Лабораторная работа 7
Графический пользовательский интерфейс с использованием JavaFX

Теоретическая часть

JavaFX представляет инструментарий для создания кроссплатформенных графических приложений на платформе Java. JavaFX позволяет создавать приложения с богатой насыщенной графикой благодаря использованию аппаратного ускорения графики и возможностей GPU.

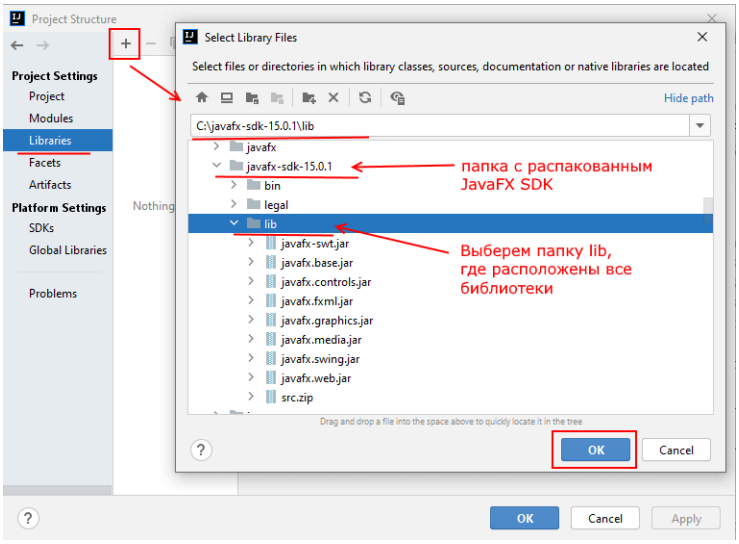
С помощью JavaFX можно создавать программы для различных операционных систем: Windows, MacOS, Linux и для самых различных устройств: десктопы, смартфоны, планшеты, встроенные устройства, ТВ.

JavaFX предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML, возможность стилизации интерфейса с помощью CSS, интеграция со Swing и многое другое.

Платформа JavaFX версии 2.0 состоит из программного интерфейса JavaFX API, альтернативного декларативного языка FXML описания GUI-интерфейса, среды выполнения JavaFX Runtime, набора разработчика JavaFX SDK.

Запуск и настройка среды

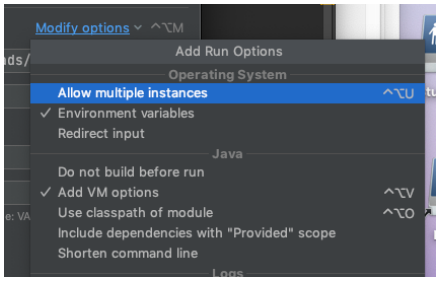
Для разработки приложения с использованием JavaFX необходимо подключить к проекту все необходимые библиотеки. Сделать это можно в окне настроек проекта File->Project Structure. В этой вкладке нажмем на знак + и в открывшемся диалоговом окне выберем путь к каталоге lib в папке, где распакован JavaFX SDK.



Далее, для запуска программы необходимо указать среде исполнения путь, который указывает на расположение модулей. В каждом конкретном случае в зависимости от того, где расположен SDK, этот путь может отличаться. Кроме того, указывается параметр `add-modules`, который указывает на используемые модули. В данном работе применяется модуль `javafx.controls`, который содержит ссылки на другие модули.

```
--module-path /Путь/до/папки/javafx-sdk/lib --add-modules javafx.controls
```

Сделать это можно в настройках *Run/Debug configuration*, в открывшемся окне перейдем к полю *VM options*. Если это поле отсутствует, то нажать на опцию *Modify options* и в контекстном меню выбрать *Add VM options*.



Практическая часть.

Точкой входа в JavaFX-приложение служит Java-класс, расширяющий абстрактный класс javafx.application.Application и содержащий метод main():

```
public class JavaFXApp extends Application {
  public static void main(String[] args) {
    launch(args);
```

```
  }
  @Override
  public void start(Stage stage) {
  }
}
```

В методе main() главного класса JavaFX-приложения вызывает метод launch() класса Application, отвечающий за загрузку JavaFX-приложения. Кроме того, главный класс JavaFX-приложения должен переопределить абстрактный метод start() класса Application, обеспечивающий создание и отображение сцены JavaFX-приложения.

Методы init() и stop() класса Application могут использоваться для инициализации данных и освобождения ресурсов JavaFX-приложения.

```
@Override
public void init() throws Exception {
  System.out.println("Application inits");
}

@Override
public void stop() throws Exception {
  System.out.println("Application stops");
}
```

Метод start() класса Application содержит в качестве параметра объект javafx.stage.Stage, представляющий графический контейнер главного окна JavaFX- приложения. Данный объект Stage создается средой выполнения при запуске JavaFX-приложения и передается в метод start() главного класса JavaFX-приложения, что позволяет использовать методы объекта Stage для установки и отображения сцены JavaFX-приложения.

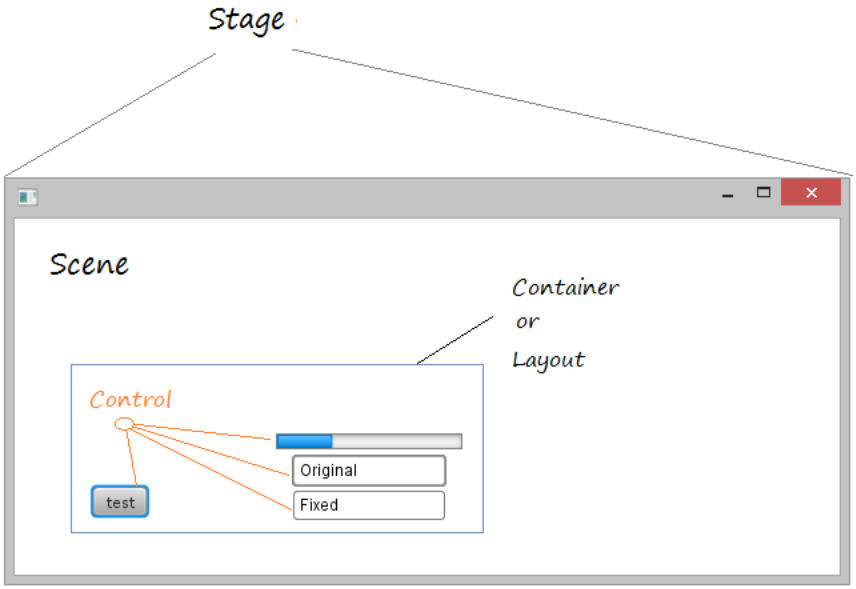
Перед установкой и отображением сцены в графическом контейнере Stage главного окна JavaFX-приложения необходимо создать граф сцены, состоящий из корневого узла и его дочерних элементов, и на его основе создать объект javafx.scene.Scene сцены.

Дочерние узлы графа сцены, представляющие графику, элементы контроля GUI- интерфейса, медиаконтент, добавляются в корневой узел с помощью метода getChildren().add() или метода getChildren().addAll(). При этом

дочерние узлы могут иметь визуальные эффекты, режимы наложения, CSS-стили, прозрачность, трансформации, обработчики событий, участвовать в анимации по ключевым кадрам, программируемой анимации и др.

Метод `Show()`, вызванный у объекта `stage` отобразит окно приложения со всеми компонентами. С этого момента можно считать приложение запущенным.

Связь между `Stage`, `Scene`, `Container` (контейнер), `Layout` (размещение) и `Control`:



Все компоненты GUI-интерфейса являются объектами Node узлов графа сцены и характеризуются идентификатором, CSS-стилем, границами, визуальными эффектами, прозрачностью, трансформациями, обработчиками событий, состоянием, ре- жимом наложения и участием в анимации.

Окно Stage

Компонент Stage представлен классом javafx.stage.Stage, экземпляр которого может быть создан с помощью конструкторов:

```
Stage stage = new Stage();
Stage stage = new Stage(StageStyle style);
```

Класс Stage представляет основной графический контейнер JavaFX-приложения, содержащий его сцену. Он является контейнером, в который помещаются все остальные компоненты интерфейса. Его конкретная реализация зависит от платформы, на которой запускается приложение.

Stage позволяет управлять позиционированием, размерами и некоторыми другими настройками окна приложения. Рассмотрим некоторые основные методы класса Stage:

close(): закрывает объект Stage (на десктопах по сути закрывает окно)

hide(): скрывает окно

centerOnScreen(): располагает окно в центре экрана

initStyle(StageStyle style): устанавливает стиль окна

setOpacity(double value): устанавливает прозрачность

setResizable(boolean value): при передачи значения true позволяет изменять размеры окна (растягивать его и сжимать)

setScene(Scene value): устанавливает сцену (объект Scene) для объекта Stage

setTitle(String value): устанавливает заголовок окна

setMaxHeight(double value): устанавливает максимальную высоту

setMaxWidth(double value): устанавливает максимальную ширину

setMinHeight(double value): устанавливает минимальную высоту

setMinWidth(double value): устанавливает минимальную ширину

setHeight(double value): устанавливает высоту окна

setWidth(double value): устанавливает ширину окна

Сцена Scene

Компонент Scene представлен классом javafx.scene.Scene, экземпляр которого может быть создан с помощью класса-фабрики SceneBuilder или посредством одного из конструкторов:

```
Scene scene = new Scene(Parent root);
```

```
Scene scene = new Scene(Parent root, double width, double height);
```

Объект Scene выступает в качестве контейнера для всех графических элементов внутри объекта Stage в виде графа, который называется Scene Graph. Все узлы этого графа, то есть по сути все вложенные элементы должны представлять класс javafx.scene.Node.

Панели компоновки

Обычно все элементы управления, которые используются в приложении, помещаются в специальные контейнеры – панели компоновки (layout panes). Панели

компоновки позволяют упорядочить вложенные элементы определенным образом и могут определять их размеры.

Поскольку все эти классы наследуются от класса Pane, то они все они имеют один важный метод – getChildren(), который возвращает коллекцию вложенных в контейнер элементов. Каждый вложенный элемент представляет объект Node.

Панель FlowPane

Компонент FlowPane представлен классом javafx.scene.layout.FlowPane, экземпляр которого может быть создан посредством одного из конструкторов:

```
FlowPane flowPane = new FlowPane();
FlowPane flowPane = new FlowPane(double hgap, double vgap);
```

Свои дочерние узлы панель FlowPane компонует в вертикальный или горизонтальный поток GUI-компонентов. Парметры hgap и vgap задают расстояния между элементами управления внутри контейнера по горизонтали и по вертикали.

Метод setAlignment(Pos value) задает выравнивание.

Панели VBox и HBox

Компонент VBox представлен классом javafx.scene.layout.VBox, компонент HBox классом javafx.scene.layout.HBox.

```
Конструкторы:
VBox vBox = new VBox();
VBox vBox = new VBox(double spacing);
HBox hBox = new HBox();
HBox hBox = new HBox(double spacing);
```

Параметр spacing определяет отступы элементов внутри контейнера.

Элементы управления

Кнопка Button

Компонент Button представлен классом javafx.scene.control.Button, экземпляр которого может быть создан с помощью конструктора:

```
Button btn = new Button();
Button btn = new Button("текст");
```

Метка Label

Компонент Label представлен классом javafx.scene.control.Label, экземпляр может быть создан с одного из конструкторов:

```
Label label = new Label();
Label label = new Label("[текст]");
```

Класс Label обеспечивает отображение текстовой подписи, изображения, текстовой подписи и изображения одновременно

Поле ввода TextField

Компонент TextField представлен классом javafx.scene.control.TextField

```
TextField textField = new TextField();
TextField textField = new TextField("[текст]");
```

Обработка событий

Ключевой возможностью кнопки является способность реагировать на нажатия пользователей и по нажатию выполнять некоторое действие. Для обработки нажатий в базовом классе определен метод setOnAction(), который устанавливает обработчик нажатия setOnAction(EventHandler<ActionEvent> handler).

Рассмотрим на примере простейшего приложения:

```
public class Main extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        FlowPane root = new FlowPane();
        Button btnExit = new Button("Выход");
        Button btnHello = new Button("Привет!");
        btnExit.setOnAction(new EventHandler<>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                Platform.exit();
            }
        });
        Label label = new Label("Введите имя:");
        TextField tfName = new TextField();
        Label result = new Label();
        btnHello.setOnAction(actionEvent -> {
            result.setText("Привет, " + tfName.getText());
        });
        root.getChildren().addAll(label, tfName, btnHello,
                                  btnExit, result);
```

```
        root.setHgap(20);
        root.setVgap(10);
        Insets insets = new Insets(30,30,30,30);
        root.setPadding(insets);
        Scene scene = new Scene(root, 420, 360);
        stage.setTitle("New window");
        stage.setScene(scene);
        stage.setMinWidth(300);
        stage.setMinHeight(200);
        stage.initStyle(StageStyle.UTILITY);
        stage.setResizable(false);
        stage.show();
    }

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

Задания на лабораторную работу

Подготовить программу, выполняющую вычисления в соответствии с вариантом.

Логику вычислений вынести в отдельный класс.

Данные для расчетов получить с формы, выполнить вычисления, вывести результат.

На форме должны быть кнопки для выполнения вычислений, очистки значений и завершения приложения.

Вариант	**Задание**
1	Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
2	Заданы стороны прямоугольника. Вычислить его периметр, площадь и длину диагонали.
3	Даны вещественные положительные числа a, b, c. Если существует треугольник со сторонами a, b, c, то определить его вид (прямоугольный, остроугольный, тупоугольный)

4	Даны вещественные положительные числа a, b, c. Выяснить, существует ли треугольник со сторонами a, b, c.
5	Даны два числа. Вычислить их сумму, разность, произведение и среднее арифметическое.
6	Даны три числа. Найти среднее арифметическое квадратов этих чисел и сумму модулей этих чисел.
7	Заданы координаты трех вершин треугольника. Найти его периметр и площадь
8	Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b
9	Вычислить площадь и периметр прямоугольника, если задана длина одной стороны (a) и коэффициент n (%), позволяющий вычислить длину второй стороны (b=n*a)
10	Если известны две стороны и угол между ними, вычислить периметр и площадь треугольника.