Міністерство освіти і науки України
Волинський національний університет імені Лесі Українки
Навчально-науковий фізико-технологічний інститут

В. П. Муляр

ПРОЄКТУВАННЯ І РОЗРОБКА КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ

Практикум

Луцьк
Вежа-друк
2022

УДК 004.438(07)
М 90

Рекомендовано до друку науково-методичною радою Волинського національного університету імені Лесі Українки (протокол № 3 від 16.11.2022 р.)

Рецензенти:

Яцюк С. М. – кандидат педагогічних наук, доцент кафедри загальної математики та методики навчання інформатики, декан факультету інформаційних технологій і математики Волинського національного університету імені Лесі Українки;

Багнюк Н. В. – кандидат технічних наук, доцент кафедри комп'ютерної інженерії та кібербезпеки Луцького національного технічного університету.

Муляр В. П.

М 90 Проєктування і розробка користувацьких інтерфейсів: практикум. Луцьк : Вежа-Друк, 2022. 72 с.

У практикумі розкрито основи проєктування та розробки користувацьких інтерфейсів на основі технології JavaFX. Описано основні компоненти графічного інтерфейсу користувача. На конкретних прикладах розглянуто їх використання при створенні JavaFX-додатків. Належну увагу приділено анімації і трансформації зображень в JavaFX.

Для студентів ЗВО педагогічного та технічного спрямування, які вивчають дисципліни «Об'єктно-орієнтоване програмування», «Програмування на Java», «Алгоритми і структури даних», «Комп'ютерна фізика», «Моделювання фізичних явищ і процесів», а також для тих, хто цікавиться проєктуванням та розробкою користувацьких інтерфейсів.

УДК 004.438(07)

ЗМІСТ

Вступ

Потужним інструментом для розробки настільних і мережевих додатків є технологія JavaFX. Вона дозволяє розробникам проєктувати, створювати, тестувати, налагоджувати і розгортати насичені клієнтські додатки, які працюють для різних платформ. Програмний код JavaFX-додатка може посилатися на Application Programming Interface (API) будь-якої бібліотеки Java.

Платформа JavaFX призначена для забезпечення додатків такими складними функціями графічного інтерфейсу користувача, як плавна анімація, веб-перегляд, відтворення аудіо та відео, стилі на основі каскадних таблиць стилів CSS.

JavaFX – це базовий набір інструментів для розробки інтерфейсу користувача, єдине узгоджене середовище програмування додатків як для вбудованих, так і для настільних систем. Іншими словами, JavaFX має на меті використання в багатьох типах пристроїв, таких як мобільні пристрої, смартфони, телевізори, планшетні комп'ютери та десктопи. Для компіляції і запуску JavaFX-додатків більше немає необхідності у встановленні додаткових програм. Усі частини одного проєкту тепер можна реалізувати на одній платформі Java, що, в свою чергу, забезпечує швидкодію, підвищує безпеку, скорочує час розробки і економить витрати на розробку.

Практикум спрямований на формування професійних компетентностей розробки і проєктування додатків на платформі JavaFX із використанням декларативного способу опису інтерфейсу за допомогою мови розмітки FXML, стилізації інтерфейсу за допомогою CSS та ін.

Навчальне видання буде корисним для студентів ЗВО педагогічного та технічного спрямування, які вивчають дисципліни «Об'єктно-орієнтоване програмування», «Програмування на Java», «Алгоритми і структури даних», «Комп'ютерна фізика», «Моделювання фізичних явищ і процесів», а також для тих, хто цікавиться проєктуванням та розробкою користувацьких інтерфейсів.

Практична робота 1. GUI-компонент Hyperlink

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компонент *Hyperlink.*

Теоретичні відомості

Компонент *Hyperlink* представлений класом javafx.scene.control.Hyperlink, екземпляр якого може бути створений за допомогою класу-фабрики HyperlinkBuilder або за допомогою одного з конструкторів:

Hyperlink hlink = New Hyperlink();

Hyperlink hlink = new Hyperlink("[текст]");

Hyperlink hlink = new Hyperlink("[текст]", [вузол значка]);

Клас Hyperlink має:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis , scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– успадковані від класу javafx.scene.control.Labeled властивості: alignment, contentDisplay, font, graphic, graphicTextGap, labelPadding, mnemonicParsing, textAlignment, textFill, textOverrun, text, underline, wrapText;

– успадковані від класу javafx.scene.control.ButtonBase властивості armed та onAction;

– власні властивості visited.

Властивості компонента *Hyperlink*, успадковані від класів *ButtonBase*, *Labeled*, *Control*, *Parent* та *Node*, аналогічні тим самим властивостям, які має і компонент *Button*.

Відмінність набору властивостей компонента *Hyperlink* від

набору властивостей компонента *Button* полягає у наявності властивості *visited* самого класу javafx.scene.control.Hyperlink, яке набуває значення *true* при активації гіперпосилання.

Основне завдання, яке стоїть при розробці компонента *Hyperlink* – це створення привабливого загального зовнішнього вигляду гіперпосилання та його зміна в залежності від того, чи гіперпосилання натиснуте чи ні, чи було воно вже активоване чи ні, а також обробка активації гіперпосилання.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить компонент *Hyperlink*, в меню *Файл* оберіть *Створити проект | Java with Ant | JavaFX | JavaFX Application*, натисніть кнопку *Далі*, введіть ім’я проекту *JavaFXApplicationHyperlink*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplicationHyperlink*, який розширює клас Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationHyperlink 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. У методі *start* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об'єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: Hyperlink” та зробіть видимим об'єкт Stage:

```
Group root = new Group();
Scene sceneOne = new Scene(root, 400, 400, Color.LIGHTGREEN);
primaryStage.setScene(sceneOne);
primaryStage.setTitle("Тестування GUI-компонентів: Hyperlink");
primaryStage.show();
```

4. Створіть екземпляр гіперпосилання *Hyperlink* із текстом. Гіперпосилання помістіть у лівий верхній кут основного вікна JavaFX-додатку, і для нього визначте обробник подій активації гіперпосилання таким чином, що при натисканні гіперпосилання змінюється сцена основного вікна JavaFX-додатку, в якому з'являється область з прокруткою, що відображає значення властивостей гіперпосилання *Hyperlink*, а також кнопка повернення попередньої сцени. При поверненні попередньої сцени змінюється колір гіперпосилання, що вказує, що гіперпосилання було активовано. Для гіперпосилання визначаються такі властивості, як режим накладання, курсор миші, візуальний ефект, переважні розміри, підказка, значок, стиль, вирівнювання, перенесення рядків, а також обробники подій входження миші в область гіперпосилання, клацання мишею на гіперпосиланні та виходу миші з області гіперпосилання, в яких змінюється стиль гіперпосилання:

```
Hyperlink hlink = new Hyperlink("Тестувати властивості");
        hlink.setLayoutX(20);
        hlink.setLayoutY(20);
        hlink.setBlendMode(BlendMode.HARD_LIGHT);
        hlink.setCursor(Cursor.CLOSED_HAND);
```

```
                DropShadow effect = new DropShadow();
                effect.setOffsetX(5);
                effect.setOffsetY(5);
                hlink.setEffect(effect);
                hlink.setPrefSize(200,50);
                hlink.setTooltip(new Tooltip("Це посилання тестування
властивостей класу Hyperlink"));
                Image im = new
Image(this.getClass().getResource("image.png").toString());
                ImageView imv = new ImageView(im);
                imv.setFitHeight(50);
                imv.setFitWidth(50);
                hlink.setGraphic(imv);
                hlink.setStyle("-fx-font: bold italic 9pt Georgia;");
                hlink.setAlignment(Pos.CENTER);
                hlink.setTextAlignment(TextAlignment.CENTER);
                hlink.setContentDisplay(ContentDisplay.RIGHT);
                hlink.setWrapText(true);
                hlink.setOnMouseEntered(e3->{
                    if(!hlink.isVisited()){
                        hlink.setTextFill(Color.BLUE);
                        hlink.setStyle("-fx-font: bold italic 12pt Georgia;");}
                });
                hlink.setOnMousePressed(e4->{
                    if(!hlink.isVisited()){
                    hlink.setTextFill(Color.RED);
                    hlink.setStyle("-fx-font: bold italic 12pt Georgia;"); }
                });
                hlink.setOnMouseExited(e5->{
                    if(!hlink.isVisited()){
                        hlink.setTextFill(Color.BLACK);
                        hlink.setStyle("-fx-font: bold italic 10pt Georgia;");}
                });
                hlink.setOnAction(e->{
                    Group root = new Group();
                    Scene sceneTwo = new Scene(root, 400, 400,
Color.AQUAMARINE);
                    ScrollPane scp = new ScrollPane();
                    Text text = new Text("Властивості, успадковані від
```

```
класу Node:"+"\n"+
                        "властивість blendMode:
"+hlink.blendModeProperty().getValue()+"\n"+
                        "властивість boundsInLocal:
"+hlink.boundsInLocalProperty().getValue()+"\n"+
                        "властивість boundsInParent:
"+hlink.boundsInParentProperty().getValue()+"\n"+
                        "властивість cacheHint:
"+hlink.cacheHintProperty().getValue()+"\n"+
                        "властивість cache:
"+hlink.cacheProperty().getValue()+"\n"+
                        "властивість clip:
"+hlink.clipProperty().getValue()+"\n"+
                        "властивість cursor:
"+hlink.cursorProperty().getValue());
                    scp.setContent(text);
                    scp.setPrefSize(390, 250);
                    scp.setLayoutX(5);
                    scp.setLayoutY(5);
                    Button btn = new Button();
                    btn.setLayoutX(20);
                    btn.setLayoutY(270);
                    btn.setText("Повернутися");
                    btn.setOnAction(e1 -> {
                        hlink.setTextFill(Color.RED);
                        Group root1 = new Group();
                        root1.getChildren().add(hlink);
                        Scene sceneOne = new Scene(root1, 400, 400,
Color.LIGHTGREEN);
                        primaryStage.setScene(sceneOne);
                    });
                    root.getChildren().add(scp);
                    root.getChildren().add(btn);
                    primaryStage.setScene(sceneTwo);
                });
```

5. Скачайте з інтернету рисунок із зображенням смайлика та збережіть його як *image.png* у папці, де розміщений файл JavaFXApplicationHyperlink.java.

6. Гіперпосилання додайте в кореневий вузол графа сцени:

root.getChildren().add(hlink);

7. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

8. В результаті можна буде побачити гіперпосилання (рис. 1), розмір шрифту та колір тексту якої змінюється при наведенні та натисканні мишею. При натисканні мишею на гіперпосиланні у вікно будуть виведені значення властивостей гіперпосилання (рис. 2), при цьому натискання кнопки «Повернутися» дозволить повернути попередню сцену, тільки вже з гіперпосиланням іншого кольору.

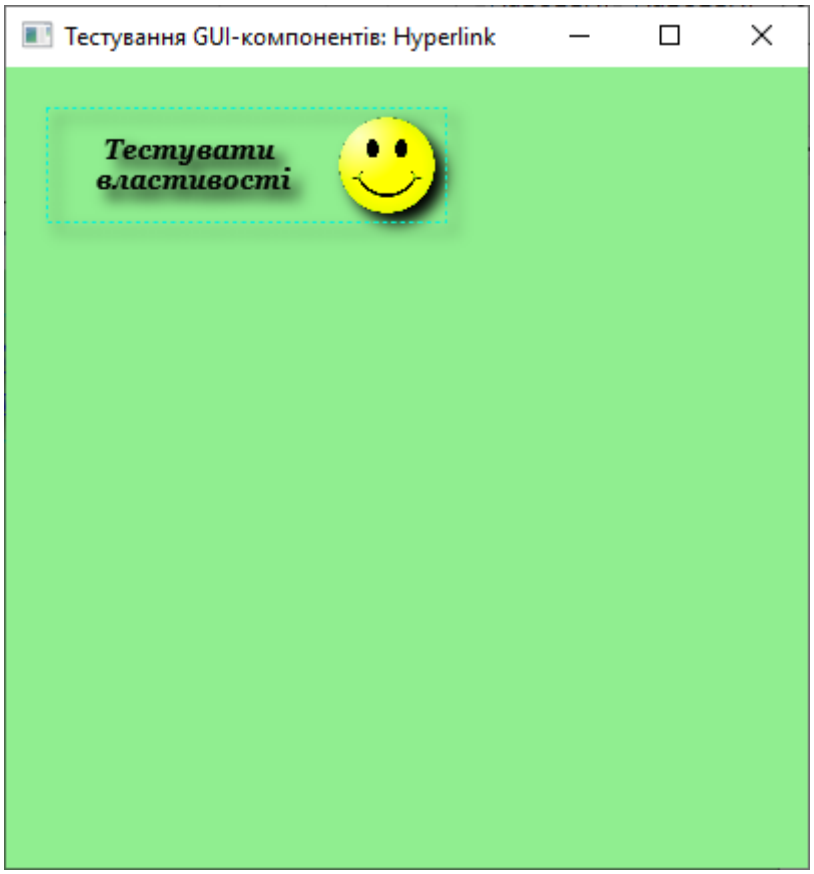

Рис. 1. JavaFX-додаток з GUI-інтерфейсом,
що містить гіперпосилання

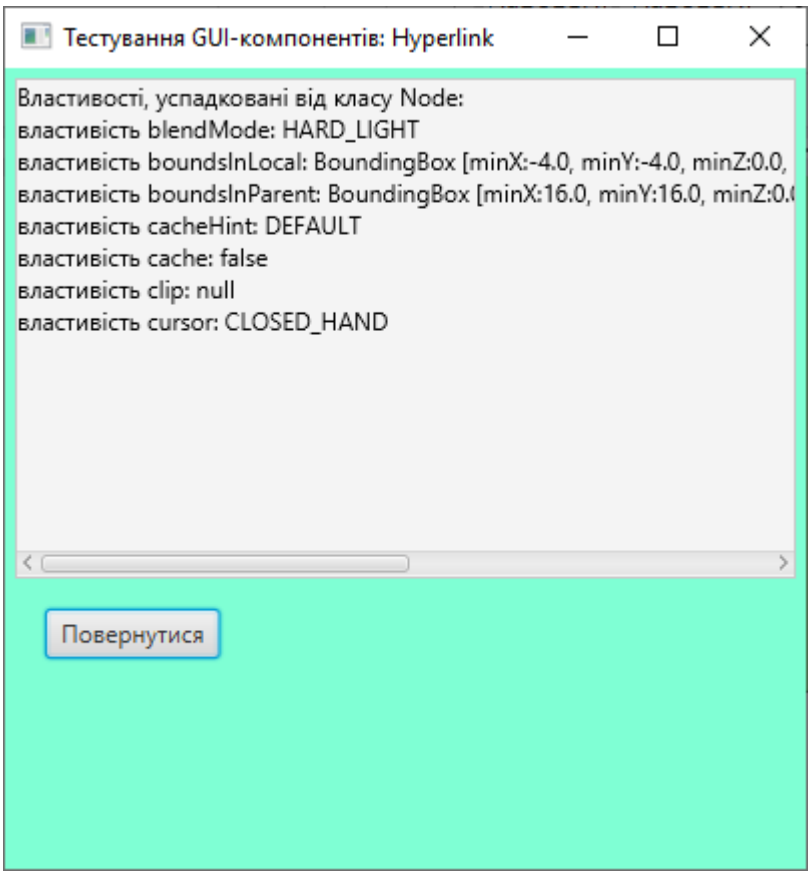

Рис. 2. Результат активації гіперпосилання

Контрольні запитання

1. Яке призначення компоненту *Hyperlink*?
2. Як створити екземпляр компоненту *Hyperlink*?
3. Які основні властивості має компонент *Hyperlink?*

Практична робота 2. GUI-компонент MenuButton

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компонент *MenuButton*.

Теоретичні відомості

Компонент *MenuButton* представлений класом javafx.scene.control.MenuButton, екземпляр якого може бути створений за допомогою класу-фабрики або за допомогою одного з конструкторів:

MenuButton btn = New MenuButton();

MenuButton btn = new MenuButton("[текст]");

MenuButton btn = new MenuButton("[текст]", [вузол значка]);

Клас MenuButton представляє кнопку, при натисканні якої з'являється меню, і має такі властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– успадковані від класу javafx.scene.control.Labeled властивості: alignment, contentDisplay, font, graphic, graphicTextGap, labelPadding, mnemonicParsing, textAlignment, textFill, textOverrun, text, underline, wrapText;

– успадковані від класу javafx.scene.control.ButtonBase властивості armed та onAction;

– власні властивості popupSide і showing.

Властивості компонента MenuButton, успадковані від класів ButtonBase, Labeled, Control, Parent та Node, аналогічні тим самим властивостям, які має і компонент Button.

Відмінність набору властивостей компонента MenuButton від набору властивостей компонента Button полягає в наявності властивостей popupSide та showing самого класу javafx.scene.control.MenuButton.

Властивість popupSide визначає розташування меню щодо кнопки, а властивість showing набуває значення true при відображенні меню.

При створенні компонента MenuButton потрібно створити меню, що є невід'ємною частиною компонента. При цьому меню складається з набору компонентів, представлених класом javafx.scene.control.MenuItem, поповнювати який можна за допомогою методу getItems().addAll() класу javafx.scene.control.MenuButton.

Екземпляр класу MenuItem можна створити за допомогою класу-

фабрики MenuItemBuilder або за допомогою одного з конструкторів:

MenuItem menuItem = new MenuItem();

MenuItem menuItem = new MenuItem("[текст]");

MenuItem menuItem = new MenuItem("[текст]", [вузол значка]);

Клас MenuItem має властивості accelerator, disabled, graphic, id, mnemonicParsing, onAction, parentMenu, parentPopup, style, text, і visible, що дозволяють визначити для елемента меню швидкі клавіші, відключити його, визначити значок, ідентифікатор, розбір тексту, обробка меню, батьківське контекстне меню, стиль, текст та видимість.

Клас MenuItem має підкласи CheckMenuItem, CustomMenuItem, Menu і RadioMenuItem, тому в меню компонента MenuButton можна додавати не тільки елементи MenuItem, але і прапорець CheckMenuItem і перемикач RadioMenuItem, а також роздільники SeparatorMenuItem, представлені класом SeparatorMenuItem, клас CustomMenuItem, та вкладені меню Menu.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить кнопку *MenuButton*, в меню *Файл* оберіть *Створити проект* | *Java with Ant* | *JavaFX* | *JavaFX Application*, натисніть кнопку *Далі*, введіть ім’я проекту *JavaFXApplicationMenuButton*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplicationMenuButton*, який розширює клас Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationMenuButton 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. У методі *start* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об'єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: MenuButton” та зробіть видимим об'єкт Stage:

```
Group root = new Group();
Scene scene = new Scene(root, 400, 400, Color.LIGHTGREEN);
primaryStage.setScene(scene);
primaryStage.setTitle("Тестування GUI-компонентів: MenuButton");
primaryStage.show();
```

4. Створіть екземпляр кнопки MenuButton із текстом. Кнопку помістіть у лівий верхній кут основного вікна JavaFX-програми, і для неї визначте такі властивості, як режим накладання, курсор миші, візуальний ефект, переважні розміри, підказка, значок, стиль, вирівнювання, перенесення рядків і розташування меню щодо кнопки. Створіть елементи меню Вирізати, Копіювати та Вставити. Для них встановіть текст, стиль, швидкі клавіші та обробники подій:

```
MenuButton btn = new MenuButton("Правка");
btn.setLayoutX(20);
btn.setLayoutY(20);
btn.setBlendMode(BlendMode.HARD_LIGHT);
btn.setCursor(Cursor.CLOSED_HAND);
DropShadow effect=new DropShadow();
```

```
        effect.setOffsetX(8);
        effect.setOffsetY(8);
        btn.setEffect(effect);
        btn.setPrefSize(200,80);
        btn.setTooltip(new Tooltip("Це кнопка контекстного меню"));
        Image im = new
Image(this.getClass().getResource("image.png").toString());
        ImageView imv = new ImageView(im);
        imv.setFitHeight(50);
        imv.setFitWidth(50);
        btn.setGraphic(imv);
        btn.setStyle("-fx-font: bold italic 14pt Georgia;");
        btn.setAlignment(Pos.CENTER);
        btn.setContentDisplay(ContentDisplay.LEFT);
        btn.setTextAlignment(TextAlignment.CENTER);
        btn.setGraphicTextGap(10);
        btn.setWrapText(true);
        btn.setPopupSide(Side.RIGHT);
        MenuItem menuItemCut = new MenuItem("Вирізати");
        menuItemCut.setStyle("-fx-font:bold italic 12pt Times;" );
        menuItemCut.setAccelerator(KeyCombination.keyCombination("Ctr
l+U"));
        menuItemCut.setOnAction(e->{
            System.out.println("Вирізаю");
        });
        MenuItem menuItemCopy = new MenuItem("Копіювати");
        menuItemCopy.setStyle("-fx-font:bold italic 12pt Times;" );
        menuItemCopy.setAccelerator(KeyCombination.keyCombination("C
trl+O"));
        menuItemCopy.setOnAction(e1-> {
            System.out.println("Копіюю");
        });
        MenuItem menuItemPaste = new MenuItem("Вставити");
        menuItemPaste.setStyle("-fx-font:bold italic 12pt Times;" );
        menuItemPaste.setAccelerator(KeyCombination.keyCombination("C
trl+P"));
        menuItemPaste.setOnAction(e2->{
            System.out.println("Вставляю");
        });
```

5. Скачайте з інтернету рисунок із зображенням смайлика та збережіть його як *image.png* у папці, де розміщений файл JavaFXApplicationMenuButton.java.

6. Додайте в меню кнопки елементи меню, а кнопку додайте в кореневий вузол графа сцени:

```
btn.getItems().addAll(menuItemCut, menuItemCopy, menuItemPaste
);
root.getChildren().add(btn);
```

7. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті можна буде побачити кнопку при натисканні якої з'являється меню (рис. 1). Елементи меню кнопки відображаються своїм текстом та позначенням швидких клавіш, і при активації елемента меню спрацьовує його обробник подій.

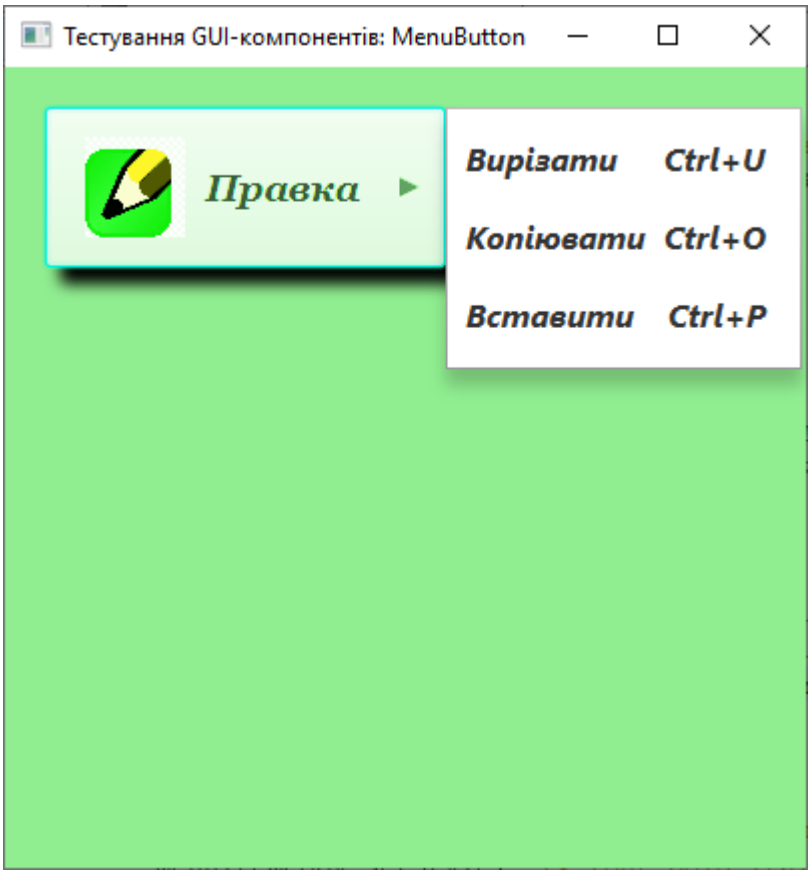

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить кнопку з меню

Контрольні запитання

1. Яке призначення компоненту *MenuButton*?
2. Як створити екземпляр компоненту *MenuButton*?
3. Які основні властивості має компонент *MenuButton?*

Практична робота 3. GUI-компонент SplitMenuButton

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компонент *SplitMenuButton*.

Теоретичні відомості

Компонент SplitMenuButton представлений класом javafx.scene.control.SplitMenuButton, екземпляр якого може бути створений за допомогою класу-фабрики SplitMenuButtonBuilder або за допомогою одного з конструкторів:

SplitMenuButton btn = New SplitMenuButton();

SplitMenuButton btn = New SplitMenuButton(MenuItem... items);

Клас SplitMenuButton представляє кнопку, розділену на дві частини, одна з яких відіграє роль звичайної кнопки, а при натисканні на іншу з'являється меню, і має такі властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– успадковані від класу javafx.scene.control.Labeled властивості: alignment, contentDisplay, font, graphic, graphicTextGap, labelPadding, mnemonicParsing, textAlignment, textFill, textOverrun, text, underline, wrapText;

– успадковані від класу javafx.scene.control.ButtonBase властивості armed та onAction;

– успадковані від класу javafx.scene.control.MenuButton властивості popupSide та showing.

Компонент SplitMenuButton за своєю функціональністю аналогічний до компоненту MenuButton. Відмінність полягає в тому, що компонент SplitMenuButton складається з двох частин, одна з яких може працювати як звичайна кнопка і для неї можна визначити окремий обробник подій onAction, а інша частина виконує основну функцію компонента MenuButton, тобто відкриває контекстне меню.

Меню компонента SplitMenuButton складається з набору компонентів, представлених класом javafx.scene.control.MenuItem, поповнювати який можна за допомогою методу getItems().addAll() класу javafx.scene.control.MenuButton.

Екземпляр класу MenuItem можна створити за допомогою класу-фабрики MenuItemBuilder або за допомогою одного з конструкторів:

MenuItem menuItem = new MenuItem();

MenuItem menuItem = new MenuItem("[текст]");

MenuItem menuItem = new MenuItem("[текст]", [вузол значка]);

Клас MenuItem має властивості accelerator, disabled, graphic, id, mnemonicParsing, onAction, parentMenu, parentPopup, style, text і visible, що дозволяють визначити для елемента меню швидкі клавіші, відключити його, визначити значок, ідентифікатор, розбір тексту, обробник подій , батьківське контекстне меню, стиль, текст та видимість.

Клас MenuItem має підкласи CheckMenuItem, CustomMenuItem, Menu і RadioMenuItem, тому в меню компонента SplitMenuButton можна додавати не тільки елементи MenuItem, але й перемикачі CheckMenuItem і RadioMenuItem, а також роздільники SeparatorMenuItem, представлені класом SeparatorMenuItem, представлені класом SeparatorMenuItem, що розширює клас CustomMenuItem, та вкладені меню Menu.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить кнопку *SplitMenuButton,* в меню *Файл* оберіть *Створити проект | Java with Ant | JavaFX | JavaFX Application*, натисніть кнопку *Далі,* введіть ім’я проекту *JavaFXApplicationSplitMenuButton,* залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplicationSplitMenuButton*, який розширює клас Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationSplitMenuButton 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. У методі *start* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об’єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: SplitMenuButton” та зробіть видимим об’єкт Stage:

```
        Group root = new Group();
        Scene scene = new Scene(root, 400, 400, Color.LIGHTGREEN);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Тестування GUI-компонентів:
SplitMenuButton");
        primaryStage.show();
```

4. Створіть екземпляр кнопки SplitMenuButton. Кнопку помістіть у лівий верхній кут основного вікна JavaFX-додатка, і для неї визначте такі властивості, як текст, режим накладання, курсор миші, візуальний ефект, переважні розміри, підказка, значок, стиль, вирівнювання, перенесення рядків, розташування меню щодо кнопки та обробник подій. Створіть елементи меню Вирізати, Копіювати та Вставити. Для них встановіть текст, стиль, швидкі клавіші та обробники подій:

```
        SplitMenuButton btn = new SplitMenuButton();
        btn.setText("Правка");
        btn.setLayoutX(20);
        btn.setLayoutY(20);
        btn.setBlendMode(BlendMode.HARD_LIGHT);
        btn.setCursor(Cursor.CLOSED_HAND);
        DropShadow effect=new DropShadow();
        effect.setOffsetX(8);
        effect.setOffsetY(8);
        btn.setEffect(effect);
        btn.setPrefSize(200,80);
        btn.setTooltip(new Tooltip("Кнопка редагування"));
        Image im = new
Image(this.getClass().getResource("image.png").toString());
        ImageView imv = new ImageView(im);
        imv.setFitHeight(50);
        imv.setFitWidth(50);
        btn.setGraphic(imv);
        btn.setStyle("-fx-font: bold italic 14pt Georgia;");
        btn.setAlignment(Pos.CENTER);
        btn.setContentDisplay(ContentDisplay.LEFT);
        btn.setTextAlignment(TextAlignment.CENTER);
        btn.setGraphicTextGap(5);
        btn.setWrapText(true);
        btn.setPopupSide(Side.BOTTOM);
        btn.setOnAction(e -> {
```

```
        System.out.println("Обробка натиснення кнопки");
    });
    MenuItem menuItemCut = new MenuItem("Вирізати");
    menuItemCut.setStyle("-fx-text-fill:green;-fx-font:bold italic 12pt
Times;");
    menuItemCut.setAccelerator(KeyCombination.keyCombination("Ctr
l+U"));
    menuItemCut.setOnAction(e1 -> {
        System.out.println("Вирізаю");
    });
    MenuItem menuItemCopy = new MenuItem("Копіювати");
    menuItemCopy.setStyle("-fx-text-fill:green;-fx-font:bold italic 12pt
Times;");
    menuItemCopy.setAccelerator(KeyCombination.keyCombination("C
trl+O"));
    menuItemCopy.setOnAction(e2 -> {
        System.out.println("Копіюю");
    });
    MenuItem menuItemPaste = new MenuItem("Вставити");
    menuItemPaste.setStyle("-fx-text-fill:green;-fx-font:bold italic 12pt
Times;");
    menuItemPaste.setAccelerator(KeyCombination.keyCombination("C
trl+P"));
    menuItemPaste.setOnAction(e3 -> {
        System.out.println("Вставляю");
    });
```

5. Скачайте з інтернету рисунок із зображенням смайлика та збережіть його як *image.png* у папці, де розміщений файл JavaFXApplicationSplitMenuButton.java.

6. У меню кнопки додайте елементи меню, додайте кнопку в кореневий вузол графа сцени:

```
btn.getItems().addAll(menuItemCut, menuItemCopy,
menuItemPaste);
root.getChildren().add(btn);
```

7. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті можна буде побачити кнопку при натисканні на праву частину якої з’являється меню (рис. 1).

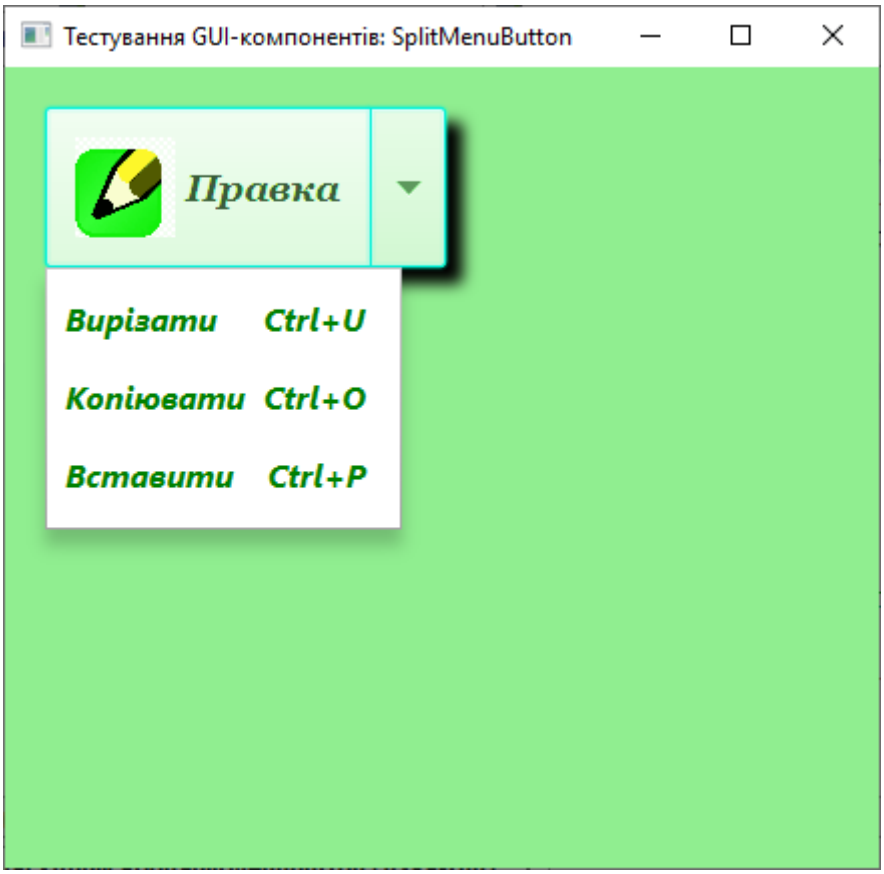

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить подвійну кнопку з меню

Контрольні запитання

1. Яке призначення компоненту *SplitMenuButton*?
2. Як створити екземпляр компоненту *SplitMenuButton*?
3. Які основні властивості має компонент *SplitMenuButton?*

Практична робота 4. GUI-компонент ToggleButton

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компонент *ToggleButton*.

Теоретичні відомості

Компонент ToggleButton представлений класом javafx.scene.control.ToggleButton, екземпляр якого може бути створений за допомогою класу-фабрики ToggleButtonBuilder або за допомогою одного з конструкторів:

ToggleButton btn = New ToggleButton();
ToggleButton btn = New ToggleButton("[текст]");
ToggleButton btn = New ToggleButton("[текст]", [вузол значка]);

Клас ToggleButton представляє кнопку, яка може перебувати в натиснутому та відтисненому станах, і має наступні властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– успадковані від класу javafx.scene.control.Labeled властивості: alignment, contentDisplay, font, graphic, graphicTextGap, labelPadding, mnemonicParsing, textAlignment, textFill, textOverrun, text, underline, wrapText;

– успадковані від класу javafx.scene.control.ButtonBase властивості armed та onAction;

– власні властивості selected і toggleGroup.

Властивості компонента ToggleButton, успадковані від класів ButtonBase, Labeled, Control, Parent та Node, аналогічні тим самим властивостям, які має і компонент Button.

Відмінність набору властивостей компонента ToggleButton від набору властивостей компонента Button полягає в наявності властивостей selected та toggleGroup самого класу javafx.scene.control.ToggleButton.

Властивість selected приймає значення true, якщо кнопка ToggleButton знаходиться в натиснутому стані, а значення false – якщо кнопка ToggleButton віджата.

Властивість toggleGroup вказує групу javafx.scene.control.ToggleGroup, до якої належить кнопка ToggleButton. Об’єднання кнопок ToggleButton у групу ToggleGroup відрізняється від об’єднання перемикачів RadioButton у групу

ToggleGroup – у групі перемикачів RadioButton щонайменше один елемент повинен перебувати у вибраному стані. У групі кнопок ToggleButton такої функціональності немає.

Об’єднання кнопок ToggleButton у групу забезпечує ефект натискання лише єдиної кнопки у групі. При натисканні іншої кнопки групи всі інші кнопки автоматично віджимаються.

Екземпляр класу ToggleGroup можна створити за допомогою класу-фабрики або за допомогою конструктора:

ToggleGroup tgroup = New ToggleGroup();

Клас ToggleGroup має властивість selectedToggle, що вказує на обраний в даний момент елемент групи.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить об’єднання кнопок *ToggleButton*, в меню *Файл* оберіть *Створити проект | Java with Ant | JavaFX | JavaFX Application*, натисніть кнопку *Далі*, введіть ім’я проекту *JavaFXApplicationToggleButton*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplication ToggleButton*, який розширює клас Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationToggleButton 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. У методі *start* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об’єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: ToggleButton” та зробіть видимим об’єкт Stage:

```
Group root = new Group();
Scene scene = new Scene(root, 430, 400, Color.LIGHTGREEN);
primaryStage.setScene(scene);
primaryStage.setTitle("Тестування GUI-компонентів:
ToggleButton");
primaryStage.show();
```

4. Створіть групу ToggleGroup, перший та другий екземпляр кнопки ToggleButton з текстом.

```
ToggleGroup tgroup = new ToggleGroup();
ToggleButton btnOn;
btnOn = new ToggleButton("JavaFX");
ToggleButton btnOf;
btnOf = new ToggleButton("Silverlight");
```

5. Перший екземпляр кнопки помістіть у лівий верхній кут основного вікна JavaFX-програми, і для неї визначте такі властивості, як курсор миші, візуальний ефект, кращі розміри, підказка, значок, стиль, вирівнювання, перенесення рядків, група та обробник подій кнопки. При визначенні стилю кнопки встановіть початковий стиль та обробник зміни значення властивості selected кнопки, в якому стиль кнопки та розмір значка змінюються залежно від того, кнопка натиснута або віджата. При цьому змінюється загальний колір сцени.

Встановіть обробник подій кнопки, який виводить у консоль значення властивості selected кнопки:

```
        btnOn.setLayoutX(10);
        btnOn.setLayoutY(10);
        btnOn.setCursor(Cursor.CLOSED_HAND);
        DropShadow effect = new DropShadow();
        effect.setOffsetX(5);
        effect.setOffsetY(5);
        btnOn.setEffect(effect);
        btnOn.setPrefSize(200,80);
        btnOn.setTooltip(new Tooltip("Це кнопка вибору платформи JavaFX"));
        Image imOn = new Image(this.getClass().getResource("imageOn.jpg").toString());
        ImageView imvOn;
        imvOn = new ImageView(imOn);
        imvOn.setFitHeight(50);
        imvOn.setFitWidth(50);
        btnOn.setGraphic(imvOn);
        btnOn.setStyle("-fx-base:#9900ff;-fx-font: bold italic 16pt Georgia;-fx-textfill:white;");
        btnOn.selectedProperty().addListener((javafx.beans.value.ObservableValue<? extends Boolean> ov, Boolean old_val, Boolean new_val) -> {
            if (new_val.equals(Boolean.TRUE)){btnOn.setStyle("-fx-base:#9900ff;-fx-font: bold italic 16pt Georgia;-fxtext-fill:white;" );
                imvOn.setFitHeight(40);
                imvOn.setFitWidth(40);
                scene.setFill(Color.web("#ffff00"));
                System.out.println("Використовую JavaFX");

            if (new_val.equals(Boolean.FALSE)){
                btnOn.setStyle("-fx-base:#9900ff;-fx-font: bold italic 18pt Georgia;-fx-text-fill:white;");
                imvOn.setFitHeight(50);
                imvOn.setFitWidth(50);
                scene.setFill(Color.LIGHTGREEN);
            }
        });
            btnOn.setAlignment(Pos.CENTER);
```

```
            btnOn.setContentDisplay(ContentDisplay.LEFT);
            btnOn.setTextAlignment(TextAlignment.CENTER);
            btnOn.setGraphicTextGap(20);
            btnOn.setWrapText(true);
            btnOn.setToggleGroup(tgroup);
            btnOn.setOnAction((ActionEvent e) -> {
        System.out.println("JavaFX:"+btnOn.selectedProperty().getValue());
System.out.println("Silverlight:"+btnOf.selectedProperty().getValue());
        });
```

5. Другий екземпляр кнопки помістіть поруч з першою кнопкою, і для неї визначте такі властивості, як курсор миші, візуальний ефект, переважні розміри, підказка, значок, стиль, вирівнювання, перенесення рядків, група і обробник подій кнопки. При визначенні стилю другої кнопки також встановіть початковий стиль та обробник зміни значення властивості selected кнопки, в якому стиль кнопки та розмір значка змінюються залежно від того, кнопка натиснута або віджата. При цьому змінюється загальний колір сцени. Встановіть обробник подій кнопки, який виводить у консоль значення властивості selected кнопки:

```
        btnOf.setLayoutX(220);
        btnOf.setLayoutY(10);
        btnOf.setCursor(Cursor.CLOSED_HAND);
        btnOf.setEffect(effect);
        btnOf.setPrefSize(200,80);
        btnOf.setTooltip(new Tooltip("Це кнопка вибору платформи
Silverlight"));
        Image imOf = new
Image(this.getClass().getResource("imageOf.jpg").toString());
        ImageView imvOf;
        imvOf = new ImageView(imOf);
        imvOf.setFitHeight(50);
        imvOf.setFitWidth(50);
        btnOf.setGraphic(imvOf);
        btnOf.setStyle("-fx-base:#ff0000;-fx-font: bold italic 14pt Georgia;-fx-
textfill:white;" );
        btnOf.selectedProperty().addListener((javafx.beans.value.ObservableV
alue<? extends Boolean> ov, Boolean old_val, Boolean new_val) -> {
            if (new_val.equals(Boolean.TRUE)) {
                btnOf.setStyle("-fx-base:#ff0000;-fx-font: bold italic 12pt
```

```
Georgia;-fxtext-fill:white;" );
                imvOf.setFitHeight(40);
                imvOf.setFitWidth(40);
                scene.setFill(Color.web("#660000"));
                System.out.println("Використовую Silverlight");
            }
            if (new_val.equals(Boolean.FALSE)) {
                btnOf.setStyle("-fx-base:#ff0000;-fx-font: bold italic 14pt
Georgia;-fxtext-fill:white;" );
                imvOf.setFitHeight(50);
                imvOf.setFitWidth(50);
                scene.setFill(Color.LIGHTGREEN);
            }
        });
        btnOf.setAlignment(Pos.CENTER);
        btnOf.setContentDisplay(ContentDisplay.LEFT);
        btnOf.setTextAlignment(TextAlignment.CENTER);
        btnOf.setGraphicTextGap(20);
        btnOf.setWrapText(true);
        btnOf.setToggleGroup(tgroup);
        btnOf.setOnAction((ActionEvent e) -> {
        System.out.println("JavaFX:"+btnOn.selectedProperty().getValue());
        System.out.println("Silverlight:"+btnOf.selectedProperty().getValue());
        });
```

6. Скачайте з інтернету рисунки, назвавши як *imageOn.jpg* та *imageOf.jpg* і збережіть їх у папці, де розміщений файл JavaFXApplicationToggleButton.java.

7. Створені кнопки додайте в кореневий вузол графа сцени:

```
root.getChildren().add(btnOn);
root.getChildren().add(btnOf);
```

8. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті можна буде побачити дві кнопки (рис. 1), при натисканні яких змінюються їх відображення та колір сцени.

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить кнопки ToggleButton

Контрольні запитання

1. Яке призначення компоненту *ToggleButton*?
2. Як створити екземпляр компоненту *ToggleButton*?
3. Які основні властивості має компонент *ToggleButton?*

Практична робота 5. GUI-компонент Label

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компонент *Label*.

Теоретичні відомості

Компонент Label представлений класом javafx.scene.control.Label, екземпляр якого може бути створений за допомогою класу-фабрики LabelBuilder або за допомогою одного з конструкторів:

Label label = new Label();

Label label = new Label("[текст]");

Label label = new Label("[текст]", [вузол значка]);

Клас Label забезпечує відображення текстового підпису, зображення, текстового підпису та зображення одночасно і має такі

властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– успадковані від класу javafx.scene.control.Labeled властивості: alignment, contentDisplay, font, graphic, graphicTextGap, labelPadding, mnemonicParsing, textAlignment, textFill, textOverrun, text, underline, wrapText;

– власна властивість labelFor, що визначає компонент, з яким зв'язується мітка.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить мітку *Label*, в меню *Файл* оберіть *Створити проект | Java with Ant | JavaFX | JavaFX Application*, натисніть кнопку *Далі*, введіть ім’я проекту *JavaFXApplicationLabel*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplicationLabel*, який розширює клас Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
```

```
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationLabel 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. У методі *start* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об’єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: Label” та зробіть видимим об’єкт Stage:

```
Group root = new Group();
Scene scene = new Scene(root, 400, 400, Color.BEIGE);
primaryStage.setScene(scene);
primaryStage.setTitle("Тестування GUI-компонентів: Label");
primaryStage.show();
```

4. Створіть перший екземпляр мітки Label. Мітку помістіть у лівий верхній кут основного вікна JavaFX-програми, і для неї визначте такі властивості, як курсор миші, візуальний ефект, переважні розміри, підказка, значок і вирівнювання:

```
Label limg = new Label();
```

```
limg.setLayoutX(10);
limg.setLayoutY(10);
limg.setCursor(Cursor.CROSSHAIR);
DropShadow effect = new DropShadow();
effect.setOffsetX(10);
effect.setOffsetY(10);
limg.setEffect(effect);
limg.setPrefSize(200, 200);
limg.setTooltip(new Tooltip("Це ті гори, де я не бував"));
Image im = new
Image(this.getClass().getResource("image.jpg").toString());
ImageView imv = new ImageView(im);
imv.setPreserveRatio(true);
imv.setFitHeight(200);
imv.setFitWidth(200);
limg.setGraphic(imv);
limg.setAlignment(Pos.TOP_LEFT);
```

5. Створіть другий екземпляр мітки Label із текстом. Другу мітку помістіть під першою міткою, і для неї визначте такі властивості, як масштабування, курсор миші, переважні розміри, стиль, вирівнювання та перенесення рядків:

```
Label label = new Label("Гори");
label.setLayoutX(45);
label.setLayoutY(155);
label.setScaleX(1.5);
label.setCursor(Cursor.TEXT);
label.setPrefSize(133,30);
label.setStyle("-fx-font: bold italic 14pt Georgia;-fx-text-
fill:#000066;-fx-background-color: lightgrey;");
label.setGraphic(null);
label.setAlignment(Pos.CENTER);
label.setWrapText(true);
```

6. Скачайте з інтернету рисунок із зображенням гір та збережіть його як *image.png* у папці, де розміщений файл JavaFXApplicationLabel.java.

7. Створені мітки додайте в кореневий вузол графа сцени:

```
root.getChildren().add(label);
root.getChildren().add(limg);
```

8. Запустіть створений JavaFX-додаток, клацнувши правою

кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті можна буде побачити дві мітки Label (рис. 1), одна з яких відображає зображення, а інша підпис до зображення.

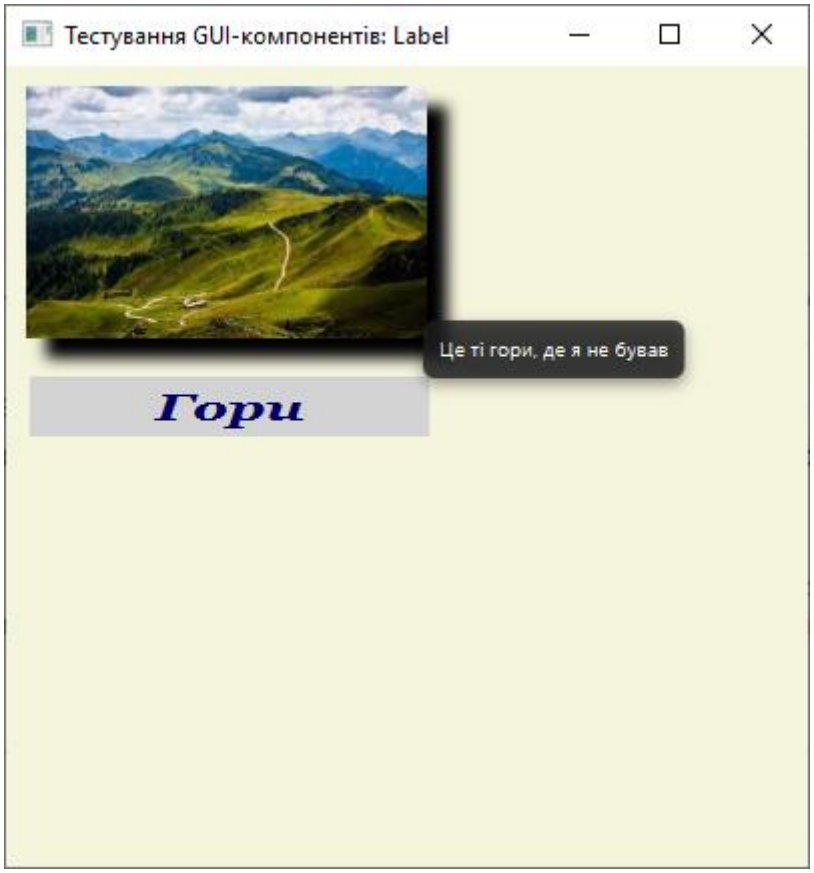

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить мітки Label

Контрольні запитання

1. Яке призначення компоненту *Label*?
2. Як створити екземпляр компоненту *Label*?
3. Які основні властивості має компонент *Label?*

Практична робота 6. GUI-компонент ListView

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компонент *ListView*.

Теоретичні відомості

Компонент ListView представлений класом javafx.scene.control.ListView<T>, екземпляр якого може бути створений за допомогою класу-фабрики ListViewBuilder або за допомогою одного з конструкторів:

ListView<String> listView = New ListView<String>();

ListView<String> listView =

new ListView<String>(ObservableList<String> list);

Набір елементів javafx.collections.ObservableList списку ListView може бути створений за допомогою статичного методу observableArrayList() класу javafx.collections.FXCollections.

Клас ListView<T> представляє список елементів, що прокручується, і має наступні властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– власні властивості: cellFactory, editable, editingIndex, focusModel, items, onEditCancel, onEditCommit, onEditStart, orientation, selectionModel.

За допомогою властивості: orientation класу ListView<T> можна встановити, чи список буде вертикальним або горизонтальним списком елементів.

Властивість cellFactory класу ListView<T> дозволяє заповнити список ListView користувачами компонентами javafx.scene.control.ListCell<T>.

Примірник класу ListCell<T> можна створити за допомогою класу-фабрики ListCellBuilder або за допомогою конструктора:

ListCell<String> cell=new ListCell<String>();

Клас ListCell<T> представляє елемент списку ListView<T> і має такі властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– успадковані від класу javafx.scene.control.Labeled властивості: alignment, contentDisplay, font, graphic, graphicTextGap, labelPadding, mnemonicParsing, textAlignment, textFill, textOverrun, text, underline, wrapText;

– успадковані від класу javafx.scene.control.Cell<T> властивості: editable, editing, empty, item, selected;

– успадкована від класу javafx.scene.control.IndexedCell<T> властивість index;

– власну властивість listView.

Властивість selectionModel класу ListView<T> дозволяє визначити множинність вибору та обробку події вибору елемента списку.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить список *ListView*, в меню *Файл* оберіть *Створити проект* | *Java with Ant* | *JavaFX* | *JavaFX Application*, натисніть кнопку *Далі*, введіть ім’я проекту *JavaFXApplicationListView*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplicationListView*, який розширює клас Application.

```
package javafxapplicationbutton;
```

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationListView 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. У методі *start* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об’єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: ListView” та зробіть видимим об’єкт Stage:

```
Group root = new Group();
Scene scene = new Scene(root, 500, 400, Color.BEIGE);
primaryStage.setScene(scene);
primaryStage.setTitle("Тестування GUI-компонентів: ListView");
primaryStage.show();
```

4. Створіть перший екземпляр списку та відповідний екземпляр набору елементів списку. Список помістіть в лівий верхній кут основного вікна JavaFX-додатка, і для нього визначте такі властивості, як курсор миші, візуальний ефект, стиль, кращі розміри, підказка, орієнтація, фабрика елементів списку, яка заповнює список осередками, що містять кнопки:

```
ObservableList<String> categories =
FXCollections.observableArrayList("Побутова техніка", "Посуд",
"Госптовари");
ListView<String> listViewC = new ListView<>(categories);
listViewC.setLayoutX(10);
listViewC.setLayoutY(10);
listViewC.setCursor(Cursor.OPEN_HAND);
final DropShadow effect = new DropShadow();
effect.setOffsetX(10);
effect.setOffsetY(10);
listViewC.setEffect(effect);
listViewC.setStyle("-fx-border-width:3pt;-fx-border-color:navy;-fx-
font:bold 12pt Georgia;");
listViewC.setPrefSize(470, 170);
listViewC.setTooltip(new Tooltip("Виберіть категорію товару"));
listViewC.setOrientation(Orientation.HORIZONTAL);
listViewC.setCellFactory((ListView<String> p) -> {
    Button btn = new Button();
    btn.setEffect(effect);
    btn.setStyle("-fx-background-color:#66ccff;");
    btn.setPrefSize(130, 50);
    btn.setCursor(Cursor.NONE);
    btn.setWrapText(true);
    final ListCell<String> cell = new ListCell<String>(){
        @Override
        public void updateItem(String item, boolean empty) {
            super.updateItem(item, empty);
            if (item != null) {
                btn.setText(item);
                this.setGraphic(btn);
            }
        }
    };
```

```
            return cell;
        });
```

5. Створіть другий екземпляр списку та відповідний йому екземпляр набору елементів списку. Другий список помістіть під першим списком, і для нього визначте такі властивості, як курсор миші, візуальний ефект, стиль, переважні розміри, орієнтація, множинність вибору та видимість:

```
        ObservableList<String> goodsH =
FXCollections.observableArrayList("Вентилятор", "М’ясорубка",
"Насос");
        ListView<String> listViewH;
        listViewH = new ListView<>(goodsH);
        listViewH.setLayoutX(10);
        listViewH.setLayoutY(220);
        listViewH.setCursor(Cursor.CROSSHAIR);
        listViewH.setEffect(effect);
        listViewH.setStyle("-fx-border-width:3pt;-fx-border-color:navy;-fx-
font:bold 12pt Georgia;");
        listViewH.setPrefSize(200, 150);
        listViewH.setOrientation(Orientation.VERTICAL);
        listViewH.getSelectionModel().setSelectionMode(SelectionMode.M
ULTIPLE);
        listViewH.setVisible(false);
```

6. Створіть третій екземпляр списку та відповідний екземпляр набору елементів списку. Третій список помістіть під першим списком, і для нього визначте такі властивості, як курсор миші, візуальний ефект, стиль, переважні розміри, орієнтація, множинність вибору та видимість:

```
        ObservableList<String> goodsU =
FXCollections.observableArrayList("Чайник", "Каструля", "Пательня");
        ListView<String> listViewU;
        listViewU = new ListView<>(goodsU);
        listViewU.setLayoutX(10);
        listViewU.setLayoutY(220);
        listViewU.setCursor(Cursor.CROSSHAIR);
        listViewU.setEffect(effect);
        listViewU.setStyle("-fx-border-width:3pt;-fx-border-color:navy;-fx-
font:bold 12pt Georgia;");
        listViewU.setPrefSize(200, 150);
```

```
listViewU.setOrientation(Orientation.VERTICAL);
listViewU.getSelectionModel().setSelectionMode(SelectionMode.M
ULTIPLE);
listViewU.setVisible(false);
```

7. Створіть четвертий екземпляр списку та відповідний екземпляр набору елементів списку. Четвертий список помістіть під першим списком і для нього визначте такі властивості, як курсор миші, візуальний ефект, стиль, переважні розміри, орієнтація, множинність вибору та видимість:

```
ObservableList<String> goodsW =
FXCollections.observableArrayList("Шланг", "Лопата", "Ваги");
ListView<String> listViewW;
listViewW = new ListView<>(goodsW);
listViewW.setLayoutX(10);
listViewW.setLayoutY(220);
listViewW.setCursor(Cursor.CROSSHAIR);
listViewW.setEffect(effect);
listViewW.setStyle("-fx-border-width:3pt;-fx-border-color:navy;-fx-
font:bold 12pt Georgia;");
listViewW.setPrefSize(200, 150);
listViewW.setOrientation(Orientation.VERTICAL);
listViewW.getSelectionModel().setSelectionMode(SelectionMode.M
ULTIPLE);
listViewW.setVisible(false);
```

8. Створіть обробник вибору елемента списку, який візуалізує ще один список, що відповідає обраному елементу:

```
listViewC.getSelectionModel().selectedItemProperty().addListener((
ObservableValue<? extends String> ov, String old_val, String new_val) ->
{
        if(new_val.equals("Побутова техніка")){
            listViewH.setVisible(true);
            listViewU.setVisible(false);
            listViewW.setVisible(false);
        }
        if(new_val.equals("Посуд")){
            listViewU.setVisible(true);
            listViewH.setVisible(false);
            listViewW.setVisible(false);
        }
```

```
        if(new_val.equals("Госптовари")){
            listViewW.setVisible(true);
            listViewH.setVisible(false);
            listViewU.setVisible(false);
        }
    });
```

9. Створені списки додайте до кореневого вузла графа сцени:

```
root.getChildren().add(listViewC);
root.getChildren().add(listViewH);
root.getChildren().add(listViewU);
root.getChildren().add(listViewW);
```

10. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті можна буде побачити горизонтальний список (рис. 1), при виборі одного з елементів якого нижче з'являється інший вертикальний список, що відповідає вибраному елементу. Вертикальний список дозволяє вибрати кілька елементів за допомогою кнопки <Ctrl>.

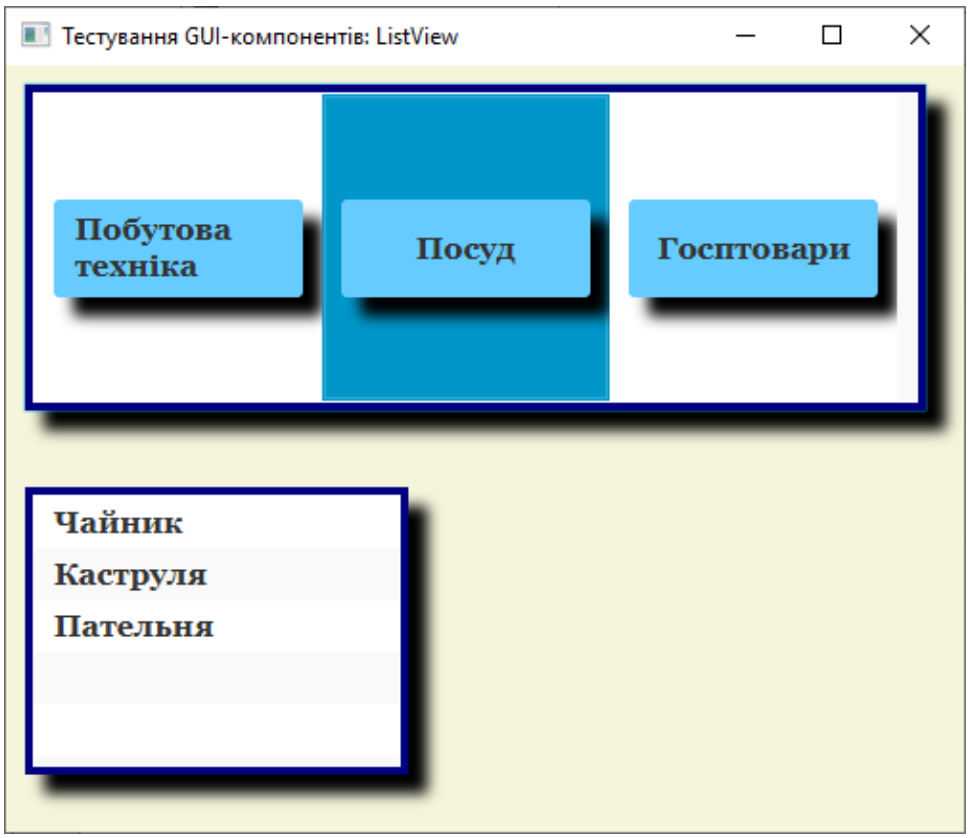

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить списки listView

Контрольні запитання

1. Яке призначення компоненту *listView*?
2. Як створити екземпляр компоненту *listView*?
3. Які основні властивості має компонент *listView?*

Практична робота 7. GUI-компонент TableView

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компонент *TableView*.

Теоретичні відомості

Компонент TableView представлений класом javafx.scene.control.TableView<S>, екземпляр якого може бути створений за допомогою класу-фабрики TableViewBuilder або за допомогою одного з конструкторів:

TableView<S> table = new TableView<S>();

TableView<S> table = new TableView<S>(ObservableList<S> list);

Набір даних javafx.collections.ObservableList для рядків таблиці TableView може бути створений за допомогою статичного методу observableArrayList() класу javafx.collections.FXCollections.

Клас TableView<S> представляє таблицю елементів і має такі властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– власні властивості: columnResizePolicy, editable, editingCell, focusModel, items, placeholder, rowFactory, selectionModel, tableMenuButtonVisible.

Таблиця TableView<S> складається з набору стовпців javafx.collections.ObservableList <TableColumn<S,?>>, який можна поповнити за допомогою методу getColumns().addAll() класу javafx.scene.control.TableView<S>.

За допомогою властивості items класу TableView<S> набір стовпців таблиці ObservableList<TableColumn<S,?>> пов'язується з набором даних для рядків таблиці ObservableList<S>.

Властивість tableMenuButtonVisible забезпечує опцію контекстного меню, що дозволяє регулювати відображення стовпців таблиці.

Властивість placeholder дає можливість встановити вузол Node, що відображається у разі відсутності даних таблиці.

Властивість rowFactory дозволяє наповнити таблицю компонентами користувача javafx.scene.control.TableRow<S>. Клас TableRow<S> розширює клас javafx.scene.control.IndexedCell<S> і представляє рядки таблиці TableView<S>.

Властивість selectionModel дає можливість визначити виділені елементи таблиці та встановити множинність вибору.

Стовпці таблиці TableView представляє клас javafx.scene.control.TableColumn<S,T>, екземпляр якого може бути створений за допомогою класу-фабрики TableColumnBuilder або за допомогою конструкторів:

TableColumn column = New TableColumn();

TableColumn column = new TableColumn("[текст]");

Клас TableColumn<S,T> має властивості: cellFactory, cellValueFactory, comparator, contextMenu, editable, maxWidth, minWidth, onEditCancel, onEditCommit, onEditStart, parentColumn, prefWidth, resizable, sortable, sortType, tableView, text, visible, width.

Властивість cellFactory класу TableColumn дає можливість наповнити стовпець користувальницькими компонентами, представленими класом javafx.scene.control.TableCell<S,T>, який розширює клас javafx.scene.control.IndexedCell<T>.

Властивості maxWidth, minWidth, prefWidth та width забезпечують визначення ширини стовпця. Властивості sortable, sortType та comparator регулюють сортування даних стовпця. Властивість text дозволяє визначити шапку стовпця, а властивість

resizable – можливість зміни ширини стовпця користувачем.

За допомогою властивості cellValueFactory можна заповнити даними осередку стовпця таблиці. Для цього можна використати клас javafx.scene.control.cell.PropertyValueFactory<S,T>, що реалізує інтерфейси Callback<TableColumn.CellDataFeatures<S,T> і ObservableValue<T>>. Клас PropertyValueFactory<S,T> має методи public ObservableValue<T> call(TableColumn.CellDataFeatures<S,T> param) та public final java.lang.String getProperty(). Екземпляр класу PropertyValueFactory<S,T> може бути створений за допомогою класу-фабрики PropertyValueFactoryBuilder або за допомогою конструктора:

PropertyValueFactory<S, T> pvf =

new PropertyValueFactory<S, T>([java.lang.String property]);

Стовпець TableColumn<S,T> може містити набір javafx.collections.ObservableList<TableColumn<S,?>> вкладених стовпців, поповнити який можна за допомогою методу getColumns().addAll() класу javafx.scene.control.TableColumn<S,T>.

Для створення стовпця з редагованими елементами необхідно створити клас, що розширює клас TableCell<S,T>, в якому перевизначаються методи startEdit(), cancelEdit(), commitEdit() і updateItem() з використанням текстових полів, що редагуються, і наповнити стовпець його екземплярами з Використовуючи властивість cellFactory. Властивості onEditCancel, onEditCommit і onEditStart забезпечать обробку подій редагування. Властивість editable класів TableColumn<S,T> і TableView<S> визначає редагування стовпця та таблиці.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить таблицю *TableView*, в меню *Файл* оберіть *Створити проект* | *Java with Ant* | *JavaFX* | *JavaFX Application*, натисніть кнопку *Далі*, введіть ім’я проекту *JavaFXApplicationTableView*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplicationTableView*, який розширює клас Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
```

```
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationTableView 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. Попередньо створіть клас *Hotel*, який представляє дані таблиці. Для цього в меню *Файл* оберіть *Створити Файл* | *Java* | *Java Class*, натисніть кнопку *Далі*, введіть ім’я класу *Hotel* та натисніть кнопку *Готово*.

```
public class Hotel {
    private final StringProperty name;
    private final StringProperty resort;
    private final StringProperty category;
    private final DoubleProperty rate;
    public Hotel(String name, String resort, String category, double rate) {
        this.name = new SimpleStringProperty(name);
        this.resort = new SimpleStringProperty(resort);
```

```
            this.category = new SimpleStringProperty(category);
            this.rate = new SimpleDoubleProperty(rate);
        }
        public StringProperty nameProperty() {
            return name;
        }
        public String getName(){
            return name.getValue();
        }
        public StringProperty resortProperty() {
            return resort;
        }
        public String getResort(){
            return resort.getValue();
        }
        public StringProperty categoryProperty() {
            return category;
        }
        public String getCategory(){
            return category.getValue();
        }
        public DoubleProperty rateProperty() {
            return rate;
        }
        public Double getRate(){
            return rate.getValue();
        }
    }
```

3. У класі JavaFXApplicationTableView у методі *start* вилучіть весь код.

4. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об’єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: TableView” та зробіть видимим об’єкт Stage:

```
Group root = new Group();
Scene scene = new Scene(root, 600, 400, Color.BEIGE);
primaryStage.setScene(scene);
primaryStage.setTitle("Тестування GUI-компонентів: TableView");
primaryStage.show();
```

5. Створіть заголовок Label таблиці. Помістіть його в лівий верхній кут основного вікна JavaFX-додатку і для нього встановіть текст, бажані розміри, стиль та вирівнювання:

```
Label label=new Label("TOP20 готелів Турції");
label.setLayoutX(10);
label.setLayoutY(10);
label.setPrefSize(500, 30);
label.setStyle("-fx-font: bold italic 16pt Georgia;-fx-text-fill:#000066;-fxbackground-color:lightgrey;");
label.setAlignment(Pos.CENTER);
```

6. Створіть набір даних таблиці та екземпляр таблиці TableView, який міститься під заголовком Label таблиці і для якого визначаються такі властивості, як контекстне меню відображення стовпців, курсор миші, підказка, стиль, краща ширина і множинність вибору. Створіть стовпці таблиці, які пов'язуються з властивостями класу даних Hotel та додаються до таблиці TableView:

```
ObservableList<Hotel> hotels = FXCollections.observableArrayList(
        new Hotel("Amara Dolce Vita","Кемер","HV1",4.5),
        new Hotel("Club Boran Mare Beach","Кемер","HV1",4.7),
        new Hotel("Delphin Botanik World of Paradise","Аланія","5*",4.4),
        new Hotel("Kamelya World Hotel Fulya","Сіде","5*",4.8),
        new Hotel("Delphin Deluxe Resort","Алания","5*",4.7));
        TableView<Hotel> table = new TableView<>();
        table.setLayoutX(10);
        table.setLayoutY(50);
        table.setTableMenuButtonVisible(true);
        table.setCursor(Cursor.TEXT);
        table.setTooltip(new Tooltip("Популярні готелі Турції"));
        table.setStyle("-fx-font: 12pt Arial;");
        table.setPrefWidth(500);
        table.setPrefHeight(200);
table.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
        TableColumn nameCol = new TableColumn("Готель");
        nameCol.setCellValueFactory(new PropertyValueFactory<Hotel, String>("name"));
        nameCol.setPrefWidth(250);
        nameCol.setResizable(false);
        nameCol.setSortable(true);
```

```
        TableColumn resortCol = new TableColumn("Курорт");
        resortCol.setCellValueFactory(new
PropertyValueFactory<Hotel, String>("resort"));
        TableColumn categoryCol = new
TableColumn("Категорія");
        categoryCol.setCellValueFactory(new
PropertyValueFactory<Hotel, String>("category"));
        TableColumn rateCol = new TableColumn("Рейтинг");
        rateCol.setCellValueFactory(new
PropertyValueFactory<Hotel, String>("rate"));
        table.setItems(hotels);
        table.getColumns().addAll(nameCol, resortCol, categoryCol,
rateCol);
```

7. Створені заголовок Label та таблицю TableView додайте до кореневого вузла графа сцени:

```
root.getChildren().add(label);
root.getChildren().add(table);
```

8. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті можна буде побачити таблицю (рис. 1), що має контекстне меню регулювання відображення стовпців, підказку, ширину стовпців, що змінюється, і опцію сортування даних стовпця при натисканні на її шапці.

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить таблицю TableView

Контрольні запитання

1. Яке призначення компоненту *TableView*?
2. Як створити екземпляр компоненту *TableView*?
3. Які основні властивості має компонент *TableView?*

Практична робота 8. GUI-компонент ChoiceBox

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компонент *ChoiceBox*.

Теоретичні відомості

Компонент ChoiceBox представлений класом javafx.scene.control.ChoiceBox<T>, екземпляр якого може бути створений за допомогою класу-фабрики ChoiceBoxBuilder або за допомогою одного з конструкторів:

ChoiceBox<String> choice=new ChoiceBox<String>();

ChoiceBox<String> choice = new ChoiceBox<String>(ObservableList<String> list);

Набір елементів javafx.collections.ObservableList списку ChoiceBox може бути створений за допомогою статичного методу observableArrayList() класу javafx.collections.FXCollections.

Клас ChoiceBox<T> представляє список елементів, що випадає, і має наступні властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width;

– власні властивості items, selectionModel і showing.

За допомогою властивості items класу ChoiceBox<T> можна встановити набір ObservableList<T> елементів списку.

Властивість selectionModel класу ChoiceBox<T> дає можливість визначити вибір елемента та обробку події вибору елемента списку.

Властивість showing класу ChoiceBox<T> приймає значення true, якщо відображається список, що випадає.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить список *ChoiceBox*, в меню *Файл* оберіть *Створити проект* | *Java with Ant* | *JavaFX* | *JavaFX Application*, натисніть кнопку *Далі*, введіть ім’я проекту *JavaFXApplicationChoiceBox*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplicationChoiceBox*, який розширює клас Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationTableView 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();
        }
        public static void main(String[] args) {
            launch(args);
        }
    }
```

2. У методі *start* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об’єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: ChoiceBox” та зробіть видимим об’єкт Stage:

```
Group root = new Group();
Scene scene = new Scene(root, 400, 300, Color.BEIGE);
primaryStage.setScene(scene);
primaryStage.setTitle("Тестування GUI-компонентів: ChoiceBox");
primaryStage.show();
```

4. Створіть екземпляр списку та відповідний екземпляр набору елементів списку. Список помістіть у лівому верхньому куті основного вікна JavaFX-додатка, і для нього визначте такі властивості, як режим накладання, курсор миші, візуальний ефект, стиль, переважні розміри, підказка, початковий вибір першого елемента списку:

```
ObservableList<String> country =
FXCollections.observableArrayList("Україна","США","Великобританія
");
ChoiceBox<String> choice = new ChoiceBox<>(country);
choice.setLayoutX(10);
choice.setLayoutY(10);
choice.setBlendMode(BlendMode.HARD_LIGHT);
choice.setCursor(Cursor.CLOSED_HAND);
DropShadow effect = new DropShadow();
effect.setOffsetX(8);
effect.setOffsetY(8);
choice.setEffect(effect);
choice.setStyle("-fx-text-fill:#000000;-fx-border-width:5pt;-fx-
bordercolor:#d2691e;-fx-font:bold italic 12pt Georgia;");
choice.setPrefSize(200, 50);
choice.setTooltip(new Tooltip("Виберіть країну"));
choice.getSelectionModel().selectFirst();
```

5. Створіть перший екземпляр вузла зображення ImageView, помістіть під списком і для нього визначте такі властивості, як розміри та збереження пропорцій вихідного зображення:

```
Image imR = new
Image(this.getClass().getResource("imageR.jpg").toString());
ImageView imvR;
imvR = new ImageView(imR);
imvR.setPreserveRatio(true);
imvR.setFitHeight(200);
imvR.setFitWidth(200);
imvR.setLayoutX(10);
imvR.setLayoutY(100);
```

6. Створіть другий екземпляр вузла зображення ImageView, помістіть під списком і для нього визначте такі властивості, як розміри, збереження пропорцій вихідного зображення та видимість:

```
Image imU = new
Image(this.getClass().getResource("imageU.jpg").toString());
ImageView imvU;
imvU = new ImageView(imU);
imvU.setPreserveRatio(true);
imvU.setFitHeight(200);
imvU.setFitWidth(200);
imvU.setLayoutX(10);
imvU.setLayoutY(100);
imvU.setVisible(false);
```

7. Створіть третій екземпляр вузла зображення ImageView, помістіть під списком і для нього визначте такі властивості, як розміри, збереження пропорцій вихідного зображення та видимість:

```
Image imG = new
Image(this.getClass().getResource("imageG.jpg").toString());
ImageView imvG;
imvG = new ImageView(imG);
imvG.setPreserveRatio(true);
imvG.setFitHeight(200);
imvG.setFitWidth(200);
imvG.setLayoutX(10);
imvG.setLayoutY(100);
imvG.setVisible(false);
```

8. Створіть обробник вибору елемента списку, який візуалізує

зображення, що відповідає вибраному елементу:

```
choice.getSelectionModel().selectedItemProperty().addListener((Obs
ervableValue<? extends String> ov, String old_val, String new_val) -> {
        if(new_val.equals("Україна")){
            imvR.setVisible(true);
            imvU.setVisible(false);
            imvG.setVisible(false);
        }
        if(new_val.equals("США")){
            imvU.setVisible(true);
            imvR.setVisible(false);
            imvG.setVisible(false);
        }
        if(new_val.equals("Великобританія")){
            imvG.setVisible(true);
            imvR.setVisible(false);
            imvU.setVisible(false);
        }
    });
```

9. Скачайте з інтернету рисунки із зображенням України, США та Великобританії, назвавши як *imageR.jpg*, *imageU.jpg* та *imageG.jpg*, і збережіть їх у папці, де розміщений файл JavaFXApplicationChoiceBox.java.

10. Створений список та вузли зображень додайте до кореневого вузла графа сцени:

```
root.getChildren().add(choice);
root.getChildren().add(imvR);
root.getChildren().add(imvU);
root.getChildren().add(imvG);
```

11. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті можна буде побачити список, що випадає (рис. 1), при виборі одного з елементів якого нижче з’являється зображення, що відповідає вибраному елементу.

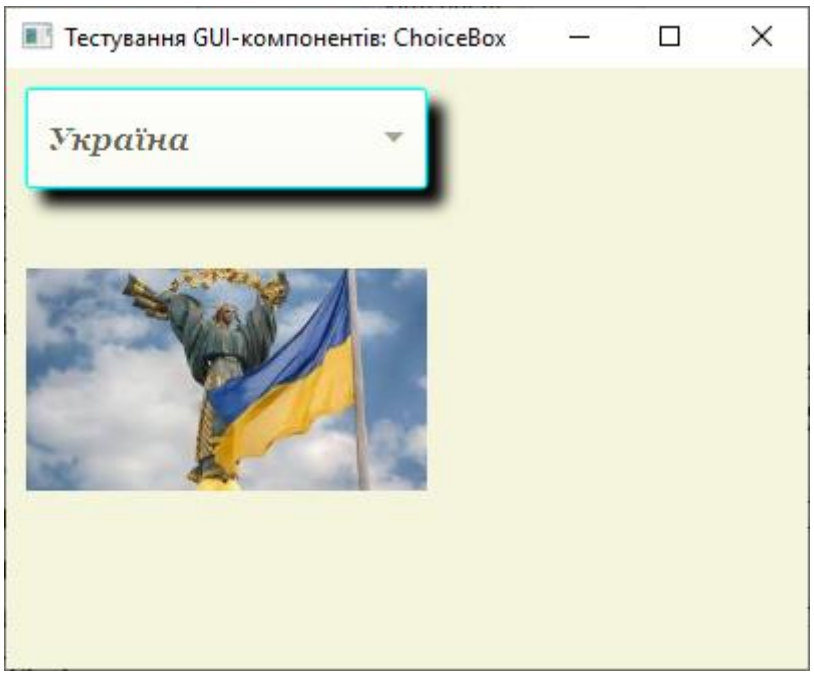

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить список, що випадає, ChoiceBox

Контрольні запитання

1. Яке призначення компоненту *ChoiceBox*?
2. Як створити екземпляр компоненту *ChoiceBox*?
3. Які основні властивості має компонент *ChoiceBox?*

Практична робота 9. GUI-компоненти: MenuBar, Menu

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять компоненти *MenuBar, Menu*.

Теоретичні відомості

Компонент MenuBar представлений класом javafx.scene.control.MenuBar, екземпляр якого може бути створений за допомогою класу-фабрики MenuBarBuilder або за допомогою конструктора:

MenuBar MenuBar = New MenuBar();

Клас MenuBar представляє панель меню і має такі властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered,

onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадкована від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.control.Control властивості: contextMenu, height, maxHeight, maxWidth, minHeight, minWidth, prefHeight, prefWidth, skin, tooltip, width.

Компонент MenuBar містить список javafx.collections.ObservableList<Menu> компонентів меню Menu, який можна поповнити за допомогою методу getMenus().addAll() класу javafx.scene.control.MenuBar.

Компоненти меню Menu, що містяться в панелі MenuBar, представлені класом javafx.scene.control.Menu, екземпляр якого може бути створений за допомогою класу-фабрики MenuBuilder або за допомогою одного з конструкторів:

Menu menu = new Menu("[текст]");

Menu menu = new Menu("[текст]", [вузол значка]);

Клас Menu має такі властивості:

– успадковані від класу javafx.scene.control.MenuItem властивості: accelerator, disabled, graphic, id, mnemonicParsing, onAction, parentMenu, parentPopup, style, text, visible;

– власні властивості: onHidden, onHiding, onShowing, onShown, showing.

Компоненти меню Menu панелі MenuBar також містять свої набори javafx.collections.ObservableList<MenuItem> компонентів MenuItem, які можна поповнити за допомогою методу getItems().addAll() класу javafx.scene.control.Menu.

Компоненти MenuItem представлені класом javafx.scene.control.MenuItem, екземпляр якого може бути створений за допомогою класу-фабрики MenuItemBuilder або за допомогою одного з конструкторів:

MenuItem menuItem = new MenuItem();

MenuItem menuItem = new MenuItem("[текст]");

MenuItem menuItem = new MenuItem("[текст]", [вузол значка]);

Клас MenuItem має властивості accelerator, disabled, graphic, id, mnemonicParsing, onAction, parentMenu, parentPopup, style, text і visible і підкласи CheckMenuItem, CustomMenuItem, Menu, RadioMenuItem. Тому до компоненту меню Menu панелі MenuBar можна додавати не

тільки елементи MenuItem, але й прапорці CheckMenuItem та перемикачі RadioMenuItem, а також роздільники SeparatorMenuItem, представлені класом SeparatorMenuItem, що розширює клас CustomMenuItem, та вкладені меню Menu.

Прапорець CheckMenuItem представлений класом javafx.scene.control.CheckMenuItem, екземпляр якого може бути створений за допомогою класу-фабрики CheckMenuItemBuilder або за допомогою одного з конструкторів:

CheckMenuItem checkMenuItem = New CheckMenuItem();

CheckMenuItem checkMenuItem = New CheckMenuItem("[текст]");

CheckMenuItem checkMenuItem = new CheckMenuItem("[текст]", [вузол значка]);

Клас CheckMenuItem має такі властивості:

– успадковані від класу javafx.scene.control.MenuItem властивості: accelerator, disabled, graphic, id, mnemonicParsing, onAction, parentMenu, parentPopup, style, text, visible;

– власну властивість selected.

Перемикач RadioMenuItem представлений класом javafx.scene.control.

RadioMenuItem, екземпляр якого може бути створений за допомогою класу-фабрики RadioMenuItemBuilder або за допомогою одного з конструкторів:

RadioMenuItem radioItem = new RadioMenuItem("[текст]");

RadioMenuItem radioItem = new RadioMenuItem("[текст]", [вузол значка]);

Клас RadioMenuItem має такі властивості:

– успадковані від класу javafx.scene.control.MenuItem властивості: accelerator, disabled, graphic, id, mnemonicParsing, onAction, parentMenu, parentPopup, style, text, visible;

– власні властивості selected та toggleGroup (дозволяє об'єднувати перемикачі у групу ToggleGroup).

Розділювач SeparatorMenuItem представлений класом javafx.scene.control.

SeparatorMenuItem, екземпляр якого може бути створений за допомогою класу-фабрики SeparatorMenuItemBuilder або за допомогою конструктора:

SeparatorMenuItem separatorMenuItem = new SeparatorMenuItem();

Клас SeparatorMenuItem має такі властивості:

– успадковані від класу javafx.scene.control.MenuItem властивості: accelerator, disabled, graphic, id, mnemonicParsing, onAction, parentMenu, parentPopup, style, text, visible;

– успадковані від класу javafx.scene.control.CustomMenuItem властивості: content та hideOnClick.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить панель *MenuBar*, в меню *Файл* оберіть *Створити проект* | *Java with Ant* | *JavaFX* | *JavaFX Application*, натисніть кнопку *Далі*, введіть ім'я проекту *JavaFXApplicationMenuBar*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з'явиться код згенерованого класу *JavaFXApplicationMenuBar*, який розширює клас Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationMenuBar 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. У методі *start* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об’єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: MenuBar” та зробіть видимим об’єкт Stage:

```
Group root = new Group();
Scene scene = new Scene(root, 450, 300, Color.BEIGE);
primaryStage.setScene(scene);
primaryStage.setTitle("Тестування GUI-компонентів: MenuBar, Menu");
primaryStage.show();
```

4. Створіть екземпляр панелі MenuBar. Панель меню помістіть у лівий верхній кут основного вікна JavaFX-додатка, і для неї визначте такі властивості, як режим накладання, курсор миші, візуальний ефект, стиль і бажані розміри:

```
MenuBar menuBar = new MenuBar();
menuBar.setLayoutX(10);
menuBar.setLayoutY(10);
menuBar.setBlendMode(BlendMode.HARD_LIGHT);
menuBar.setCursor(Cursor.CLOSED_HAND);
DropShadow effect = new DropShadow();
effect.setOffsetX(5);
effect.setOffsetY(5);
menuBar.setEffect(effect);
menuBar.setStyle("-fx-base:skyblue;-fx-border-width:4pt;-fx-border-color:navy;-fx-font:bold 16pt Georgia;");
menuBar.setPrefSize(350, 50);
```

5. Створіть перший екземпляр компонента меню Menu з текстом, для якого створюється набір елементів меню, що складається з компонента MenuItem, роздільника SeparatorMenuItem, перемикачів RadioMenuItem та прапорця CheckMenuItem. Для компонента MenuItem визначте текст, стиль, швидкі клавіші та обробник вибору. Перемикачі RadioMenuItem об'єднайте в групу, і для них встановіть

текст та стиль. Для прапорця CheckMenuItem визначте текст, стиль та вибір:

```
Menu menuF = new Menu("Файл");
MenuItem menuItemP = new MenuItem("Друк");
menuItemP.setStyle("-fx-text-fill:navy;-fx-font:bold italic 14pt Georgia;");
menuItemP.setAccelerator(KeyCombination.keyCombination("Ctrl+P"));
menuItemP.setOnAction((ActionEvent e) -> {
    System.out.println("Триває друк...");
});
SeparatorMenuItem sep = new SeparatorMenuItem();
RadioMenuItem radioItemY = new RadioMenuItem("З номерами сторінок");
radioItemY.setStyle("-fx-text-fill:navy;-fx-font:bold italic 12pt Georgia;");
ToggleGroup tgroup = new ToggleGroup();
radioItemY.setToggleGroup(tgroup);
radioItemY.setSelected(true);
RadioMenuItem radioItemN = new RadioMenuItem("Без номерів сторінок");
radioItemN.setStyle("-fx-text-fill:navy;-fx-font:bold italic 12pt Georgia;");
radioItemN.setToggleGroup(tgroup);
CheckMenuItem checkMenuItem = new CheckMenuItem("Покращена якість");
checkMenuItem.setSelected(true);
checkMenuItem.setStyle("-fx-text-fill:navy;-fx-font:bold italic 14pt Georgia;");
menuF.getItems().addAll(menuItemP, radioItemY, radioItemN, sep, checkMenuItem);
```

6. Створіть другий екземпляр компонента меню Menu з текстом, для якого формується набір елементів меню, що складається з компонентів MenuItem. Для компонентів MenuItem встановіть текст та стиль:

```
Menu menuE = new Menu("Правка");
MenuItem menuItemCut = new MenuItem("Вирізати");
menuItemCut.setStyle("-fx-text-fill:navy;-fx-font:bold italic 14pt Georgia;");
```

MenuItem menuItemCopy = new MenuItem("Копіювати");
menuItemCopy.setStyle("-fx-text-fill:navy;-fx-font:bold italic 14pt Georgia;");
MenuItem menuItemPaste = new MenuItem("Вставити");
menuItemPaste.setStyle("-fx-text-fill:navy;-fx-font:bold italic 14pt Georgia;");
menuE.getItems().addAll(menuItemCut, menuItemCopy, menuItemPaste);

7. Створіть третій примірник компонента меню Menu з текстом, для якого створюється набір елементів меню, що складається з компонента MenuItem. Для компонента MenuItem встановіть текст та стиль:

Menu menuV = new Menu("Вид");
MenuItem menuItemS = new MenuItem("Масштаб");
menuItemS.setStyle("-fx-text-fill:navy;-fx-font:bold italic 14pt Georgia;");
menuV.getItems().addAll(menuItemS);

8. Створені екземпляри компонента меню Menu додайте в панель MenuBar, яка в свою чергу додається до кореневого вузла графа сцени:

menuBar.getMenus().addAll(menuF, menuE, menuV);
root.getChildren().add(menuBar);

9. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті, можна буде побачити панель меню, при виборі одного з елементів якої з'являється відповідне підменю (рис. 1).

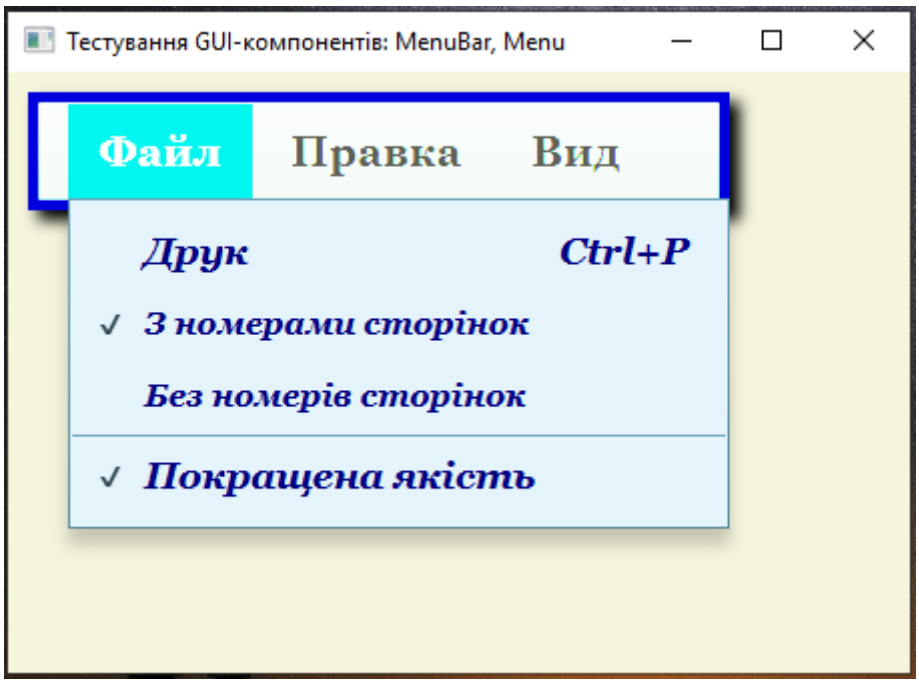

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить панель меню

Контрольні запитання

1. Яке призначення компонентів *MenuBar*, *Menu*?
2. Як створити екземпляри компонентів *MenuBar*, *Menu*?
3. Які основні властивості мають компоненти *MenuBar*, *Menu?*

Практична робота 10. GUI-компонент PieChart

Мета: набуття вмінь та навичок створення JavaFX-додатків, які містять кругову діаграму *PieChart*.

Теоретичні відомості

Платформа JavaFX 2.0 забезпечує створення двовимірних діаграм за допомогою базового класу javafx.scene.chart.Chart та його підкласів PieChart, AreaChart, BarChart, BubbleChart, LineChart та ScatterChart.

Клас Chart має такі властивості:

– успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased,

opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

– успадковану від класу javafx.scene.Parent властивість needsLayout;

– успадковані від класу javafx.scene.layout.Region властивості: height, maxHeight, maxWidth, minHeight, minWidth, padding, prefHeight, prefWidth, snapToPixel, width;

– власні властивості: animated, legend, legendSide, legendVisible, title, titleSide.

За допомогою властивості title встановлюється заголовок діаграми, а за допомогою властивості titleSide – розташування заголовку вгорі, внизу, праворуч або ліворуч самої діаграми.

Властивості legendSide та legendVisible визначають розташування та видимість панелі пояснень до діаграми.

Якщо властивість animated встановити зі значенням true, тоді діаграма динамічно реагуватиме на зміни своїх властивостей.

У той час як клас PieChart безпосередньо розширює клас Chart, класи AreaChart, BarChart, BubbleChart, LineChart та ScatterChart є підкласами класу javafx.scene.chart.XYChart<X,Y>, що розширює клас Chart.

Клас XYChart є базовим класом для класів, що представляють діаграми з двома осями, і має, крім успадкованих від класу Chart властивостей, власні властивості: data, verticalGridLinesVisible, horizontalGridLinesVisible, alternativeColumnFillVisible, alternativeRowFillVisible, verticalZeroLineVisible, horizontalZeroLineVisible.

Властивості alternativeColumnFillVisible і alternativeRowFillVisible визначають виділення через один у сітці діаграми стовпців і рядків, якщо обидві властивості встановлені зі значенням true, сітка діаграми схожа на шахівницю.

Властивості verticalGridLinesVisible та horizontalGridLinesVisible визначають відображення вертикальних та горизонтальних ліній сітки діаграми, а властивості verticalZeroLineVisible та horizontalZeroLineVisible – відображення додаткових вертикальних та горизонтальних ліній нульових позначок за умови відображення вертикальних та горизонтальних ліній сітки діаграми.

Властивість data визначає набір javafx.collections.ObservableList<XYChart.Series<X,Y>> даних, з яких формується сама діаграма, розмітка осей діаграми, набір підписів до

міток осей діаграми та вміст легенди. Поповнити набір ObservableList<XYChart.Series<X,Y>> даних діаграми можна за допомогою методу getData().addAll() класу javafx.scene.chart.XYChart<X,Y>.

Дані діаграми XYChart є класом javafx.scene.chart.XYChart.Series<X,Y>, екземпляр якого можна створити за допомогою одного з конструкторів:

XYChart.Series series= new XYChart.Series();

XYChart.Series series = new XYChart.Series(ObservableList<XYChart.Data<X,Y>> data);

XYChart.Series series = new XYChart.Series(java.lang.String name, ObservableList<XYChart.Data<X,Y>> data);

Клас XYChart.Series представляє серію даних діаграми XYChart і має властивості: chart, data, name, node.

Властивість name визначає назву серії, що відображається у легенді до діаграми.

Після того, як серія додається в набір даних діаграми XYChart, формується вузол Node, який відповідає за відображення серії, і за допомогою властивості node можна отримати доступ до властивостей цього вузла.

Властивість data визначає набір javafx.collections.ObservableList<XYChart.

Data<X,Y>> даних серії XYChart.Series, поповнити який можна за допомогою методу getData().addAll() класу javafx.scene.chart.XYChart.Series<X,Y>.

Дані серії XYChart.Series представлені класом javafx.scene.chart.XYChart.Data<X,Y>, екземпляр якого можна створити за допомогою одного з конструкторів:

XYChart.Data data = new XYChart.Data();

XYChart.Data data = new XYChart.Data (X xValue, Y yValue);

XYChart.Data data= new XYChart.Data(X xValue, Y yValue,

java.lang.Object extraValue);

Клас XYChart.Data представляє дані серії XYChart.Series та має властивості: extraValue, node, xValue, yValue.

Властивості xValue та yValue визначають значення елемента даних по осі x та y, властивість extraValue — додаткове значення, наприклад радіус для бульбашкової діаграми.

Після того, як елемент даних додається в набір даних діаграми XYChart, формується вузол Node, який відповідає за відображення

елемента даних, і за допомогою властивості node можна отримати доступ до властивостей цього вузла. До додавання елемента даних до набору даних діаграми XYChart можна визначити свій вузол Node, який відповідає за відображення елемента даних.

Осі діаграми XYChart представлені базовим класом javafx.scene.chart.Axis<T> та його підкласами CategoryAxis та NumberAxis.

Клас Axis має наступні властивості:

 успадковані від класу javafx.scene.Node властивості: blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, managed, mouseTransparent, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragged, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible;

успадкована від класу javafx.scene.Parent властивість needsLayout;

успадковані від класу javafx.scene.layout.Region властивості: height, maxHeight, maxWidth, minHeight, minWidth, padding, prefHeight, prefWidth, snapToPixel, width;

власні властивості: animated, autoRanging, label, side, tickLabelFill, tickLabelFont, tickLabelGap, tickLabelRotation, tickLabelsVisible, tickLength, tickMarkVisible.

Властивість label визначає підпис осі, а властивість side – сторону відображення осі.

За допомогою характеристики autoRanging встановлюється автоматичний вибір діапазону значень осі на основі набору даних діаграми.

Якщо властивість animated встановити рівним true, тоді вісь динамічно реагуватиме на зміни своїх властивостей.

Властивості tickLabelFill, tickLabelFont, tickLabelGap, tickLabelRotation, tickLabelsVisible, tickLength та tickMarkVisible визначають колір підписів до міток осі, шрифт підписів до міток осі, інтервал між лініями міток на осі та підписами до міток осі, поворот підписів до міток осі, відображення підписів міток осі, довжину ліній

міток на осі та відображення міток осі.

Клас CategoryAxis є вісь, що відображає дискретні рядкові значення окремих категорій. Примірник класу CategoryAxis може бути створений за допомогою класу-фабрики CategoryAxisBuilder або за допомогою одного з конструкторів:

CategoryAxis axis = new CategoryAxis();

CategoryAxis axis = new CategoryAxis(ObservableList<java.lang.String> categories);

Набір рядкових значень ObservableList<java.lang.String> осі CategoryAxis може бути створений за допомогою статичного методу observableArrayList() класу javafx.collections.FXCollections.

Клас CategoryAxis має, окрім успадкованих від класу Axis<T> властивостей, власні властивості category початком осі та першою міткою.

Клас NumberAxis представляє числову вісь діаграми XYChart.

Примірник класу NumberAxis може бути створений за допомогою класу-фабрики NumberAxisBuilder або за допомогою одного з конструкторів:

NumberAxis = NumberAxis();

NumberAxis = NumberAxis(double lowerBound, double upperBound, double tickUnit);

NumberAxis axis = new NumberAxis(java.lang.String axisLabel, double lowerBound, double upperBound, double tickUnit);

Клас NumberAxis має, крім успадкованих від класу Axis<T> властивостей, такі властивості:

– успадковані від класу javafx.scene.chart.ValueAxis<T> властивості: lowerBound, minorTickCount, minorTickLength, minorTickVisible, scale, tickLabelFormatter, upperBound;

– власні властивості forceZeroInRange та tickUnit.

Властивості lowerBound, minorTickCount, minorTickLength, minorTickVisible, tickLabelFormatter та upperBound дозволяють визначити мінімальне значення осі, кількість допоміжних міток, довжину допоміжних міток, відображення допоміжних міток, форматування підписів до міток осі та максимальне

Властивості forceZeroInRange та tickUnit визначають включення нульової мітки у видимий діапазон при його автоналаштуванні та інтервал між головними мітками осі.

Кругова діаграма PieChart

Діаграма PieChart представлена класом

javafx.scene.chart.PieChart, екземпляр якого можна створити за допомогою класу-фабрики PieChartBuilder або за допомогою одного з конструкторів:

PieChart chart = new PieChart();

PieChart chart = new PieChart(ObservableList<PieChart.Data> data);

Клас PieChart представляє кругову (секторну) діаграму і має, крім успадкованих від класу Chart властивостей, власні властивості: clockwise, data, labelLineLength, labelsVisible, startAngle.

Властивість data визначає набір javafx.collections.ObservableList<PieChart.Data> даних, з яких формуються сектори діаграми, підписи до них та вміст панелі пояснень до діаграми.

Набір даних ObservableList<PieChart.Data> діаграми PieChart може бути створений за допомогою статичного методу observableArrayList() класу javafx.collections.FXCollections. При цьому дані діаграми PieChart є класом javafx.scene.chart.PieChart.Data, екземпляр якого може бути створений за допомогою конструктора public PieChart.Data(java.lang.String name, double value), де name – це підпис до сектора діаграми, а value – частка сектора діаграми.

Отримати доступ до властивостей вузла Node, що є сектором PieChart.Data, дозволяє метод getNode() класу PieChart.Data, а до властивостей діаграми PieChart, до якої належить сектор – метод getChart() класу PieChart.Data.

За допомогою властивості clockwise класу PieChart встановлюється розташування секторів діаграми за годинниковою стрілкою, а за допомогою властивості startAngle – кут початку першого сектора діаграми.

Властивості labelsVisible і labelLineLength визначають відображення підписів до секторів діаграми та довжину лінії від сектора діаграми до підпису до сектора.

Завдання

1. Відкрийте середовище *NetBeans* з підтримкою платформи JavaFX 2.0. Для створення JavaFX-додатку з GUI-інтерфейсом, що містить кругову діаграму *PieChart*, в меню *Файл* оберіть *Створити проект | Java with Ant | JavaFX | JavaFX Application*, натисніть кнопку *Далі*, введіть ім’я проекту *JavaFXApplicationPieChart*, залишивши прапорець *Create Main Class*, та натисніть кнопку *Готово*.

У вікні редактора середовища NetBeans з’явиться код згенерованого класу *JavaFXApplicationPieChart*, який розширює клас

Application.

```
package javafxapplicationbutton;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class JavaFXApplicationPieChart 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();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

2. У методі *start()* вилучіть весь код.

3. Створіть кореневий вузол графа сцени та на його основі екземпляр сцени, який встановлюється для об’єкта Stage. Установіть заголовок основного вікна JavaFX-додатку “Тестування GUI-компонентів: PieChart” та зробіть видимим об’єкт Stage:

```
Group root = new Group();
Scene scene = new Scene(root, 600, 500, Color.BEIGE);
primaryStage.setScene(scene);
```

```
primaryStage.setTitle("Тестування GUI-компонентів: PieChart") ;
primaryStage.show();
```

4. Створіть кругову діаграму PieChart. Діаграму помістіть у лівий верхній кут, і для неї встановіть набір даних і такі властивості, як курсор миші, стиль, переважні розміри, динамічне реагування на зміну властивостей, заголовок та його розташування, відображення та розташування панелі пояснень до діаграми, розташування секторів за годинною стрілки, відображення підписів і довжина лінії до підпису, кут початку першого сектора, обробник клацання кнопкою миші на секторі діаграми, який відображає значення сектора, та обробник перетягування мишею, що повертає діаграму навколо своєї осі. Такий поворот відображення діаграми навколо своєї осі можливий, тому що властивість animated діаграми встановлено рівним true:

```
final ObservableList<PieChart.Data> pieChartData =
        FXCollections.observableArrayList(new
PieChart.Data("Долар США", 45.9),
        new PieChart.Data("Євро", 42.5),
        new PieChart.Data("Японська єна", 1.6),
        new PieChart.Data("Фунт стерлінгів", 9.2),
        new PieChart.Data("Канадський долар", 0.8));
        final PieChart chart = new PieChart(pieChartData);
        chart.setLayoutX(50);
        chart.setLayoutY(10);
        chart.setCursor(Cursor.CROSSHAIR);
        chart.setStyle("-fx-font:bold 14 Arial; -fx-text-fill:brown;");
        chart.setPrefSize(500, 400);
        chart.setAnimated(true);
        chart.setTitle("Розподіл валютних активів\n Ощадбанку по
валюті у 2022 р.");
        chart.setTitleSide(Side.TOP);
        chart.setLegendVisible(true);
        chart.setLegendSide(Side.BOTTOM);
        chart.setClockwise(true);
        chart.setLabelsVisible(true);
        chart.setLabelLineLength(20);
        chart.setStartAngle(150);
        final Popup popup = new Popup();
        popup.setAutoHide(true);
        final Label label = new Label("");
```

```
            label.setStyle("-fx-font: bold 20 Arial;-fx-text-fill:brown");
            popup.getContent().addAll(label);
            chart.getData().forEach(data -> {
      data.getNode().addEventHandler(MouseEvent.MOUSE_PRESSED,
(MouseEvent e) -> {
                  label.setText(String.valueOf(data.getPieValue())     +
"%");
                  popup.setX(e.getScreenX());
                  popup.setY(e.getScreenY());
                  popup.show(primaryStage);
               });
            });
            chart.addEventHandler(MouseEvent.DRAG_DETECTED,
(MouseEvent e) -> {
               chart.setStartAngle(chart.getStartAngle()+2);
            });
```

5. Додайте кругову діаграму в кореневий вузол графа сцени:

```
root.getChildren().add(chart);
```

6. Запустіть створений JavaFX-додаток, клацнувши правою кнопкою миші на піктограмі *Очистити та побудувати проект* (Shift+F11), а потім – *Виконати проект* (F6).

У результаті можна буде побачити кругову діаграму (рис. 1), у якій відображено розподіл валютних активів Ощадбанку по валюті у 2022 році. При натисканні лівою кнопкою миші на секторі з’являється числове значення у валюті.

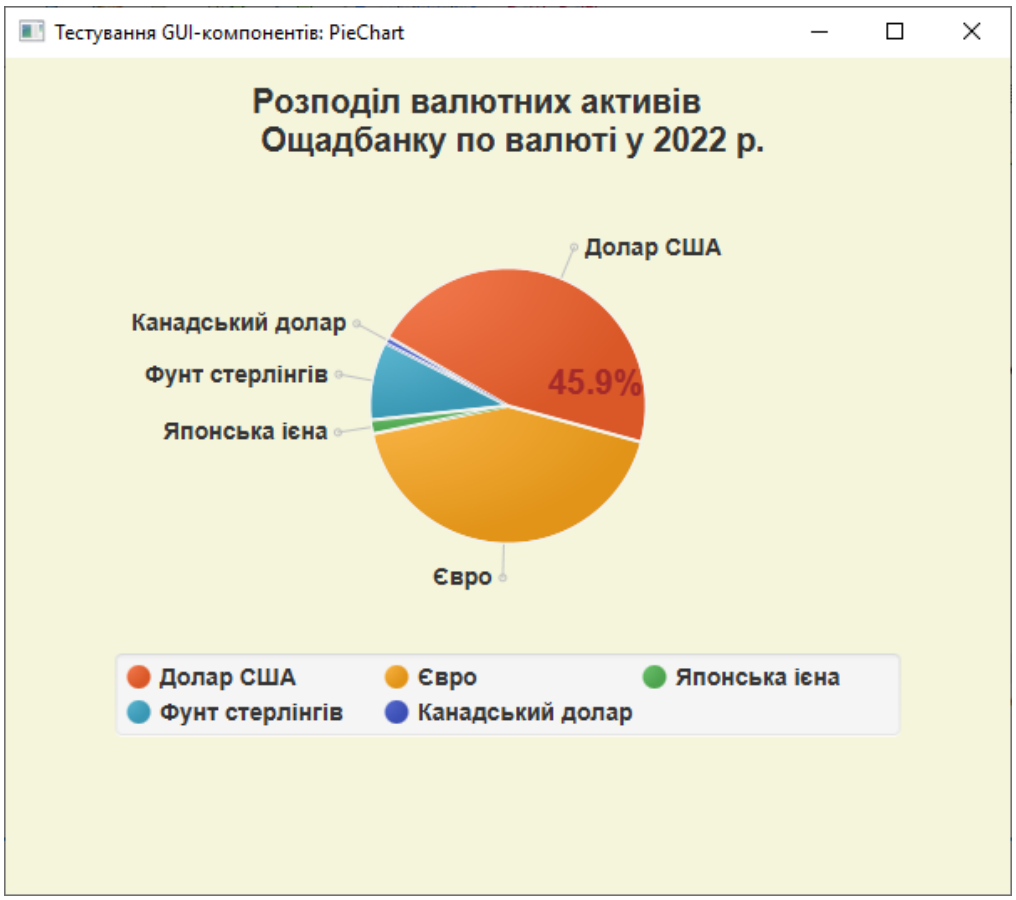

Рис. 1. JavaFX-додаток з GUI-інтерфейсом, що містить кругову діаграму

Контрольні запитання

1. Яке призначення компоненту *PieChart*?
2. Як створити екземпляр компоненту *PieChart*?
3. Які основні властивості має компонент *PieChart?*

Список використаних джерел

1. Машнин Т. С. JavaFX 2.0: разработка RIA-приложений. СПб.: БХВ-Петербург, 2012. 320 с.
2. Муляр В. П. Основи розробки додатків з використанням технології JavaFX. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2018. Вип. № 30-31. С. 104–110.
3. Муляр В. П. Розробка JavaFX-додатків із використанням Scene Builder. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2020. Вип. № 39. С. 181–189.
4. Муляр В. П., Яцюк С. М. Елементи комп'ютерної графіки у візуалізації результатів моделювання фізичних явищ і процесів. Комп'ютерно-орієнтовані технології: освіта, наука, виробництво. 2016. № 23. С. 80–84.
5. Java Tutorial. URL: https://www.w3schools.com/java/default.asp
6. Java. Классы. Объектно-ориентированное программирование. URL: https://metanit.com/java/tutorial/3.1.php
7. Підручник з Java. URL: https://www.javatpoint.com/java-tutorial
8. GDB online Debugger / Compiler. URL: https://www.onlinegdb.com/
9. Java – Учебники по программированию. URL: https://betacode.net/
10. Apache NetBeans. URL: https://netbeans.apache.org/download/index.html
11. Java Course. URL: http://java-course.ru/begin/introduce/
12. Java SE Downloads. URL: https://www.oracle.com/java/technologies/javase-downloads.html
13. JavaFX. URL: https://gluonhq.com/products/javafx/
14. Scene Builder. URL: https://gluonhq.com/products/scene-builder/
15. Введение в Java FX. URL: https://metanit.com/java/javafx/1.1.php
16. Руководство JavaFX для начинающих – Hello JavaFX. URL: https://betacode.net/10623/javafx-tutorial-for-beginners

Для нотаток

Навчальне видання

Муляр Вадим Петрович

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

Практикум