Лабораторная работа 3

Отображение информации в табличом виде

Цель работы: изучить способы оформления приложения с помощью компонента TableView и панели компоновки AnchorPane.

Теоретическая часть

1. Разработка интерфейса

Элемент разметки *AnchorPane* позволяет позиционировать вложенные элементы вдоль одной из сторон контейнера. При этом можно задать отступы от границы окна, которые будут постоянными.

Важная особенность данного контейнера – если пользователь будет растягивать или сжимать окно приложения, то отступы останутся неизменными, а элементы также будут растягиваться и сжиматься соразмерно изменению окна.

Создайте новый проект JavaFX. Для размещения элементов будем использовать корневой контейнер AnchorPane. На вкладке Layout установите значение характеристикам Pref Width и Pref Height - 600 и 400 соответственно.

Создайте разметку, как показано в примере на рисунке 1.

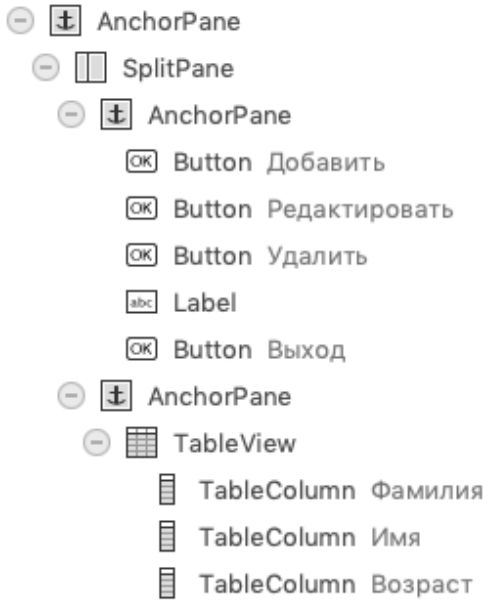

Рисунок 1 – Разметка основного окна

Установив соответствующие значения свойств компонентов, должно получиться окно, как показано на рисунке 2.

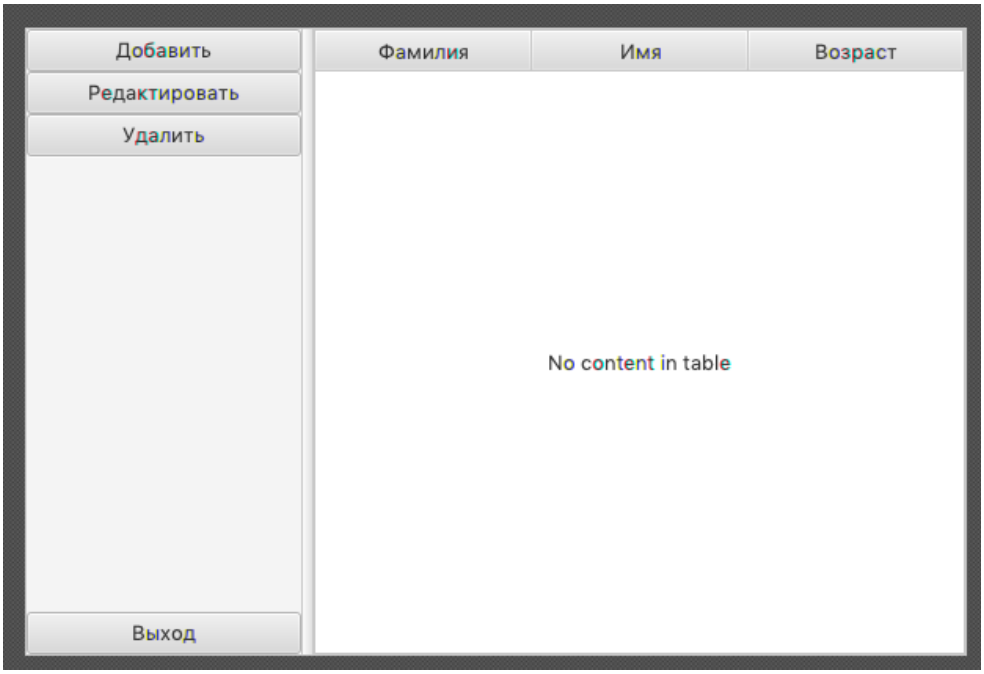

Рисунок 2 – Внешний вид основного окна

Для того, чтобы какой-то вложенный контейнер занял весь размер окна, можно выбрать пункт “*Fit To Parent*” (рис.3)

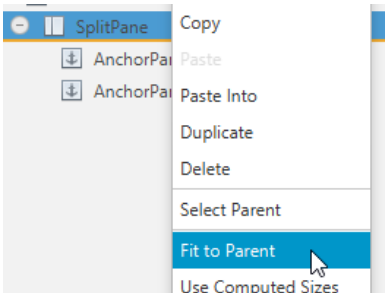

Рисунок 3 – Изменение размера контейнера

В контейнере AnchorPane для любого вложенного компонента можно установить привязки и отступы. Для компонента SplitPane установите отступы равные 0 и 27. Размеры, отличные от 0 выбираются исходя из особенностей текущей компоновки интерфейса. На рисунке 4 показан пример отступов для кнопки «Редактировать»

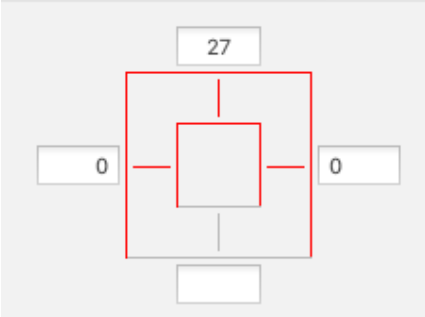

Рисунок 4 – Установка отступов

Для компонента TableView во вкладке Properties измените значение Column Resize Policy на constrained-resize. Выбор этой характеристики гарантирует, что колонки таблицы всегда будут занимать всё доступное пространство.

Если в процессе работы приложения необходимо отобразить новое окно, то необходимо создать и наполнить компонентами новый объект Stage.

При отображении нового окна, ему необходимо установить родительское окно. Так как для этого будет использоваться основное окно, то для его получения необходимо добавит метод, возвращающий объект Stage:

```
private static Stage primaryStage;

public static Stage getPrimaryStage() {
    return primaryStage;
}
```

Программа будет хранить и отображать информацию о студентах. Для этого создадим класс `Student` с полями *Имя*, *Фамилия*, *Возраст*. Текст класса приведен в приложении.

В JavaFX для всех полей класса-модели предпочтительно использовать различные классы, реализующие интерфейс `Property`.

Они позволяет автоматические получать уведомления при любых изменениях переменных. Это позволяет поддерживать синхронность представления и данных.

Для строковых значений используется класс `StringProperty`, для целочисленных `IntegerProperty` и т.д. Фактически эти типы представляю надстройку над стандартными типами String и Integer. Эти и многие другие классы для свойств находятся в пакете `javafx.beans.property`. Каждый подобный тип определяет метод `get()`, который возвращает хранимые данные, и метод `set()`, которые устанавливает данные.

Геттеры и сеттеры для полей класса можно генерировать автоматический через меню *Code->Generate*.

Для синхронизации данных между представлениями и сущностями, их необходимо информировать при любых изменениях. Для этого используется специальная коллекция `ObservableList`. Экземпляра данного класса хранится в коде контроллера:

```
private ObservableList<Student> students =
```

```
                                    FXCollections.observableArrayList();
```

Далее необходимо добавить в контроллер все элементы формы, с которыми предстоит работать. При добавлении таблицы, она создается не типизированная какими-либо данными. Необходимо указать ей, что она будет хранить объекты класса `Student`.

```
private TableView<Student> studentsTable;
private TableColumn<Student, String> surnameColumn;
private TableColumn<Student, String> nameColumn;
private TableColumn<Student, Integer> ageColumn;
```

При загрузке контроллера необходимо инициализировать таблицу и связать ее со списком студентов:

```
@FXML
void initialize(){
//заполнение списка начальными данными
students.add(new Student("Миша", "Иванов",20));
students.add(new Student("Леша", "Петров",21));
//установка обработчиков для колонок таблицы
surnameColumn.setCellValueFactory(studentStringCellDataFeatures ->
     studentStringCellDataFeatures.getValue().surnameProperty());
nameColumn.setCellValueFactory(studentStringCellDataFeatures ->
     studentStringCellDataFeatures.getValue().nameProperty());
ageColumn.setCellValueFactory(cellData ->
     cellData.getValue().ageProperty().asObject());
studentsTable.setItems(students);
```

Здесь мы указываем таблице, какие поля необходимо брать из класса, которым типизированы столбцы. Далее указываем самой таблице, из какого списка брать объекты соответствующего класса.

Далее необходимо добавить слушатель для строк таблицы, чтобы можно было автоматически получать выбранную строку.

```
studentsTable.getSelectionModel().selectedItemProperty()
     .addListener(new ChangeListener<Student>() {
           @Override
           public void changed(ObservableValue<? extends Student>
observableValue, Student student, Student t1) {
```

```
                if(t1 != null){
                    showStudent(t1);
                }
            }
        });
```

Выводить данные будем в `label` на форме. Чтобы получаемая строка полностью умещалась на форме, установите свойство *Wrap Text*. Код форматирования текста и вывода на форму расположен в методе `showStudent()`.

```
    private void showStudent(Student student){
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("…");
        …
        lblLog.setText(stringBuilder.toString());
    }
```

Удаление строки

В обработчике кнопки получаем выбранную строку, и тут же удаляем ее.

```
int selectedIndex = studentsTable.getSelectionModel()
                    .getSelectedIndex();
studentsTable.getItems().remove(selectedIndex);
lblLog.setText("Строка удалена");
```

Добавление и редактирование

Добавление и редактирование будет выполняться в отдельном окне. Для этого необходимо создать новый *fxml* файл и наполнить его как показано на рисунке 6.

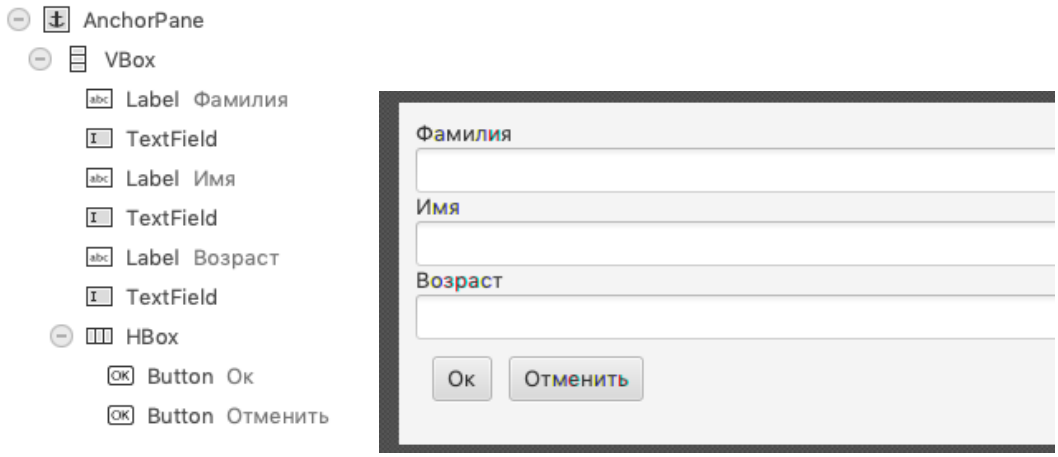

Рисунок 6 – Окно добавления и редактирования.

В описании добавим имя нового контроллера `fx:controller="sample.NewStudentController"`, который также необходимо создать вручную.

В нем должны быть обработчики кнопок. А также объекты для создания окна и передачи объекта класс `Student`.

```
private Stage dialogStage;
private Student student;
```

Для этих полей создадим сеттеры:

```
public void setAddStage(Stage addStage) {
        this.dialogStage = addStage;
   }

public void setStudent(Student student) {
        this.student = student;
        tfSurname.setText(student.getSurname());
        tfName.setText(student.getName());
        tfAge.setText(String.valueOf(student.getAge()));
}
```

Для кнопки *Ок* добавим в обработчик следующий текст:

```
public void onOk() {
        student.setName(tfName.getText());
        student.setSurname(tfSurname.getText());
        student.setAge(Integer.parseInt(tfAge.getText()));
        dialogStage.close();
    }
```

Полный код контроллера приведен в приложении.

Далее необходимо создать окно *добавления* или *изменения*. Отличаться они будут только тем, что при изменении будем передавать объект, полученный из выбранной строки. А при добавлении создавать пустой объект класса `Student`, наполнять его значениями с формы и потом добавлять в список.

Так как при добавлении или редактировании будет создаваться одно и тоже окно, то создадим общий метод showDialog(Student student):

```
private void showDialog(Student student) throws IOException {
      FXMLLoader loader = new FXMLLoader();
      loader.setLocation(
                  Main.class.getResource("new-student.fxml")
                        );
        Parent page =  loader.load();
        Stage addStage = new Stage();
        addStage.setTitle("Информация о студенте");
        addStage.initModality(Modality.WINDOW_MODAL);
        addStage.initOwner(Main.getPrimaryStage());
        Scene scene = new Scene(page);
        addStage.setScene(scene);
        NewStudentController controller = loader.getController();
        controller.setAddStage(addStage);
        controller.setStudent(student);
        addStage.showAndWait();
    }
```

При нажатии на кнопку Редактировать, проверим выбор строки, и передадим новому контроллеру этот объект:

```
   public void onEdit(ActionEvent actionEvent) throws IOException {
        Student selectedStudent =
                           studentsTable.getSelectionModel()
                           .getSelectedItem();
        if(selectedStudent != null){
            showDialog(selectedStudent);
        }
    }
```

При добавлении нового создаем пустой объект, открываем новое окно. После его закрытия добавляем в список студентов новый объект.

```
public void onAdd(ActionEvent actionEvent) throws IOException {
        Student student = new Student();
        showDialog(student);
        students.add(student);
    }
```

Полный код контроллера приведен в приложении.

Задание.

1. Подготовить приложение из рассмотренного в работе примера.

2. Добавить поля для хранения и отображения отчества, города проживания, названия группы.

Содержание отчета по лабораторной работе

- Титульный лист
- Описание задания
- Скриншоты разработанных экранных форм
- Разметка экранных форм на языке fxml
- Программный код основного класса, классов контроллеров и модели.

Вопросы для самоконтроля

1. Что такое компонент TableView?
2. Чем характеризуется контейнер AnchorPane?
2. Порядок определения *привязок* в AnchorPane?
3. Какие способы открытия окна в проекте JavaFX?

Список литературы

1. Вязовик, Н. А. Программирование на Java : учебное пособие / Н. А. Вязовик. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 601 c. — ISBN 978-5-4497-0852-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/102048.html

2. Мухаметзянов, Р. Р. Основы программирования на Java : учебное пособие / Р. Р. Мухаметзянов. — Набережные Челны : Набережночелнинский государственный педагогический университет, 2017. — 114 c. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/66812.html

3. Блох, Дж. Java. Эффективное программирование / Дж. Блох ; перевод В. Стрельцов ; под редакцией Р. Усманов. — 2-е изд. — Саратов : Профобразование, 2019. — 310 c. — ISBN 978-5-4488-0127-3. — Текст : электронный // Цифровой

образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/89870.html

Приложение 1

Класс **Student.class**

```
public class Student {
    private StringProperty name;
    private StringProperty surname;
    private IntegerProperty age;

    public Student(String name, String surname, int age) {
        this.name = new SimpleStringProperty(name);
        this.surname = new SimpleStringProperty(surname);
        this.age = new SimpleIntegerProperty(age);
    }

    public Student() {
        this("", "", 0);
    }

    public String getName() {
        return name.get();
    }

    public StringProperty nameProperty() {
        return name;
    }

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

    public String getSurname() {
        return surname.get();
    }

    public StringProperty surnameProperty() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname.set(surname);
    }
```

```
    public int getAge() {
        return age.get();
    }

    public IntegerProperty ageProperty() {
        return age;
    }

    public void setAge(int age) {
        this.age.set(age);
    }
}
```

Класс **NewStudentController.class**

```
public class NewStudentController {
    public TextField tfSurname;
    public TextField tfName;
    public TextField tfAge;

    private Stage dialogStage;
    private Student student;

    public void setAddStage(Stage addStage) {
        this.dialogStage = addStage;
    }

    public void setStudent(Student student) {
        this.student = student;
        tfSurname.setText(student.getSurname());
        tfName.setText(student.getName());
        tfAge.setText(String.valueOf(student.getAge()));
    }


    public void onOk() {
        student.setName(tfName.getText());
        student.setSurname(tfSurname.getText());
        student.setAge(Integer.parseInt(tfAge.getText()));
        dialogStage.close();
    }

    public void onCancel() {
        dialogStage.close();
    }
}
```

Класс **Controller.java**

```
public class Controller implements Initializable{
    public TableView<Student> studentsTable;
    public TableColumn<Student, String> surnameColumn;
    public TableColumn<Student, String> nameColumn;
    public TableColumn<Student, Integer> ageColumn;
    public Label lblLog;
    public SplitPane splitPane;
    private ObservableList<Student> students =
                            FXCollections.observableArrayList();


    public void onExit(ActionEvent actionEvent) {
        Platform.exit();
    }

    public void onAdd(ActionEvent actionEvent) throws IOException {
        Student student = new Student();
        showDialog(student);
        students.add(student);
    }
    private void showStudent(Student student){
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append("Студент ");
        stringBuilder.append(student.getSurname());
        stringBuilder.append(" ");
        stringBuilder.append(student.getName());
        stringBuilder.append(", возраст ");
        stringBuilder.append(student.getAge());
        stringBuilder.append(" лет.");
        lblLog.setText(stringBuilder.toString());
    }

    @FXML
    void initialize(){
        students.add(new Student("Миша", "Иванов",20));
        students.add(new Student("Леша", "Петров",21));

surnameColumn.setCellValueFactory(studentStringCellDataFeatures          ->
studentStringCellDataFeatures.getValue().surnameProperty());
```

```
            nameColumn.setCellValueFactory(studentStringCellDataFeatures
-> studentStringCellDataFeatures.getValue().nameProperty());
            ageColumn.setCellValueFactory(cellData                    ->
cellData.getValue().ageProperty().asObject());
            studentsTable.setItems(students);

studentsTable.getSelectionModel().selectedItemProperty().addListener(new
ChangeListener<Student>() {
                @Override
                public  void  changed(ObservableValue<?  extends  Student>
observableValue, Student student, Student t1) {
                    if(t1 != null){
                        showStudent(t1);
                    }
                }
            });
        }

        public void onEdit(ActionEvent actionEvent) throws IOException {
            Student                  selectedStudent                  =
studentsTable.getSelectionModel().getSelectedItem();
            if(selectedStudent != null){
                showDialog(selectedStudent);
            }
        }

        private void showDialog(Student student) throws IOException {
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class.getResource(
                                        "new-student.fxml"));
            Parent page =  loader.load();
            Stage addStage = new Stage();
            addStage.setTitle("Информация о студенте");
            addStage.initModality(Modality.WINDOW_MODAL);
            addStage.initOwner(Main.getPrimaryStage());
            Scene scene = new Scene(page);
            addStage.setScene(scene);
            NewStudentController controller = loader.getController();
            controller.setAddStage(addStage);

            controller.setStudent(student);

            addStage.showAndWait();
```

```
        }

        public void onDelete(ActionEvent actionEvent) {
            int                      selectedIndex                      =
studentsTable.getSelectionModel().getSelectedIndex();
            studentsTable.getItems().remove(selectedIndex);
            lblLog.setText("Строка удалена");
        }
    }
```

Код основной программы **Main.java**

```
public class Main extends Application {
    private static Stage primaryStage;


    public static Stage getPrimaryStage() {
        return primaryStage;
    }

    @Override
    public void start(Stage primaryStage) throws Exception{
        this.primaryStage = primaryStage;
        Parent root = FXMLLoader
                .load(getClass().getResource("main-view.fxml"));
        primaryStage.setTitle("Работа с таблицами");
        primaryStage.setScene(new Scene(root, 600, 400));
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }
}
```