УДК 004.415.2
Муляр В. П., к. пед. н., доцент
Східноєвропейський національний університет імені Лесі Українки

ОСНОВИ РОЗРОБКИ ДОДАТКІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ JAVAFX

Муляр В. П. Основи розробки додатків з використанням технології JavaFX. У статті розглянуто особливості розробки графічного інтерфейсу користувача з використанням технології JavaFX. Розкрито структуру JavaFX-додатків. Розглянуто використання лямбда-виразів для обробки подій. Висвітлено питання компонування елементів керування графічного інтерфейсу з використанням мови розмітки FXML та середовища SceneBuilder. Розглянуто особливості зміни зовнішнього вигляду інтерфейсу користувача за допомогою таблиць стилів CSS. Особливу увагу приділено анімації і трансформації зображень в JavaFX. Доведено високу ефективність технології JavaFX для розробки додатків із насиченим графічним інтерфейсом, які розгортаються на різних платформах.
Ключові слова: JavaFX, графічний інтерфейс користувача, компонування, мова розмітки FXML, таблиці стилів CSS, анімація, трансформація.

Муляр В. П. Основы разработки приложений с использованием технологии JavaFX. В статье рассмотрены особенности разработки графического интерфейса пользователя с использованием технологии JavaFX. Раскрыта структура JavaFX-приложений. Рассмотрено использование лямбда-выражений для обработки событий. Освещены вопросы компоновки элементов управления графического интерфейса с использованием языка разметки FXML и среды Scene Builder. Рассмотрены особенности изменения внешнего вида интерфейса с помощью таблиц стилей CSS. Особое внимание уделено анимации и трансформации изображений в JavaFX. Доказана высокая эффективность технологии JavaFX для разработки приложений с насыщенным графическим интерфейсом, которые разворачиваются на разных платформах.
Ключевые слова: JavaFX, графический интерфейс пользователя, компоновки, язык разметки FXML, таблицы стилей CSS, анимация, трансформация.

Muliar V. P. Fundamentals of development of applications with use JavaFX technology. The article discusses the features of developing a graphical user interface using JavaFX technology. The structure of JavaFX applications is revealed. The use of lambda expressions for processing events is considered. Issues related to the layout of GUIs using the FXML markup language and the Scene Builder environment. The features of changing the appearance of the interface using CSS style sheets are considered. Particular attention is paid to animation and image transformation in JavaFX. The high efficiency of JavaFX technology is proven for developing rich graphical user-friendly applications, which unfold on different platforms.
Keywords: JavaFX, graphical user interface, layouts, FXML markup language, CSS style sheets, animation, transformation.

Постановка проблеми в загальному вигляді. Потужним інструментом для розробки настільних і мережевих додатків є технологія JavaFX, що дозволяє розробникам проектувати, 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.

Починаючи з JavaFX 2.2, всі пізніші версії повністю інтегровані з Java SE Runtime Environment (JRE) і комплектом Java Development Kit (JDK). Оскільки JDK доступний для всіх основних настільних платформ (Windows, Mac OS X і Linux), то JavaFX-додатки, скомпільовані в JDK, можуть працювати на даних платформах. Крос-платформна сумісність дає змогу використовувати досвід узгодженого виконання додатків для JavaFX-розробників і користувачів. 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 [2, с. 386].

Платформа JavaFX забезпечує графічні й анімаційні можливості для додатків, які дозволяють створювати і відображати зображення за допомогою програмного забезпечення і комп'ютерного устаткування. Компоненти графічного інтерфейсу користувача (Graphical User Interface, GUI) JavaFX-додатка утворюють сцену, логічна структура якої описується графом сцени. Відображенням GUI-інтерфейсу JavaFX-додатка є графічне зображення графа сцени. Для відображення GUI-інтерфейсу, розробленого на основі платформи JavaFX, і створення графічного зображення графа сцени, середовище виконання JavaFX Runtime представляє графічну систему, яка містить відповідні модулі і створює набір паралельних потоків. Основний потік JavaFX-додатка JavaFX Application Thread відповідає за оновлення сцени, обробку анімації і подій. Потік рендеринга системи Prism відповідає за відрисовку сцени, а фоновий медіапотік відповідає за декодування, буферизацію і відтворення аудіо та відео. Синхронізація графа сцени з його графічним представленням здійснюється за допомогою подій Pulse, які генеруються середовищем виконання JavaFX Runtime з максимальною частотою 1/60 секунди і посилаються в чергу подій

під час анімації і кожен раз, коли змінюється граф сцени, викликаючи перерисовку сцени з використанням компонування і стилів [3, с. 14].

Аналіз досліджень і публікацій. Розробці RIA-додатків (Rich Internet Applications) із використанням технології JavaFX присвячено дослідження Ю. Парфенова та В. Федорченко [4]. Розгляду архітектури платформи JavaFX 2.0, її основним компонентам графічного інтерфейсу користувача, використанню CSS-стилів, створенню візуальних ефектів, трансформації й анімації зображень, використанню компонентів JavaFX NetBeanes, мові FXML присвячена робота Т. Машніна [3]. Аналізу технологій розробки насичених інтернет-додатків на платформі Java присвячено дослідження В. Герасимова та В. Левицької [1]. Особливості побудови графічного контенту додатків із використанням JavaFX і Swing компонентів і даних, взятих із баз даних, розкрито в дослідженні В. Карашецького [2]. Більшість дослідників вважають, що у Java-розробників з'явився потужний конкурентоспроможний інструмент – JavaFX. Технологія має багато різних реалізованих елементів інтерфейсу та класів, які дуже прискорюють розробку RIA-додатків.

Формулювання цілей статті. Мета статті – розглянути основи розробки графічного інтерфейсу користувача засобами JavaFX.

Виклад основного матеріалу. JavaFX – це базовий набір інструментів для розробки інтерфейсу користувача, єдине узгоджене середовище програмування додатків як для вбудованих, так і для настільних систем [10]. Іншими словами, JavaFX має на меті використання в багатьох типах пристроїв, таких як мобільні пристрої, смартфони, телевізори, планшетні комп'ютери та десктопи [6]. Для компіляції і запуску JavaFX-програм більше немає необхідності у встановленні додаткових програм. Усі частини одного проекту тепер можна реалізувати на одній платформі Java, що, в свою чергу, забезпечує швидкодію, підвищує безпеку, скорочує час розробки і економить витрати на розробку.

Розглянемо основні можливості JavaFX як засобу розробки графічного інтерфейсу користувача.

Структура додатків JavaFX. Загалом, JavaFX-додаток має три основні компоненти, а саме: підмостки, сцену та вузли [8]. Усе, що потрібно відобразити засобами JavaFX, розміщується на сцені, яку можна оформити і оживити «акторами», в ролі яких виступають елементи керування і форми. Сцена повинна знаходитися на підмостках, які є вікном верхнього рівня, якщо програма виконується на робочому столі операційної системи, або прямокутною областю, якщо програма виконується в вигляді аплета. Підмостки передаються у вигляді параметра методу start(), який необхідно перевизначити в підкласі, похідному від класу Application, як показано в наведеному нижче прикладі.

```
public class HelloWorld extends Application {
public void start(Stage stage) {
Label message = new Label("Hello, JavaFX!");
message.setFont(new Font(100));
stage.setScene(new Scene(message));
stage.setTitle("Hello");
stage.show();
}
}
public class MyApp extends Application {
public static void main(String[] args) {
launch(args);
}
...
}
```

Обробка подій. Керування графічними інтерфейсами здійснюється за допомогою подій. Коли користувач вибирає кнопки клацанням на них, регулює повзунки і виконують інші дії, графічний інтерфейс користувача реагує і оновлюється. Окремий елемент управління (наприклад, кнопка) забезпечується обробником подій, щоб отримувати повідомлення про вибір кнопки клацанням на ній. Все це істотно спрощується завдяки використанню лямбда-виразів, як показано в наведеному нижче прикладі коду.

```
red.setOnAction(event -> message.setTextFill(Color.RED));
```

При виборі кнопки клацанням на ній викликається лямбда-вираз. У даному випадку задається червоний колір тексту.

Властивості JavaFX. Властивість є атрибутом класу, значення якого можна читати або записувати. Як правило, властивість підтримується полем, а метод отримання і установки просто читає і записує дані у властивість відповідно. Властивість в JavaFX, крім методів отримання і установки, забезпечується третім методом, що повертає об'єкт класу, що реалізовує інтерфейс Property. Наприклад, у властивості text в JavaFX є метод Property <String> textProperty(). До об'єкту властивості можна приєднати приймач подій. Цим JavaFX відрізняється від колишньої технології JavaBeans. У JavaFX об'єкт властивості, а не компонент JavaBeans, посилає повідомлення про зміни. І для таких змін є вагомі підстави. Для реалізації прив'язаних властивостей в JavaBeans був потрібний шаблонний код, який виконував введення, видалення і запуск приймачів подій. А в JavaFX все зроблено набагато простіше, оскільки все навантаження беруть на себе бібліотечні класи [5, с. 86].

Прив'язка. Головне призначення властивостей в JavaFX полягає в повідомленні про прив'язку – автоматичному оновленні однієї властивості при зміні іншої. Механізм прив'язки дозволяє подолати труднощі, які виникають під час програмування інтерфейсу користувача. Як приклад розглянемо поле дати і селектор дати з календаря. Коли користувач вибирає дату з календаря, поле дати має бути оновлено автоматично, а разом із ним і властивість дати в моделі.

Компонування. Якщо графічний інтерфейс користувача містить кілька елементів управління, вони повинні бути розташовані на екрані як з точки зору виконуваних ними функцій, так і з точки зору зручності застосування. Компонування елементів управління графічного інтерфейсу користувача можна здійснити за допомогою спеціального інструментального засобу, який призначений для розробки інтерфейсу. З іншого боку, компоновка може бути досягнута програмним способом, тобто написанням коду в методі установки, що вводить елементи управління в призначений для користувача інтерфейс на окремих позиціях. Ще один спосіб полягає в тому, щоб визначити компонування на декларативній, непроцедурній мові. Наприклад, веб-сторінки компонуються засобами HTML і CSS. Аналогічно на платформі Android є окрема мова XML для визначення компоновок. У JavaFX підтримуються всі три згаданих вище способи. Зокрема, Scene Builder виконує в JavaFX роль візуального конструктора графічного інтерфейсу користувача [9].

Мова розмітки FXML. Для опису компонувань в JavaFX застосовується мова розмітки FXML. Файли FXML-документів можна створити вручну або ж скористатися для цієї мети інструментальним засобом Scene Builder. Отримавши такий файл, його можна завантажити в такий спосіб:

```
public void start(Stage stage) {
try {
Parent root = FXMLLoader.load(getClass().getResource("dialog.fxml"));
stage.setScene(new Scene(root));
stage.show();
} catch (IOException ex) {
ex.printStackTrace();
System.exit(0);
}
}
```

Установити зв'язок між елементами управління інтерфейсу користувача і програмою можна, зокрема, скориставшись таким способом. Зокрема, анотацію @FXML можна використовувати для «впровадження» керуючих об'єктів у клас контролера. А в класі контролера повинен бути реалізований інтерфейс Initializable. У методі initialize() з класу контролера засоби прив'язки зв'язуються з обробниками подій. Як контролер можна використовувати будь-який клас – навіть сам JavaFX-додаток.

У файлі FXML-документа слід надати імена змінних екземпляра контролера для відповідних елементів управління в цьому файлі. Для цієї мети слугує атрибут fx: id. У кореневому елементі необхідно також оголосити клас контролера за допомогою атрибуту fx: controller. При завантаженні файлу FXML-документа конструюється граф сцени, а посилання на іменовані об'єкти управління впроваджуються в анотовані поля об'єкта контролера. Потім викликається його метод initialize(). Більшу частину ініціалізації можна зробити в файлі FXML-документа. Зокрема, можна визначити прості прив'язки і задати анотовані методи контролера в якості

приймачів подій. Такий підхід дозволяє відокремити візуальний дизайн від поведінки програми. При цьому розробник призначеного для користувача інтерфейсу може займатися графічним оформленням програми, а програміст – реалізацією її поведінки.

Таблиці стилів CSS. У JavaFX допускається зміна зовнішнього вигляду інтерфейсу користувача за допомогою таблиць стилів CSS, що, як правило, зручніше, ніж надання атрибутів FXML-розмітки або виклик методів в Java. Таблицю стилів CSS можна завантажити програмно і застосувати її до графу сцени наступним чином:

```
Scene scene = new Scene(pane);
scene.getStylesheets().add("scene.css");
```

У таблиці стилів можна звертатися до будь-яких елементів управління, які мають ідентифікатор. Як приклад розглянемо, як організувати управління зовнішнім виглядом панелі типу GridPane. Ідентифікатор цієї панелі встановлюється в коді наступним чином:

```
GridPane pane = new GridPane();
pane.setId("pane");
```

А будь-яке заповнення проміжків або розстановка встановлюється не в коді, а в таблиці CSS, як показано нижче.

```
#pane {
-fx-padding: 0.5em;
-fx-hgap: 0.5em;
-fx-vgap: 0.5em;
-fx-background-image: url("metal.jpg")
}
```

На жаль, замість відомих атрибутів CSS доведеться освоїти і застосовувати спеціальні атрибути JavaFX, імена яких починаються із префікса -fx- і утворюються з імен властивостей в нижньому регістрі і дефісів замість змішаного написання.

Замість стилізації за окремими ідентифікаторами можна скористатися класами стилів. Із цією метою клас стилю вводиться в вузловий об'єкт наступним чином:

```
HBox buttons = new HBox();
buttons.getStyleClass().add("buttonrow");
```

Потім він стилізується за допомогою наступного позначення класу CSS:

```
.buttonrow {
-fx-spacing: 0.5em;
}
```

Крім того, таблиці стилів CSS можна застосовувати в FXML-компоновках. Для цього досить приєднати таблицю стилів до кореневої панелі наступним чином:

```
<GridPane id = "pane" stylesheets = "scene.css">
```

Код FXML-розмітки забезпечується атрибутами id або styleClass, як показано в наведеному нижче прикладі.

```
<HBox styleClass = "buttonrow">
```

Потім більшу частину стилізації можна вказати в таблиці стилів CSS, а FXML-розмітку використовувати тільки для перегляду [5, с. 102].

Анімація і трансформація. Основною ідеєю анімації в JavaFX є зміна властивості вузла протягом певного періоду часу. Якщо ця змінна визначає розташування вузла, результуюча анімація – це рух. Якщо з часом змінити властивості заповнення форми, ілюзія анімації змінює свою перспективу. Отже, анімація не завжди означає певну форму руху. Це може бути зміна кольору, прозорості тощо.

Платформа JavaFX забезпечує створення двох видів анімації – анімацію по ключовим кадрам і анімацію з вбудованою тимчасової шкалою. JavaFX-анімацію представляє пакет javafx.animation, базовим класом якого є клас Animation. Клас Animation розширюється класами Timeline і Transition, при цьому клас Timeline представляє анімацію по ключовим кадрам, а клас Transition – анімацію з вбудованою тимчасової шкалою.

Клас Animation має набір властивостей, що дозволяють управляти швидкістю і напрямком анімації, затримкою і кількістю циклів анімації, встановлювати автореверс анімації, зчитувати статус анімації, обробляти завершення анімації та ін.

Швидкість і напрям анімації можна встановити за допомогою метода setRate(double value), затримку анімації – за допомогою методу setDelay(Duration value), зчитати статус Animation.Status.PAUSED, Animation.Status.RUNNING або Animation.Status.STOPPED анімації –

методом getStatus(), встановити обробник завершення анімації – методом setOnFinished(EventHandler<ActionEvent>value). Також клас Animation містить методи управління життєвим циклом анімації:

jumpTo(Duration time) – перехід анімації до вказаної позиції на часовій шкалі;
playFrom(Duration time) – запуск анімації, починаючи з вказаної позиції на часовій шкалі;
play() – запуск анімації з поточної позиції на часовій шкалі;
playFromStart() – запуск анімації з початкової позиції на часовій шкалі;
stop() – зупинка анімації;
pause() – пауза анімації [3, с. 247].

Анімація по ключовим кадрам дозволяє створити видимі зміни значення будь-якої JavaFX-властивості за певний проміжок часу за допомогою класу Timeline. Примірник класу Timeline можна створити за допомогою класу-фабрики TimelineBuilder або за допомогою одного з конструкторів, що дозволяють встановити частоту кадрів і набір ключових кадрів анімації:

public Timeline(double targetFramerate)
public Timeline(double targetFramerate, KeyFrame ... keyFrames)
public Timeline(KeyFrame ... keyFrames) і public Timeline()

Набір ключових кадрів Timeline-анімації можна поповнити методом getKeyFrames(). AddAll(), а зупинити Timeline-анімацію і повернути її до початкової позиції – методом stop(). Ключовий кадр Timeline-анімації представлений класом javafx.animation.KeyFrame і визначає зміну значень JavaFX-властивостей за певний проміжок часу. Примірник класу KeyFrame можна створити за допомогою набору конструкторів, що дозволяють встановити час відтворення ключового кадру, ім'я ключового кадру, обробник закінчення ключового кадру і набір зміни значень відповідних JavaFX-властивостей:

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

Зміна значення JavaFX-властивості представлена класом javafx.animation. KeyValue, екземпляр якого можна створити за допомогою конструкторів, дозволяє встановити змінювану JavaFX-властивість, її кінцеве значення в результаті анімації і спосіб її зміни протягом анімації:

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

Спосіб зміни значення JavaFX-властивості протягом анімації представлений класом javafx.animation.Interpolator.

Transition-анімація з вбудованою часовою шкалою також використовує об'єкт Interpolator як значення властивості interpolator класу Transition.

Transition-анімація з вбудованою часовою шкалою, на відміну від Timeline-анімації, описує зміну в часі обмеженого набору JavaFX-властивостей, таких як прозорість (клас FadeTransition), положення в просторі (класи PathTransition і TranslateTransition), обертання (клас RotateTransition) і масштабування вузла графа сцени (клас ScaleTransition), а також колір заповнення (клас FillTransition) і колір контуру форми Shape (клас StrokeTransition). Іншими словами, Transition-анімація визначає цілий ряд монтажних переходів, що змінюють властивості вузлів у часі. Як приклад нижче показано, яким чином в коді реалізується збільшення розмірів вузла наполовину в напрямку обох осей координат x і y протягом трьох секунд.

```
ScaleTransition st = new ScaleTransition(Duration.millis(3000));
st.setByX(1.5);
st.setByY(1.5);
st.setNode(yesButton);
st.play();
```

Клас PauseTransition дозволяє зробити паузу в послідовності анімацій. Клас PauseTransition має властивість duration (тривалість паузи) і конструктори public PauseTransition(Duration duration) і public PauseTransition(), а також клас-фабрику PauseTransitionBuilder. Клас AnimationTimer дозволяє створювати таймер, що викликається в кожному кадрі анімації. Створити таймер можна, якщо розширивши абстрактний клас AnimationTimer і перевизначити його метод handle (long

now), що викликається в кожному кадрі. Для управління таймером клас AnimationTimer пропонує методи start() і stop().

Пакет javafx.scene.transform платформи JavaFX забезпечує трансформацію вузлів графа сцени, яка складається з афінних перетворень: обертання, переміщення, масштабування і зсуву. На відмінну від анімації, трансформація графічних об'єктів не має плавного видимого переходу від початкової до кінцевої точки під час певного проміжку часу, а виконується відразу.

Базовим класом JavaFX трансформацій є клас javafx.scene.transform.Transform, який має реалізацію в вигляді класів Affine, Rotate, Scale, Shear і Translate. Застосувати JavaFX трансформації до вузла графа сцени можна двома способами. Перший спосіб – використати метод getTransforms() класу javafx.scene.Node, який повертає список ObservableList<Transform> об'єктів Transform, доповнити який можна методом addAll(). Інший спосіб – це застосування методів setRotate(), setRorationAxis(), setScaleX(), setScaleY(), setScaleZ(), setTranslateX(), setTranslateY(), setTranslateZ() класу javafx.scene.Node, який забезпечує трансформацію обертання, переміщення і масштабування для вузла графа сцени [3, с. 247].

Клас Affine представляє афінні перетворення матриці:

$$\begin{bmatrix} mxx & mxy & mxz & tx \\ myx & myy & myz & ty \\ mzx & mzy & mzz & tz \end{bmatrix}$$

за допомогою властивостей: mxx – X-множник матриці; mxy – XY-множник матриці; mxz – XZ-множник матриці; tx – зсув по осі x; myx – YX-множник матриці; myy – Y-множник матриці; myz – YZ-множник матриці; ty – зсув по осі y; mzx – ZX-множник матриці; mzy – ZY-множник матриці; mzz – Z-множник матриці; tz – зсув по осі z.

Екземпляр класу Affine можна створити за допомогою класу-фабрики AffineBuilder, за допомогою конструктора public Affine() або за допомогою статичного методу affine класу Transform, який повертає Affine-об'єкт. Афінні перетворення відображають n-вимірний об'єкт в n-вимірний, зберігають паралельність ліній і площин, а також пропорції паралельних об'єктів. За допомогою афінних перетворень можна створювати трансформації обертання, зсуву, масштабуванні і переміщення.

Для забезпечення обертання, масштабування, зсуву, переміщення вузла графа сцени використовуються класи Rotate, Scale, Shear та Translate відповідно.

Удосконалені елементи управління. JavaFX передбачає створення і використання комбінованих списків, панелей вкладок, дерев, таблиць, а також нові елементи управління графічного інтерфейсу, в тому числі селектор дат і меню-гармошку. Окрім того, засобами JavaFX можна побудувати багато видів діаграм. І для цього не потрібно спеціально встановлювати сторонні бібліотеки [7].

Розгортання додатків. JavaFX містить пакувальник додатків, який дозволяє створити простий у розгортанні автономний настільний додаток, що містить усі Java-бібліотеки часу виконання, необхідні для установки і запуску додатків.

Висновки. JavaFX є потужним інструментальним засобом для розробки графічного інтерфейсу користувача високої продуктивності, що включає аудіо, відео, графіку та анімацію. Для обробки подій використовуються лямбда-вирази, що істотно спрощує процес керування графічними інтерфейсами. Механізм прив'язки дозволяє реалізувати автоматичне оновлення однієї властивості при зміні іншої. Для побудови графічного інтерфейсу JavaFX пропонує використовувати декларативну мову розмітки FXML. Дизайнер може кодувати в FXML або використовувати JavaFX Scene Builder для інтерактивного проектування графічного інтерфейсу користувача. Scene Builder генерує розмітку FXML, яку можна перенести на IDE, де розробник може додати бізнес-логіку. JavaFX Scene Builder є ключовим засобом розробки для платформи JavaFX. Технологія JavaFX надає можливість змінювати зовнішній вигляд інтерфейсу користувача за допомогою таблиць стилів CSS, що, як правило, зручніше, ніж надання атрибутів FXML-розмітки або виклик методів в Java. Платформа JavaFX забезпечує створення двох видів анімації – анімацію по ключовим кадрам і анімацію з вбудованою часовою шкалою. Пакет javafx.scene.transform забезпечує трансформацію вузлів графа сцени, яка складається з афінних перетворень: обертання, переміщення, масштабування і зсуву. На відмінну від анімації, трансформація графічних об'єктів не має плавного видимого переходу від початкової до кінцевої точки під час певного проміжку часу, а виконується відразу. JavaFX містить удосконалені елементи управління, які забезпечують побудову діаграм, опрацювання мультимедійних даних тощо.

Напрямом подальшого дослідження є розробка насичених мережевих додатків, які розгортаються на різних платформах.

1. Герасимов В. В. Аналіз технологій розробки насичених інтернет-додатків на платформі Java / В. В. Герасимов, В. Я. Левицька // ИТ проектирования, моделирования, дизайна, WEB, 2017. – С. 355–363.
2. Карашецький В. П. Побудова графічного контенту додатків з використанням JavaFX і Swing компонентів і даних, взятих із баз даних / В. П. Карашецький. – Науковий вісник НЛТУ. – 2015. – Вип. 25.1. – С. 386–392.
3. Машнин Т. С. JavaFX 2.0: разработка RIA-приложений / Т. С. Машнин. – СПб.: БХВ-Петербург, 2012. – 320 с.
4. Парфенов Ю. Э. Разработка «насыщенных» интернет-приложений с помощью JavaFX / Ю. Э. Парфенов, В. Н. Федорченко // Системи обробки інформації. – 2012. – Вип. 8 (106). – С. 40–46.
5. Хорстманн К. Java SE 8. Вводный курс / К. Хорстманн. – М.: Вильямс, 2014. – 208 с.
6. Carl Dea. JavaFX 2.0: Introduction by Example, apress, 2011. – 181 p.
7. Carl□David Granbäck. Rich Internet Applications (RIAs): A Comparison Between Adobe Flex, JavaFX and Microsoft Silverlight. – Chalmers University of Technology – Göteborg, 2009. – P. 1–70.
8. JavaFX – Application [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/javafx/javafx_application.htm
9. JavaFX Scene Builder [Електронний ресурс]. – Режим доступу: http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html
10. Kishori Sharan. Learn JavaFX 8 (The Expert's Voice in Java): Building User Experience and Interfaces with Java 8. – New York, 2015. – P. 1–1173.