Университет ИТМО
Кафедра вычислительной техники

Лабораторная работа №6
по дисциплине «Программирование»
Вариант: 900.

Выполнил: Юсюмбели Владислав Иванович
Проверил: Письмак Алексей Евгеньевич

1. Для вывода списка объектов должен использоваться компонент (виджет) TreeView.
2. Необходимо реализовать возможность сортировки и фильтрации значений в таблице по каждому из столбцов.
3. Интерфейс приложения должен использовать layout VBox, и содержать, как минимум, следующие компоненты (виджеты):

   ListView

   ColorPicker

   FileChooser

   Label
4. Все действия над коллекцией, реализованные в предыдущей лабораторной работе, в новой версии программы должны быть реализованы в виде отдельных кнопок, полей ввода или пунктов меню в интерфейсе (по согласованию с преподавателем).
5. При запуске приложения коллекция должна автоматически заполняться значениями из файла. Все операции файлового ввода / вывода должны выполняться в отдельных потоках.

Исходный код программы:

class lab5

```
import GUI.LoginWindow;
import javafx.application.Application;
import javafx.stage.Stage;
public class lab5 extends Application {
    public static void main(String[] args) {
        launch(args);
    }
    @Override
    public void start(Stage primaryStage) throws Exception {
        new LoginWindow(primaryStage).run();
    }
}
```

class AddObjects

```
package commands;
```

class AddObjects

```
import com.google.gson.Gson;
```

```
import com.google.gson.GsonBuilder;
import com.google.gson.JsonSyntaxException;

import java.io.*;

import GUI.*;
import deprecated.People;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.stage.Stage;

/**
 * Created by slavik on 22.02.17.
 */

public class AddObjects {
    private static Gson gson = new GsonBuilder().create();
    private Stage dataStage = null;
    private TextField keyTextField = new TextField();


    /**
     * Команда: add_if_max.
     * Добавляет новый элемент в коллекцию, если его значение превышает значение наибольшего элемента этой
коллекции.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     */
    public void addIfMax(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readDataFromTextField(1, peopleTree);
        } else {
            dataStage.toFront();
        }
    }

    /**
     * Команда add_if_min.
     * Добавляет новый элемент в коллекцию, если его значение меньше, чем у наименьшего элемента этой
коллекции.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 2.0
     */
    public void addIfMin(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readDataFromTextField(2, peopleTree);
        } else {
            dataStage.toFront();
        }
    }

    private void readDataFromTextField(int min, TreeView<Container> peopleTree) {
        dataStage = new Stage();


        Label keyLabel = new Label("Please, enter Object");
        keyLabel.setFont(Font.font("Helvetica", FontWeight.LIGHT, 16));

        keyTextField.setPromptText("{name=\"name\";age=1}");


        Button buttonOK = new Button("OK");
        HBox buttonOKHBox = new HBox(buttonOK);
        buttonOKHBox.setPadding(new Insets(0, 9, 0, 255));
```

```
        buttonOK.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ENTER) {
                addToCollection(min, peopleTree, keyTextField);
            }
        });

        buttonOK.setOnMouseClicked(event -> addToCollection(min, peopleTree, keyTextField));

        VBox enterKeyVBox = new VBox(keyLabel, keyTextField, buttonOKHBox);
        enterKeyVBox.setSpacing(5);

        dataStage.setTitle("Object");
        dataStage.centerOnScreen();
        dataStage.setScene(new Scene(enterKeyVBox, 300, 90));
        dataStage.toFront();
        dataStage.setMaximized(false);
        dataStage.setResizable(false);
        dataStage.show();
        dataStage.setOnCloseRequest(event -> dataStage = null);
    }

    private void addToCollection(int min, TreeView<Container> peopleTree, TextField keyTextField) {
        String data = keyTextField.getText();
        try {
            People people = gson.fromJson(data, People.class);
            if (people == null) {
                throw new NullPointerException();
            }

            boolean flag = true;

            switch (min) {
                case 1:
                    for (People peopleCollection : Storage.getInstanceOf().getFamily().values()) {
                        if (peopleCollection.getAge() > people.getAge()) {
                            flag = false;
                            break;
                        }
                    }
                    break;
                case 2:
                    for (People peopleCollection : Storage.getInstanceOf().getFamily().values()) {
                        if (peopleCollection.getAge() < people.getAge()) {
                            flag = false;
                            break;
                        }
                    }
                    break;
            }


            if (flag) {
                switch (min) {
                    case 1: {
                        Storage.getInstanceOf().getFamily().put(String.valueOf(Storage.getInstanceOf().getFamily().size()),
people);
                        break;
                    }
                    case 2: {
                        Storage.getInstanceOf().getFamily().put(String.valueOf(Storage.getInstanceOf().getFamily().size()),
people);
                    }
                }
                new ShowAlert(Alert.AlertType.INFORMATION,"Done", "Объект успешно добавлен \n ");
            } else {
                new ShowAlert(Alert.AlertType.INFORMATION, "Done", "Объект не добавлен\n ");
            }
        } catch (JsonSyntaxException ex) {
            new ShowAlert(Alert.AlertType.ERROR, "Error", "Не удалось распознать объект,\nпроверьте корректность
данных");
        } catch (NullPointerException ex) {
            new ShowAlert(Alert.AlertType.ERROR, "Error", "\nНе ввели данные об объекте");
        }
```

```
        peopleTree.setRoot(MainWindow.getTreeForPeople());
        dataStage.close();
        dataStage = null;
    }
}
```

class ImportObjects

```
package commands;

import GUI.Container;
import GUI.NoFileSelectedExceprion;
import javafx.scene.control.Alert;
import javafx.scene.control.TreeView;
import javafx.stage.FileChooser;

import java.io.File;

public class ImportObjects {

    /**
     * Команда import.
     * добавляет в коллекцию все данные из файла.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     */
    public void importAllFromFile(TreeView<Container> peopleTree) {
        FileChooser fileChooser = new FileChooser();
        File selectedFile = fileChooser.showOpenDialog(null);

        if (selectedFile == null) {
            try {
                throw new NoFileSelectedExceprion("Произошла ошибка, не был выбран файл\n ");
            } catch (NoFileSelectedExceprion noFileSelected) {
                new ShowAlert(Alert.AlertType.ERROR, "Error", noFileSelected.getMessage());
            }
        } else {
            ReadFromFile readFromFile = new ReadFromFile();
            readFromFile.fieldData(selectedFile, peopleTree);
            Thread thread = new Thread(readFromFile);
            thread.start();
            try {
                thread.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
//            new Thread(readFromFile).start();
        }
    }
}
```

class IsertObjects

```
package commands;

import GUI.Container;
import GUI.MainWindow;
import GUI.Storage;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonSyntaxException;
import deprecated.People;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
```

```
import javafx.scene.text.FontWeight;
import javafx.stage.Stage;

public class InsertObject {
    private static Gson gson = new GsonBuilder().create();
    private Stage dataStage = null;

    /**
     * Команда insert.
     * Добавляет новый элемент с заданным ключом.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     */
    public void insertNewObject(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readKeyAndObject(peopleTree);
        } else {
            dataStage.toFront();
        }
    }


    private void readKeyAndObject(TreeView<Container> peopleTree) {
        dataStage = new Stage();


        Label keyLabel = new Label("Please, enter Data");
        keyLabel.setFont(Font.font("Helvetica", FontWeight.LIGHT, 16));


        TextField keyTextField = new TextField();
        keyTextField.setPromptText("Key");

        TextField objectTextField = new TextField();
        objectTextField.setPromptText("{name=\"name\";age=1}");


        Button buttonOK = new Button("OK");
        HBox buttonOKHBox = new HBox(buttonOK);
        buttonOKHBox.setPadding(new Insets(0, 9, 0, 255));


        buttonOK.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ENTER) {
                insertToCollection(peopleTree, objectTextField, keyTextField);
            }
        });


        buttonOK.setOnMouseClicked(event -> insertToCollection(peopleTree, objectTextField, keyTextField));

        VBox enterKeyVBox = new VBox(keyLabel, keyTextField, objectTextField, buttonOKHBox);
        enterKeyVBox.setSpacing(5);

        dataStage.setTitle("Object");
        dataStage.centerOnScreen();
        dataStage.setScene(new Scene(enterKeyVBox, 300, 120));
        dataStage.toFront();
        dataStage.setMaximized(false);
        dataStage.setResizable(false);
        dataStage.show();
        dataStage.setOnCloseRequest(event -> dataStage = null);
    }

    private void insertToCollection(TreeView<Container> peopleTree, TextField objectTextField, TextField
keyTextField) {
        String object = objectTextField.getText();
        String key = keyTextField.getText();

        try {
            People people = gson.fromJson(object, People.class);
            if (people == null) {
```

```
            throw new NullPointerException();
        }
        Storage.getInstanceOf().getFamily().put(key, people);
        peopleTree.setRoot(MainWindow.getTreeForPeople());
        new ShowAlert(Alert.AlertType.INFORMATION, "Done", "\nОбъект успешно добавлен");
      } catch (NullPointerException ex) {
        new ShowAlert(Alert.AlertType.ERROR,"Error", "Не ввели данные об объекте");
      } catch (JsonSyntaxException ex) {
        new ShowAlert(Alert.AlertType.ERROR, "Error", "Не удалось распознать объект, \nпроверьте корректность
данных");
      }
      dataStage.close();
      dataStage = null;
    }
}
```

class OtherMethods

```
package commands;

import GUI.Container;
import GUI.MainWindow;
import GUI.Storage;
import deprecated.People;
import deprecated.Place;
import javafx.scene.control.Alert;
import javafx.scene.control.TreeView;
import javafx.stage.Stage;

/**
 * Created by slavik on 05.04.17.
 */
public class OtherMethods {
    private Stage dataStage = null;


    /**
     * Команда clear.
     * Очищает коллекцию.
     *@param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     * @since 1.0
     */
    public void clear(TreeView<Container> peopleTree) {
        int size = Storage.getInstanceOf().getFamily().size();
        Storage.getInstanceOf().getFamily().clear();
        new ShowAlert(Alert.AlertType.INFORMATION, "Done", "Коллекция очищена. \nУдалено " + size + "
объектов");
        peopleTree.setRoot(MainWindow.getTreeForPeople());
    }

    /**
     * Команда save.
     * Сохраняет весь объект типа {@link Storage} в файл.
     *
     * @version 3.0
     */
    public void save() {
        new Thread(new SaveDataToFile()).start();
        new ShowAlert(Alert.AlertType.INFORMATION, "Done", "Коллекция 'family' будет сохранена в файл:
\nobjects");
    }

    /**
     * Команда load.
     * Загружает дефолтные объекты типа {@link Storage} данные в коллекцию.
     *
     *@param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     */
    public void loadDefaultObjects(TreeView<Container> peopleTree) {
```

```
        People x = new People();

        for (int i = 0; i < Storage.getInstanceOf().getAllPlaces(); i++) {
            Storage.getInstanceOf().getPlaces().add(new Place());
        }

        x.setName("Малыш");
        x.setAge((byte) 7);
        Storage.getInstanceOf().getPlaces().get(0).setFull(x);
        Storage.getInstanceOf().getFamily().put("0", x);

        x = new People();
        x.setName("мама");
        x.setAge((byte) 31);
        Storage.getInstanceOf().getPlaces().get(1).setFull(x);
        Storage.getInstanceOf().getFamily().put("1", x);
        Storage.getInstanceOf().getFamilyOfChild().put(0, x);

        x = new People();
        x.setName("Папа");
        x.setAge((byte) 32);
        Storage.getInstanceOf().getPlaces().get(2).setFull(x);
        Storage.getInstanceOf().getFamily().put("2", x);
        Storage.getInstanceOf().getFamilyOfChild().put(1, x);

        x = new People();
        x.setName("Босс");
        x.setAge((byte) 23);
        Storage.getInstanceOf().getPlaces().get(3).setFull(x);
        Storage.getInstanceOf().getFamily().put("3", x);
        Storage.getInstanceOf().getFamilyOfChild().put(2, x);

        x = new People();
        x.setName("Бетан");
        x.setAge((byte) 26);
        Storage.getInstanceOf().getPlaces().get(4).setFull(x);
        Storage.getInstanceOf().getFamily().put("4", x);
        Storage.getInstanceOf().getFamilyOfChild().put(3, x);

        x = new People();
        x.setName("Бок");
        x.setAge((byte) 38);
        Storage.getInstanceOf().getPlaces().get(5).setFull(x);
        Storage.getInstanceOf().getFamily().put("5", x);

        x = new People();
        x.setName("Фрид");
        x.setAge((byte) 45);
        Storage.getInstanceOf().getPlaces().get(6).setFull(x);
        Storage.getInstanceOf().getFamily().put("6", x);

        new ShowAlert(Alert.AlertType.INFORMATION, "Done","\nДанные загружены \n");
        peopleTree.setRoot(MainWindow.getTreeForPeople());
    }
}
```

class ReadFromFile

```
package commands;

import GUI.Container;
import GUI.MainWindow;
import GUI.Storage;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonSyntaxException;
import com.google.gson.reflect.TypeToken;
import deprecated.People;
import javafx.scene.control.TreeView;
import old.school.Botherable;
import old.school.Chatable;
import old.school.InterfaceAdapter;
```

```
import old.school.Missable;

import java.io.*;
import java.lang.reflect.Type;
import java.util.Map;
import java.util.Properties;

public class ReadFromFile implements Runnable {
    private TreeView<Container> peopleTree;
    private File selectedFile;

    public synchronized void fieldData(File selectedFile, TreeView<Container> peopleTree) {
        this.peopleTree = peopleTree;
        this.selectedFile = selectedFile;
    }

    @Override
    public synchronized void run() {
        GsonBuilder builder = new GsonBuilder();
        builder.registerTypeAdapter(Missable.class, new InterfaceAdapter<People>());
        builder.registerTypeAdapter(Chatable.class, new InterfaceAdapter<People>());
        builder.registerTypeAdapter(Botherable.class, new InterfaceAdapter<People>());
        Gson gson = builder.create();

        Properties runProperties = getProperties(ReadFromFile.class.getResourceAsStream("/properties/run.properties"));
        try (PrintWriter errorWriter = new PrintWriter(runProperties.getProperty("file.err"))) {
            try (FileReader reader = new FileReader(selectedFile)) {
                int c;
                StringBuilder string = new StringBuilder();
                while ((c = reader.read()) != -1) {
                    string.append((char) c);
                }

                int size = Storage.getInstanceOf().getFamily().size();
                Type typeMap = new TypeToken<Map<String, People>>() {
                }.getType();
                Storage.getInstanceOf().getFamily().putAll(gson.fromJson(string.toString(), typeMap));
                errorWriter.println(Storage.getInstanceOf().getFamily().size() - size + " объекта считано с файла:" + "\n" +
selectedFile.getName());
                peopleTree.setRoot(MainWindow.getTreeForPeople());
            } catch (JsonSyntaxException e) {
                errorWriter.println("Не удалось распознать объект, \nпроверьте корректность данных");
            } catch (IOException ex) {
                errorWriter.println("Произошла ошибка при чтении файла");
            } catch (NullPointerException ex) {
                errorWriter.println("Произошла ошибка, возможно файл пуст\n ");
            }
        }catch (FileNotFoundException e) {
            //do nothing
        }
    }

    private Properties getProperties(InputStream file) {
        Properties properties = new Properties();

        try {
            properties.load(file);
        } catch (IOException e) {
            //do nothing
        }
        return properties;
    }
}
```

class RemoveObjects

```
package commands;

import GUI.Container;
import GUI.MainWindow;
import GUI.Storage;
import com.google.gson.Gson;
```

```
import com.google.gson.GsonBuilder;
import com.google.gson.JsonSyntaxException;
import deprecated.People;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.stage.Stage;
import javafx.scene.control.Alert;

import java.io.PrintWriter;
import java.util.Map;
import java.util.function.Predicate;

/**
 * Created by slavik on 21.02.17.
 */
public class RemoveObject {
    private static Gson gson = new GsonBuilder().create();
    private static PrintWriter printWriter = new PrintWriter(System.out, true);
    private String data;
    private People people;
    private Stage dataStage = null;
    private TextField keyTextField = new TextField();
    private Button buttonOK = new Button("OK");

    /**
     * Команда: remove_greater_key.
     * Удаляет из коллекции все элементы, ключ которых превышает заданный.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     */
    public void removeGreaterKey(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readDataFromTextField("Key");
        } else {
            dataStage.toFront();
        }
        buttonOK.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ENTER) {
                removeFromCollectionWithKey(peopleEntry -> peopleEntry.getKey().compareTo(data) > 0, peopleTree,
keyTextField);
            }
        });

        buttonOK.setOnMouseClicked(event -> removeFromCollectionWithKey(peopleEntry ->
peopleEntry.getKey().compareTo(data) > 0, peopleTree, keyTextField));

    }

    /**
     * Команда remove.
     * Удаляет элемент из коллекции по его ключу.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     */
    public void removeWithKey(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readDataFromTextField("Key");
        } else {
            dataStage.toFront();
        }

        buttonOK.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ENTER) {
                removeFromCollectionWithKey(peopleEntry -> peopleEntry.getKey().equals(data), peopleTree,
keyTextField);
```

```
            }
        });

        buttonOK.setOnMouseClicked(event -> removeFromCollectionWithKey(peopleEntry ->
peopleEntry.getKey().equals(data), peopleTree, keyTextField));

    }

    /**
     * Команда remove_greater.
     * Удаляет из коллекции все элементы, превышающие заданный.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     * @since 1.0
     */
    public void removeGreater(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readDataFromTextField("{name=\"name\";age=1}");
        } else {
            dataStage.toFront();
        }

        buttonOK.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ENTER) {
                removeFromCollectionWithObject(peopleEntry -> peopleEntry.getValue().getAge() > people.getAge(),
peopleTree, keyTextField);
            }
        });

        buttonOK.setOnMouseClicked(event -> removeFromCollectionWithObject(peopleEntry ->
peopleEntry.getValue().getAge() > people.getAge(), peopleTree, keyTextField));

    }

    /**
     * Команда remove_all.
     * Удалят из коллекции все элементы, эквивалентные заданному.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     * @since 1.0
     */
    public void removeAll(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readDataFromTextField("{name=\"name\";age=1}");
        } else {
            dataStage.toFront();
        }

        buttonOK.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ENTER) {
                removeFromCollectionWithObject(peopleEntry -> peopleEntry.getValue().getAge() == people.getAge(),
peopleTree, keyTextField);
            }
        });

        buttonOK.setOnMouseClicked(event -> removeFromCollectionWithObject(peopleEntry ->
peopleEntry.getValue().getAge() == people.getAge(), peopleTree, keyTextField));

    }

    /**
     * Команда remove_lower.
     * Удаляет из коллекции все элементы, меньшие, чем заданный.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 3.0
     * @since 1.0
     */
    public void removeLowerObject(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readDataFromTextField("{name=\"name\";age=1}");
```

```
        } else {
            dataStage.toFront();
        }

        buttonOK.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ENTER) {
                removeFromCollectionWithObject(peopleEntry -> peopleEntry.getValue().getAge() < people.getAge(),
peopleTree, keyTextField);
            }
        });

        buttonOK.setOnMouseClicked(event -> removeFromCollectionWithObject(peopleEntry ->
peopleEntry.getValue().getAge() < people.getAge(), peopleTree, keyTextField));

    }

    /**
     * Команда remove_lower.
     * Удаляет из коллекции все элементы, ключ которых меньше, чем заданный.
     *
     * @param peopleTree Ожидается TreeView<Container> для изменения содержимого
     * @version 1.0
     */
    public void removeLowerKey(TreeView<Container> peopleTree) {
        if (dataStage == null) {
            readDataFromTextField("Key");
        } else {
            dataStage.toFront();
        }

        buttonOK.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.ENTER) {
                removeFromCollectionWithKey(peopleEntry -> peopleEntry.getKey().compareTo(data) < 0, peopleTree,
keyTextField);
            }
        });

        buttonOK.setOnMouseClicked(event -> removeFromCollectionWithKey(peopleEntry ->
peopleEntry.getKey().compareTo(data) < 0, peopleTree, keyTextField));

    }

    private void readDataFromTextField(String element) {
        dataStage = new Stage();


        Label keyLabel = new Label("Please, enter " + element);
        keyLabel.setFont(Font.font("Helvetica", FontWeight.LIGHT, 16));

        keyTextField.setPromptText(element);


        HBox buttonOKHBox = new HBox(buttonOK);
        buttonOKHBox.setPadding(new Insets(0, 18, 0, 245));

        VBox enterKeyVBox = new VBox(keyLabel, keyTextField, buttonOKHBox);
        enterKeyVBox.setSpacing(5);

        dataStage.setTitle(element);
        dataStage.centerOnScreen();
        dataStage.setScene(new Scene(enterKeyVBox, 300, 90));
        dataStage.toFront();
        dataStage.setMaximized(false);
        dataStage.setResizable(false);
        dataStage.show();
        dataStage.setOnCloseRequest(event -> dataStage = null);
    }

    private void removeFromCollectionWithKey(Predicate<Map.Entry<String, People>> predicate,
TreeView<Container> peopleTree, TextField textField) {
        this.data = textField.getText();
        dataStage.close();
        dataStage = null;
```

```
        int size = Storage.getInstanceOf().getFamily().size();
        Storage.getInstanceOf().getFamily().entrySet().removeIf(predicate);
        peopleTree.setRoot(MainWindow.getTreeForPeople());
        new ShowAlert(Alert.AlertType.INFORMATION, "Done", "Операция выполнена успешно. \nУдалено " + (size
- Storage.getInstanceOf().getFamily().size()) + " объекта.");
    }


    private void removeFromCollectionWithObject(Predicate<Map.Entry<String, People>> predicate,
TreeView<Container> peopleTree, TextField textField) {
        this.data = textField.getText();
        dataStage.close();
        dataStage = null;

        try {
            people = gson.fromJson(data, People.class);
            int size = Storage.getInstanceOf().getFamily().size();
            Storage.getInstanceOf().getFamily().entrySet().removeIf(predicate);
            peopleTree.setRoot(MainWindow.getTreeForPeople());
            new ShowAlert(Alert.AlertType.INFORMATION, "Done", "Операция выполнена успешно. \nУдалено " +
(size - Storage.getInstanceOf().getFamily().size()) + " объекта.");
        } catch (JsonSyntaxException ex) {
            new ShowAlert(Alert.AlertType.ERROR, "Error", "Не удалось распознать объект, \nпроверьте корректность
данных");
        } catch (NullPointerException ex) {
            new ShowAlert(Alert.AlertType.ERROR, "Error", "Не ввели данные об объекте\n ");
        }
    }


}
```

class SaveDataToFile

```
package commands;

import GUI.Storage;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import deprecated.People;
import old.school.Botherable;
import old.school.Chatable;
import old.school.InterfaceAdapter;
import old.school.Missable;

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Properties;

public final class SaveDataToFile implements Runnable {

    @Override
    public void run() {
        GsonBuilder builder = new GsonBuilder();

        builder.registerTypeAdapter(Missable.class, new InterfaceAdapter<People>());
        builder.registerTypeAdapter(Chatable.class, new InterfaceAdapter<People>());
        builder.registerTypeAdapter(Botherable.class, new InterfaceAdapter<People>());

        Gson gson = builder.create();

        try (PrintWriter printWriter = new PrintWriter("objects")) {
            printWriter.println(gson.toJson(Storage.getInstanceOf().getFamily()));
        }catch (FileNotFoundException ex ){
            try(PrintWriter writeLog = new PrintWriter("../out/log")) {
                writeLog.write(ex.getMessage()+"\n");
            } catch (FileNotFoundException e) {
                //doNothing();
            }
        }
    }
```

}

class TreeTextFieldEditor

```
package commands;

import GUI.Container;
import GUI.ContainerType;
import GUI.Storage;
import javafx.scene.control.TextField;
import javafx.scene.control.TreeCell;
import javafx.scene.control.TreeView;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;

public class TreeTextFieldEditor extends TreeCell<Container> {
    private TextField textField;
    private TreeView<Container> tree;

    public TreeTextFieldEditor(TreeView<Container> tree) {
        this.tree = tree;
    }

    @Override
    public void cancelEdit() {
        super.cancelEdit();
    }

    @Override
    public void startEdit() {
        super.startEdit();
        if (textField == null)
            createTextField();
        setText(null);
        setGraphic(textField);
        textField.selectAll();
    }

    private void createTextField() {
        textField = new TextField(getString());
        textField.setOnKeyReleased(e -> {
            if (e.getCode() == KeyCode.ENTER) {
                if (getItem().getType() == ContainerType.ELEMENT) {
                    commitEdit(getItem());
                    getItem().setValue(textField.getText());
                    Storage.getInstanceOf().getFamily().get(getItem().getKey()).setName(getItem().getValue());
                }
            } else if (e.getCode() == KeyCode.ESCAPE) {
                cancelEdit();
            }
        });
    }

    private String getString() {
        return getItem() == null ? "" : getItem().getValue();
    }

    @Override
    protected void updateItem(Container string, boolean empty) {
        super.updateItem(string, empty);
        if (empty) {
            setText(null);
            setGraphic(null);
        } else {
            if (isEditing()) {
                if (textField != null) {
                    textField.setText(getString());
                }
                setText(null);
                setGraphic(textField);
            } else {
                setText(getString());
```

```
                setGraphic(getTreeItem().getGraphic());
            }
        }
    }
}
```

interface ButtonOkListener

```
package GUI;

/**
 * Created by slavik on 08.04.17.
 */
public interface ButtonOKListener {
    void buttonOKListener();
}
```

class Container

```
package GUI;

import deprecated.People;

public class Container {
    private String key;
    private String value;
    private ContainerType type;

    public Container(String key, String value, ContainerType type) {
        this.key = key;
        this.value = value;
        this.type = type;
    }

    public String getKey() {
        return key;
    }

    public String getValue() {
        return value;
    }

    public void setKey(String key) {
        this.key = key;
    }

    public void setValue(String value) {
        this.value = value;
    }

    public ContainerType getType() {
        return type;
    }

    public void setType(ContainerType type) {
        this.type = type;
    }
}
```

enum ContainerType

```
package GUI;

public enum ContainerType {
    COLLECTION,
    ELEMENT,
    AGE

}
```