

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КІРОВОГРАДСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ ТА ДИСТАНЦІЙНОЇ ОСВІТИ

КАФЕДРА ПРОГРАМУВАННЯ ТА ЗАХИСТУ ІНФОРМАЦІЇ

Комп'ютерна графіка

Методичні вказівки до виконання лабораторних робіт

з елементами кредитно - модульної
системи організації навчального процесу

*для студентів заочної форми навчання (повні, друга вища)
за напрямками підготовки 6.050102 «Комп'ютерна інженерія»*

7.05010202 «Системне програмування»

7.05010201 «Комп'ютерні системи та мережі»

Укладачі:

к.т.н, доцент

к.т.н, доцент

Н.В. Смірнова

В.В. Смірнов

Кіровоград 2015

Комп'ютерна графіка: Методичні вказівки до виконання лабораторних робіт для студентів заочної форми / Укл.: Смірнова Н.В. Смірнов В.В. - Кіровоград: КНТУ, 2015. – 41 с.

Затверджено на засіданні кафедри ПЗ: 2 липня 2015 р. протокол № 21.

Укладачі:

Смірнова Наталя Володимирівна, к.т.н., доцент кафедри ПЗІ.

Смірнов Володимир Вікторович, к.т.н., доцент кафедри ПЗІ,

Для студентів заочної форми навчання, що вивчають навчальну дисципліну "Комп'ютерна графіка" за напрямками підготовки 6.050102 «Комп'ютерна інженерія», 7.05010202 «Системне програмування», 7.05010201 «Комп'ютерні системи та мережі».

У стислій формі викладені основні принципи побудови об'єктно-орієнтованих подієво-керованих інтерактивних графічних додатків мовою програмування Java у середовищі розробки NetBeansIDE.

Представлені практичні рішення навчальних завдань для кращого освоєння досліджуваного матеріалу, представлені варіанти навчальних завдань для самостійного придбання практичних навичок.

Примітка: Дані методичні вказівки є інтелектуальною власністю авторів. Методичні вказівки можуть використовуватися у навчальному процесі КНТУ, копіюватися, розмножуватися і поширюватися без обмежень.

Будь-яке внесення змін і доповнень до тексту методичних вказівок без письмового дозволу авторів на підставі законів України "Про інтелектуальну власність" і "Про авторське право і суміжні права" кваліфікується як порушення авторських прав.

© Н.В. Смірнова, В.В. Смірнов / 2015

© КНТУ, кафедра " КАФЕДРА ПРОГРАМУВАННЯ ТА ЗАХИСТУ ІНФОРМАЦІЇ "

ЗМІСТ

1. Опис навчальної дисципліни "Комп'ютерна графіка"	4
2. Шкала оцінювання	5
3. Теми лабораторних робіт з дисципліни "Комп'ютерна графіка"	6

Лабораторні роботи:

№1 Освоєння середовища розробки графічних додатків NetBeansIDE	7
№2 Створення графічних примітивів 2D графіки	22
№3 Анімація об'єктів комп'ютерної графіки. Клас TranslateTransition і події миші	32
№4 Анімація об'єктів комп'ютерної графіки. Клас Timeline і виявлення колізій	36
Список літератури	41

Опис навчальної дисципліни

"Комп'ютерна графіка"

Основна мета курсу полягає в придбанні знань і навичок розробки та побудови об'єктно-орієнтованих подійно-керованих інтерактивних графічних Java додатків із застосуванням сучасних технологій і інструментальних засобів.

У результаті проведення лекцій студенти повинні одержати теоретичні знання і методику ефективної роботи із сучасними методами створення об'єктно-орієнтованих керованих подіями графічних додатків.

Завдання вивчення дисципліни

- Вивчення теоретичних основ проектування додатків комп'ютерної графіки;
- Вивчення теоретичних основ програмування додатків комп'ютерної графіки;
- Вивчення теоретичних основ методів створення графічних 2D - об'єктів;
- Вивчення теоретичних основ методів створення графічних 3D - об'єктів;
- Розв'язок завдань створення елементів керування графічними 2D і 3D об'єктами з використанням платформи JavaFx;
- Придбання практичних навичок в області програмування 2D і 3D додатків комп'ютерної графіки на основі технології Java.

Предметом навчальної дисципліни є створення об'єктно-орієнтованих подійно-керованих інтерактивних додатків комп'ютерної графіки в середовищі програмування NetBeansIDE на платформі Java.

У результаті вивчення навчальної дисципліни студент повинен

знати:

1. Основи створення 2D і 3D графічних об'єктів.
2. Принципи створення подійно-керованих програм.
3. Основи керування 2D і 3D графічними об'єктами.

уміти:

1. Вирішувати завдання створення 2D і 3D додатків комп'ютерної графіки на основі технології Java.
2. Вирішувати завдання створення об'єктно-орієнтованих додатків комп'ютерної графіки в середовищі розробки NetBeansIDE на платформі JavaFx.
3. Вирішувати завдання керування 2D і 3D графічними об'єктами в додатках комп'ютерної графіки.

Шкала оцінювання

За шкалою ECTS	За національною шкалою	За шкалою навчального закладу
A	відмінно	90-100
B-C	добре	75-89
D-E	задовільно	60-74
F-X	незадовільно з можливістю повторної здачі	35-59
F	незадовільно з обов'язковим повторним курсом	1-34

**Лабораторні заняття по дисципліні
"Комп'ютерна графіка"**

№ заняття	Теми лабораторних занять	Кількість годин
	5 семестр	
1.	Освоєння середовища розробки графічних додатків NetBeansIDE	2
2.	Створення графічних примітивів 2D графіки	2
	6 семестр	
3.	Анімація об'єктів комп'ютерної графіки. Клас TranslateTransition і події миші	2
4.	Анімація об'єктів комп'ютерної графіки. Клас Timeline і виявлення колізій	2
	усього годин	8

Лабораторна робота №1

Тема: Освоєння середовища розробки графічних додатків
NetBeansIDE

Ціль роботи

Одержання навичок роботи з NetBeansIDE на платформі JavaFx, створення проекту, введення і компіляція програми, створення файлу *.jar, що виконується, запуск програми.

Завдання:

- Створити простий проект JavaFx в NetBeansIDE.
- Ввести текст демонстраційної програми. Відкомпілювати програму, усунути ймовірні помилки. Створити файл *.jar, що виконується, запустити програму на виконання.

Теоретичні відомості:

Перший приклад простої програми

Розглянемо просту програму, написану на Java. Почнемо з компіляції і запуску прикладу програми.

```
//=== Лабораторна робота № 1
public class Lab1 extends Application {
    //=== Метод launch() запускає програму викликом методу start()
    @Override
    public void start(Stage stage) {
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction((ActionEvent event) -> {
            System.out.println("Hello World!");
        });
        Stackpane root = new Stackpane();
        root.getChildren().add(btn);
        Scene scene = new Scene(root, 300, 250);
        stage.setTitle("Hello World!");
        stage.setScene(scene);
    }
}
```

```
        stage.show();
    }
    //=== Крапка входу в програму
    public static void main(String[] args) {
        //Метод launch() запускає програму викликаючи метод start()
        launch(args); //
    }
}
```

Введення коду програми

Синтаксис мова програмування Java аналогічний синтаксису мови програмування C, C++ і C#.

В Java вихідний файл називається *модулем компіляції*. Він містить опис одного або декількох класів. Компілятор Java вимагає, щоб вихідний файл мав розширення *.java.

В Java увесь код повинен розміщатися усередині класу.

Ім'я головного класу, який містить функцію main() повинне збігатися з іменем файлу, що містить програму. Необхідно також, щоб написання імені файлу відповідало імені класу, включаючи малі і прописні букви.

Це обумовлене тим, що код Java чутливий до регістру символів.

Компіляція і виконання програми в NetBeansIDE

- Компіляція програми здійснюється натисканням кнопки **F9**.
- Компіляція і запуск програми здійснюється натисканням кнопки **F6**.
- Файл, що виконується, має розширення (*.jar).

Більш докладний розгляд першого прикладу програми

Програма Lab1 має кілька важливих особливостей, характерних для всіх програм Java. Розглянемо кожну частину цієї програми більш докладно.

Програма починається з наступних строк:

```
// Лабораторна робота № 1
```

В Java підтримується три стилі коментарів.

```
/*Comment*/;  
//Comment;  
/** Comment*/.
```

Перші два являють собою звичайні коментарі, застосовувані як в Java, так і в C++. Останній введений для автоматичного документування тексту програми. Після написання вихідного тексту утиліта автоматичної генерації документації збирає тексти таких коментарів в один файл.

Наступний рядок програми має такий вигляд:

```
public class Lab1 extends Application {
```

У цьому рядку ключове слово `class` використовується для оголошення про те, що виконується визначення нового класу. `Lab1` – це *ідентифікатор*, що є іменем класу.

Усе визначення класу, у тому числі його членів, буде розміщатися між відкриваючою `{` і закриваючою `}` фігурними дужками.

Наступний рядок коду виглядає так:

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

Виконання всіх додатків Java починається з виклику методу `main()`.

Ключове слово `public` – *модифікатор доступу*, який дозволяє програмісту управляти видимістю елементів класу. Коли елементу класу передує ключове слово `public`, він є доступним іншим об'єктам програми.

У цьому випадку метод `main()` повинен бути визначений як `public`, оскільки при запуску програми він повинен викликатися кодом, визначеним поза його класом.

Ключове слово `static` дозволяє викликати метод `main()` без явного створення екземпляра класу.

Ключове слово `void` повідомляє компілятору, що метод `main()` не повертає ніяких значень.

Для передачі інформації, необхідної методу, служать змінні, які називають *параметрами*.

Якщо для даного методу ніякі параметри не потрібні, слід вказувати порожні дужки. Метод `main()` містить тільки один параметр, але досить складний.

Частина `String[] args` повідомляє параметр `args`, який являє собою масив екземплярів класу `String`.

У цьому випадку параметр `args` приймає будь-які аргументи командного рядка.

Увесь код, що утворює метод, розташований між відкриваючою і закриваючою фігурними дужками методу.

Ще один важливий момент: метод `main()` служить усього лише початком програми. Складна програма може включати десятки класів, тільки один з яких повинен містити метод `main()`, щоб виконання було можливим.

Метод `launch()` запускає програму викликом методу `start()`.

Модель програмування додатків платформи JavaFx

Один і той самий код JavaFx - додатка може запускатися в якості настільного додатка, який розвертається на клієнтському комп'ютері автономно, може розвертатися як додаток Java Web Start або відображатися в Web - браузері як JavaFx - аплет, вбудований в HTML - сторінку.

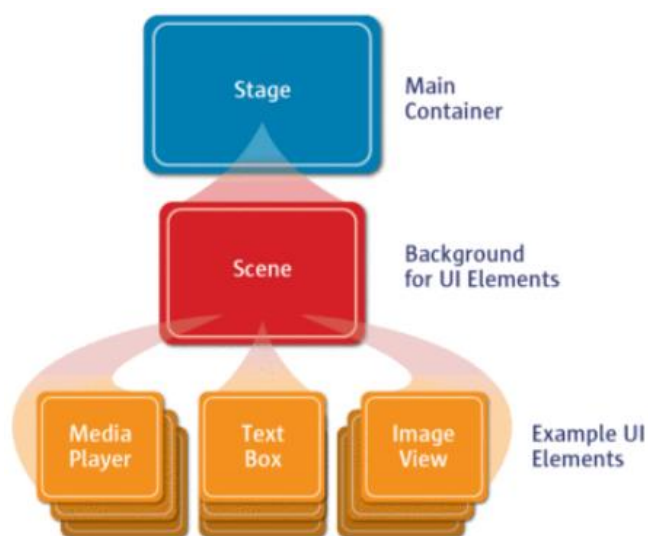
Крапкою входу в JavaFx-додаток служить `Java` - клас, що розширює абстрактний клас `Javafx.application.Application` і який містить метод `main()`:

```
public class JavaFxapp extends Application {
    public static void main(String[] args) {
        launch(args);
    }
    //
    @Override
    public void start(Stage stage) {
        //Установка параметрів сцени
        . . . .
        stage.setscene(scene);
        stage.setVisible(true);
    }
}
```

Метод `start()` класу `Application` містить у якості параметра об'єкт `Stage`, що представляє графічний контейнер головного вікна JavaFx-дodatка. Даний об'єкт

Stage створюється середовищем виконання при запуску JavaFx-додатка і передається в метод `start()` головного класу JavaFx-додатка, що дозволяє використовувати методи об'єкта `Stage` для установки і відображення сцени JavaFx-додатка.

Перед установкою і відображенням сцени в графічному контейнері `Stage` головного вікна JavaFx-додатка необхідно створити граф сцени, що полягає з кореневого вузла і його дочірніх елементів, і на його основі створити об'єкт `Scene` сцени.



Дочірні вузли графа сцени, що представляють графіку, елементи контролю GUI- інтерфейсу, медіаконтент, додаються в кореневий вузол за допомогою методу `getChildren().add()` або методу `getChildren().addAll()`. При цьому дочірні вузли можуть мати візуальні ефекти, режими накладення, CSS-стилі, прозорість, трансформації, оброблювачі подій, брати участь в анімації по ключових кадрах, програмувальній анімації та ін.

Розробка програмного забезпечення

Розробка програмного забезпечення здійснюється мовою програмування Java в інтегрованому середовищі розробки NetBeansIDE фірми Oracle.

Порядок роботи:

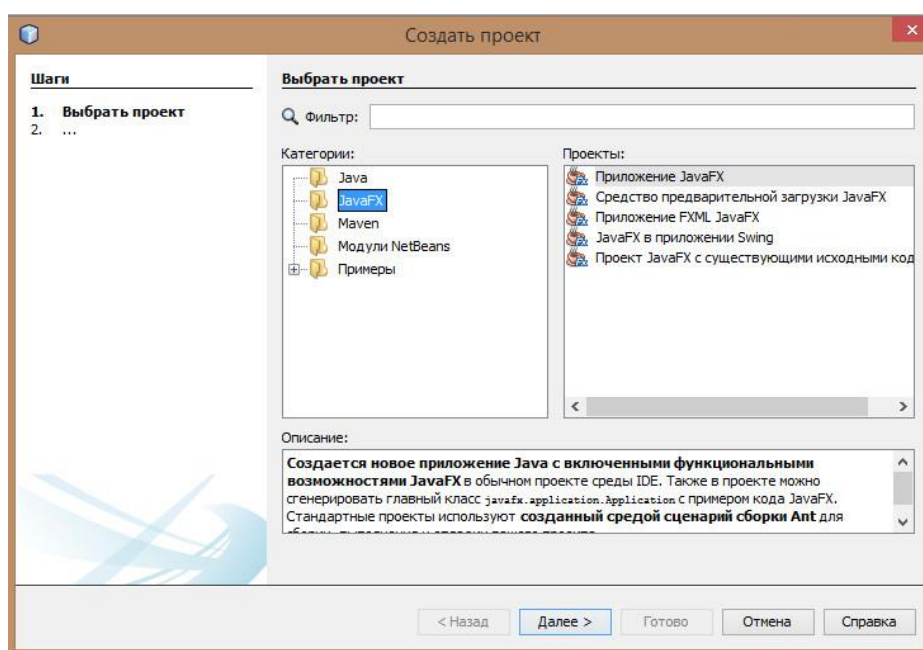
1. Створити свій директорій для файлів проекту.
2. Створити проект в інтегрованому середовищі розробки NetBeansIDE (c:\Temp\Ваша_Папка).
3. Написати програму мовою програмування Java (демонстраційна програма).
4. Скомпілювати програму.
5. Створити файл *.jar, який виконується.

Робота з NetBeansIDE

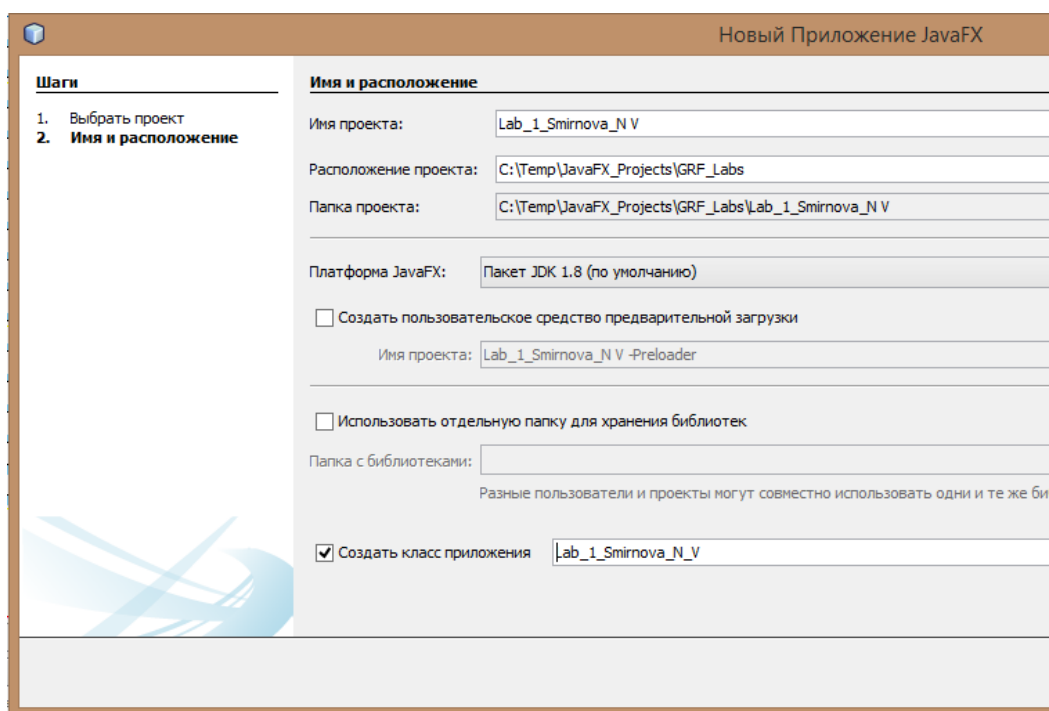
Створення проекту.

Запустити NetBeansIDE, у меню «Файл» вибрати пункт «Створити проект».

Вибрати пункт JavaFx, Додаток JavaFx і натиснути кнопку «Далі».



Папку та ім'я робочого проекту вказати латинським шрифтом, наприклад:



Буде створений проект і шаблон програми:

```
public class Lab_1_Smirnova_N_V extends Application {

    @Override
    public void start(Stage primaryStage) {
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

        StackPane root = new StackPane();
        root.getChildren().add(btn);

        Scene scene = new Scene(root, 300, 250);

        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }
}
```

Шаблон програми привести до нормального вигляду:

```

//=====
// Лабораторная работа № 1
// class: Lab_1_Smirnova_N_V
// Copyright (c) 2015 Smirnova N.V., гр. |группа|
//=====
package lab_1_smirnova_n_v;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;

...

//=====
// Главный класс
//=====
public class Lab_1_Smirnova_N_V extends Application {

    //===== объявления
    final int THICKNESS_MAX = 50;    //максимальная толщина линии
    //
    Group root = new Group();        //корень узлов сцены
    //
    TextArea display;                 //область вывода
    TextField in;                     //поле ввода
    Text txt;                         //текстовая надпись
    Slider slider;                    //слайдер

    //=====
    // Начало
    //=====
    @Override
    public void start(Stage stage) {
        //===== заголовок окна
        stage.setTitle("Лабораторная работа №1. доц. Смирнова Н.В.");
        stage.setResizable(false);    //фиксированный размер
        //===== создать узлы сцены с графикой
        CreateGraphNodes();
        //===== создать узлы сцены с элементами управления
        CreateControlNodes();
        //===== создать сцену с размерами и цветом фона
        Scene scene = new Scene(root, 460, 250, Color.CORNSILK);
        //===== поместить сцену в окно
        stage.setScene(scene);
        //===== отобразить окно
        stage.show();
    }

    //=====
    // Точка входа в программу (запускает метод start)
    //=====
    public static void main(String[] args) {
        launch(args);
    }
}

```

Ввести код демонстраційної програми:

```
//=====
// Лабораторная работа № 1
// class: Lab_1_Smirnova_N_V
// Copyright (c) 2015 Smirnova N.V., гр. |группа|
//=====
package lab_1_smirnova_n_v;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ColorPicker;
import javafx.scene.control.Slider;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.effect.DropShadow;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.StrokeLineCap;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

/**
 *
 * @author Smirnova N.V.
 */
//=====
// Главный класс
//=====
public class Lab_1_Smirnova_N_V extends Application {

    final int THICKNESS_MAX = 50;           //максимальная толщина линии
    Group root = new Group();              //группа для узлов сцены

    //===== элементы управления
    Slider slider;                          //слайдер
    Button btn_inc;                          //кнопка инкремент
    Button btn_dec;                          //кнопка декремент
    ColorPicker cp_1;                        //колорпикер для 1-й линии
    ColorPicker cp_2;                        //колорпикер для 2-й линии
    ColorPicker cp_3;                        //колорпикер для 3-й линии
    //===== графика
    DropShadow dropShadow;                  //тень
    Line line_1;                             //линия 1
    Line line_2;                             //линия 2
    Line line_3;                             //линия 3
    Text text;                               //текстовая надпись
    //===== размерности
    double lines_X = 30;                     //начало линий по X

```

```

double lines_Y = 30;           //начало линий по Y
double space_Y = 50;          //расстояние между линиями
//
double line_1_Y = lines_Y;    //начало линии 1 по Y
double line_2_Y = line_1_Y + space_Y; //начало линии 2 по Y

double line_3_Y = line_2_Y + space_Y; //начало линии 3 по Y
double line_thickness = 5;    //толщина линии
double line_length = 380;     //длина линии

#####
// Создание узлов (nodes)
#####
//=====
// Тень
//=====
private void Shadow() {
    dropShadow = new DropShadow(); //создать объект тень
    dropShadow.setRadius(5.0);     //закругление углов тени
    dropShadow.setOffsetX(5.0);    //смещение тени по X и Y
    dropShadow.setOffsetY(5.0);
    dropShadow.setColor(Color.GRAY); //цвет тени
}
//=====
// Графика
//=====
private void CreateGraphNodes() {
    //===== создать тень для узлов сцены
    Shadow();
    //===== применить эффект для всех узлов группы
    root.setEffect(dropShadow);
    //////////////////////////////////////
    // Создать линии
    //=== линия 1
    line_1 = new Line(lines_X, line_1_Y, line_length,
line_1_Y); //beg_x,beg_y,end_x,end_y
    line_1.setStroke(Color.RED); //цвет линии
    line_1.setStrokeWidth(line_thickness); //ширина линии

    //=== линия 2
    line_2 = new Line(lines_X, line_2_Y, line_length,
line_2_Y); //beg_x,beg_y,end_x,end_y
    line_2.setStroke(Color.GREEN); //цвет линии
    line_2.setStrokeWidth(line_thickness); //ширина линии
    line_2.setStrokeLineCap(StrokeLineCap.ROUND); //закругление линии

    //=== линия 3
    line_3 = new Line(lines_X, line_3_Y, line_length,
line_3_Y); //beg_x,beg_y,end_x,end_y
    line_3.setStroke(Color.BLUE); //цвет линии
    line_3.setStrokeWidth(line_thickness); //ширина линии
    line_3.setStrokeLineCap(StrokeLineCap.ROUND);

    //=== надпись
    text = new Text(); //создать текст
    text.setX(20); //координаты
    text.setY(175);
}

```

```

text.setFont(Font.font("Arial", FontWeight.BOLD, 16));
text.setFill(Color.BLUE); //цвет текста

////////////////////////////////////
////////// поместить все узлы в корень
root.getChildren().addAll(text, line_1, line_2, line_3);
}
//=====
// Элементы управления
//=====
private void CreateControlNodes() {

    //===== кнопка инкремент
    btn_inc = new Button(); //создать кнопку
    btn_inc.setText("Увеличение"); //надпись на кнопке
    btn_inc.setLayoutX(120); //расположение кнопки на сцене
    btn_inc.setLayoutY(200);
    btn_inc.setTextFill(Color.BROWN);
    btn_inc.setFont(Font.font("Arial", FontWeight.BOLD, 12));

    //===== кнопка декремент
    btn_dec = new Button(); //создать кнопку
    btn_dec.setText("Уменьшение"); //надпись на кнопке
    btn_dec.setLayoutX(20); //расположение кнопки на сцене
    btn_dec.setLayoutY(200);
    btn_dec.setTextFill(Color.BROWN);
    btn_dec.setFont(Font.font("Arial", FontWeight.BOLD, 12));

    //===== Слайдер
    slider = new Slider(); //создать слайдер
    slider.setLayoutX(250); //координаты
    slider.setLayoutY(200);
    slider.setPrefWidth(200); //длина слайдера
    slider.setBlockIncrement(1.0); //единица изменения
    slider.setMajorTickUnit(10); //большие деления на шкале
    slider.setMinorTickCount(5); //малые деления на шкале
    slider.setMax(50); //максимальное значение
    slider.setMin(1); //минимальное значение
    slider.setShowTickLabels(true); //показать значения шкалы
    slider.setShowTickMarks(true); //показать деления шкалы
    slider.setSnapToTicks(false); //не привязывать к делениям
    //=== инициализация слайдера
    slider.setValue(line_thickness); //установить ползунок слайдера

    //===== создать колорпикеры
    cp_1 = new ColorPicker(Color.RED); //линия 1
    cp_1.setMaxWidth(0); //минимальная ширина
    cp_1.setLayoutX(line_length + 30); //координаты по X.Y
    cp_1.setLayoutY(line_1_Y - 10);
    //
    cp_2 = new ColorPicker(Color.GREEN); //линия 2
    cp_2.setMaxWidth(0); //минимальная ширина
    cp_2.setLayoutX(line_length + 30); //координаты по X.Y
    cp_2.setLayoutY(line_2_Y - 10);
    //
    cp_3 = new ColorPicker(Color.BLUE); //линия 3
    cp_3.setMaxWidth(0); //минимальная ширина

```

```

cp_3.setLayoutX(line_length + 30);          //координаты по X.Y
cp_3.setLayoutY(line_3_Y - 10);

//===== поместить все узлы в корень
root.getChildren().addAll(slidebar, btn_dec, btn_inc, cp_1, cp_2,
cp_3);
//===== отобразить установки параметров
toDisplay(line_thickness);
//===== обработчики событий
onAction();
}

//=====
// Обработчики событий
//=====
private void onAction() {
    //=== обработчик события нажатия кнопки
    btn_inc.setOnAction((ActionEvent event) -> {
        if (line_thickness < THICKNESS_MAX) {

            line_thickness += 1;          //увеличить толщину линии
        }
        setThicness(line_thickness);    //установить толщину линии
        toDisplay(line_thickness);      //отобразить толщину линии
    });
    //=== обработчик события нажатия кнопки
    btn_dec.setOnAction((ActionEvent event) -> {
        if (line_thickness > 1) {
            line_thickness -= 1;          //уменьшить толщину линии
        }
        setThicness(line_thickness);    //установить толщину линии
        toDisplay(line_thickness);
    });
    //=== обработчик события ползунка слайдера
    slider.valueProperty().addListener((observable) -> {
        if (slider.isValueChanging()) {
            line_thickness = (int) slider.getValue(); //получить значение
ползунка

            setThicness(line_thickness);    //установить толщину линии
            toDisplay(line_thickness);
        }
    });
    //////////////// колорпикеры
    //===== колорпикер линия 1
    cp_1.setOnAction((ActionEvent t) -> {
        //=== получить цвет
        line_1.setStroke(cp_1.getValue());
    });
    //===== колорпикер линия 2
    cp_2.setOnAction((ActionEvent t) -> {
        //=== получить цвет
        line_2.setStroke(cp_2.getValue());
    });
    //===== колорпикер линия 3
    cp_3.setOnAction((ActionEvent t) -> {
        //=== получить цвет
        line_3.setStroke(cp_3.getValue());
}

```

```

    });
}
//=====
// установка значения толщины линии
//=====
public void setThickness(double val) {
    line_1.setStrokeWidth(val);           //ширина синей линии
    line_2.setStrokeWidth(val);           //ширина синей линии
    line_3.setStrokeWidth(val);           //ширина синей линии
}

//=====
// Вывод значения толщины линии
//=====
public void toDisplay(double val) {
    text.setText("Толщина линии: " +
Double.toString(line_thickness)); //текст
    slider.setValue(line_thickness);      //установить ползунок
слайдера
}
//=====
// Начало
//=====
@Override
public void start(Stage stage) {
    //===== заголовок окна
    stage.setTitle("Лабораторная работа №1. доц. Смирнова Н.В.");
    stage.setResizable(false);           //фиксированный размер

    //===== создать узлы сцены с графикой
    CreateGraphNodes();
    //===== создать узлы сцены с элементами управления
    CreateControlNodes();
    //===== создать сцену с размерами и цветом фона
    Scene scene = new Scene(root, 460, 250, Color.CORNSILK);
    //===== поместить сцену в окно
    stage.setScene(scene);
    //===== отобразить окно
    stage.show();
}
//=====
// Точка входа в программу (запускает метод start)
//=====
public static void main(String[] args) {
    launch(args);
}
} //:0~

```

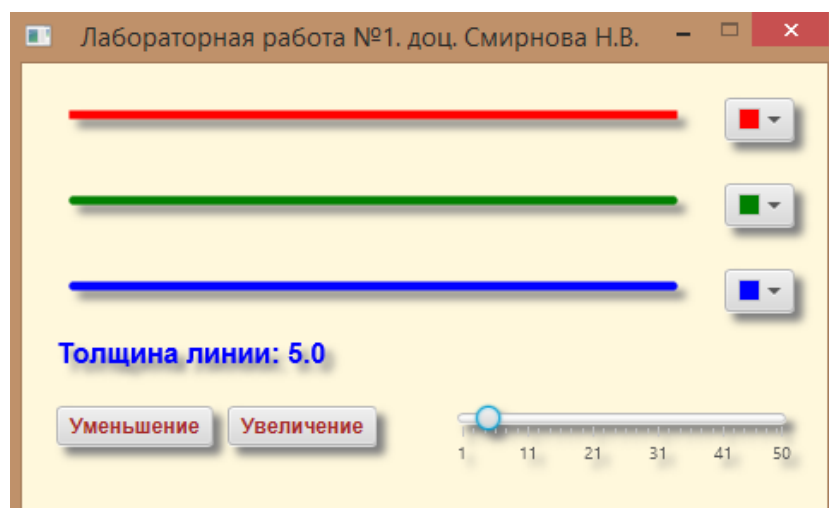
Код демонстраційної програми знаходиться в папці лабораторної роботи №1.

Компіляція програми здійснюється натисканням кнопки **F9**.

Компіляція і запуск програми здійснюється натисканням кнопки **F6**.

Створення файлу, що виконується, здійснюється натисканням кнопки **F11**.

Результат виконання програми:



Методичні вказівки до виконання лабораторної роботи:

1. Створити свій директорій для файлів проекту.
 2. Створити проект в інтегрованому середовищі розробки NetBeansIDE (c:\Temp\Ваша_Папка).
 3. **Створити алгоритм програми**
 4. Написати програму мовою програмування Java (демонстраційна програма).
 5. Змінити прізвище викладача на свою. Указати групу.
 6. Скомпілювати програму.
 7. Створити файл *.jar, що виконується, виконати.
- Файл *.jar буде знаходитися в папці dist проекту.

Контрольні питання:

1. Дати коротку характеристику мові програмування Java.
2. Призначення методу `launch (args)` .
3. Призначення методу `start (Stage primarystage)` .

4. Призначення об'єкта Stage .

5. Призначення об'єкта Scene .

Зміст звіту

У звіті повинні бути представлені:

- лістинг програми.
- висновки за результатами роботи.

Примітки: атрибути тексту програми

1. Шрифт – Courier New 10 пт.

2. Одиночний інтервал.

3. Файл програми, що виконується, з розширенням *.jar і демонстраційний ролик з розширенням *.wmv знаходяться у папці “Приклади виконання”

Лабораторна робота №2

Тема: Створення графічних примітивів 2D графіки

Ціль роботи:

Одержати навички створення графічних примітивів 2D графіки.

Завдання:

- Створити проект.
- Створити три графічні примітиви відповідно до варіанту.
- Вибрати колір і товщину лінії для малювання примітива.
- Вибрати колір заливання.
- Виконати програму

Теоретичні відомості:

Платформа JavaFx 2.0 забезпечує можливості 2D-графіки за допомогою базового класу `Javafx.scene.shape.Shape` і його підкласів `Arc`, `Circle`, `CubicCurve`, `Ellipse`, `Line`, `Path`, `Polygon`, `Polyline`, `QuadCurve`, `Rectangle`, `SVGPath` і `Text` що представляють геометричні примітиви, і текст.

За допомогою властивості `fill` визначається колір внутрішньої області графічного примітиву, а за допомогою властивості `stroke` - колір її контуру.

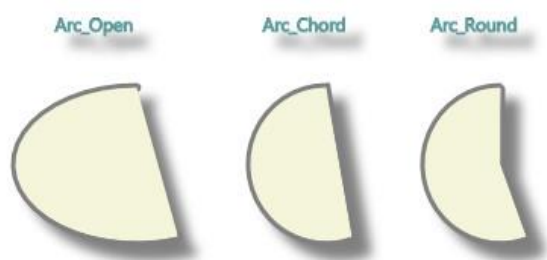
Властивості `strokeLineCap`, `strokeLineJoin`, `strokeMiterLimit`, `strokeType` і `strokeWidth` забезпечують регулювання стилю закінчення ліній контуру графічного примітиву, стилю з'єднання країв сегментів форми, розташування контуру графічного примітиву щодо її границь і ширину контуру графічного примітиву.

За допомогою властивості `smooth` можна встановити згладжування при відображенні форми.

Якщо потрібно відобразити контур графічного примітиву у вигляді пунктирної лінії, необхідно скористатися методом `getStrokeDashArray().addAll()`

класу `Javafx.scene.shape.Shape` для визначення набору пар: довжина пунктиру - інтервал між пунктирами, і властивістю `strokeDashOffset`, що встановлюють інтервал до першого пунктиру. Якщо для набору методом, що повертається `getStrokeDashArray()`, визначити тільки одне значення, тоді воно буде задавати і довжину пунктиру, і інтервал між пунктирами.

Дуга *Arc*



Геометрична форма `Arc` представлена класом `Javafx.scene.shape.Arc`, екземпляр якого можна створити за допомогою класу-фабрики `ArcBuilder` або за допомогою одного з конструкторів:

```
Arc arc = new Arc();  
Arc arc = new Arc(double centerX, double centerY, double radiusX,  
double radiusY, double startAngle, double length);
```

Коло *Circle*



Геометрична форма `Circle` представлена класом `Javafx.scene.shape.Circle`, екземпляр якого можна створити за допомогою класу-фабрики `CircleBuilder` або за допомогою одного з конструкторів:

```
Circle circle = new Circle();  
Circle circle = new Circle(double radius);
```

```
Circle circle = new Circle(double centerX, double centerY, double  
radius);  
Circle circle = new Circle(double centerX, double centerY, double  
radius, Paint fill);  
Circle circle = new Circle(double radius, Paint fill);
```

Клас `Circle` представляє геометричний примітив - коло і має, крім успадкованих від класу `Shape` властивостей, власні властивості: `centerX`, `centerY` і `radius`, що визначають координати центру кола і його радіус.

Лінія *Line*

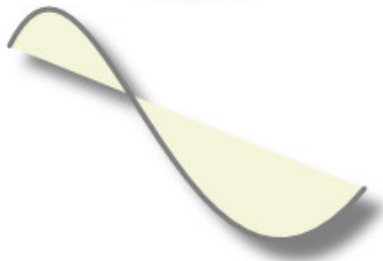


Геометрична форма `Line` представлена класом `Javafx.scene.shape.Line`, екземпляр якого можна створити за допомогою класу-фабрики `LineBuilder` або за допомогою одного з конструкторів:

```
Line line = new Line();  
Line line = new Line(double startX, double startY, double endX, double  
endY);
```

Клас `Line` представляє геометричний примітив - пряму лінію і має, крім успадкованих від класу `Shape` властивостей, власні властивості: `startX`, `startY`, `endX` і `endY`, що визначають координати початку і кінця прямої лінії.

Кубічна крива Без'є *CubicCurve*



Геометрична форма `CubicCurve` представлена класом `Javafx.scene.shape.CubicCurve`, екземпляр якого можна створити за допомогою конструктора:

```
CubicCurve cubicCurve = new CubicCurve();
```

Клас `CubicCurve` представляє геометричний примітив - кубічну криву Без'є і має, крім успадкованих від класу `Shape` властивостей, власні властивості: `controlX1`, `controlX2`, `controlY1`, `controlY2`, `startX`, `startY`, `endX` і `endY`.

За допомогою властивостей `startX`, `startY`, `endX` і `endY` встановлюються початкові і кінцеві координати кривої, а властивості `controlX1`, `controlX2`, `controlY1` і `controlY2` визначають проміжні координати проходження кривої.

Квадратична крива Без'є *QuadCurve*



Геометрична форма `QuadCurve` представлена класом `Javafx.scene.shape.QuadCurve`, екземпляр якого можна створити за допомогою класу-фабрики `QuadCurveBuilder` або за допомогою одного з конструкторів:

```
QuadCurve quadCurve = new QuadCurve();
```

```
QuadCurve quadCurve = new QuadCurve(double startX, double startY,  
double controlX, double controlY, double endX, double endY);
```

Клас `QuadCurve` представляє геометричний примітив - квадратичну криву Без'є і має, крім успадкованих від класу `Shape` властивостей, власні властивості: `controlX`, `controlY`, `startX`, `startY`, `endX` і `endY`.

За допомогою властивостей `startX`, `startY`, `endX` і `endY` встановлюються початкові і кінцеві координати кривої, а властивості `controlX` і `controlY` визначають проміжні координати проходження кривої.

Еліпс *Ellipse*



Геометрична форма `Ellipse` представлена класом

`Javafx.scene.shape.Ellipse`, екземпляр якого можна створити за допомогою класу-фабрики `EllipseBuilder` або за допомогою одного з конструкторів:

```
Ellipse ellipse = new Ellipse();
Ellipse ellipse = new Ellipse(double radiusX, double radiusY);
Ellipse ellipse = new Ellipse(double centerX, double centerY,
double radiusX, double radiusY);
```

Клас `Ellipse` представляє геометричний примітив - еліпс і має, крім успадкованих від класу `Shape` властивостей, власні властивості: `centerX`, `centerY`, `radiusX`, `radiusY` координати, що визначають, центр еліпса та його ширину і висоту.

Прямокутник *Rectangle*



Геометрична форма `Rectangle` представлена класом

`Javafx.scene.shape.Rectangle`, екземпляр якого можна створити за допомогою класу-фабрики `RectangleBuilder` або за допомогою одного з конструкторів:

```
Rectangle rectangle = new Rectangle();
Rectangle rectangle = new Rectangle(double width, double height);
```

```
Rectangle rectangle = new Rectangle(double width, double height, Paint fill);
```

Клас `Rectangle` представляє геометричний примітив - прямокутник і має, крім успадкованих від класу `Shape` властивостей, власні властивості:

```
arcHeight, arcWidth, height, width, x, y.
```

Властивості `x`, `y`, `height` і `width` визначають координати і розміри прямокутника, а властивості `arcHeight` і `arcWidth` дозволяють створювати прямокутник із закругленими кутами, встановлюючи висоту і ширину дуги кута прямокутника.

Ламана лінія *Polyline*

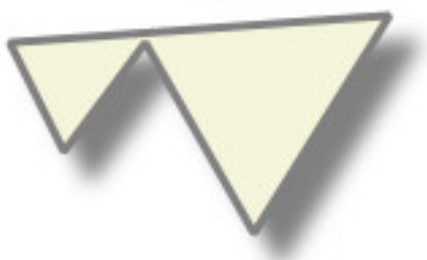


Геометрична форма `Polyline` представлена класом `Javafx.scene.shape.Polyline`, екземпляр якого можна створити за допомогою класу-фабрики `PolyLineBuilder` або за допомогою одного з конструкторів:

```
Polyline polyline = new Polyline();  
Polyline polyline = new Polyline(double[] points);
```

Клас `Polyline` представляє геометричний примітив - ламану лінію. Цей примітив визначається набором `javafx.collections.ObservableList<java.lang.Double>` координат, через які повинна проходити лінія, включаючи початкові і кінцеві координати `x`, `y`.

Багатокутник *Polygon*



Геометрична форма `Polygon` представлена класом `Javafx.scene.shape.Polygon`, екземпляр якого можна створити за допомогою класу-фабрики `PolygonBuilder` або за допомогою одного з конструкторів:

```
Polygon polygon = new Polygon();
Polygon polygon = new Polygon(double[] points);
```

Клас `Polygon` представляє геометричний примітив - багатокутник, по суті, є ламаною лінією `Polyline`, кінці якої з'єднані лінією, і так само, як і лінія `Polyline`, визначається набором `javafx.collections.ObservableList<java.lang.Double>` координат кутів багатокутника.

Фігура *Path*



Геометрична форма `Path` представлена класом `Javafx.scene.shape.Path`, екземпляр якого можна створити за допомогою класу-фабрики `PathBuilder` або за допомогою конструктора:

```
Path path = new Path();
```

Геометрична форма `SVGPath` представлена класом `Javafx.scene.shape.SVGPath`, екземпляр якого можна створити за допомогою класу-фабрики `SVGPathBuilder` або за допомогою конструктора:

```
SVGPath SVG = new SVGPath();
```

Фігура *SVGPath*



Клас `SVGPath` представляє фігуру, створену на основі SVG-шляху, і має, крім успадкованих від класу `Shape` властивостей, власні властивості: `fillRule` - визначає, як області перетинання геометричних форм комбінуються для утворення фігури і `content` - рядок SVG-шляху, що складається з букв, які визначають команди, і числа - параметрів команд.

Стандартні команди SVG-шляхи це:

M - перемістити в позицію (X, Y);

L - провести лінію від поточної крапки до зазначеної крапки (X, Y);

H - провести горизонтальну лінію від поточної крапки до зазначеної крапки (X);

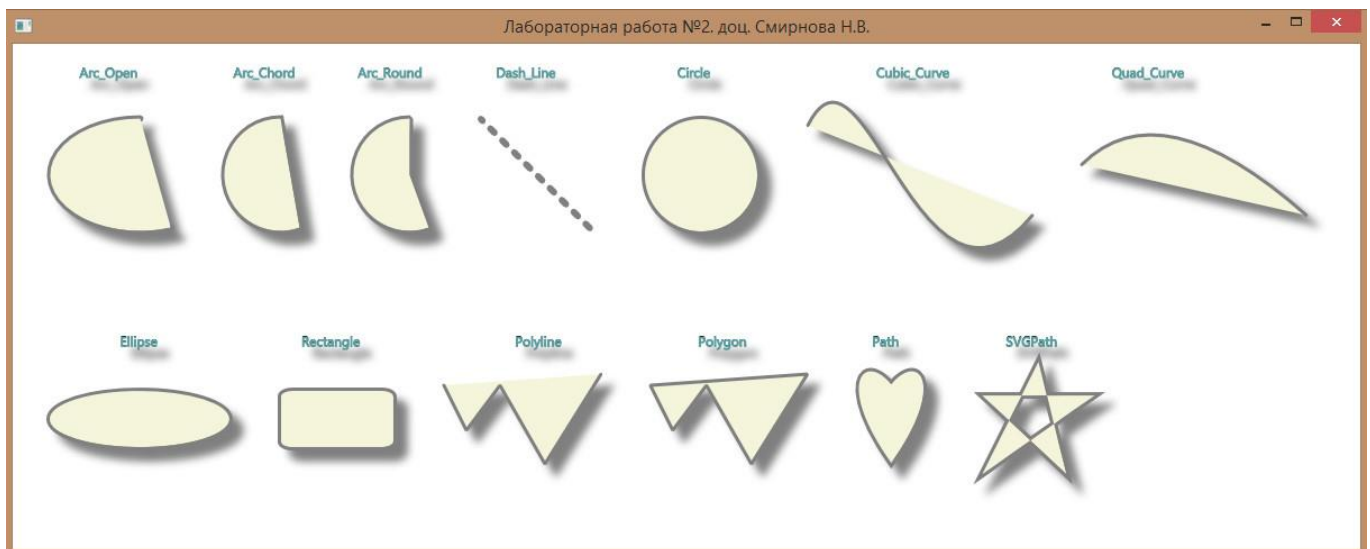
V - провести вертикальну лінію від поточної крапки до зазначеної крапки (Y);

Z - замкнути фігуру;

C - провести криву Без'є (x_1, y_1, x_2, y_2, x, y).

Методичні вказівки до виконання лабораторної роботи:

1. Створити свій директорій для файлів проекту.
2. Створити проект в інтегрованому середовищі розробки NetBeansIDE.
3. Реалізувати завдання - створити графічні примітиви 2D графіки в узгодженні з варіантом.
4. Скомпілювати програму.

Результат виконання програми:**Контрольні питання:**

1. Порядок створення графічного примітива (відповідно до варіанта).
2. Параметри графічного примітива.
3. Атрибути графічного примітива.
4. Параметри і властивості графічного примітива.

Зміст звіту:

- лістинг програми.
- висновки за результатами роботи.

Варіанти:

	Дуга Arc	Полігон Polygon	Лінія Line	кубічна крива CubicCurve	Еліпс Ellipse	Коло Circle	квадратична крива QuadCurve	Прямокутник Rectangle	Полілінія Polyline	Шлях SVGPath
0	+	+	+							
1		+	+	+						
2			+	+	+					
3				+	+	+				
4					+	+	+			
5						+	+	+		
6							+	+	+	
7								+	+	+
8	+								+	+
9		+			+		+			

В заголовку вікна вказати № лабораторної роботи, групу, ПІБ студента

Примітки: файл програми, що виконується, з розширенням *.jar і

демонстраційний ролік з розширенням *.wmv знаходяться у папці “Приклади виконання”.

Лабораторна робота № 3

Тема: Анімація об'єктів комп'ютерної графіки. Клас `TranslateTransition` і події миші

Ціль: Одержати навички роботи з інтерактивною анімацією об'єктів комп'ютерної графіки на основі класу `TranslateTransition`. Навчитися виконувати дії з об'єктом в оброблювачі подій миші.

Завдання:

- Створити проект.
- Створити об'єкт комп'ютерної графіки відповідно до варіанта.
- Створити екземпляр класів анімації об'єкта.
- Створити оброблювач події миші.
- Запустити анімацію.
- Здійснити керування поведінкою об'єкта анімації за допомогою миші.

Теоретичні відомості:

Клас `TranslateTransition` дозволяє створювати переміщення графічного об'єкта з однієї точки в іншу точку за допомогою властивостей:

- `node` – цільовий вузол для анімації;
- `duration` – тривалість анімації;
- `fromx` – початкова координата переміщення по осі x;
- `fromy` – початкова координата переміщення по осі y;
- `fromz` – початкова координата переміщення по осі z;
- `tox` – кінцева координата переміщення по осі x;
- `toy` – кінцева координата переміщення по осі y;
- `toz` – кінцева координата переміщення по осі z;
- `byx` – крок переміщення по осі x;

- byy – крок переміщення по осі y;
- byz – крок переміщення по осі z.

Екземпляр класу `TranslateTransition` можна створити за допомогою конструкторів:

```
public TranslateTransition(Duration duration, Node node)
public TranslateTransition(Duration duration)
public TranslateTransition()
```

Методичні вказівки до виконання лабораторної роботи:

Після створення проекту, у проекті необхідно реалізувати наступне:

- Створити об'єкт анімації відповідно до варіанта.
- Створити екземпляр класу `TranslateTransition`.
- Здійснити прив'язку екземпляра класу `TranslateTransition` до об'єкта анімації.
- Створити оброблювач подій миші.
- Запустити анімацію.

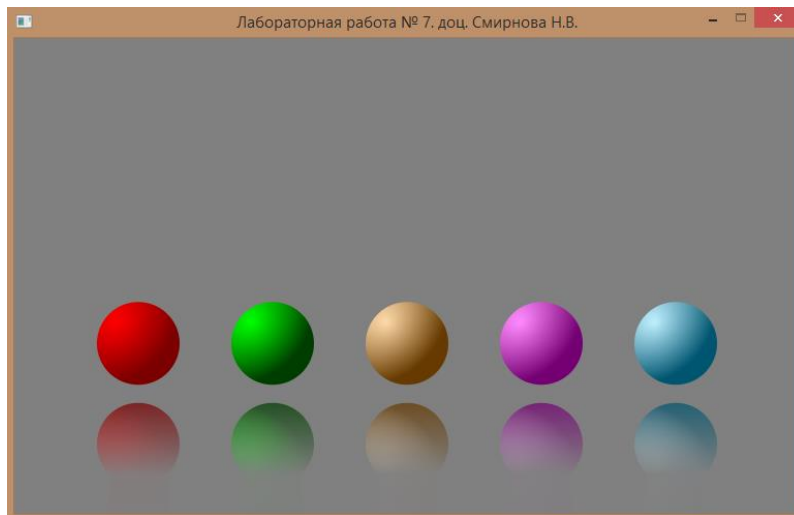
Приклад створення екземпляра класу `TranslateTransition`:

```
private TranslateTransition getTransition(double fromX, double fromY, double
toY, double duration) {
    //===== створити об'єкт керування анімацією
    TranslateTransition transition = new TranslateTransition(new
Duration(duration));
    transition.setFromX(fromX); //координати початкової крапки по X
    transition.setFromY(fromY); //координати початкової крапки по Y
    transition.setToY(toY); //координати кінцевої крапки по Y
    transition.setAutoReverse(true); //автореверс
    transition.setCycleCount(Animation.INDEFINITE); //поки немає stop()
    transition.setInterpolator(Interpolator.EASE_IN); //гальмування в
кінцевих крапках
```

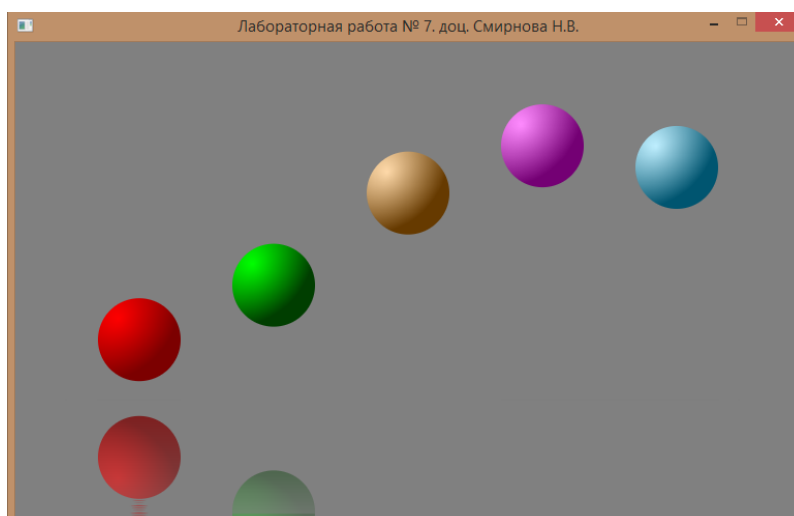
Приклад створення оброблювача події миші:

```
//===== оброблювачі подій куля 1
ball_1.setOnMousePressed(me -> {
    if (ball_1_stopped == true) {
        ball_animation_1.pause();
        ball_1_stopped = false;
    } else {
        ball_animation_1.play();
        ball_1_stopped = true;
    }
});
```

Результат виконання програми:



Реакція об'єкта анімації на подію миші:



Контрольні питання:

1. Призначення і створення класу `TranslateTransition`.
2. Порядок створення анімації об'єкта з використанням `TranslateTransition`.
3. Створення оброблювача події миші.
4. Керування об'єктом анімації в оброблювачі подій миші.

Зміст звіту:

У звіті повинні бути представлені:

- Лістинг програми.
- Висновки за результатами роботи.

Варіанти:

	Еліпс	Сфера	Куб
0	+		
1		+	
2			+
3	+		
4		+	
5			+
6	+		
7		+	
8			+
9	+		

В заголовку вікна вказати № лабораторної роботи, групу, ПІБ студента

Примітки: файл програми, що виконується з розширенням `*.jar` і демонстраційний ролик з розширенням `*.wmv` знаходяться у папці “Приклади виконання”.

Лабораторна робота № 4

Тема: Анімація об'єктів комп'ютерної графіки. Клас `Timeline` і виявлення колізій

Ціль: Одержати навички роботи з інтерактивною анімацією об'єктів комп'ютерної графіки на основі класу `Timeline`. Навчитися відслідковувати зіткнення (колізії) об'єкта анімації з іншими об'єктами сцени.

Завдання:

- Створити проект.
- Створити об'єкт комп'ютерної графіки відповідно до варіанта.
- Створити екземпляр класів анімації об'єкта.
- Запустити анімацію.
- Здійснити керування поведінкою об'єкта анімації за допомогою миші, використовуючи виявлення колізій.

Теоретичні відомості:

Клас `Animation` надає методи керування життєвим циклом анімації:

- `jumpTo(Duration time)` – перехід анімації до зазначеної позиції на тимчасовій шкалі;
- `playFrom(Duration time)` – запуск анімації, починаючи із зазначеної позиції на тимчасовій шкалі;
- `play()` – запуск анімації з поточної позиції на тимчасовій шкалі;
- `playFromStart()` – запуск анімації з первісної позиції на тимчасовій шкалі;
- `stop()` – зупинка анімації;
- `pause()` – пауза анімації.

Анімація по ключових кадрах дозволяє створити видиму зміну значення будь-якої `JavaFx`-властивості за визначений проміжок часу за допомогою класу `Timeline`.

Екземпляр класу `Timeline` можна створити за допомогою одного з конструкторів, що дозволяють установити частоту кадрів і набір ключових кадрів анімації:

```
public Timeline(double targetFramerate)
public Timeline(double targetFramerate, KeyFrame... KeyFrames)
public Timeline(KeyFrame... KeyFrames) и public Timeline()
```

Набір ключових кадрів `Timeline` - анімації можна поповнити методом `getKeyFrames().addAll()`, а зупинити `Timeline` - анімацію і повернути її в початкову позицію – методом `stop()`.

Ключовий кадр `Timeline` - анімації представлений класом `javafx.animation.KeyFrame` і визначає зміни значень JavaFx- властивостей за визначений проміжок часу.

Екземпляр класу `KeyFrame` можна створити за допомогою набору конструкторів, що дозволяють установити час відтворення ключового кадра, ім'я ключового кадра, оброблювач закінчення ключового кадра і набір змін значень JavaFx- властивостей:

```
public KeyFrame(Duration time, java.lang.String name,
EventHandler<ActionEvent> onFinished, java.util.Collection<KeyValue<?>>
values)
public KeyFrame(Duration time, java.lang.String name,
EventHandler<ActionEvent> onFinished, KeyValue<?>... values)
public KeyFrame(Duration time, EventHandler<ActionEvent> onFinished,
KeyValue<?>... values)
public KeyFrame(Duration time, java.lang.String name, KeyValue<?>...
values)
public KeyFrame(Duration time, KeyValue<?>... values)
```

Зміна значення JavaFx - властивості представлена класом `javafx.animation.KeyValue`, екземпляр якого можна створити за допомогою конструкторів, що дозволяють установити змінювану JavaFx- властивість, його кінцеве значення в результаті анімації і спосіб його зміни протягом анімації:

```
public KeyValue(WritableValue<T> target, T endValue, Interpolator<?
super T> interpolator)
public KeyValue(WritableValue<T> target, T endValue)
```

Методичні вказівки до виконання лабораторної роботи:

Після створення проекту, у проекті необхідно реалізувати наступне:

- Створити об'єкт анімації відповідно до варіанта.
- Створити екземпляр класу `KeyFrame` і `Timeline`.
- Здійснити прив'язку екземпляра класу `Timeline` до екземпляра класу `KeyFrame`.
- Створити оброблювач подій миші.
- Створити метод, що реалізує розпізнавання колізій і реакцію миші на ці колізії.
- Запустити анімацію.

Приклад створення екземпляра класу `KeyFrame` і `Timeline`:

```
//===== створити Timeline для анімації, у ньому KeyFrame з
оброблювачем
ball animator = new Timeline(
    new KeyFrame(new Duration(8.0), t -> {
        //=== перевірити зіткнення
        checkForCollision();
        //=== піксел по вертикалі +\ -
        double dy = flag_moving_down ? 1 : -1;
        //=== змінити координати центру кулі

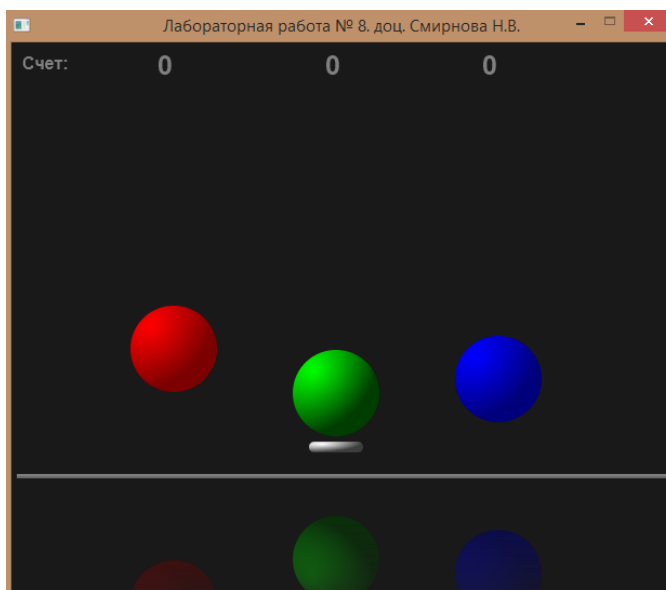
moving_ball_center_Y.setValue(moving_ball_center_Y.getValue() + dy);
    })
);

//===== встановити кількість циклів - нескінченне
ball_animator.setCycleCount(Timeline.INDEFINITE);
```

Приклад коду для створення методу виявлення колізій:

```
//=== зіткнення з верхньою стіною
if (ball.intersects(top_wall.getBoundsInLocal())) {
    //===== варіант анімації FadeTransition
    //=== куля пропущена - значить змінювати прозорість
    if (flag_ball_passed) {
        opacity_up_animation.play();
    }
    //=== поміняти прапор напрямку руху вгору / вниз на
зворотний
    flag_moving_down = !flag_moving_down;
}
```

Результат виконання програми:



Контрольні питання:

1. Призначення і створення класу `KeyFrame`.
2. Порядок створення анімації об'єкта з використанням класу `Timeline`.
3. Створення методу виявлення колізій.
4. Керування об'єктом при виявленні колізій.

Зміст звіту:

У звіті повинні бути представлені:

- Лістинг програми.
- Висновки за результатами роботи.

Варіанти:

	Еліпс	Сфера	Куб
0	+		
1		+	
2			+
3	+		
4		+	
5			+
6	+		
7		+	
8			+
9	+		

В заголовку вікна вказати № лабораторної роботи, групу, ПІБ студента

Примітки: файл програми, що виконується з розширенням *.jar і демонстраційний ролик з розширенням *.wmv знаходяться у папці “Приклади виконання”.

Рекомендована література

1. <http://www.oracle.com/>
2. <http://docs.oracle.com/apps/search/search.jsp?category=java&product=&q=javafx>
3. Johan Vos Pro JavaFX 8. A Definitive Guide to Building. Desktop, Mobile, and Embedded Java Clients / Johan Vos, Weiqi Gao, 2014. – APRESS, 579 с.
4. Carl Dea JavaFX 8 Introduction by Example Second Edition / Carl Dea Mark Heckler, 2014. – APRESS, 383 с.
5. Тимур Машнин JavaFX 2.0: разработка RIA-приложений / Тимур Машнин – СПб.: БХВ-Петербург, 2012. - 320 с.

Методичне забезпечення

1. Приходькіна А.І., Смірнова Н.В. Робоча програма навчальної дисципліни для студентів заочної форми навчання / Укл.: Приходькіна А.І., Смірнова Н.В. – Кіровоград: КНТУ, 2015.
2. Приходькіна А.І. Комп'ютерна графіка: Лекції для студентів заочної форми навчання / Укл.: Приходькіна А.І. – Кіровоград: КНТУ
3. Смірнова Н.В. Комп'ютерна графіка: Методичні вказівки до виконання лабораторних робіт для студентів заочної форми навчання / Укл.: Смірнова Н.В. Смірнов В.В. –Кіровоград: КНТУ, 2015.
4. Смірнова Н.В. Комп'ютерна графіка: Методичні вказівки до виконання контрольних робіт для студентів заочної форми навчання / Укл.: Смірнова Н.В. Смірнов В.В. – Кіровоград: КНТУ, 2015.
5. Смірнова Н.В. Комп'ютерна графіка: методичні вказівки до виконання курсового проектування для студентів заочної форми навчання / Укл.: Смірнова Н.В., Смірнов В.В. – Кіровоград: КНТУ, 2015.
6. Смірнова Н.В. Комп'ютерна графіка: Методичні вказівки до виконання самостійних робіт для студентів заочної навчання/ Укл.: Смірнова Н.В., Смірнов В.В. – Кіровоград: КНТУ, 2015.
7. Смірнов.В.В., Смирнова Н.В. Навчальні відеоролики з прикладами реалізації програм 2D і 3D комп'ютерної графіки
<http://forum.kntu.kr.ua/viewtopic.php?f=633&t=8051&sid=40e4eb7fca9dc189965c7bf352a3a36d>