

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Информационной безопасности

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Модели безопасности компьютерных систем»
Тема: Матрица доступа.

Студенты гр. 1361

Горбунова Д. А.

Кравцов И. Ю.

Преподаватель

Шкляр Е. В.

Санкт-Петербург

2024

ТЕОРЕТИЧЕСКАЯ СПРАВКА

Матрица доступа (МД) – это готовая модель, позволяющая регламентировать доступ к информационным ресурсам компании, на основании которой можно оценить состояние и структуру защиты данных в информационных системах. В матрице четко устанавливаются права для каждого субъекта по отношению ко всем объектам информации.

Визуально это можно представить в качестве некоего массива данных со множеством ячеек, которые формируются пересечением строки, указывающей на субъект и столбика, указывающего на объект. Получается, что при таком подходе к управлению доступом ячейка содержит определенную запись, характерную для пары субъект-объект и указывает на режим доступа, разрешенный или запрещенный, или его характеристику для каждого конкретного случая. В матрице доступа к информационным ресурсам столбец отождествляется с перечнем контроля доступа, строка выполняет роль профиля доступа присущего объекту.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Задача. написать две программы для обработки матрицы доступа.

Первая программа — администратор. Реализуйте:

- Добавление, удаление и изменение субъектов и объектов (буквы и цифры), а также выдача прав на доступ к объектам.
- Ввод имён субъектов и объектов. Имена регистрозависимы, то есть G и g — разные объекты. Предусмотрите обработку ошибок при вводе.
- Сохранение и загрузку матрицы доступа из файла;
- Отображение матрицы доступа в окне программы и возможность её интерактивного изменения без перезагрузки программы.

Вторая программа — пользователь. Реализуйте:

- Ввод имени пользователя. Предусмотрите обработку ошибок.
- Ввод текстовой строки и её фильтрацию в соответствии с правами доступа для выбранного субъекта. Например, для субъекта Ivan с правами доступа к объектам AaBb, введённая строка ABCDEaF превратится в ABa.
- Предусмотрите ситуацию, когда матрица доступа меняется администратором после запуска программы.

Важно: оба приложения — оконные, с пользовательским интерфейсом. Язык программирования любой. Все необходимые для работы текстовые поля должны быть подписаны. Консольные приложения в этой работе не принимаются.

ХОД РАБОТЫ

1) В первую очередь, была реализована программа администратора, которая может добавлять, удалять, изменять субъекты и объекты(а именно буквы и цифры). Интерфейс программы представлен на рисунке 1.

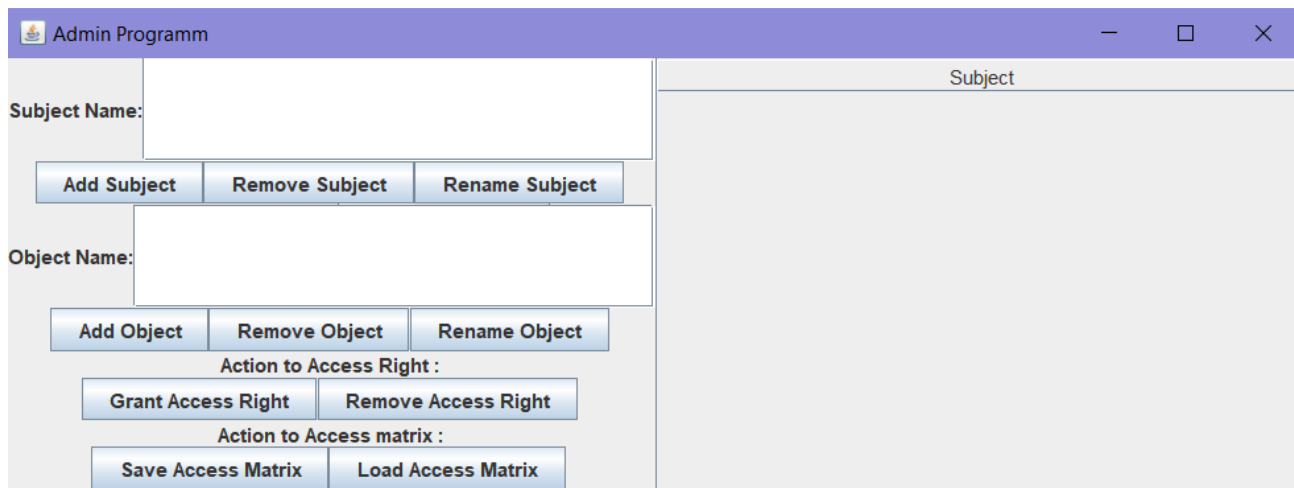


Рисунок 1 – Интерфейс программы администратора

Создадим пару объектов и субъектов для примера. Для этого нужно в соответствующие поля написать названия субъектов и объектов. После нажатия кнопок «Add subject» и «Add Object» соответственно, данные добавятся в матрицу прав доступа и отобразятся в левой части программы, где строки — это субъекты, а столбцы – объекты. Результат представлен на рисунке 2.

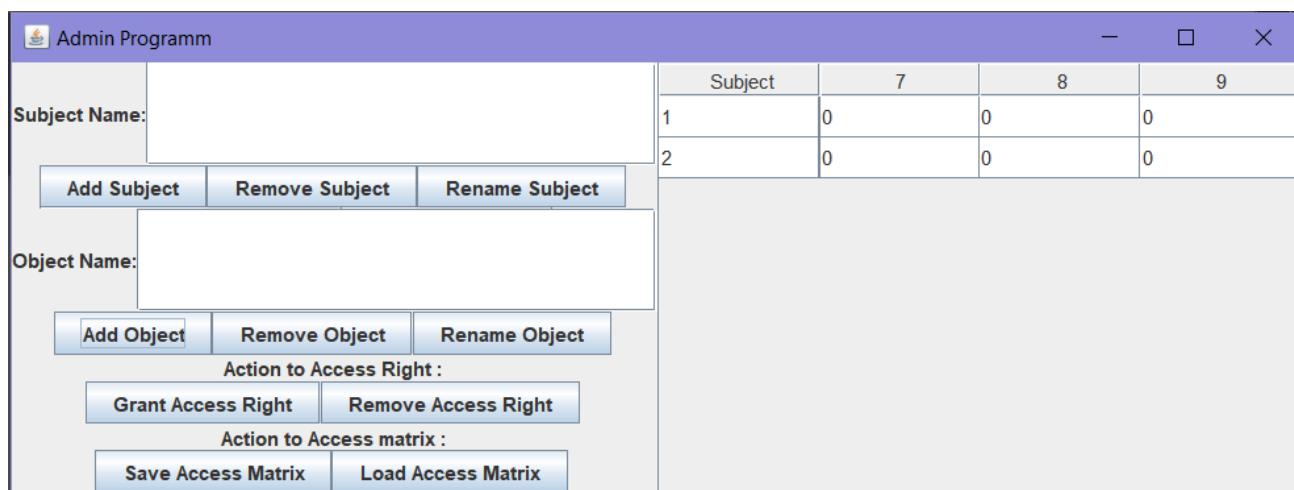


Рисунок 2 – Добавление элементов в матрицу

Как видно из рисунка 2, элементы добавились в матрицу. Теперь попробуем удалить объект «9», для этого достаточно ввести в поле «Object

Name» название элемента и нажать кнопку «Remove Object». Результат на рисунке 3.

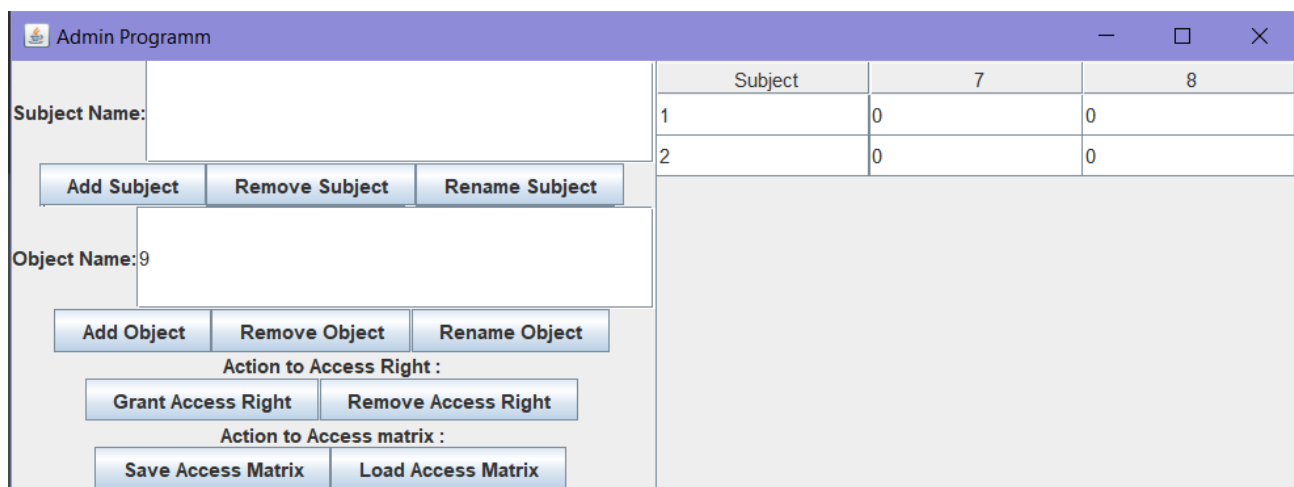


Рисунок 3 – Удаление объекта

Как видно из рисунка 3 – файл удаленся корректно. Теперь выдадим права доступа субъекту «1» к объекту «8». Для этого в соответствующие поля вводим названия элементов и нажимаем кнопку «Grant Access Right». Результат представлен на рисунке 4.

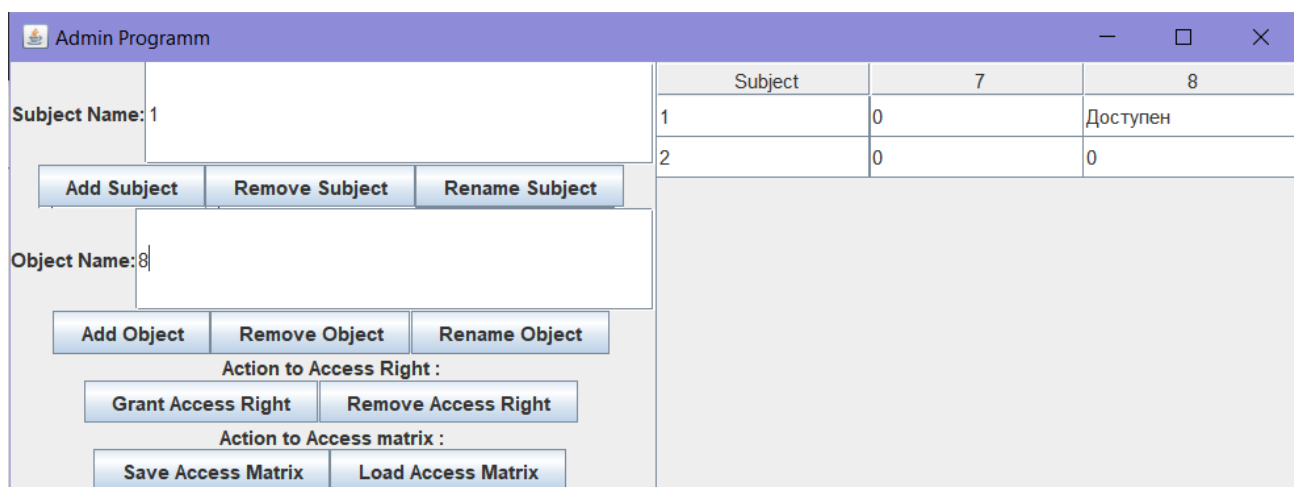


Рисунок 4 – Выдача прав доступа

Теперь попробуем убрать те права что, только что выдали. Для этого в соответствующие поля вводим названия элементов и нажимаем кнопку «Remove Access Right» Результат на рисунке 5.

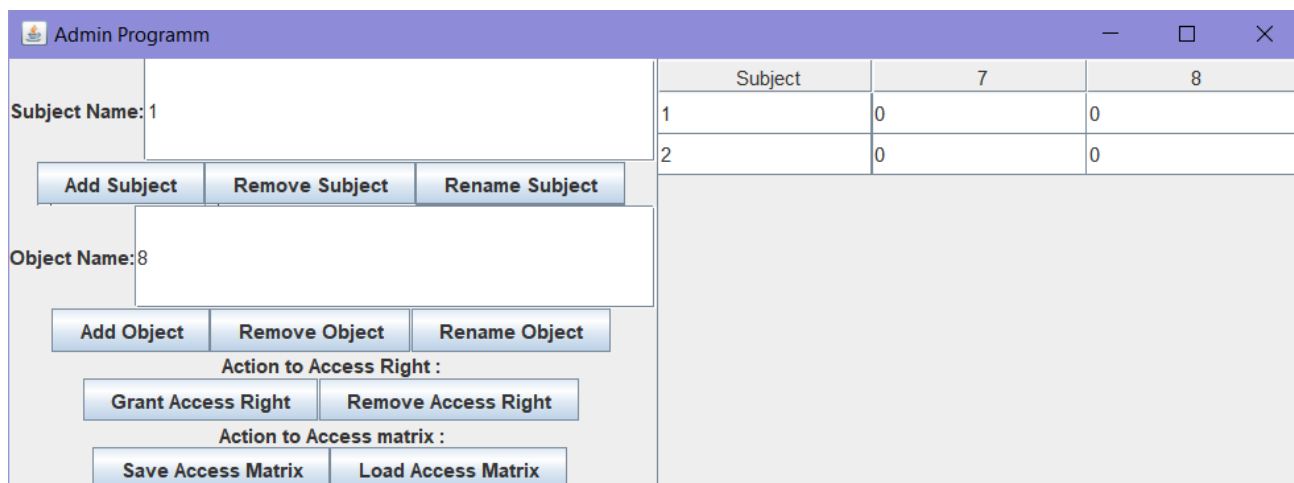


Рисунок 5 – Удаление прав доступа

Так же можно переименовать субъекты и объекты, за это отвечает кнопки «Rename Subject» и «Rename Object» соответственно. Результат после нажатия на рисунке 6.

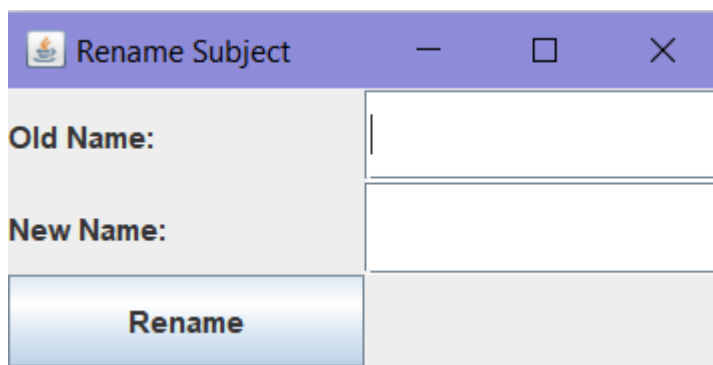


Рисунок 6 – окно кнопки «Rename Subject».

Чтобы переименовать субъект или объект нужно ввести название актуального элемента и новое название этого элемента. Нажав кнопку «Rename» изменения вступят в силу. Результат представлен на рисунках 7-9.

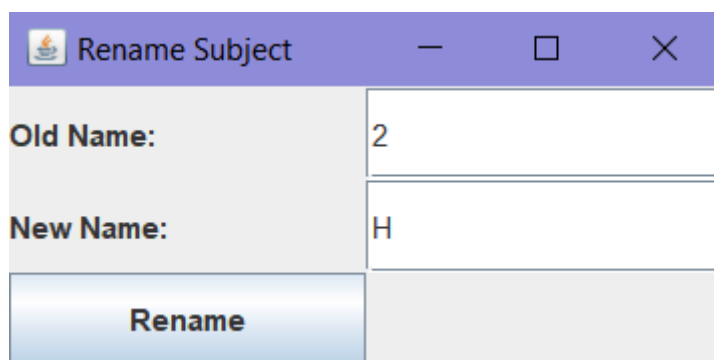


Рисунок 7 – действие переименования субъекта

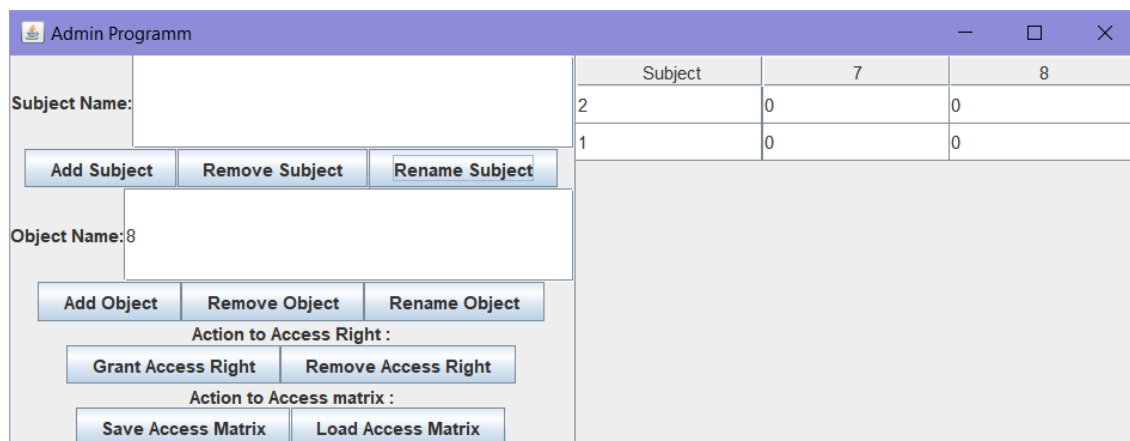


Рисунок 8 -- Предыдущая матрица прав доступа

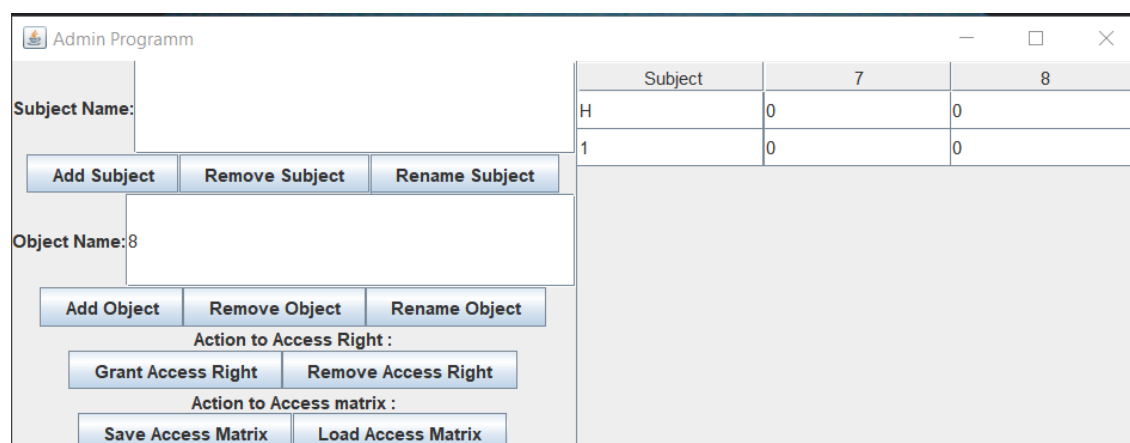


Рисунок 9 -- Нынешняя матрица прав доступа

Также в программе есть кнопки «Save Access Matrix» и «Load Access Matrix». Далее мы рассмотрим их возможности.

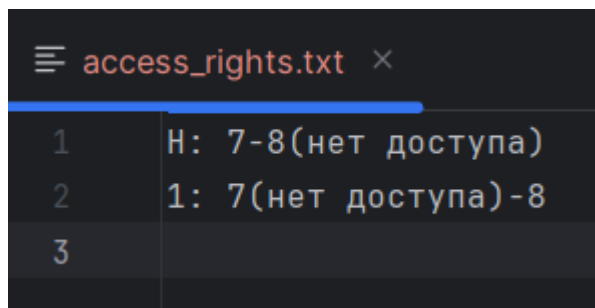
Кнопка «Save Access Matrix» позволяет сохранять в файл нынешнюю матрицу прав доступа. Пример работы данной кнопки приведен ниже на рисунках 10-12.

Subject	7	8
H	Доступен	0
1	0	Доступен

Рисунок 10 – Матрица прав доступа, для записи в файл.



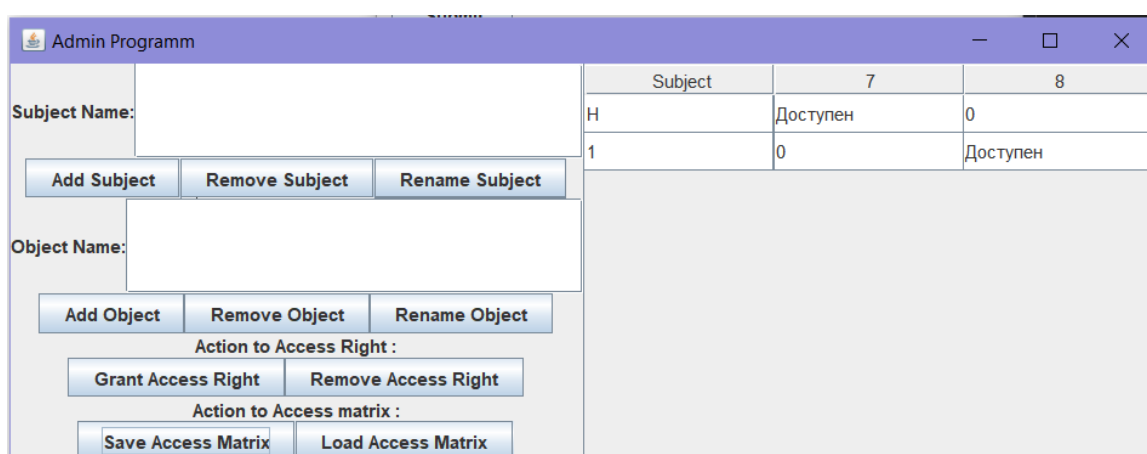
Рисунок 11 – Сообщение о выполнении действия.



1	H: 7-8(нет доступа)
2	1: 7(нет доступа)-8
3	

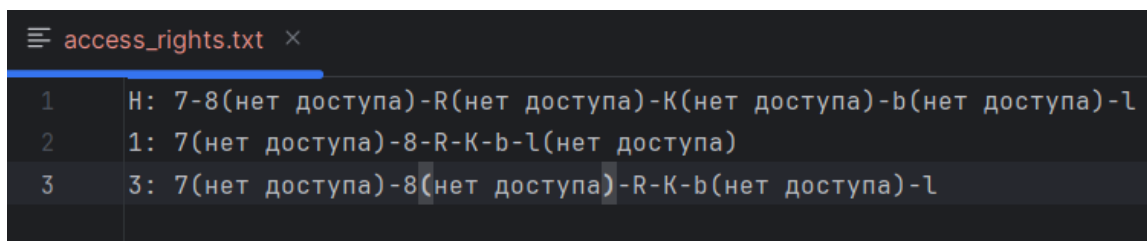
Рисунок 12 – Результат записи в файл.

Кнопка «Load Access Matrix» позволяет извлечь из файла матрицу прав доступа. Пример работы данной кнопки приведен ниже на рисунках 13-15.



Subject	7	8
H	Доступен	0
1	0	Доступен

Рисунок 13 – старая матрица прав доступа.



1	H: 7-8(нет доступа)-R(нет доступа)-K(нет доступа)-b(нет доступа)-l
2	1: 7(нет доступа)-8-R-K-b-l(нет доступа)
3	3: 7(нет доступа)-8(нет доступа)-R-K-b(нет доступа)-l

Рисунок 14 – файл, из которого будем записывать

The screenshot shows a window titled "Admin Programm". On the left, there are two input fields: "Subject Name:" and "Object Name:". Below "Subject Name:" are three buttons: "Add Subject", "Remove Subject", and "Rename Subject". Below "Object Name:" are three buttons: "Add Object", "Remove Object", and "Rename Object". Further down are two pairs of buttons: "Grant Access Right" and "Remove Access Right", and "Save Access Matrix" and "Load Access Matrix". On the right side of the window is a table representing an access matrix.

Subject	7	8	R	K	b	l
H	Доступен	0	0	0	0	Доступен
1	0	Доступен	Доступен	Доступен	Доступен	0
3	0	0	Доступен	Доступен	0	Доступен

Рисунок 15 – обновленная матрица доступа

2) Была реализована программа пользователя с графическим интерфейсом. Интерфейс предоставлен на рисунке 16.

The screenshot shows a window with a title bar. Inside, there are two input fields: "User Name:" and "Input Text:". Below these fields is a "Submit" button. The bottom half of the window is a large, empty rectangular area, likely for displaying output or a list.

Рисунок 16 – Интерфейс программы

- Теперь введём имя пользователя и строку, права доступа в которой надо выделить для пользователя (рисунок 17).

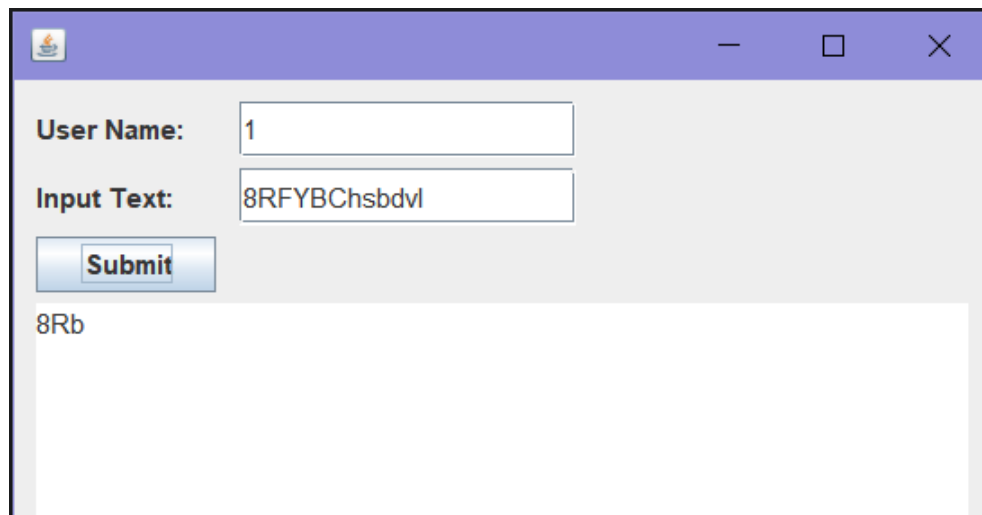


Рисунок 17 – Проверка прав доступа User.

Как видно, программа пользователя выводит только те объекты введенной строки, которые совпадают с объектами, на которые имеют права данные субъекты.

ВЫВОДЫ.

В ходе данной лабораторной работы были реализованы две программы. Первая программа задаёт объекты, субъекты, а также доступ субъектов к отдельным объектам. Вторая программа проверяет наличие прав доступа у введённого субъекта на введенные в виде строки объекты. Также в ходе лабораторной работы была рассмотрена модель матрицы доступа.

Код приложения.

1. Программа пользователя.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

public class UserProgram {
    private JFrame frame;
    private JTextField userNameField;
    private JTextField inputTextField;
    private JTextArea filteredTextArea;
    private JButton submitButton;

    // Точка входа в программу
    public static void main(String[] args) {
        // Инициализация графического интерфейса в отдельном потоке
        EventQueue.invokeLater(() -> {
            try {
                UserProgram window = new UserProgram();
                window.frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }

    // Конструктор класса UserProgram, который вызывает метод
    инициализации
    public UserProgram() {
        initialize();
    }

    // Метод инициализации, который создает и настраивает графический
    интерфейс
    private void initialize() {
        // Создание и настройка окна
```

```

frame = new JFrame();
frame.setBounds(100, 100, 450, 300);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.getContentPane().setLayout(null);

// Создание и настройка метки для имени пользователя
JLabel lblUserName = new JLabel("User Name:");
lblUserName.setBounds(10, 10, 80, 25);
frame.getContentPane().add(lblUserName);

// Создание и настройка текстового поля для ввода имени
пользователя
userNameField = new JTextField();
userNameField.setBounds(100, 10, 150, 25);
frame.getContentPane().add(userNameField);
userNameField.setColumns(10);

// Создание и настройка метки для ввода текста
JLabel lblInputText = new JLabel("Input Text:");
lblInputText.setBounds(10, 40, 80, 25);
frame.getContentPane().add(lblInputText);

// Создание и настройка текстового поля для ввода текста
inputTextField = new JTextField();
inputTextField.setBounds(100, 40, 150, 25);
frame.getContentPane().add(inputTextField);
inputTextField.setColumns(10);

// Создание и настройка кнопки для отправки текста
submitButton = new JButton("Submit");
submitButton.setBounds(10, 70, 80, 25);
frame.getContentPane().add(submitButton);

// Создание и настройка области для отображения отфильтрованного
текста
filteredTextArea = new JTextArea();
filteredTextArea.setBounds(10, 100, 414, 150);
frame.getContentPane().add(filteredTextArea);

// Добавление слушателя событий для кнопки отправки текста
submitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Получение имени пользователя и введенного текста
        String userName = userNameField.getText();
        String inputText = inputTextField.getText();

```

```

        // Фильтрация текста по правам доступа пользователя
        String filteredText = filterText(userName, inputText);
        // Отображение отфильтрованного текста
        filteredTextArea.setText(filteredText);
    }
    });
}

// Метод для фильтрации текста по правам доступа пользователя
private String filterText(String userName, String inputText) {
    // Путь к файлу с правами доступа
    String accessRightsFilePath = "access_rights.txt";
    String filteredText = "";
    try {
        // Чтение всех строк из файла с правами доступа
        List<String> lines =
Files.readAllLines(Paths.get(accessRightsFilePath));
        // Проход по каждой строке файла
        for (String line : lines) {
            // Разделение строки на части по символу ":"
            String[] parts = line.split(":");
            // Проверка, что строка содержит имя пользователя и права
доступа

            if (parts.length == 2 && parts[0].equals(userName)) {
                // Получение прав доступа пользователя
                var objects = List.of(parts[1].split("-"));
                // Фильтрация введенного текста по правам доступа
todo
                for (char obj : inputText.toCharArray()) {
                    if (objects.stream().anyMatch(o -> o.charAt(0) ==
obj && ! o.contains("("))) {
                        filteredText += obj;
                    }
                }
                break;
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
        // Обработка ошибки чтения файла
        filteredText = "Error reading access rights file.";
    }
    return filteredText;
}
}

```

2. Программа администратора.

```
import dto.MyObject;
import dto.Subject;
import tools.AccessTableModel;
import tools.IoTools;

import javax.swing.*.*;
import javax.swing.table.TableColumn;
import java.awt.*.*;

public class AccessControlGUI extends JFrame {
    private JTextField subjectNameField;
    private JTextField objectNameField;
    private JButton addSubjectButton;
    private JButton addObjectButton;
    private JButton grantAccessRightButton;
    private JButton saveAccessMatrixButton;
    private JButton loadAccessMatrixButton;
    private JButton removeSubjectButton;
    private JButton removeObjectButton;
    private JButton removeAccessRightButton;
    private JButton renameSubjectButton;
    private JButton renameObjectButton;

    private JTable accessTable;
    private AccessTableModel tableModel;
    public AccessControlGUI() {
        setLayout(new GridLayout(1, 2));
        setSize(800, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Admin Programm");

        subjectNameField = new JTextField();
        objectNameField = new JTextField();
        addSubjectButton = new JButton("Add Subject");
        addObjectButton = new JButton("Add Object");
        grantAccessRightButton = new JButton("Grant Access Right");
        saveAccessMatrixButton = new JButton("Save Access Matrix");
        loadAccessMatrixButton = new JButton("Load Access Matrix");
        removeSubjectButton = new JButton("Remove Subject");
        removeObjectButton = new JButton("Remove Object");
        removeAccessRightButton = new JButton("Remove Access Right");
        renameSubjectButton = new JButton("Rename Subject");
```

```

renameObjectButton = new JButton("Rename Object");

tableModel = new AccessTableModel();
accessTable = new JTable(tableModel);
setTableBounds();

tableModel.addTableModelListener(e -> {
    refreshTable();
});

JPanel PanelInputSubject = new JPanel();
PanelInputSubject.setLayout(new BoxLayout(PanelInputSubject,
BoxLayout.X_AXIS));
PanelInputSubject.add(new JLabel("Subject Name:"));
PanelInputSubject.add(subjectNameField);

JPanel PanelSubject = new JPanel();
PanelSubject.setLayout(new BoxLayout(PanelSubject,
BoxLayout.X_AXIS));
PanelSubject.add(addSubjectButton);
PanelSubject.add(removeSubjectButton);
PanelSubject.add(renameSubjectButton);

JPanel PanelInputObject = new JPanel();
PanelInputObject.setLayout(new BoxLayout(PanelInputObject,
BoxLayout.X_AXIS));
PanelInputObject.add(new JLabel("Object Name:"));
PanelInputObject.add(objectNameField);

JPanel PanelObject = new JPanel();
PanelObject.setLayout(new BoxLayout(PanelObject,
BoxLayout.X_AXIS));
PanelObject.add(addObjectButton);
PanelObject.add(removeObjectButton);
PanelObject.add(renameObjectButton);

JPanel PanelAccessRightLabel = new JPanel();
PanelAccessRightLabel.setLayout(new
BoxLayout(PanelAccessRightLabel, BoxLayout.X_AXIS));
PanelAccessRightLabel.add(new JLabel("Action to Access Right
:"));

JPanel PanelAccessRight = new JPanel();
PanelAccessRight.setLayout(new BoxLayout(PanelAccessRight,
BoxLayout.X_AXIS));

```



```

PanelAccessRight.add(grantAccessRightButton);
PanelAccessRight.add(removeAccessRightButton);

JPanel PanelMatrixLabel = new JPanel();
PanelMatrixLabel.setLayout(new BoxLayout(PanelMatrixLabel,
BoxLayout.X_AXIS));
PanelMatrixLabel.add(new JLabel("Action to Access matrix :"));

JPanel PanelMatrix = new JPanel();
PanelMatrix.setLayout(new BoxLayout(PanelMatrix,
BoxLayout.X_AXIS));
PanelMatrix.add(saveAccessMatrixButton);
PanelMatrix.add(loadAccessMatrixButton);

Box buttonBox = Box.createVerticalBox();
buttonBox.add(PanelInputSubject);
buttonBox.add(PanelSubject);
buttonBox.add(PanelInputObject);
buttonBox.add(PanelObject);
buttonBox.add(PanelAccessRightLabel);
buttonBox.add(PanelAccessRight);
buttonBox.add(PanelMatrixLabel);
buttonBox.add(PanelMatrix);
add(buttonBox, BorderLayout.WEST);
add(new JScrollPane(accessTable), BorderLayout.EAST);

addSubjectButton.addActionListener(e ->
{
    String subjectName = subjectNameField.getText();
    if (subjectName.length() == 1){
        if (subjectName.isEmpty()) {
            JOptionPane.showMessageDialog(null,
                "Subject name cannot be empty.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (tableModel.subjectExists(subjectName)) {
            JOptionPane.showMessageDialog(null,
                "Subject name already exists.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }
    }
}

```

```

        tableModel.addSubject(new Subject(subjectName));
        subjectNameField.setText("");
        refreshTable();}
        else {
            JOptionPane.showMessageDialog(null,
                "Subject name is long.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    });

    addObjectButton.addActionListener(e ->
    {
        String objectName = objectNameField.getText();
        if (objectName.length() == 1){
            if (objectName.isEmpty()) {
                JOptionPane.showMessageDialog(null,
                    "Object name cannot be empty.", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            if (tableModel.objectExists(objectName)) {
                JOptionPane.showMessageDialog(null,
                    "Object name already exists.", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            tableModel.addObject(new MyObject(objectName));
            objectNameField.setText("");
            refreshTable();}
            else {
                JOptionPane.showMessageDialog(null,
                    "Object name is long.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    grantAccessRightButton.addActionListener(e -> {
        String subjectName = subjectNameField.getText();
        String objectName = objectNameField.getText();
        if (subjectName.isEmpty() || objectName.isEmpty()) {
            JOptionPane.showMessageDialog(null,
                "Subject or object or right names cannot be
empty.", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
    }

```

```

        Subject subject = tableModel.findSubjectByName(subjectName);
        MyObject myObject = tableModel.findObjectByName(objectName);
        if (subject == null) {
            JOptionPane.showMessageDialog(null,
                "Subject not found.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (myObject == null) {
            JOptionPane.showMessageDialog(null,
                "Object not found.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }
        tableModel.grantAccessRight(subject, myObject);
        subjectNameField.setText("");
        objectNameField.setText("");
        refreshTable();
    });
    saveAccessMatrixButton.addActionListener(e -> {
        IoTools.saveAccessMatrixToFile(tableModel,
"access_rights.txt");
    });
    loadAccessMatrixButton.addActionListener(e -> {
        IoTools.loadAccessMatrixFromFile(tableModel,
"access_rights.txt");
        refreshTable();
    });

    removeSubjectButton.addActionListener(e -> {
        String subjectName = subjectNameField.getText();
        if (subjectName.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Subject name cannot
be empty.", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        tableModel.removeSubject(subjectName);
        subjectNameField.setText("");
        refreshTable();
    });

    removeObjectButton.addActionListener(e -> {
        String objectName = objectNameField.getText();
        if (objectName.isEmpty()) {
            JOptionPane.showMessageDialog(null,

```

```

        "Object name cannot be empty.", "Error",
JOptionPane.ERROR_MESSAGE);
        return;
    }
    tableModel.removeObject(objectName);
    objectNameField.setText("");
    refreshTable();
});

removeAccessRightButton.addActionListener(e -> {
    String subjectName = subjectNameField.getText();
    String objectName = objectNameField.getText();
    if (subjectName.isEmpty() || objectName.isEmpty()) {
        JOptionPane.showMessageDialog(null,
            "Subject and object names cannot be empty.",
"Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    tableModel.removeAccessRight(subjectName, objectName);
    subjectNameField.setText("");
    objectNameField.setText("");
    refreshTable();
});

renameSubjectButton.addActionListener(e -> renameSubject());
renameObjectButton.addActionListener(e -> renameObject());

}

private void syncColumnModel() {
    var columnModel = accessTable.getColumnModel();
    var model = accessTable.getModel();
    while (columnModel.getColumnCount() > 0) {
        columnModel.removeColumn(columnModel.getColumn(0));
    }
    for (int i = 0; i < model.getColumnCount(); i++) {
        var col = new TableColumn(i);
        col.setHeaderValue(model.getColumnName(i));
        columnModel.addColumn(col);
    }
}

private void refreshTable(){
    syncColumnModel();
    accessTable.revalidate();
}

```

```

        accessTable.repaint();
    }

    private void setTableBounds() {
        accessTable.setRowHeight(25);
        for (int i = 0; i < accessTable.getColumnCount(); i++) {
            accessTable.getColumnModel().getColumn(i).setPreferredWidth(100);
        }
    }

    private void renameSubject() {
        JFrame renameSubjectFrame = new JFrame("Rename Subject");
        renameSubjectFrame.setSize(300, 150);
        renameSubjectFrame.setLayout(new GridLayout(3, 2));

        renameSubjectFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        JLabel oldNameLabel = new JLabel("Old Name:");
        JTextField oldNameField = new JTextField();
        JLabel newNameLabel = new JLabel("New Name:");
        JTextField newNameField = new JTextField();
        JButton renameButton = new JButton("Rename");

        renameButton.addActionListener(e -> {
            String oldName = oldNameField.getText();
            String newName = newNameField.getText();
            if (!oldName.isEmpty() && !newName.isEmpty() &&
newName.length() <= 1) {
                tableModel.renameSubject(oldName, newName);
                renameSubjectFrame.dispose();
            } else {
                JOptionPane.showMessageDialog(null, "Both fields must be
filled.", "Error", JOptionPane.ERROR_MESSAGE);
            }

            refreshTable();
        });

        renameSubjectFrame.add(oldNameLabel);
        renameSubjectFrame.add(oldNameField);
        renameSubjectFrame.add(newNameLabel);
        renameSubjectFrame.add(newNameField);
        renameSubjectFrame.add(renameButton);
    }

```

```

        renameSubjectFrame.setVisible(true);
    }

    private void renameObject() {
        JFrame renameObjectFrame = new JFrame("Rename Object");
        renameObjectFrame.setSize(300, 150);
        renameObjectFrame.setLayout(new GridLayout(3, 2));

        renameObjectFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        JLabel oldNameLabel = new JLabel("Old Name:");
        JTextField oldNameField = new JTextField();
        JLabel newNameLabel = new JLabel("New Name:");
        JTextField newNameField = new JTextField();
        JButton renameButton = new JButton("Rename");

        renameButton.addActionListener(e -> {
            String oldName = oldNameField.getText();
            String newName = newNameField.getText();
            if (!oldName.isEmpty() && !newName.isEmpty() &&
newName.length() <= 1) {
                tableModel.renameObject(oldName, newName);
                renameObjectFrame.dispose();
            } else {
                JOptionPane.showMessageDialog(null, "Both fields must be
filled.", "Error", JOptionPane.ERROR_MESSAGE);
            }

            refreshTable();
        });

        renameObjectFrame.add(oldNameLabel);
        renameObjectFrame.add(oldNameField);
        renameObjectFrame.add(newNameLabel);
        renameObjectFrame.add(newNameField);
        renameObjectFrame.add(renameButton);
        renameObjectFrame.setVisible(true);

    }

    public static void main(String[] args) {
        AccessControlGUI gui = new AccessControlGUI();
        gui.setVisible(true);
    }
}

```

3. Вспомогательные классы

```
package tools;
import dto.AccessRight;
import dto.MyObject;
import dto.Subject;
import javax.swing.*;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
public class IoTools {
    private IoTools() {
    }
    public static void saveAccessMatrixToFile(AccessTableModel
tableModel, String fileName) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName))) {
            List<Subject> subjects = tableModel.getSubjects();
            List<MyObject> myObjects = tableModel.getObjects();
            for (Subject subject : subjects) {
                List<String> accessibleObjects = new ArrayList<>();
                for (MyObject myObject : myObjects) {
                    String accessRight =
tableModel.findAccessRight(subject.getName(), myObject.getName());
                    if (!"0".equals(accessRight)) {
                        accessibleObjects.add(myObject.getName());
                    } else {
                        accessibleObjects.add(myObject.getName() + "(нет
доступа)");
                    }
                }
                String objectsAccess = String.join("-",
accessibleObjects);
                writer.write(subject.getName() + ": " + objectsAccess +
"\n");
            }

            JOptionPane.showMessageDialog(null,
                "Matrix is save in " + fileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * почему мы не добавляем в список субъект из файла и объект?
     */
    public static void loadAccessMatrixFromFile(AccessTableModel
tableModel, String fileName) {
        try (BufferedReader reader = new BufferedReader(new
FileReader(fileName))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split(":");

```

```

        if (parts.length == 2) {
            String subjectName = parts[0];
            String[] objects = parts[1].split("-");
            Subject subject =
tableModel.findSubjectByName(subjectName);
            if (subject == null) {
                subject = new Subject(subjectName);
                tableModel.getSubjects().add(subject);
            }

            for (var obj : objects) {
                String objectName = obj.replace("(нет доступа)",
"".trim());
                MyObject myObject =
tableModel.findObjectByName(objectName);
                if (myObject == null) {
                    myObject = new MyObject(objectName);
                    tableModel.getObjects().add(myObject);
                }
                if (!obj.contains("(нет доступа)")) {
                    tableModel.grantAccessRight(subject,
myObject);
                }
            }
        }
    }
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

```
package tools;
```

```
import dto.AccessRight;
import dto.MyObject;
import dto.Subject;
```

```
import javax.swing.table.AbstractTableModel;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
```

```
public class AccessTableModel extends AbstractTableModel {
    private List<Subject> subjects;
    private List<MyObject> myObjects;
    private Set<AccessRight> accessRights;

    public AccessTableModel() {
        this.subjects = new ArrayList<Subject>();
        this.myObjects = new ArrayList<>();
        this.accessRights = new HashSet<>();
    }
}

```



```

    public void addSubject(Subject subject) {
        subjects.add(subject);
    }

    public void addObject(MyObject myObject) {
        myObjects.add(myObject);
    }

    public void grantAccessRight(Subject subject, MyObject myObject) {
        AccessRight accessRight = new AccessRight(subject, myObject);
        accessRights.add(accessRight);
    }

    public void removeSubject(String name) {
        subjects.removeIf(subject -> subject.getName().equals(name));
    }

    public void removeObject(String name) {
        myObjects.removeIf(myObject -> myObject.getName().equals(name));
    }

    public void removeAccessRight(String subjectName, String objectName)
    {
        accessRights.removeIf(accessRight ->
            accessRight.getSubject().getName().equals(subjectName) &&
            accessRight.getObject().getName().equals(objectName)
        );
    }

    @Override
    public int getRowCount() {
        return subjects.size();
    }

    @Override
    public int getColumnCount() {
        return myObjects.size() + 1;
    }

    @Override
    public String getColumnName(int column) {
        if (column == 0) {
            return "Subject";
        }
        return myObjects.get(column - 1).getName();
    }

    @Override
    public String getValueAt(int rowIndex, int columnIndex) {
        if (columnIndex == 0) {
            return subjects.get(rowIndex).getName();
        }
    }

```

```

        String subjectName = subjects.get(rowIndex).getName();
        String objectName = myObjects.get(columnIndex - 1).getName();
    }

    @Override
    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return false;
    }

    @Override
    public void setValueAt(Object aValue, int rowIndex, int columnIndex)
{ // Метод для установки значения ячейки
    }

    public List<Subject> getSubjects() {
        return subjects;
    }

    public List<MyObject> getObjects() {
        return myObjects;
    }

    public Set<AccessRight> getAccessRights() {
        return accessRights;
    }

    public String findAccessRight(String subjectName, String objectName)
{
        for (AccessRight accessRight : accessRights) {
            if (accessRight.getSubject().getName().equals(subjectName) &&
                accessRight.getObject().getName().equals(objectName))
{return "Доступен";}
            }
        return "0";
    }

    public Subject findSubjectByName(String name) {
        for (Subject subject : subjects) {
            if (subject.getName().equals(name)) {
                return subject;
            }
        }
        return null;
    }

    public MyObject findObjectByName(String name) {
        for (MyObject myObject : myObjects) {
            if (myObject.getName().equals(name)) {
                return myObject;
            }
        }
        return null;
    }
}

```

```

public boolean subjectExists(String name) {
    for (Subject subject : subjects) {
        if (subject.getName().equals(name)) {
            return true;
        }
    }
    return false;
}

public boolean objectExists(String name) {
    for (MyObject myObject : myObjects) {
        if (myObject.getName().equals(name)) {
            return true;
        }
    }
    return false;
}

public void renameSubject(String oldName, String newName) {
    Subject subjectToRename = findSubjectByName(oldName);
    if (subjectToRename != null) {
        if (!subjectExists(newName)) {
            subjectToRename.setName(newName);
            updateSubjectReferences(oldName, newName);
        } else {
            System.out.println("A subject with the new name already
exists.");
        }
    } else {
        System.out.println("Subject not found.");
    }
}

public void renameObject(String oldName, String newName) {
    MyObject objectToRename = findObjectByName(oldName);
    if (objectToRename != null) {
        if (!objectExists(newName)) {
            objectToRename.setName(newName);
            updateObjectReferences(oldName, newName);
        } else {
            System.out.println("An object with the new name already
exists.");
        }
    } else {
        System.out.println("Object not found.");
    }
}

private void updateSubjectReferences(String oldName, String newName)
{
    for (AccessRight accessRight : accessRights) {
        if (accessRight.getSubject().getName().equals(oldName)) {
            accessRight.getSubject().setName(newName);

```

```

        }
    }
}

private void updateObjectReferences(String oldName, String newName) {
    for (AccessRight accessRight : accessRights) {
        if (accessRight.getObject().getName().equals(oldName)) {
            accessRight.getObject().setName(newName);
        }
    }
}
}

package dto;

public class Subject {
    private String name;

    public Subject(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {return name;}
}

package dto;

public class MyObject {
    private String name;

    public MyObject(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {return name;}
}

```

```

package dto;

import dto.MyObject;

import java.util.Objects;

public class AccessRight {
    private Subject subject;
    private MyObject myObject;

    public AccessRight(Subject subject, MyObject myObject) {
        this.subject = subject;
        this.myObject = myObject;
    }

    public Subject getSubject() {
        return subject;
    }

    public MyObject getObject() {
        return myObject;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        AccessRight that = (AccessRight) o;
        return subject.getName().equals(that.subject.getName())
            && myObject.getName().equals(that.myObject.getName());
    }

    @Override
    public int hashCode() {
        return Objects.hash(subject.getName(), myObject.getName());
    }
}

```