Лабораторная работа №6

API-ИНТЕРФЕЙС JAVAFX CANVAS

Цель работы:

Познакомиться с API-интерфейсом JavaFX Canvas. Научиться рисовать основные фигуры.

Рассмотрим API-интерфейс **JavaFX** Canvas. Он определяется по классам **Canvas**, **CanvasBuilder** и **GraphicsContextв javafx.scene.canvas**. Использование этого API включает в себя создание Canvas объекта, его получение **GraphicsContext** и запуск операций рисования для отображения ваших пользовательских фигур на экране. Поскольку **Canvas** это **Node** подкласс, его можно использовать в графе сцены **JavaFX**.

- **Рисование основных фигур**

Нарисуем некоторые основные формы (линии, овалы, закругленные прямоугольники, дуги и многоугольники возможны при использовании методов **GraphicsContext** класса).

Создайте новый проект JavaFx (jdk1.8). Удалите создавшиеся автоматически пакеты. Добавьте свой пакет и в нем новый класс. Вставьте в него следующий код для рисования основных фигур на холсте.

```
/*
 * Copyright (c) 2012 Oracle and/or its affiliates...
 */
package laba_6_1;

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import javafx.scene.shape.ArcType;
import javafx.stage.Stage;

public class laba_6_1 extends Application {

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

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Рисование основных фигур");
        Group root = new Group();
        Canvas canvas = new Canvas(500, 300);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        drawShapes(gc);
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }
    /**
     * Рисует ряд базовых фигур, gc GraphicsContext объект для рисования
     */
    private void drawShapes(GraphicsContext gc) {
        gc.setFill(Color.RED);
        gc.setStroke(Color.AQUAMARINE);
        gc.setLineWidth(5);
```

```
        gc.strokeLine(40, 10, 10, 40);
        gc.fillOval(10, 60, 30, 30);
        gc.strokeOval(60, 60, 30, 30);
        gc.fillRoundRect(110, 60, 30, 30, 10, 10);
        gc.strokeRoundRect(160, 60, 30, 30, 10, 10);
        gc.fillArc(10, 110, 30, 30, 45, 240, ArcType.OPEN);
        gc.fillArc(60, 110, 30, 30, 45, 240, ArcType.CHORD);
        gc.fillArc(110, 110, 30, 30, 45, 240, ArcType.ROUND);
        gc.strokeArc(10, 160, 30, 30, 45, 240, ArcType.OPEN);
        gc.strokeArc(60, 160, 30, 30, 45, 240, ArcType.CHORD);
        gc.strokeArc(110, 160, 30, 30, 45, 240, ArcType.ROUND);
        gc.fillPolygon(new double[]{10, 40, 10, 40},
                new double[]{210, 210, 240, 240}, 4);
        gc.strokePolygon(new double[]{60, 90, 60, 90},
                new double[]{210, 210, 240, 240}, 4);
        gc.strokePolyline(new double[]{110, 140, 110, 140},
                new double[]{210, 210, 240, 240}, 4);
    }
}
```

Экземпляр **Canvas** (холст) создается с шириной 500 и высотой 300. Затем его получают с помощью вызова **canvas.getGraphicsContext2D().** После этого, несколько основных операций рисований осуществляются с помощью методов **strokeLine, fillOval, strokeArc, и fillPolygon.**

ArcType.CHORD Тип закрытия для дуги, закрытой путем рисования отрезка прямой линии от начала сегмента дуги до конца сегмента дуги.

ArcType.OPEN Тип закрытия для открытой дуги без сегментов пути, соединяющих два конца сегмента дуги.

ArcType.ROUND Тип замыкания для дуги, закрытой путем рисования отрезков прямых линий от начала сегмента дуги до центра полного эллипса и от этой точки до конца сегмента дуги.

- **Применение градиентов и теней**

Следующий пример тестирует GraphicsContext метод, рисуя пользовательскую фигуру, вместе с некоторыми градиентами и тенями.

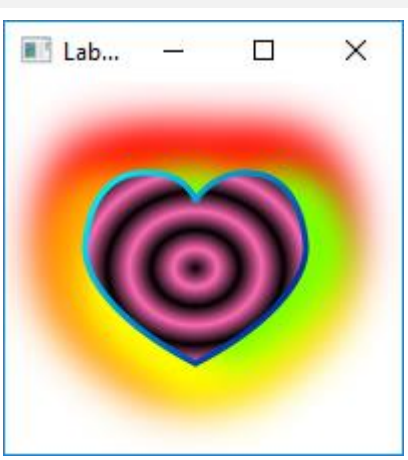
Код для этого примера организован так, что каждая операция рисования выполняется в своем собственном частном методе. Это позволяет вам тестировать различные функции, просто вызывая (или комментируя) интересующие вас методы. Просто имейте в виду, что с точки зрения изучения Canvas API код, на котором следует сосредоточиться, - это базовые вызовы объектов Canvas или GraphicsContext.

Создайте новый проект.

```
/*
 * Copyright (c) 2012 Oracle and/or its affiliates...
 */
package laba_6_2;

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.effect.DropShadow;
import javafx.scene.paint.Color;
import javafx.scene.paint.CycleMethod;
import javafx.scene.paint.LinearGradient;
import javafx.scene.paint.RadialGradient;
import javafx.scene.paint.Stop;
import javafx.stage.Stage;

public class laba_6_2 extends Application {
```



```
private Canvas canvas = new Canvas(200, 200);
private GraphicsContext gc = canvas.getGraphicsContext2D();
private Group root = new Group();

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

@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("Laba_6_1");
    moveCanvas(0,0);
    drawHeart();
    drawRadialGradient(Color.BLACK, Color.HOTPINK);
    drawLinearGradient(Color.AQUA, Color.DARKBLUE);
    drawDropShadow(Color.CHARTREUSE, Color.YELLOW, Color.ORANGE, Color.RED);
    root.getChildren().add(canvas);
    primaryStage.setScene(new Scene(root, 200, 200));
    primaryStage.show();
}

/**
  * Перемещаем холст на новое место в сцене
 */
private void moveCanvas(int x, int y) {
    canvas.setTranslateX(x);
    canvas.setTranslateY(y);
}

/**
 * Рисуем сердце с помощь кубических кривых Безье
 */
private void drawHeart() {
    gc.beginPath();
    gc.moveTo(95,60);
    gc.bezierCurveTo(95,57,90,45,70,45);
    gc.bezierCurveTo(40,45,40,82.5,40,82.5);
    gc.bezierCurveTo(40,100,60,122,95,140);
    gc.bezierCurveTo(130,122,150,100,150,82.5);
    gc.bezierCurveTo(150,82.5,150,45,120,45);
    gc.bezierCurveTo(105,45,95,57,95,60);
    gc.closePath();
}

/**
 * Радиальный градиетн
 */
private void drawRadialGradient(Color firstColor, Color lastColor) {
    gc.setFill(new RadialGradient(0, 0, 0.5, 0.5, 0.1, true,
            CycleMethod.REFLECT,
            new Stop(0.0, firstColor),
            new Stop(1.0, lastColor)));
    gc.fill();
}

/**
 * Линейный градиент
 */
private void drawLinearGradient(Color firstColor, Color secondColor) {
    LinearGradient lg = new LinearGradient(0, 0, 1, 1, true,
            CycleMethod.REFLECT,
            new Stop(0.0, firstColor),
            new Stop(1.0, secondColor));
    gc.setStroke(lg);
    gc.setLineWidth(3);
    gc.stroke();
}
```

```
    /**
     * Тень из 4 цветов
     */
    private void drawDropShadow(Color firstColor, Color secondColor,
            Color thirdColor, Color fourthColor) {
        gc.applyEffect(new DropShadow(20, 20, 0, firstColor));
        gc.applyEffect(new DropShadow(20, 0, 20, secondColor));
        gc.applyEffect(new DropShadow(20, -20, 0, thirdColor));
        gc.applyEffect(new DropShadow(20, 0, -20, fourthColor));
    }
}
```

Разберем пример.

- **Canvas** задается в координатах **(0,0)**. Вы можете передать другие значения в качестве параметров **setTranslateX** и **setTranslateY**, при этом **Canvas** переместится в заданное место.

Начало координат в - левом верхнем углу, Y вниз X вправо.

- **RadialGradient** обеспечивает круговой узор на заднем плане (внутри), **setFill** метод **GraphicsContext** принимает **RadialGradient** объект в качестве параметра.

- **LinearGradient** линейный градиент для кривых. Код устанавливает штрих **GraphicsContext** для использования **LinearGradient**, а затем отображает шаблон с помощью **gc.stroke**().

- **applyEffectна** разноцветная тень (снаружи). Создания **DropShadow** объекта с указанным цветом, который передается **applyEffect** методу **GraphicsContext** объекта.

- **Взаимодействие с пользователем**

Создадим простейшую «рисовалку», когда пользователь зажимая мышь — рисует на холсте, при двойном клике сцена отчищается.

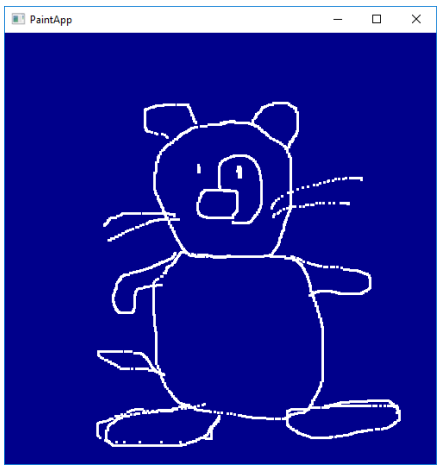
```
/*
 * Copyright (c) 2012 Oracle and/or its affiliates...
 */
package Laba_6_3;

import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class Laba_6_3 extends Application {

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

    /**
     * Сбрасывает холст в его первоначальный вид, заполняя полотно
     */
    private void reset(Canvas canvas, Color color) {
        GraphicsContext gc = canvas.getGraphicsContext2D();
```



```
        gc.setFill(color);
        gc.fillRect(0, 0, canvas.getWidth(), canvas.getHeight());
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("PaintApp");
        Group root = new Group();

        // Задний фон
        Rectangle rect = new Rectangle(500, 500, Color.WHITE);
        root.getChildren().add(rect);

        // Передний фон
        final Canvas canvas = new Canvas(500, 500);
       // canvas.setTranslateX(0);
       // canvas.setTranslateY(0);
        reset(canvas, Color.DARKBLUE);

        final GraphicsContext gc = canvas.getGraphicsContext2D();

        // Удалние переднего фона при нажатии ЛКМ и перемещении по холсту
        canvas.addEventHandler(MouseEvent.MOUSE_DRAGGED, new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent e) {
                gc.clearRect(e.getX() - 1, e.getY() - 1, 2, 2);
            }
        });

        // Возвращение переднего фона при двойном клике
        canvas.addEventHandler(MouseEvent.MOUSE_CLICKED, new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent t) {
                if (t.getClickCount() >1) {
                    reset(canvas, Color.DARKBLUE);
                }
            }
        });

        // Добавление холста к сцене и отражение сцены
        root.getChildren().add(canvas);
        primaryStage.setScene(new Scene(root, 500, 500));
        primaryStage.show();
    }
}
```

- **reset** метод заполняет весь холст синим цветом, **start** метод добавляет обработчик событий **MouseEvent** объектов, когда пользователь перетаскивает мышь. При каждом перетаскивании вызывается **clearRect** метод **GraphicsContext** объекта, передавая текущие координаты мыши, а также размер области, которую необходимо убрать. Когда это произойдет, задний фон будет просвечивать. Оставшийся код просто считает количество нажатий и сбрасывает синий фон в исходное состояние, если пользователь дважды щелкает мышью.

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

1. Нарисовать и стилизовать свой двумерный рисунок на холсте.

2. Реализовать удаление вашего рисунка при проведении по нему мышью.

3. Возвращать в исходное состояние при тройном клике.

4. (Задание на «5») Добавить возможность выбора размера удаляющей кисти.