Második feladat – Iluska

1. NetBeans előkészítése

1. **Indítsd el** az Apache NetBeans IDE-t.
2. **Hozz létre egy új projektet**:
    a. Kattints a **File -> New Project...** menüpontra.
    b. Válaszd ki a **"Java with Ant"** kategóriát.
    c. Válaszd a **"Java Application"** típust.
    d. **Adj nevet** a projektnek (pl. `MaintenanceRegistry`) és állítsd be a helyét.
    e. Kattints a **Finish** gombra.

2. Projekt struktúra beállítása

A következő fájlokat kell létrehoznod a `src/maintenanceregistry/` mappán belül:

- **MaintenanceRegistry.java** (fő osztály)
- **MainFrame.java** (grafikus felület)
- **MaintenanceRecord.java (adatmodell)**

MaintenanceRegistry.java

```
package maintenanceregistry;
 import javax.swing.*;
/**
A karbantartási nyilvántartó alkalmazás fő osztálya.
*/
public class MaintenanceRegistry { public static void main(String[] args) {
SwingUtilities.invokeLater(() -> new MainFrame().setVisible(true)); } }
```

MainFrame.java

```
package maintenanceregistry; import javax.swing.*; import
javax.swing.table.DefaultTableModel; import java.awt.*; import java.io.*; import
java.util.ArrayList; import java.util.List;
/**
A fő grafikus keret, amely a táblázatot és a hozzáadási funkciókat kezeli.
*/ public class MainFrame extends JFrame { private JTable table; private
DefaultTableModel tableModel; private JTextField itemIdField,
maintenanceDateField, actionField, commentField; private List records = new
ArrayList<>();

public MainFrame() {
  setTitle("Karbantartási Nyilvántartás");
  setSize(800, 600);
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setLayout(new BorderLayout());

  // Táblázat modell
  tableModel = new DefaultTableModel(new String[]{"ID", "Tárgy ID", "Dátum",
"Művelet", "Megjegyzés"}, 0);
  table = new JTable(tableModel);
  add(new JScrollPane(table), BorderLayout.CENTER);

  // Adatbeviteli panel
  JPanel inputPanel = new JPanel(new GridLayout(5, 2));
  inputPanel.add(new JLabel("Tárgy ID:"));
  itemIdField = new JTextField();
  inputPanel.add(itemIdField);

  inputPanel.add(new JLabel("Dátum:"));
  maintenanceDateField = new JTextField();
  inputPanel.add(maintenanceDateField);

  inputPanel.add(new JLabel("Művelet:"));
  actionField = new JTextField();
  inputPanel.add(actionField);

  inputPanel.add(new JLabel("Megjegyzés:"));
  commentField = new JTextField();
```

```
    inputPanel.add(commentField);

    JButton addButton = new JButton("Hozzáadás");
    addButton.addActionListener(e -> addEntry());
    inputPanel.add(addButton);

    add(inputPanel, BorderLayout.SOUTH);

    // Adatok betöltése
    loadCsvData();
}

private void loadCsvData() {
    try (BufferedReader reader = new BufferedReader(new
FileReader("maintenance_records.csv"))) {
        String line;
        boolean isHeader = true;
        while ((line = reader.readLine()) != null) {
            if (isHeader) {
                isHeader = false; // Az első sor fejléc
                continue;
            }
            String[] data = line.split(";");
            int id = Integer.parseInt(data[0]);
            MaintenanceRecord record = new MaintenanceRecord(id, data[1], data[2],
data[3], data[4]);
            records.add(record);
            tableModel.addRow(new Object[]{record.getId(), record.getItemId(),
record.getMaintenanceDate(), record.getAction(), record.getComment()});
        }
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this, "Nem sikerült betölteni az adatokat:
" + e.getMessage());
    }
}

private void addEntry() {
    String itemId = itemIdField.getText();
    String maintenanceDate = maintenanceDateField.getText();
    String action = actionField.getText();
    String comment = commentField.getText();
```

```
  if (itemId.isEmpty() || maintenanceDate.isEmpty() || action.isEmpty()) {
    JOptionPane.showMessageDialog(this, "Kérlek, tölts ki minden mezőt!");
    return;
  }

  int id = records.size() + 1;
  MaintenanceRecord newRecord = new MaintenanceRecord(id, itemId,
maintenanceDate, action, comment);
  records.add(newRecord);

  tableModel.addRow(new Object[]{newRecord.getId(), newRecord.getItemId(),
newRecord.getMaintenanceDate(), newRecord.getAction(),
newRecord.getComment()});
  saveCsvData();
}

private void saveCsvData() {
  try (PrintWriter writer = new PrintWriter(new
FileWriter("maintenance_records.csv"))) {
    writer.println("ID;ItemID;MaintenanceDate;Action;Comment");
    for (MaintenanceRecord record : records) {
      writer.println(record.toCsvRow());
    }
  } catch (IOException e) {
    JOptionPane.showMessageDialog(this, "Nem sikerült menteni az adatokat: "
+ e.getMessage());
  }
}


}
```

MaintenanceRecord.java

```
package maintenanceregistry;
/**
Adatmodell osztály, amely egy karbantartási műveletet reprezentál.
*/ public class MaintenanceRecord { private int id; private String itemId; private
String maintenanceDate; private String action; private String comment;
```

```
public MaintenanceRecord(int id, String itemId, String maintenanceDate, String
action, String comment) {
   this.id = id;
   this.itemId = itemId;
   this.maintenanceDate = maintenanceDate;
   this.action = action;
   this.comment = comment;
}

public int getId() {
   return id;
}

public String getItemId() {
   return itemId;
}

public String getMaintenanceDate() {
   return maintenanceDate;
}

public String getAction() {
   return action;
}

public String getComment() {
   return comment;
}

public String toCsvRow() {
   return id + ";" + itemId + ";" + maintenanceDate + ";" + action + ";" + comment;
}

}
```