# JAVA CSV DESIGN – USE CASES & CODE

Στην παρούσα εργασία παρουσιάζεται η αναλυτική **περιγραφή** του **θεωρητικού σχεδιασμού** ενός προγράμματος που παρέχει **προβολή**, **επεξεργασία** και **διαχείριση CSV αρχείων** από τον χρήστη.

## 1. ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ – USE CASES

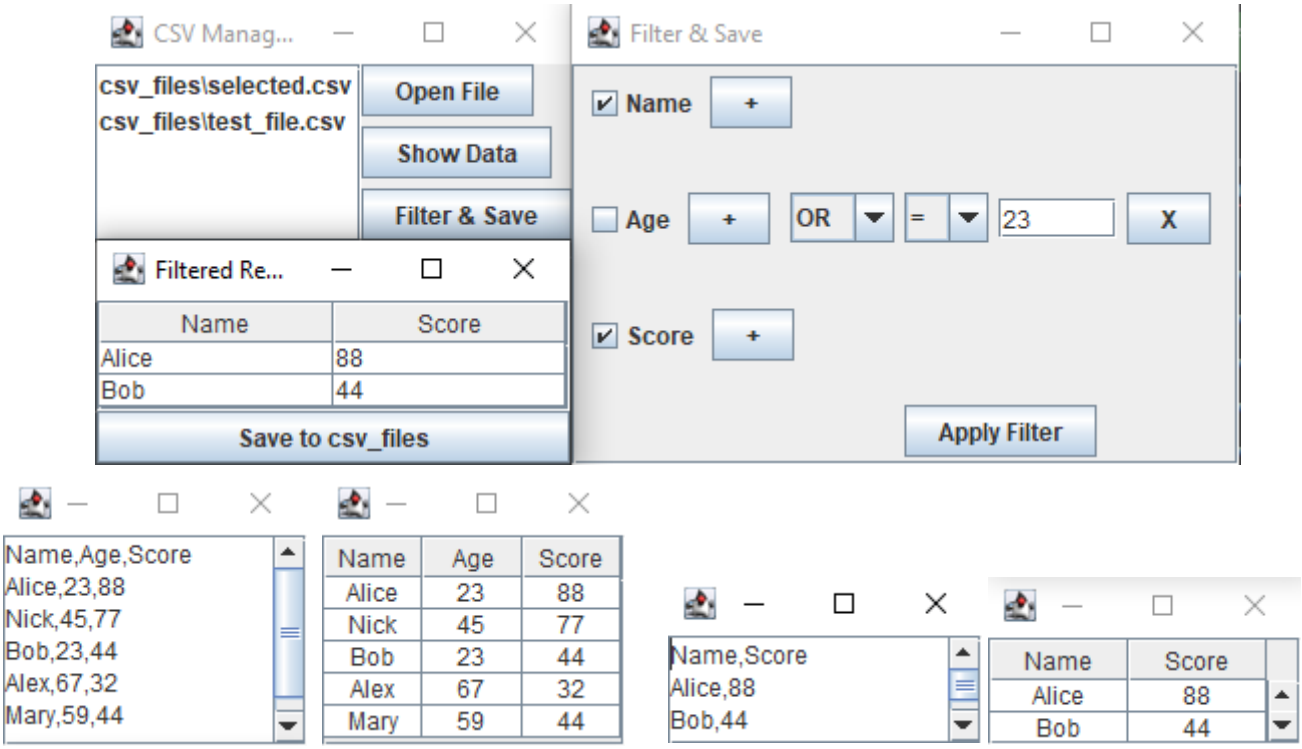
| ID : UC_1 | |
|---|---|
| **FUNCTION NAME** | **ΕΚΤΕΛΕΣΕ ΕΞΟΔΟ** |
| **DESCRIPTION AND GOAL** | Επιλογή για αποχώρηση από την εφαρμογή, ανά πάσα στιγμή |
| **PRIMARY ACTOR** | Ο χρήστης |
| **PRECONDITIONS** | Πρέπει να έχει ήδη εισέλθει ο χρήστης στην εφαρμογή |
| **BASIC FLOW** | 1. Αρχικοποίηση όταν ο χρήστης επιλέγει να αποχωρήσει<br>2. Το σύστημα κλείνει όλα τα επιμέρους παράθυρα της εφαρμογής |
| **EXTENSIONS / VARIATIONS** | - |
| **POST CONDITIONS** | Έχει γίνει πλήρης διακοπή της λειτουργίας της εφαρμογής |

| ID : UC_2 | |
|---|---|
| **FUNCTION NAME** | **ΕΜΦΑΝΙΣΕ CSV ΑΡΧΕΙΟ** |
| **DESCRIPTION AND GOAL** | Επιλογή CSV αρχείου και εμφάνιση του σε νέο παράθυρο |
| **PRIMARY ACTOR** | Ο χρήστης |
| **PRECONDITIONS** | Πρέπει να υπάρχουν αποθηκευμένα αρχεία CSV στο σύστημα |
| **BASIC FLOW** | 1. Αρχικοποίηση με επιλογή αρχείου CSV, με χρήση “File Chooser”<br>2. Το σύστημα ανοίγει νέο αναδυόμενο παράθυρο<br>3. Τα δεδομένα του αρχείου παρουσιάζονται σε tabular form |
| **EXTENSIONS / VARIATIONS** | Σε επιλογή άδειου αρχείου εμφανίζεται άδειο παράθυρο |
| **POST CONDITIONS** | - |

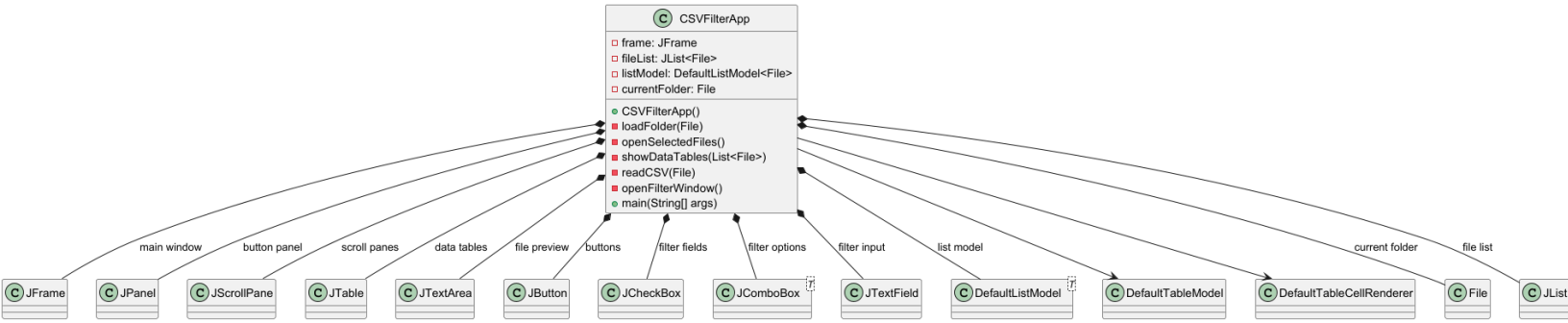
| ID : UC_3 | |
|---|---|
| **FUNCTION NAME** | **ΦΟΡΤΩΣΕ & ΚΑΤΕΓΡΑΨΕ CSV ΑΡΧΕΙΟ** |
| **DESCRIPTION AND GOAL** | Φόρτωση CSV αρχείου και καταγραφή του ως σύνολο δεδομένων |
| **PRIMARY ACTOR** | Ο χρήστης |
| **PRECONDITIONS** | Πρέπει να υπάρχουν αποθηκευμένα αρχεία CSV στο σύστημα |
| **BASIC FLOW** | 1. Αρχικοποίηση με επιλογή αρχείου CSV, με χρήση “File Chooser”<br>2. Το σύστημα φορτώνει τα δεδομένα του αρχείου<br>3. Τα δεδομένα καταγράφονται ως σύνολο δεδομένων<br>4. Σε νέο παράθυρο το σύστημα ζητά να δοθεί όνομα για το σύνολο<br>5. Αφότου πατηθεί “Submit” με κατάλληλο όνομα, αποθηκεύεται |
| **EXTENSIONS / VARIATIONS** | Μπορεί να χρειαστεί αποθήκευση επιπρόσθετων αρχείων |
| **POST CONDITIONS** | Το επιλεγμένο αρχείο αποθηκεύτηκε ως σύνολο δεδομένων |

| ID : UC_4 | |
|---|---|
| **FUNCTION NAME** | **ΕΜΦΑΝΙΣΕ ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ** |
| **DESCRIPTION AND GOAL** | Εμφάνιση καταγεγραμμένου συνόλου δεδομένων |
| **PRIMARY ACTOR** | Ο χρήστης |
| **PRECONDITIONS** | Πρέπει να υπάρχει αποθηκευμένο το αρχείο CSV στο σύστημα |
| **BASIC FLOW** | 1. Αρχικοποίηση με προσθήκη ονόματος του αρχείου από το χρήστη<br>2. Το back-end κομμάτι του συστήματος εκτελεί αναζήτηση αρχείου<br>3. Το σύστημα φορτώνει και εμφανίζει τα δεδομένα του αρχείου |
| **EXTENSIONS / VARIATIONS** | Για ανύπαρκτο αρχείο δεν εκτελείται λειτουργία |
| **POST CONDITIONS** | - |

| ID : UC_5 | |
|---|---|
| **FUNCTION NAME** | **ΦΙΛΤΡΑΡΕ ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ** |
| **DESCRIPTION AND GOAL** | Καταγραφή ως νέο σύνολο δεδομένων των στοιχείων<br>ενός υπάρχοντος συνόλου, με βάση τα επιλεγμένα φίλτρα |
| **PRIMARY ACTOR** | Ο χρήστης |
| **PRECONDITIONS** | Πρέπει να υπάρχει αποθηκευμένο σύνολο δεδομένων στο σύστημα |
| **BASIC FLOW** | 1. Αρχικοποίηση κατά την επιλογή φίλτρου (ενός ή περισσότερων) της μορφής “Attribute = Value” σε ένα σύνολο δεδομένων<br>2. Το αποτέλεσμα καταγράφεται ως νέο σύνολο δεδομένων<br>3. Το back-end κομμάτι του συστήματος επιστρέφει στο front-end την αριθμητική τιμή 0, ως ένδειξη επιτυχίας |
| **EXTENSIONS / VARIATIONS** | Σε αποτυχία το back-end σύστημα επιστρέφει αρνητική τιμή |
| **POST CONDITIONS** | Έχει αποθηκευτεί το παραγόμενο σύνολο δεδομένων στο σύστημα |



## 2. ΣΧΕΔΙΑΣΗ ΛΟΓΙΣΜΙΚΟΥ




```
package com.org.example;
```

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;
import java.io.*;
import java.nio.file.*;
import java.util.*;
import java.util.List;
```

```
public class CSVFilterApp
{
   private JFrame frame;
   private JList<File> fileList;
   private DefaultListModel<File> listModel;
   private File currentFolder = new File("csv_files");

   // Constructor initializes the main window
   public CSVFilterApp()
   {
      frame = new JFrame("CSV Manager");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(800, 600);

      listModel = new DefaultListModel<>();
      fileList = new JList<>(listModel);
      fileList.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
      JScrollPane scrollPane = new JScrollPane(fileList);

      JButton openFileButton = new JButton("Open File");
      JButton showDataButton = new JButton("Show Data");
      JButton filterButton = new JButton("Filter & Save");

      JPanel buttonPanel = new JPanel();
      buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.Y_AXIS));
      buttonPanel.add(openFileButton);
      buttonPanel.add(Box.createVerticalStrut(5));
      buttonPanel.add(showDataButton);
      buttonPanel.add(Box.createVerticalStrut(5));
```

```
    buttonPanel.add(filterButton);

    frame.setLayout(new BorderLayout());
    frame.add(scrollPane, BorderLayout.CENTER);
    frame.add(buttonPanel, BorderLayout.EAST);

    loadFolder(currentFolder);

    // Button actions
    openFileButton.addActionListener(e -> openSelectedFiles());
    showDataButton.addActionListener(e -> showDataTables(fileList.getSelectedValuesList()));
    filterButton.addActionListener(e -> openFilterWindow());

    frame.setVisible(true);
}
```

// Load CSV files from the folder into the list

```
private void loadFolder(File folder)
{
    if (!folder.exists()) folder.mkdirs();
    listModel.clear();
    File[] files = folder.listFiles();
    if (files != null)
    {
        Arrays.sort(files);
        for (File f : files) listModel.addElement(f);
    }
}
```

// Open selected CSV files in raw text view

```
private void openSelectedFiles()
{
    for (File file : fileList.getSelectedValuesList())
    {
        try
        {
            JTextArea textArea = new JTextArea();
            textArea.read(new FileReader(file), null);
            textArea.setEditable(false);
            JScrollPane scrollPane = new JScrollPane(textArea);
            JFrame textFrame = new JFrame(file.getName());
            textFrame.setSize(600, 400);
            textFrame.add(scrollPane);
            textFrame.setVisible(true);
        }
        catch (IOException ex)
        {
            ex.printStackTrace();
        }
    }
}
```

// Show selected CSV files in JTable format

```
  private void showDataTables(List<File> files)
  {
    for (File file : files)
    {
      try
      {
        List<String[]> data = readCSV(file);
        if (data.isEmpty()) continue;

        String[] headers = data.get(0);
        DefaultTableModel tableModel = new DefaultTableModel(headers, 0);
        for (int i = 1; i < data.size(); i++) tableModel.addRow(data.get(i));

        JTable table = new JTable(tableModel);
        DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment(SwingConstants.CENTER);
        for (int i = 0; i < table.getColumnCount(); i++)
table.getColumnModel().getColumn(i).setCellRenderer(centerRenderer);

        JScrollPane tableScroll = new JScrollPane(table);
        JFrame tableFrame = new JFrame(file.getName() + " - Data");
        tableFrame.setSize(800, 400);
        tableFrame.add(tableScroll);
        tableFrame.setVisible(true);
      }
      catch (Exception ex)
      {
        ex.printStackTrace();
      }
    }
  }
```

```
  // Read CSV file into a list of String arrays
  private List<String[]> readCSV(File file) throws IOException
  {
    List<String[]> rows = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(new FileReader(file)))
    {
      String line;
      while ((line = br.readLine()) != null) {
        rows.add(line.split(","));
      }
    }
    return rows;
  }
```

```
  // Open filter window to select fields and conditions
  private void openFilterWindow()
  {
    List<File> selectedFiles = fileList.getSelectedValuesList();
    if (selectedFiles.isEmpty()) return;

    JFrame filterFrame = new JFrame("Filter & Save");
```

```
    filterFrame.setSize(700, 500);
    filterFrame.setLayout(new BorderLayout());

    File firstFile = selectedFiles.get(0);
    List<String[]> data;
    try
    {
       data = readCSV(firstFile);
    }
    catch (IOException e) {
       e.printStackTrace();
       return;
    }
    if (data.isEmpty()) return;

    String[] headers = data.get(0);
    JPanel filterPanel = new JPanel();
    filterPanel.setLayout(new BoxLayout(filterPanel, BoxLayout.Y_AXIS));

    List<JCheckBox> fieldChecks = new ArrayList<>();
    List<JPanel> valuePanels = new ArrayList<>();

    // Build filter rows for each column
    for (String header : headers)
    {
       JPanel row = new JPanel(new FlowLayout(FlowLayout.LEFT));

       JCheckBox fieldCheck = new JCheckBox(header);
       fieldChecks.add(fieldCheck);

       JButton addValueButton = new JButton("+");
       row.add(fieldCheck);
       row.add(addValueButton);

       JPanel valuesContainer = new JPanel();
       valuesContainer.setLayout(new BoxLayout(valuesContainer, BoxLayout.Y_AXIS));
       row.add(valuesContainer);
       valuePanels.add(valuesContainer);

       // Add condition row
       addValueButton.addActionListener(ev ->
       {
          JPanel valRow = new JPanel(new FlowLayout(FlowLayout.LEFT));
          JComboBox<String> logicBox = new JComboBox<>(new String[]{"=", "!=", ">", "<", ">=", "<="});
          JTextField valField = new JTextField(5);
          JComboBox<String> andOrBox = new JComboBox<>(new String[]{"AND", "OR"});
          JButton removeBtn = new JButton("X");
          valRow.add(andOrBox);
          valRow.add(logicBox);
          valRow.add(valField);
          valRow.add(removeBtn);
          valuesContainer.add(valRow);
          valuesContainer.revalidate();
```

```
            valuesContainer.repaint();
            removeBtn.addActionListener(rem ->
            {
                valuesContainer.remove(valRow);
                valuesContainer.revalidate();
                valuesContainer.repaint();
            });
        });

        filterPanel.add(row);
    }
```

```
    JButton applyBtn = new JButton("Apply Filter");
    filterPanel.add(applyBtn);

    filterFrame.add(new JScrollPane(filterPanel), BorderLayout.CENTER);
    filterFrame.setVisible(true);
```

```
    // Apply filter logic
    applyBtn.addActionListener(ev ->
    {
        List<Integer> visibleCols = new ArrayList<>();
        List<String[]> filteredData = new ArrayList<>();

        // Determine which columns should be visible
        for (int col = 0; col < headers.length; col++)
        {
            if (fieldChecks.get(col).isSelected())
            {
                visibleCols.add(col);
            }
        }
```

```
        // Add header row
        List<String> newHeader = new ArrayList<>();
        for (int col : visibleCols) newHeader.add(headers[col]);
        filteredData.add(newHeader.toArray(new String[0]));
```

```
        // Evaluate each row based on conditions
        for (int i = 1; i < data.size(); i++)
        {
            String[] row = data.get(i);
            boolean rowPass = true;

            for (int col = 0; col < headers.length; col++)
            {
                JPanel valuePanel = valuePanels.get(col);
                if (valuePanel.getComponentCount() == 0) continue;

                boolean colPass = false;
                for (Component comp : valuePanel.getComponents())
                {
                    if (comp instanceof JPanel)
```

```
            {
                JPanel valRow = (JPanel) comp;
                JComboBox<String> andOrBox = (JComboBox<String>) valRow.getComponent(0);
                JComboBox<String> opBox = (JComboBox<String>) valRow.getComponent(1);
                JTextField valField = (JTextField) valRow.getComponent(2);

                String op = (String) opBox.getSelectedItem();
                String val = valField.getText();
                String cellVal = row[col];
                boolean cond = false;

                try
                {
                    double cellNum = Double.parseDouble(cellVal);
                    double valNum = Double.parseDouble(val);
                    switch (op)
                    {
                        case "=" -> cond = cellNum == valNum;
                        case "!=" -> cond = cellNum != valNum;
                        case ">" -> cond = cellNum > valNum;
                        case "<" -> cond = cellNum < valNum;
                        case ">=" -> cond = cellNum >= valNum;
                        case "<=" -> cond = cellNum <= valNum;
                    }
                }
                catch (NumberFormatException nfe)
                {
                    if (op.equals("=")) cond = cellVal.equals(val);
                    else if (op.equals("!=")) cond = !cellVal.equals(val);
                }

                String logic = (String) andOrBox.getSelectedItem();
                if (logic.equals("AND"))
                {
                    if (!cond) { colPass = false; break; }
                    else colPass = true;
                } else
                {
                    colPass |= cond;
                }
            }
        }
        rowPass &= colPass;
    }

    if (rowPass)
    {
        List<String> newRow = new ArrayList<>();
        for (int col : visibleCols) newRow.add(row[col]);
        filteredData.add(newRow.toArray(new String[0]));
    }
}
```

```
        // Show results in a new window
        DefaultTableModel tableModel = new DefaultTableModel(filteredData.get(0), 0);
        for (int i = 1; i < filteredData.size(); i++) tableModel.addRow(filteredData.get(i));
        JTable table = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(table);

        JFrame resultFrame = new JFrame("Filtered Result");
        resultFrame.setSize(800, 400);
        resultFrame.setLayout(new BorderLayout());
        resultFrame.add(scrollPane, BorderLayout.CENTER);

        JButton saveBtn = new JButton("Save to csv_files");
        resultFrame.add(saveBtn, BorderLayout.SOUTH);

        // Save results to new CSV file
        saveBtn.addActionListener(sv ->
        {
           String name = JOptionPane.showInputDialog(resultFrame, "File name:");
           if (name != null && !name.isBlank())
           {
              File outFile = new File(currentFolder, name + ".csv");
              try (PrintWriter pw = new PrintWriter(outFile))
              {
                 for (String[] r : filteredData) pw.println(String.join(",", r));
                 JOptionPane.showMessageDialog(resultFrame, "Saved: " + outFile.getAbsolutePath());
                 loadFolder(currentFolder);
              } catch (Exception ex) { ex.printStackTrace(); }
           }
        });

        resultFrame.setVisible(true);
     });
  }

  public static void main(String[] args)
  {
     SwingUtilities.invokeLater(CSVFilterApp::new);
  }
}
```