = nameField.getText().trim();
128         String caloriesStr = caloriesField.getText
    ().trim();
129         String category = (String) categoryBox.
    getSelectedItem();
130
131         // upozorenje u slučaju praznog unosa
132         if (name.isEmpty() || caloriesStr.isEmpty
```

```
132 ()) {
133             JOptionPane.showMessageDialog(this, "
    Molimo unesite ime i kalorije.");
134             return;
135         }
136 
137         // unos kalorija i obrada grešaka u slučaju
    negativnog broja ili vrijednosti koja nije broj
138         int calories;
139         try {
140             calories = Integer.parseInt(caloriesStr
    );
141             if (calories < 0) {
142                 JOptionPane.showMessageDialog(this,
    "Kalorije ne mogu biti negativne.");
143                 return;
144             }
145         } catch (NumberFormatException e) {
146             JOptionPane.showMessageDialog(this, "
    Kalorije moraju biti cijeli broj.");
147             return;
148         }
149 
150         // upozorenje za obroke s više od 1000
    kalorija
151         if (calories > 1000) {
152             Toolkit.getDefaultToolkit().beep();
153             JOptionPane.showMessageDialog(this, "
    UPOZORENJE: Kalorijska bomba detektirana!");
154         }
155 
156         // ispuna tablice
157         tableModel.addRow(new Object[]{name,
    calories, category});
158         nameField.setText("");
159         caloriesField.setText("");
160         // poziv metode za ažuriranje kalorija
161         updateTotalCalories();
162     }
163 
164     // privatna void metoda za brisanje
```

```
165     private void deleteSelectedRow() {
166         // odabir reda za brisanje
167         int selectedRow = foodTable.getSelectedRow
    ();
168         // brisanje reda ukoliko je nađen i
    ažuriranje kalorija
169         if (selectedRow != -1) {
170             tableModel.removeRow(selectedRow);
171             updateTotalCalories();
172         } else {
173             // obrada slučaja u kojem korisnik
    stišće tipku za brisanje bez da odabere obrok
174             JOptionPane.showMessageDialog(this, "
    Odaberite red za brisanje.");
175         }
176     }
177
178     // metoda za filtriranje
179     private void applyFilter() {
180         // odabir kategorije i postavljanje filter
    parametara
181         String selectedCategory = (String) filterBox
    .getSelectedItem();
182         TableRowSorter<TableModel> sorter = new
    TableRowSorter<>(foodTable.getModel());
183         // ako je kategorija "Sve", vraća sve obroke
184         if (!selectedCategory.equals("Sve")) {
185             sorter.setRowFilter(RowFilter.
    regexFilter(selectedCategory, 2));
186             // ako je odabrana kategorija, vraća
    samo nju
187         } else {
188             sorter.setRowFilter(null);
189         }
190         foodTable.setRowSorter(sorter);
191     }
192
193     // filter se resetira tako da se postavi na sve
    i dohvati sve obroke
194     private void resetFilter() {
195         filterBox.setSelectedItem("Sve");
```

```
196         applyFilter();
197     }
198 
199     private void setGoalCalories() {
200         try {
201             int value = Integer.parseInt(
    goalCaloriesField.getText().trim()); // Dohvati i
    parsiraj unos korisnika
202             if (value <= 0) {
203                 JOptionPane.showMessageDialog(this,
    "Cilj mora biti pozitivan broj."); // Provjera da
    cilj nije nula ili negativan
204                 return;
205             }
206             goalCalories = value; // Postavi cilj
207             goalLabel.setText("Cilj: " +
    goalCalories + " kalorija"); // Ažuriraj prikaz
    cilja
208             updateTotalCalories(); // Ažuriraj
    ukupne kalorije
209         } catch (NumberFormatException e) {
210             JOptionPane.showMessageDialog(this, "
    Unesite ispravnu vrijednost cilja."); // Greška ako
    unos nije broj
211         }
212     }
213 
214     private void updateTotalCalories() {
215         int total = 0;
216         for (int i = 0; i < tableModel.getRowCount
    (); i++) {
217             try {
218                 total += Integer.parseInt(tableModel
    .getValueAt(i, 1).toString()); // Zbroji kalorije iz
     tablice
219             } catch (NumberFormatException e) {
220                 // Ignoriraj neispravne unose
221             }
222             repaint(); // Osvježi tortni graf
223         }
224 
```

```
225         totalCaloriesLabel.setText("Ukupno kalorija: " + total); // Prikaži ukupan broj kalorija
226
227         if (goalCalories > 0) {
228             int percent = (int) ((total / (double) goalCalories) * 100); // Izračunaj postotak unesenih kalorija
229             progressBar.setValue(Math.min(percent, 100)); // Postavi vrijednost progress bara
230             progressBar.setString(total + " / " + goalCalories + " kcal"); // Prikaži tekst na progress baru
231
232             if (total > goalCalories) {
233                 progressBar.setForeground(Color.RED); // Ako je prekoračeno, prikaži crveno
234             } else {
235                 progressBar.setForeground(Color.GREEN.darker()); // Inače zeleno
236             }
237         } else {
238             progressBar.setValue(0); // Ako cilj nije postavljen
239             progressBar.setString("Postavi cilj kalorija");
240             progressBar.setForeground(Color.GRAY); // Siva boja ako nema cilja
241         }
242     }
243
244     private class PieChartPanel extends JPanel {
245         @Override
246         protected void paintComponent(Graphics g) {
247             super.paintComponent(g);
248
249             Map<String, Integer> categoryCalories = new HashMap<>();
250             int total = 0;
251
252             for (int i = 0; i < tableModel.getRowCount(); i++) {
```

```
253                 String category = tableModel.
    getValueAt(i, 2).toString(); // Dohvati kategoriju
254                 int cal = Integer.parseInt(
    tableModel.getValueAt(i, 1).toString()); // Dohvati
    kalorije
255
256                 categoryCalories.put(category,
    categoryCalories.getOrDefault(category, 0) + cal);
    // Zbroji kalorije po kategorijama
257                 total += cal; // Zbroji ukupne
    kalorije
258             }
259
260             if (total == 0) return; // Ako nema
    kalorija, ništa ne crtamo
261
262             Graphics2D g2d = (Graphics2D) g;
263             g2d.setRenderingHint(RenderingHints.
    KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON
    ); // Omogući glatko iscrtavanje
264
265             int x = 20, y = 20, diameter = 200;
266             int startAngle = 0;
267             Random rand = new Random();
268             Map<String, Color> colorMap = new
    HashMap<>();
269
270             for (Map.Entry<String, Integer> entry :
    categoryCalories.entrySet()) {
271                 String category = entry.getKey();
272                 int value = entry.getValue();
273
274                 int angle = (int) Math.round(360.0
     * value / total); // Izračunaj kut sektora
275                 Color color = colorMap.
    computeIfAbsent(category, k -> new Color(rand.
    nextInt(255), rand.nextInt(255), rand.nextInt(255
    ))); // Dodijeli nasumičnu boju
276
277                 g2d.setColor(color);
278                 g2d.fillArc(x, y, diameter, diameter
```

```
278 , startAngle, angle); // Nacrtaj sektor
279                 startAngle += angle; // Pomakni
    početni kut
280             }
281
282             // Crtaj legendu
283             int legendY = y + diameter + 20;
284             for (Map.Entry<String, Color> entry :
    colorMap.entrySet()) {
285                 g2d.setColor(entry.getValue());
286                 g2d.fillRect(x, legendY, 20, 20);
    // Boja kvadrata
287                 g2d.setColor(Color.BLACK);
288                 g2d.drawString(entry.getKey(), x +
    30, legendY + 15); // Tekst kategorije
289                 legendY += 25;
290             }
291         }
292     }
293
294     private void showPieChartDialog() {
295         JDialog dialog = new JDialog(this, "Tortni
    graf kalorija", true); // Novi modalni prozor
296         dialog.setSize(300, 400);
297         dialog.setLocationRelativeTo(this);
298         dialog.setLayout(new BorderLayout());
299
300         PieChartPanel pieChartPanel = new
    PieChartPanel(); // Panel s grafom
301         dialog.add(pieChartPanel, BorderLayout.
    CENTER);
302
303         JButton closeButton = new JButton("Zatvori"
    ); // Gumb za zatvaranje
304         closeButton.addActionListener(e -> dialog.
    dispose());
305         JPanel closePanel = new JPanel();
306         closePanel.add(closeButton);
307         dialog.add(closePanel, BorderLayout.SOUTH);
308
309         dialog.setVisible(true); // Prikaži prozor
```

```
310     }
311
312     public static void main(String[] args) {
313         SwingUtilities.invokeLater(() -> new
    FoodTrackerApp().setVisible(true)); // Pokreni
    aplikaciju na GUI niti
314     }
315 }
```

```
1 Manifest-Version: 1.0
2 Main-Class: FoodTrackerApp
3
4
```