Оглавление

ОПИСАНИЕ ОБЪЕКТА И ПРЕДМЕТА ИССЛЕДОВАНИЯ

В данной работе под объектом исследования мы будем понимать процесс разработки десктопных приложений на языке Java, на который и будет направлена текущая исследовательская деятельность. С понятием объекта тесно связано понятие предмета исследования.

Предмет исследования – это конкретная часть объекта, внутри которой ведется поиск. В данном случае в качестве предмета исследования будет выступать разработка графических интерфейсов пользователя (GUI) для десктопных приложений на языке Java.

ОБОСНОВАНИЕ АКТУАЛЬНОСТИ

Java, «чистокровный» объектно-ориентированный язык с прекрасным набором библиотек, изначально переносимый между различными платформами и практически не зависящий от конкретного производителя [4].

В наши дни язык Java остается одной из популярнейших платформ для разработки приложений. Это особенно справедливо в таких областях, как веб-службы, фреймворки веб-приложений и инструменты для работы с XML. Также Java и его основные API можно использовать в программировании для мобильной операционной системы Google Android, используемой на миллиардах устройств по всему миру [2].

Важнейшую часть любой программы составляет ее пользовательский интерфейс. На самом деле пользователя мало интересуют сложные алгоритмы обработки данных и удачные находки, реализованные внутри программы, также мало его занимают и технологии, примененные для создания самого приложения и его пользовательского интерфейса. Пользователь видит только то, что видит, и именно этот факт следует учитывать при решении всех задач по проектированию и созданию пользовательского интерфейса. Интерфейс должен быть максимально быстрым и максимально удобным для пользователя, который должен также получать и эстетическое удовольствие от работы с программой [4].

В ходе разработки десктопных приложений программисты сталкиваются с изобилием существующего Java-контента для графических интерфейсов (GUI, или просто UI) [2].

Кроме десктопных приложений проблемы выбора GUI-фреймворков для языка Java характерны и для RIA (Rich Internet Application) — Web-приложений, предоставляющих большую интерактивность для клиента. В данном контексте RIA-приложение является промежуточным между "тонким" клиентом и "толстым" клиентом — настольным приложением [3].

Технологии создания RIA-приложений платформы Java берут свое начало от Java-апплетов, GUI-интерфейсы которых использовали графические системы AWT и Swing для организации взаимодействия с пользователем и отображения ему данных, текста, графики и анимации [3].

СФОРМИРОВАННЫЙ ПЛАН ИССЛЕДОВАНИЯ

Целью данной исследовательской работы является анализ особенностей наиболее популярных GUI-фреймворков (Swing и JavaFX) для языка Java при создании пользовательского интерфейса десктопных приложений на примере десктопного приложения простейшего текстового редактора «Блокнот».

Задачи исследования:

1. Описание использованных программных технологий и средств (среды разработки, библиотек фреймворков).
2. Описание проведенных экспериментов по разработке и использованию программных средств.
3. Представление полученных выводов в наглядной форме.

Для сравнительной оценки GUI-фреймворков Swing и JavaFX, используемых при разработке текстового редактора «Блокнот» на языке Java как на этапе разработки программных продуктов, так и на этапе их использования могут быть использованы следующие методы исследования.

1) измерение времени, затраченного на разработку программных продуктов;

2) измерение количества файлов и папок, необходимых для функционирования программных продуктов;
3) измерение количества строк кода, написанных при разработке программных продуктов;
4) описание визуальных различий элементов графический интерфейсов приложений;
5) описание времени отклика элементов графический интерфейсов приложений;
6) выборочное измерение времени отклика элементов интерфейсов на основании результатов п. 5.

ОБЗОР ЛИТЕРАТУРЫ

В Java для создания графического интерфейса обычно используются библиотеки AWT, Swing и JavaFX [1, 3, 5].

Графическая библиотека AWT (Abstract Window Toolkit) исторически была самой первой и базовой графической Java-системой набора JDK 1.0, дополненной затем библиотекой Java 2D двумерной графики и изображений [1, 3].

Библиотека AWT предоставляет разработчику возможность использования таких основных компонентов GUI-интерфейса, как кнопки, переключатели, списки, метки, окна выбора файла, меню, компоненты визуализации и редактирования текста, функции drag and drop, возможность обработки событий UIкомпонентов, компоновки компонентов в рабочей области, работы с цветом, шрифтом, графикой, рисования и печати.

Несмотря на весьма неплохую первоначальную идею библиотеки AWT, реализация этой идеи подразумевала довольно ограниченный набор компонентов, что очень сильно вредило Java как языку создания настольных приложений (desktop applications) [4].

Кроме того, библиотека AWT является тяжеловесной [3]. Что же это значит?

Исходное назначение AWT — предоставить набор графических компонентов, который вобрал бы в себя наиболее характерные черты современных элементов управления и позволил бы однократно создавать пользовательские интерфейсы, подходящие для любой платформы. Компоненты AWT на самом деле не выполняют никакой работы и очень просты — это просто «Java-оболочки» для элементов управления той операционной системы, на которой работает пользователь. Все запросы к этим компонентам незаметно перенаправляются к операционной системе, которая и выполняет всю работу. Чтобы сделать классы AWT независимыми от конкретной платформы, каждому из них соответствует своеобразный помощник (peer) [4]. То есть AWT содержит собственную (родную, native) библиотеку java.awt.peer, через которую взаимодействует с операционной системой компьютера, поэтому отображение AWT GUI-интерфейса зависит от операционной системы, в которой приложение развернуто [3]. Такие компоненты называют тяжелыми. Главная проблема здесь связана с тем, что у разных операционных систем разный инструментарий и разные возможности [1].

Ограниченность набора GUI-компонентов библиотеки AWT и ее тяжеловесность послужили причиной создания графической системы Swing, которая основывается на библиотеке AWT (компоненты, созданные с помощью библиотеки Swing, реализуются средствами Java) и поэтому является уже легковесной (lightweight) [3, 4].

Можно сказать, что в библиотеках Swing и AWT по-разному решается проблема универсальности кода, краеугольная для концепции, положенной в основу Java [1].

Swing – это набор графических компонентов для создания пользовательских интерфейсов приложений и апплетов, а также вспомогательные классы и инструменты для работы с этими компонентами [4].

Многие решения, принятые разработчиками Swing, были направлены на то, чтобы расширить излишне лаконичный набор компонентов AWT и сделать библиотеку Swing пригодной для создания с минимальными усилиями современных пользовательских интерфейсов [4].

В частности, библиотека Swing дополняет библиотеку AWT такими компонентами GUI-интерфейса, как панель выбора цвета, индикатор состояния, переключатель, слайдер и спиннер, панель с вкладками, таблицы и деревья, расширенными возможностями компоновки GUI-компонентов, таймером, возможностью изменения внешнего вида LookAndFeel GUI-интерфейса, отображения HTML-контента [3].

Кроме того, Swing имеет подключаемые внешний вид и поведение. Это свойство позволяет компонентам Swing вести себя максимально гибко, приспосабливаясь к любым условиям, принимать необходимый вид, а также предоставляет программистам такой удобный инструмент, как модели. Реализованы подключаемые внешний вид и поведение в Swing на основе уже ставшей классической архитектуры MVC [3, 4].

В основании модели Swing лежит паттерн проектирования, известный под названием «Модель — представление — контроллер» (Model/View/Controller, MVC). Авторы пакета Swing основательно потрудились над последовательным применением этого паттерна, чтобы при знакомстве с новыми компонентами их поведение и использование казались вам знакомыми. Архитектура MVC стремится отделить то, что видит пользователь (представление), от внутреннего состояния (модели) и от совокупности взаимодействий (контроллер), которые вызывают изменения в этих частях. Такое разделение обязанностей позволяет программисту сосредоточиться на том, чтобы правильно разработать каждую часть. Сетевой трафик может незаметно обновлять модель. Представление может синхронизироваться через постоянные интервалы времени, благодаря чему интерфейс «не тормозит» и быстро реагирует на действия пользователя.

Паттерн MVC предоставляет мощную, но хорошо управляемую архитектуру, которая подходит для разработки любых десктопных приложений [2, 4].

Компоненты, созданные на основе библиотеки Swing, более функциональны, поэтому для создания графических компонентов обычно используют их. При этом важно понимать, что библиотека Swing не заменяет библиотеку AWT, а дополняет ее. В настоящее время при использовании библиотеки Swing без использования библиотеки AWT в любом случае не обойтись [1].

Для каждого графического компонента (например, кнопки, текстового поля, метки) существует определенный класс, причем это может быть как класс из библиотеки AWT, так и класс из библиотеки Swing [1].

Начиная с выпуска Java 2, стандартными библиотеками языка Java являются как библиотека AWT (которая считается устаревшей), так и основанная на ней библиотека Swing. Поэтому никаких препятствий для совместного использования графических компонентов этих двух библиотек нет [4].

Несмотря на богатые возможности графических систем AWT и Swing, они не удовлетворяют современным требованиям работы с медиаконтентом, что и послужило причиной создания платформы JavaFX, которая предоставляет современные GUI компоненты, богатый набор библиотек графического и медиапрограммного API интерфейса, а также высокопроизводительную среду выполнения приложений [3].

Один и тот же Java-код, созданный на базе платформы JavaFX, может запускаться как настольное приложение, которое разворачивается на клиентском компьютере автономно, или разворачиваться как Java Web Start-приложение, или отображаться в Web-браузере как JavaFX-апплет, встроенный в HTML-страничку [3].

Технология JavaFX обеспечивает создание мощного графического интерфейса пользователя (Graphical User Interface, GUI) для крупномасштабных приложений, ориентированных на обработку данных,

насыщенных медиаприложений, поставляющих разнообразный медиаконтент пользователю, Mashup-приложений, объединяющих различные Web-ресурсы для пользователя, компонентов высококачественной графики и анимации для Web-сайтов, различного рода пользовательских программ с графикой, анимацией и интерактивными элементами [3].

Платформа JavaFX версии состоит из программного интерфейса JavaFX API, альтернативного декларативного языка FXML описания GUI-интерфейса, среды выполнения JavaFX Runtime и набора разработчика JavaFX SDK [3].

Программный интерфейс JavaFX API дает возможность разрабатывать как десктопные приложения, так и приложения Rich Client Application (RIA) с насыщенным графическим интерфейсом пользователя, код которых сочетает широкие возможности платформы Java с богатой графической и медиафункциональностью платформы JavaFX [3].

То есть один и тот же код JavaFX-приложения может разворачиваться как настольное приложение, как приложение Java Web Start или отображаться в Web-браузере как JavaFX-апплет, встроенный в HTML-страницу. Такая универсальность использования Java-кода обеспечивается моделью развертывания приложений платформы JavaFX или, точнее, сборкой JavaFX-приложения [3].

Компоненты графического интерфейса пользователя (Graphical User Interface, GUI) JavaFX-приложения образуют сцену, логическая структура которой описывается графом сцены. Отображением GUI-интерфейса JavaFX-приложения является графическое представление графа сцены [3].

Программный интерфейс JavaFX API дает возможность разрабатывать приложения с насыщенным графическим интерфейсом пользователя, код которых сочетает широкие возможности платформы Java с богатой графической и медиафункциональностью платформы JavaFX [3].

Классы пакетов программного интерфейса JavaFX 2.0 API предоставляют для использования в разработке свои публичные свойства,

поля, конструкторы и методы. При этом свойства записи/чтения JavaFX-классов доступны с помощью традиционных методов getXXX() и setXXX(), а также методов JavaFX Beans свойств [3].

Компоненты графического интерфейса пользователя платформы JavaFX представлены такими пакетами JavaFX API, как javafx.scene.control, javafx.scene.chart, javafx.scene.image, javafx.scene.layout, javafx.scene.media, javafx.scene.shape, javafx.scene.text, javafx.scene.web и javafx.stage [3].

Пакет javafx.embed.swing платформы JavaFX 2.0 обеспечивает встраивание JavaFX сцены в Swing-приложение. Таким образом платформа JavaFX может использоваться совместно с технологией Swing, а также с другими языками — JRuby, Groovy и JavaScript для создания больших и комплексных приложений с насыщенным графическим интерфейсом пользователя [3].

Список использованных источников

1. Васильев А. Java для всех. СПб., Питер, 2020. 512 с.

2. Лой М., Нимайер П., Лук Д. Программируем на Java. СПб., Питер, 2023. 544 с.

3. Машнин Т.С. JavaFX 2.0: разработка RIA-приложений. СПб., БХВ-Петербург, 2012. 320 с.

4. Портянкин И. Swing: Эффектные пользовательские интерфейсы. Москва, Лори, 2011. 590 с.

5. Прохоренок Н.И. JavaFX. СПб., БХВ-Петербург, 2020. 768 с.