Лабораторная работа № 6

Цель:

Углубить знания о табличных компонентах языка Java, научиться выполнять операции с датами.

Постановка задачи:

С использованием табличной компоненты (например, JTable) разработать приложение, позволяющее работать с датами (например, используя классы GregorianCalendar, Calendar, Date) и выполнять следующие операции:
1. Вычисления типа *=операнд операция числовая_константа*, где операция – + -, операнд – дата (например, =20.12.17+27).
2. Вычисления типа *=операнд операция числовая_константа*, где операция – + -, операнд – число или адрес ячейки (например, =B3-72).
3. Вычисления функций *МИН, МАКС* над датами и адресами ячеек (например, *=мин(21.03.12,B2,C4)*; количество параметров функции по собственному усмотрению (переменное (предпочтительнее) либо фиксированное).
4. Обработать ошибки (в т.ч. циклические).
Реализовать паттерн проектирования MVC.
Формулу можно ввести в ячейку таблицы (предпочтительнее) либо в другой элемент.
P.S. В Excel поддерживаются операции редактирования и пересчета ячеек.
Задача имеет больший вес.

Решение задачи:

В первую очередь следует реализовать класс “MyDate”, наследующийся от базового класса “GregorianCalendar” для удобной работы с датами.

```
public class MyDate extends GregorianCalendar {
    private String record;
    private HashSet<int[]> usingList;

    MyDate(MyDate date) {
        super(date.get(Calendar.YEAR), date.get(Calendar.MONTH), date.get(Calendar.DAY_OF_MONTH));
        this.record = date.getRecord();
        this.usingList = new HashSet<>();
    }

    MyDate(int year, int month, int dayOfMonth) {
        super(year, month, dayOfMonth);
        this.record = getFormula();
        this.usingList = new HashSet<>();
    }

    void setHash(HashSet<int[]> set) {
        if (set!=null)
            this.usingList = set;
    }
```

Рис.1. Конструкторы класса “MyDate”.

В поле “record” будем хранить запись даты в виде шаблона “yyyy-mm-dd”, а в “Hashset” запишем номер столбца и строки таблицы, на пересечении которых находится объект, чтобы впоследствии обращаться к дате, исходя из полученных значений ячейки.

```
    private String getFormula() {
        SimpleDateFormat form = new SimpleDateFormat( pattern: "yyyy-MM-dd");
        form.setCalendar(this);
        return form.format(this.getTime());
    }

    void addUsingDependency(int i, int j) {
        int[] array = new int[2];
        array[0] = i;
        array[1] = j;
        this.usingList.add(array);
    }

    HashSet getHash() { return usingList; }

    String getRecord() {
        return record;
    }

    void setRecord(String record) { this.record = record; }

    @Override
    public String toString() {
        SimpleDateFormat form = new SimpleDateFormat( pattern: "dd.MM.yyyy");
        form.setCalendar(this);
        return form.format(this.getTime());
    }
}
```

Рис.2. Методы класса “MyDate”.

Следующим шагом будет реализация класса “View”, отвечающий за отображение данных (согласно паттерну MVC). Создаем модель таблицы по принципу “DefaultTableModel”. Затем организуем все компоненты окна.

```
private View(){
    super( title: "Excel lite");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (ClassNotFoundException | InstantiationException |
            IllegalAccessException | UnsupportedLookAndFeelException ignored) {
    }
    tableModel = new DefaultTableModel() {
        public boolean isCellEditable(int row, int column) {
            if (column == 0)
                return false;
            else
                return super.isCellEditable(row, column);
        }

        public Class getColumnClass(int column) { return MyDate.class; }
    };
    JPanel panel = new JPanel(new BorderLayout());
    table = new JTable(tableModel);
    table.setRowHeight(25);
    table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
    table.getTableHeader().setReorderingAllowed(false);

    JScrollPane js = new JScrollPane(table, JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
            JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
    js.setVisible(true);
    panel.add(js, BorderLayout.CENTER);
    panel.add(result, BorderLayout.NORTH);
    result.setEditable(false);
```

Рис.3.Распределение компонентов оконного приложение.

Известно, что первый столбец в “Excel” представлен в виде упорядоченной последовательности чисел - номеров строк. Поэтому инициализирует вектор целых чисел, куда поочередно добавляем числа от единицы до значения переменной “rowNumber”, в которой хранится количество строк. После добавления первого столбца дополняем таблицу остальными по заданному количеству столбцов “colNumber”. Затем создаем объект класса “Controller”, который выступает в качестве модели редактора ячеек.

```
Vector<Integer> v = new Vector<>();
int startNumber = 11;
rowNumber = startNumber;
columnNumber = startNumber+10;
for (int i = 1; i <= rowNumber-1; i++) {
    v.add(i);
}
table.getTableHeader().setReorderingAllowed(false);
tableModel.addColumn( columnName: "", v);
table.getColumnModel().getColumn( columnIndex: 0).setPreferredWidth(100);
for (int i = 0; i < columnNumber; i++) {
    tableModel.addColumn((char) (columnName + i));
    table.getColumnModel().getColumn(i).setPreferredWidth(100);
}
Controller controller = new Controller();
table.setDefaultEditor(MyDate.class, controller);
```

Рис.4. Инициализация таблицы.

Чтобы добавить возможность выполнения операций через выделение ячеек, реализуем слушателя “SelectionListener”. Таким образом, ненулевые значения ячеек будут добавляться в “ArrayList” объектов класса “MyDate” и выводиться через запятую в текстовое поле “result”.

```
ArrayList<MyDate> dates = new ArrayList<>();
ListSelectionModel selModel = table.getSelectionModel();
selModel.addListSelectionListener(e -> {
    dates.clear();
    StringBuilder res = new StringBuilder();
    int[] selectedRows = table.getSelectedRows();
    int[] selectedColumns = table.getSelectedColumns();
    for (int selectedRow : selectedRows) {
        for (int selectedColumn : selectedColumns) {
            TableModel model = table.getModel();
            MyDate value = (MyDate) model.getValueAt(selectedRow, selectedColumn);
            if(value != null){ dates.add(value); }
        }
    }
    for(int i = 0; i < dates.size(); i++){
        if(dates.get(i) != null){
            res.append(dates.get(i).toString());
            if(i != dates.size()-1){ res.append(","); }
        }
    }
```

Рис.5.Считывание значений с помощью выделения ячеек.

Слушатели для кнопок “JRadioButton” “min”,“max” вызывают соответствующие функции интерфейса “ControllerInterface”.

```
result.setText(res.toString());
min.addActionListener(e2 -> {
    if(min.isSelected()){
        result.removeAll();
        result.setText(controller.parseMin(dates));
    }
});
max.addActionListener(e1 -> {
    if(max.isSelected()){
        result.removeAll();
        result.setText(controller.parseMax(dates));
    }
});
```

Рис.6.Реализация слушателей для поиска минимального и максимального значений выделенных ячеек.

“ControllerInterface” есть не что иное, как “проводник” между функциями класса “Controller” и объектами “View”.

```
import java.util.ArrayList;

public interface ControllerInterface {
    String parseMin(ArrayList<MyDate> container);
    String parseMax(ArrayList<MyDate> container);
}
```

Рис.7.Класс “ControllerInterface”.

Создадим меню, выбрав одну из вкладок которого, пользователь получит возможность изменить количество строк и столбцов. Слушатели для вкладок “edit rows” и “edit columns” работают аналогично: в новом диалоговом окне в редактируемое текстовое поле можно задать количество столбцов / ячеек, которые надо добавить. Контроль за вводом текста производим с помощью “NumberFormatException” и класса “BadInputException”.

```
JMenuBar menuBar = new JMenuBar();
JMenu options = new JMenu( s: "Options");
JMenuItem row = new JMenuItem( text: "edit rows");
JMenuItem col = new JMenuItem( text: "edit columns");
options.add(col);
options.add(row);
menuBar.add(options);
setJMenuBar(menuBar);

col.addActionListener(e ->{
    JTextField org = new JTextField( columns: 20);
    JPanel inputPanel = new JPanel();
    inputPanel.add(new JLabel( text: "Count:"));
    inputPanel.add(org);
    if (JOptionPane.showConfirmDialog( parentComponent: this, inputPanel,
            title: "Edit the number of columns", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
        try {
            int count = Integer.valueOf(org.getText());
            if(Math.abs(count) < rowNumber) {
                columnNumber += count;
                tableModel.setColumnCount(columnNumber);
                tableModel.setRowCount(rowNumber);
                table.setModel(tableModel);
            }
            else{
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Incorrect input");
            }
        } catch (NumberFormatException exc) {
            JOptionPane.showMessageDialog( parentComponent: null, exc.getMessage());
        }
```

Рис.8.Слушатель для изменения количества столбцов таблицы.

Класс “Controller” наследуется от “TableCellEditor”, поэтому необходимо переопределить методы “getTableCellEditor” и “getCellEditorValue”. В первый метод по умолчанию передается значение ячейки и номера столбца и строки, на пересечении которых она находится. Все данные запоминаем в соответствующие переменные класса, с которыми будем работать.

```
public Component getTableCellEditorComponent(JTable table, Object value,
                                             boolean isSelected, int row, int column) {
    currentValue = (MyDate) value;
    this.table = table;
    curCol = column;
    curRow = row;
    table.setToolTipText(MESSAGE);
    if (currentValue != null) {
        editor.setText(currentValue.getRecord());
        needToNotify = true;
    } else {
        editor.setText("");
        needToNotify = false;
    }
    return this.editor;
}
public Object getCellEditorValue() {
    MyDate date = currentValue;
    try {
        date = recordDate(editor.getText());
        if (currentValue != null)
            date.setHash(currentValue.getHash());
        currentValue = date;
        table.getModel().setValueAt(date, curRow, curCol);
        if (needToNotify) notifyTable(); }
    catch (BadInputException exc) {
        date = null;
        needToNotify = false;
    } catch (NumberFormatException exc) {
        JOptionPane.showMessageDialog( parentComponent: null, MESSAGE);
    }
    return date;
}
```

Рис.9.Обработка поступающих данных после изменения значения ячейки.

Метод “parseMaxOrMin” ищет минимальное и максимальное значение в зависимости от введенной в ячейку формулы. Формулу определяем с помощью регулярных выражений. После занесения всех объектов в массив, вызываем “Collections.min” (или “Collections.max”).

```
private MyDate parseMaxOrMin(String str) throws BadInputException, NumberFormatException {
    ArrayList<MyDate> container = new ArrayList<>();
    boolean isMin;
    if (str.charAt(1) == 'i' && str.charAt(2) == 'n') {
        isMin = true;
    } else if (str.charAt(1) == 'a' && str.charAt(2) == 'x') {
        isMin = false;
    } else throw new BadInputException("Incorrect input");
    String cells = "";
    if (str.charAt(3) != '(') throw new BadInputException("Incorrect input");
    Matcher m = Date.matcher(str);
    while (m.find()) {
        cells = m.group();
        container.add(recordFullDate(cells));
    }
    cells = str.substring(str.indexOf(cells) + cells.length());
    m = CELL.matcher(cells);
    while (m.find()) {
        MyDate myDate = recordCellDate(m.group());
        if (myDate == null) continue;
        container.add(myDate);
    }
    if (cells.charAt(cells.length() - 1) != ')') throw new BadInputException("Incorrect input");

    if (isMin){
        return Collections.min(container);
    }
    else return Collections.max(container);
}
```

Рис.10.Поиск минимального и максимального значений через ввод формулы в ячейку.

Следующие две функции - поиск минимального и максимального значений выделенных ячеек - переопределенные методы “ControllerInterface”.

```
@Override
public String parseMin(ArrayList<MyDate> container) throws BadInputException, NumberFormatException {
    for(MyDate date : container){
        while (date == null){
            container.remove(date);
        }
    }
    return Collections.min(container).toString();
}
@Override
public String parseMax(ArrayList<MyDate> container) throws BadInputException, NumberFormatException {
    for(MyDate date : container){
        while (date == null){
            container.remove(date);
        }
    }
    return Collections.max(container).toString();
}
```

Рис.11.Поиск минимального и максимального значений выделенных ячеек.

Если же на вход поступает значение, начинающееся с символа “=”, то вызываем функцию “parseEqualFormula”. В соответствии со считанным

знаком (“+” или “-”) происходит операция сложения или вычитания дат, выполняемая встроенным в “GregorianCalendar” методом “add”.

```
private MyDate parseEqualFormula(String str) throws BadInputException, NumberFormatException {
    MyDate first;
    int second;
    boolean isMinus;
    if (str.charAt(1) >= '1' && str.charAt(1) <= '9') {
        String usualDate = str.substring(1, 11);
        Matcher m = Date.matcher(usualDate);
        if (m.matches()) {
            first = recordFullDate(usualDate);
            second = Integer.valueOf(str.substring(12));
            if (str.charAt(11) == '+') isMinus = false;
            else if (str.charAt(11) == '-') isMinus = true;
            else throw new BadInputException("Incorrect input");
            if (isMinus) second *= -1;
            first.add(Calendar.DAY_OF_MONTH, second);
            return first;
        } else throw new BadInputException("Incorrect input");
    } else if (str.charAt(1) >= 'A' && str.charAt(1) <= 'Z') {
        int ind = str.indexOf('-');
        isMinus = true;
        if (ind == -1) {
            ind = str.indexOf('+');
            isMinus = false;
        }
        if (ind == -1) throw new BadInputException("Incorrect input");
        String cell = str.substring(1, ind);
        Matcher m = CELL.matcher(cell);
        if (m.matches()) {
            first = recordCellDate(cell);
            if (first == null) throw new BadInputException("Incorrect input");
            second = Integer.valueOf(str.substring(cell.length() + 2));
            if (isMinus) second *= -1;
            first.add(Calendar.DAY_OF_MONTH, second);
            return first;
        } else throw new BadInputException("Incorrect input");
    } else throw new BadInputException("Incorrect input");
```

Рис.12.Операции сложения и вычитания дат.

Если же формула не была обнаружена, а на вход поступает значение в формате даты (проверяем с помощью метода “recordUsualDate”), то вызываем функцию “recordFullDate”, которая обращается к конструктору класса “MyDate”.

```
private MyDate recordFullDate(String str) throws NumberFormatException {
    String delimiter = "[-]";
    String[] subString = str.split(delimiter);
    int day, month, year;
    day = Integer.valueOf(subString[0]);
    month = Integer.valueOf(subString[1]);
    year = Integer.valueOf(subString[2]);
    return new MyDate(year,  month: month-1, day);
}

private MyDate recordUsualDate(String str) throws NumberFormatException, BadInputException {
    Matcher m = Date.matcher(str);
    if (m.matches()) {
        return recordFullDate(str);
    } else throw new BadInputException("Incorrect input");
}
```

Рис.13.Занесение дат в ячейки.

Функция “recordCellDate” записывает в “HashSet” данные изменяемой ячейки и связывает их с объектом.

```
private MyDate recordCellDate(String str) throws NumberFormatException, BadInputException {
    int curCol = 1, curRow=1;
    if(str.length()==2){
        curCol = str.charAt(0) - 'A' + 1;
        curRow = Integer.valueOf(str.substring(1)) - 1;
    }
    else if(str.length()==3){
        curCol = str.charAt(0) - 'A'  + 27;
        curRow = Integer.valueOf(str.substring(2)) - 1;
    }

    if (curRow >= table.getModel().getRowCount() || curCol >= table.getModel().getColumnCount())
        throw new BadInputException("Incorrect input");
    if (!notifier) {
        if (curRow == this.curRow && curCol == this.curCol)
            throw new BadInputException("Incorrect input");
    }
    if (table.getValueAt(curRow, curCol) == null) throw new BadInputException("Incorrect input");
    MyDate cellDate = (MyDate) table.getValueAt(curRow, curCol);
    if (cellDate == null)
        throw new BadInputException("Incorrect input");
    MyDate date = new MyDate(cellDate);
    if (!notifier) {
        date.addUsingDependency(this.curRow, this.curCol);
    }
    return date;
}
```

Рис.14.Запись пути к ячейке определенного объекта.

Следующий метод “recordDate”, организованный конструкцией “switch-case”, - посредник между всеми предыдущими функциями.

```
private MyDate recordDate(String str) throws NumberFormatException, BadInputException {
    MyDate myDate = currentValue;
    if (str.length() >= 5) {
        switch (str.charAt(0)) {
            case 'm': {
                myDate = parseMaxOrMin(str);
                myDate.setRecord(str);
                break;
            }
            case '=': {
                myDate = parseEqualFormula(str);
                myDate.setRecord(str);
                break;
            }
            default: {
                myDate = recordUsualDate(str);
                myDate.setRecord(str);
            }
        }
    }
    else if(str.length() < 5 && str.length() >=1)
        throw new BadInputException("Incorrect input");
    return myDate;
}
```

Рис.15.Функция определения вызываемого метода согласно полученному значению ячейки.

Следующие два метода уведомляют таблицу и ячейку о том, что произошло редактирование, записывая новые значения.

```
    private void notifyCell(MyDate cell, int i, int j) throws BadInputException {
        HashSet<int[]> dependencies = cell.getHash();
        MyDate tempCell = cell;

        cell = recordDate(cell.getRecord());
        cell.setHash(tempCell.getHash());
        table.setValueAt( aValue: "", i, j);
        for (var elem : dependencies) {
            notifyCell((MyDate) table.getModel().getValueAt(elem[0], elem[1]), elem[0], elem[1]);
        }
    }

    private void notifyTable() throws BadInputException, NumberFormatException {
        notifier = true;
        MyDate cell;
        HashSet<int[]> dependencies = currentValue.getHash();
        for (var elem : dependencies) {
            cell = (MyDate) table.getModel().getValueAt(elem[0], elem[1]);
            notifyCell(cell, elem[0], elem[1]);
        }
        notifier = false;
    }
}
```

Рис.16.Запись новых значений ячейки.

Результат:

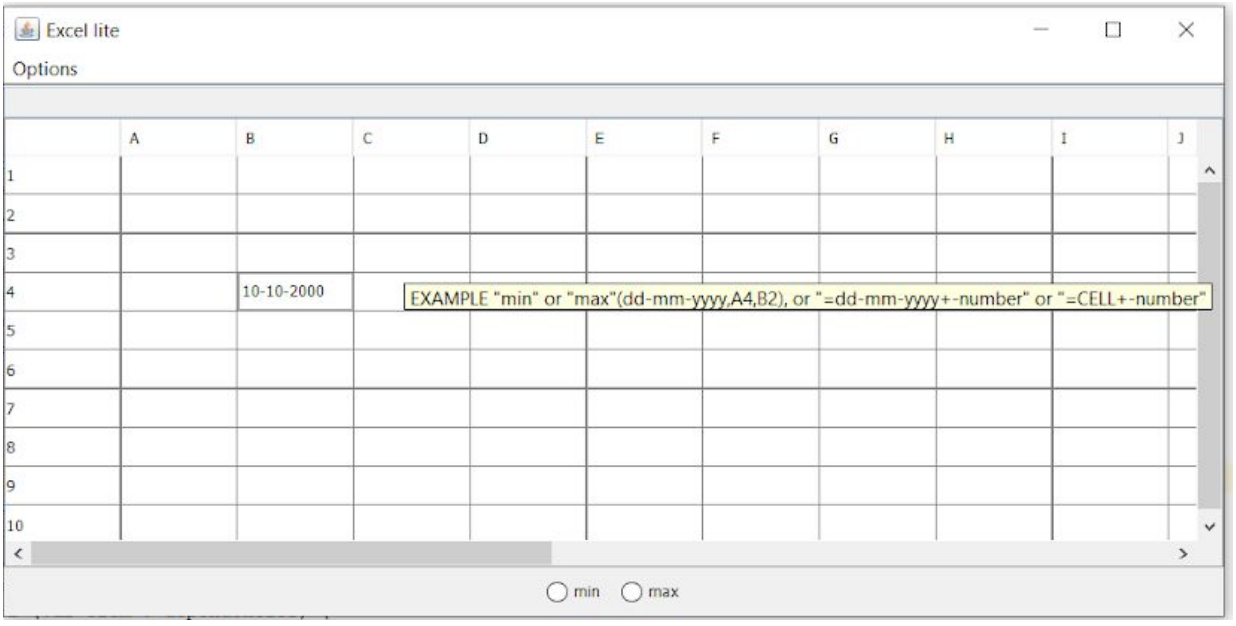

Рис.17.Вид оконного приложения после долгого удержания курсора мыши на ячейке (подсказка для ввода).

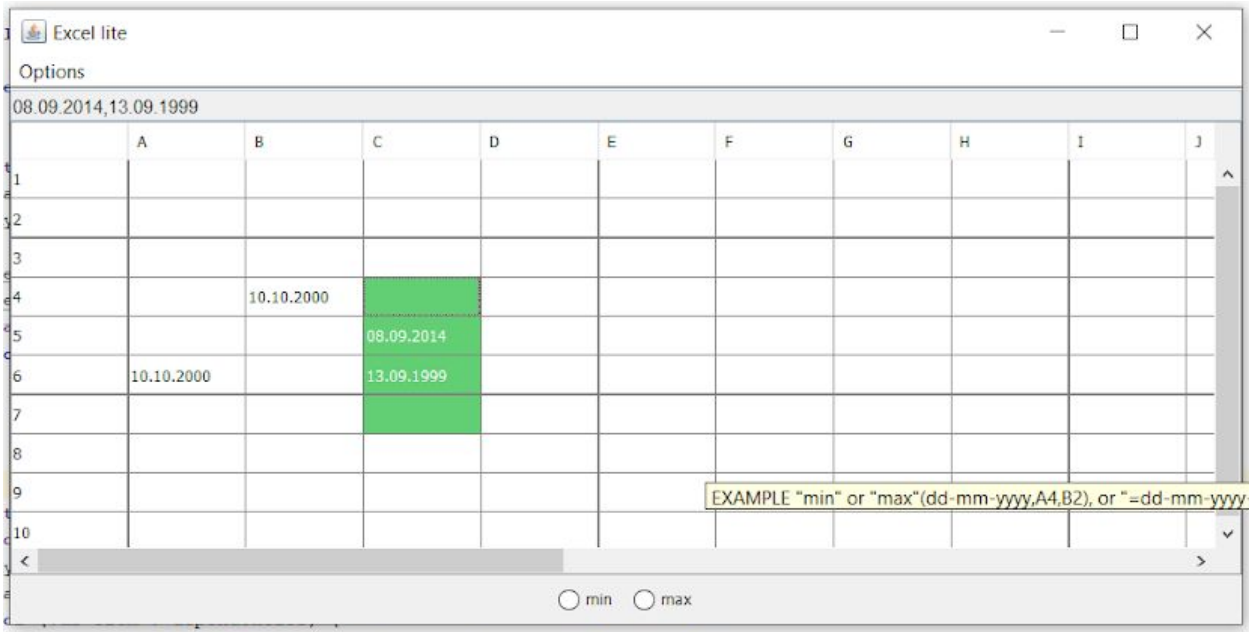

Рис.18.Результат работы программы (выделение ячеек).

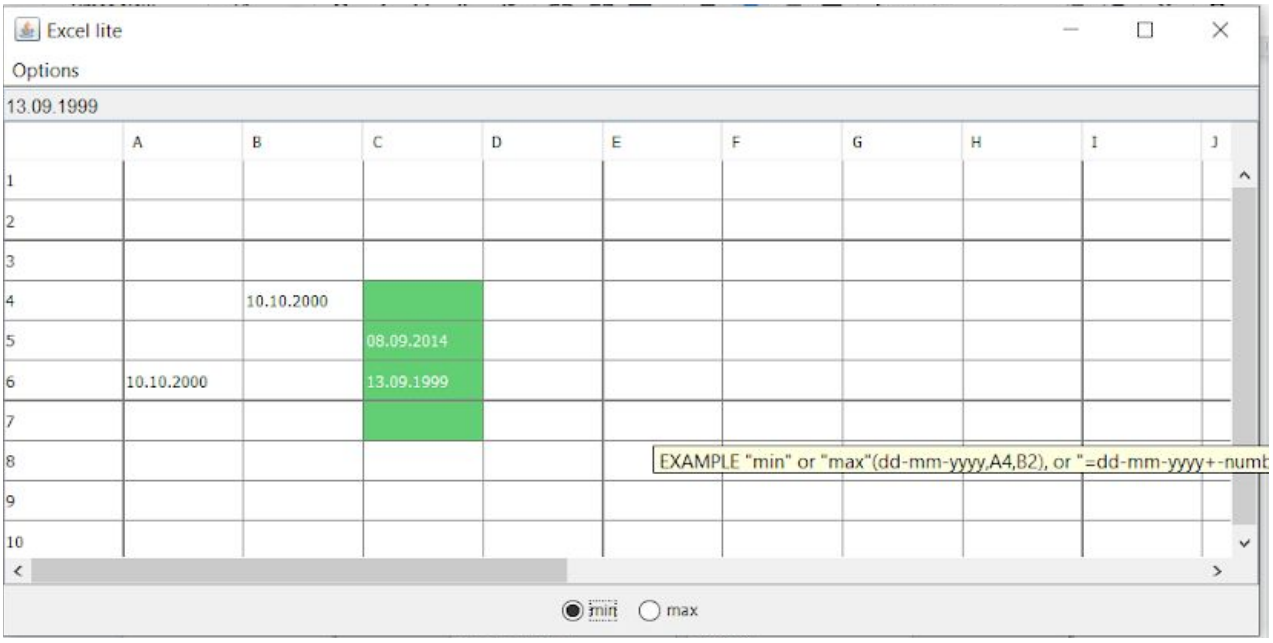

Рис.19.Результат работы программы.

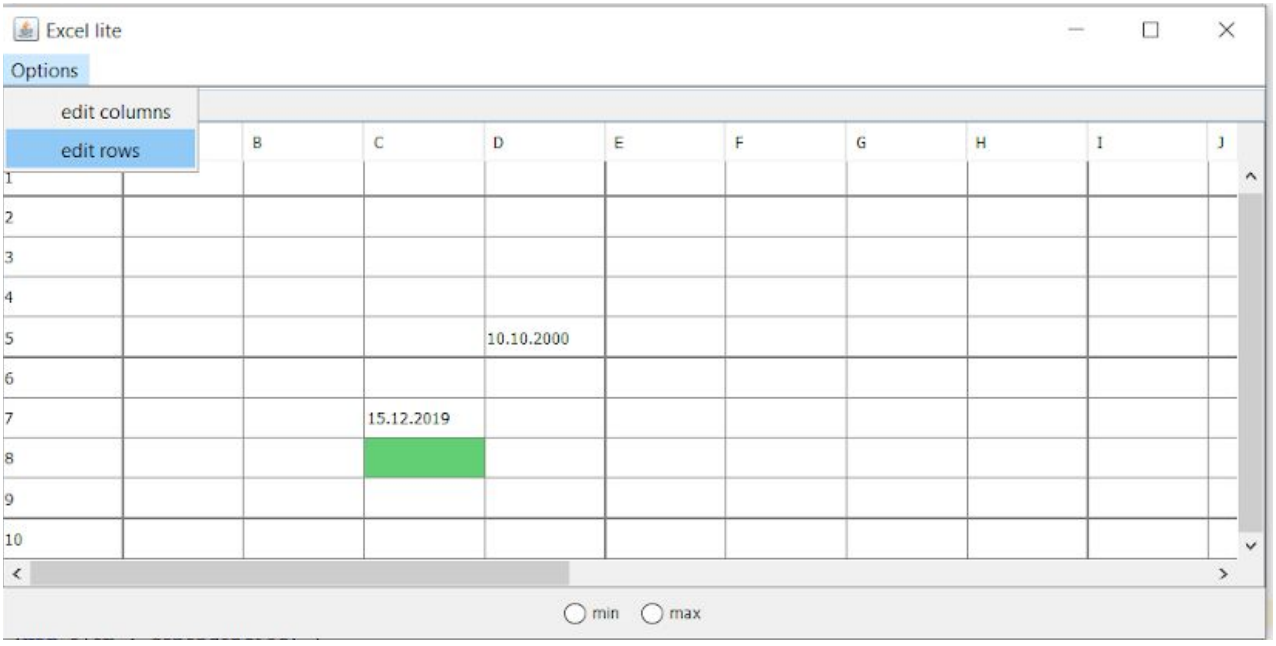

Рис.20.Результат работы программы (меню редактора количества строк/столбцов).

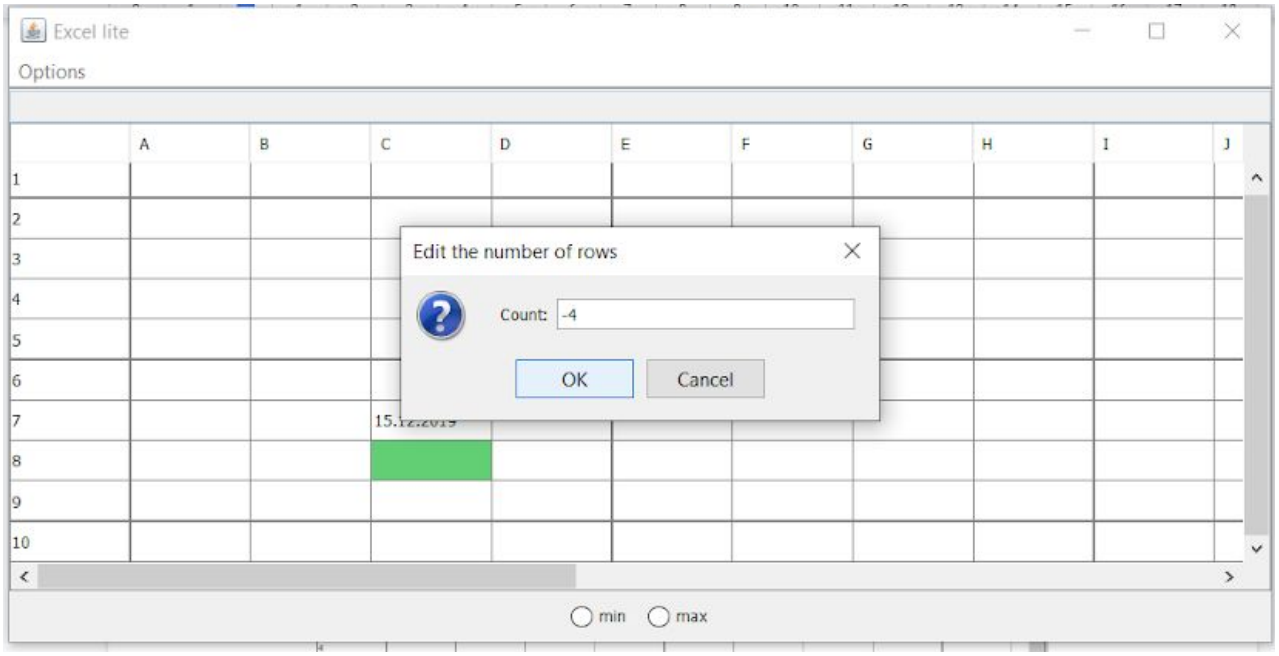

Рис.21.Результат работы программы (редактирование количества строк).

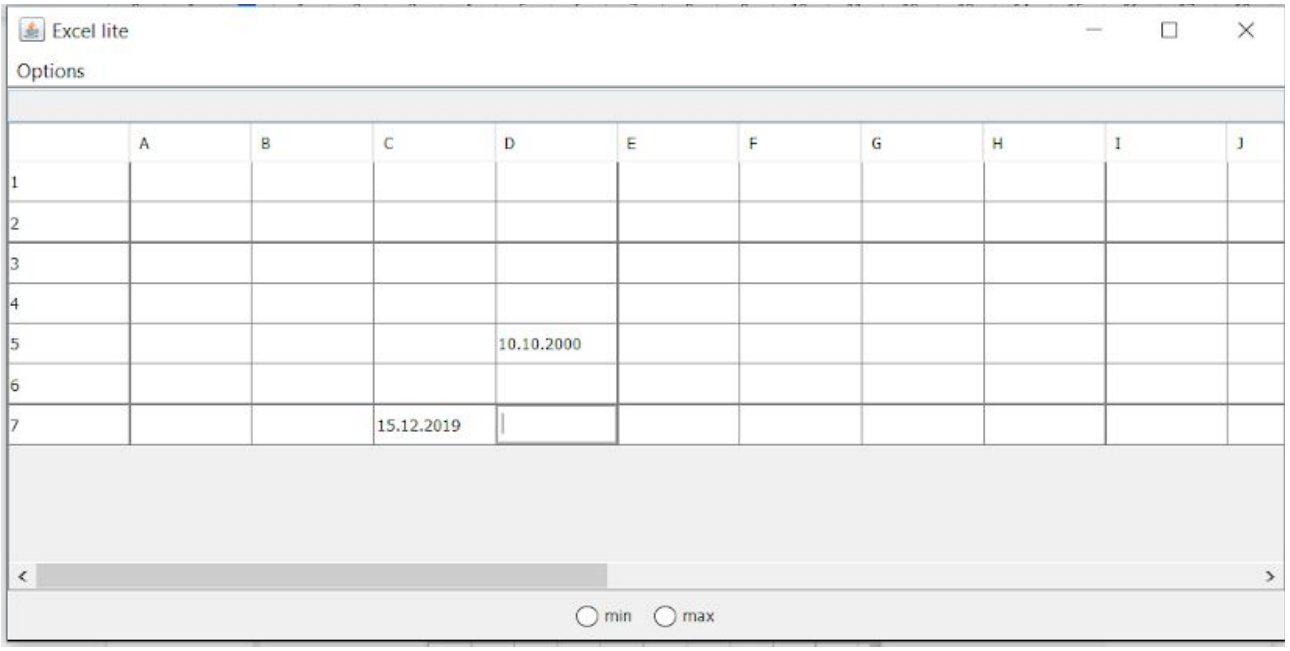

Рис.22.Результат работы программы (редактирование количества строк).