Николай Прохоренок

JavaFX

Санкт-Петербург
«БХВ-Петербург»
2019

УДК 004.438Java
ББК 32.973.26-018.1
П84

Прохоренок Н. А.

П84 JavaFX. — СПб.: БХВ-Петербург, 2019. — 768 с.: ил. — (В подлиннике)

ISBN 978-5-9775-4072-8

Описываются базовые возможности библиотеки JavaFX, позволяющей создавать приложения с графическим интерфейсом на языке Java. Рассматриваются способы обработки событий, управление свойствами окна, создание формы с помощью программы Scene Builder, а также все основные компоненты (кнопки, текстовые поля, списки, таблицы, меню и др.) и варианты их размещения внутри окна. Описаны трансформации и эффекты, графики и диаграммы, аудио и видео, стили JavaFX CSS. Книга ориентирована на тех, кто уже знаком с языком программирования Java и хотел бы научиться разрабатывать оконные приложения, насыщенные графикой, анимацией и интерактивными элементами. Большое количество практических примеров помогает начать разработку самостоятельно. Весь материал тщательно подобран, хорошо структурирован и компактно изложен, что позволяет использовать книгу как удобный справочник. Электронный архив с примерами находится на сайте издательства.

Для программистов

УДК 004.438Java
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Сависте*
Компьютерная верстка	*Ольги Сергиенко*
Дизайн серии	*Марины Дамбиевой*
Оформление обложки	*Карины Соловьевой*

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9775-4072-8

Оглавление

Предисловие

Добро пожаловать в мир JavaFX!

JavaFX — это библиотека для создания на языке Java приложений с графическим интерфейсом, насыщенных графикой, анимацией и интерактивными элементами. Библиотека является кроссплатформенной, поэтому мы можем создавать оконные приложения под Windows, Linux и Mac, а также мобильные приложения под Android и iOS (подробную информацию о создании мобильных приложений можно найти на странице **https://gluonhq.com/products/mobile/**). В этой книге мы рассмотрим процесс создания оконных приложений, применительно к операционной системе Windows.

До версии Java 8 библиотека JavaFX была доступна отдельно. В версиях Java с 8 по 10 включительно библиотека JavaFX входит в состав стандартной библиотеки, поэтому ее дополнительно устанавливать не нужно. Начиная с Java 11, JavaFX поставляется отдельно в виде набора модулей, доступных на странице **https://gluonhq.com/products/javafx/**. В этой книге мы будем изучать возможности библиотеки JavaFX версий 11 и 12.

Давайте рассмотрим структуру книги:

- *Глава 1* является вводной. Мы установим необходимое программное обеспечение, настроим среду разработки, скомпилируем и запустим первую программу — как из командной строки, так и из редактора Eclipse. Кроме того, вкратце разберемся со структурой программы, научимся управлять объектом приложения, а также рассмотрим процесс создания формы с помощью программы Scene Builder.
- В *главе 2* рассматривается жизненный цикл главного окна приложения: создание окна, его отображение, сокрытие и закрытие, сворачивание и разворачивание до максимального размера или во весь экран, отображение окна поверх всех окон, а также возможность блокировки других окон приложения до момента закрытия окна. Мы научимся изменять размеры окна, управлять его местоположением на экране, менять значок в заголовке окна и др.
- *Глава 3* полностью посвящена менеджерам компоновки (сцене, контейнерам и панелям), позволяющим управлять размещением компонентов внутри окна.

- При взаимодействии пользователя с окном происходят *события*. В ответ на события генерируются определенные сигналы — своего рода извещения о том, что пользователь выполнил какое-либо действие, или в самой системе возникло некоторое условие. События являются важнейшей составляющей приложения с графическим интерфейсом, поэтому необходимо знать, как назначить обработчик события, как удалить обработчик, а также уметь правильно обработать событие. Обработку событий мы рассмотрим в *главе 4*.
- В *главе 5* мы научимся работать с 2D-графикой, а в *главе 6* — с изображениями.
- В *главе 7* будет рассмотрен компонент `Canvas`, позволяющий программно рисовать на поверхности, которую мы называем *холстом*. Можно нарисовать любую геометрическую фигуру с заливкой или обводкой, вывести текст различными шрифтами, добавить на холст изображение, а также применить к графическим объектам эффекты или трансформации.
- В *главе 8* мы научимся применять различные трансформации — как к элементам холста, так и к обычным узлам, — например, сдвигать их, масштабировать и вращать. Помимо трансформаций мы рассмотрим применение различных эффектов — например, добавление внешней или внутренней тени, создание зеркального отражения и др.
- Все рассмотренные в *главе 8* трансформации применяются сразу. Однако часто нужно, чтобы трансформации применялись постепенно. Например, при наведении указателя мыши на пункт меню надо плавно увеличить значок и изменить его прозрачность, а при отведении указателя также плавно вернуть настройки в первоначальное положение. В *главе 9* мы как раз и научимся это делать, а также рассмотрим создание произвольной анимации.
- В *главе 10* мы научимся работать с 3D-графикой.
- *Глава 11* посвящена основным компонентам пользовательского интерфейса, таким как надписи, кнопки, переключатели, флажки и др. Кроме того, мы рассмотрим запуск задач в отдельных потоках, с отображением хода выполнения задачи, а также использование JavaFX-свойств.
- В *главе 12* мы рассмотрим различные компоненты, позволяющие работать с текстом и Web-страницами, — например, однострочные и многострочные текстовые поля.
- В JavaFX имеется широкий выбор компонентов, позволяющих отображать одномерный список строк в свернутом или развернутом состояниях. Списки поддерживают концепцию «модель/представление», позволяющую отделить данные от их отображения. В роли модели выступает объект, реализующий интерфейс `ObservableList<E>`, а в роли представления — компоненты `ChoiceBox<T>`, `ComboBox<T>` и `ListView<T>`. Одномерные списки мы рассмотрим в *главе 13*, а в *главе 14* научимся работать с таблицами и иерархическими списками.
- *Глава 15* посвящена созданию главного меню приложения — основного компонента пользовательского интерфейса, позволяющего компактно разместить множество команд, объединяя их в логические группы. Кроме того, мы рассмот-

рим возможность отображения контекстного меню, а также различных всплывающих окон, например, с текстом подсказки.

- В *главе 16* мы научимся создавать различные диалоговые окна, которые предназначены для информирования пользователя, а также для получения данных от него.
- Библиотека JavaFX содержит множество классов, реализующих различные виды диаграмм, — например, круговую диаграмму, линейный график, гистограмму и т. д. Эти классы мы подробно рассмотрим в *главе 17*.
- В *главе 18* мы научимся воспроизводить аудио и видео.
- Библиотека JavaFX позволяет изменять внешний вид окна, контейнеров и компонентов с помощью *каскадных таблиц стилей* (CSS, Cascading Style Sheets), точно так же, как это делается в Web-программировании. Применение стилей позволяет задавать точные характеристики практически всех элементов пользовательского интерфейса. Применять стили мы научимся в *главе 19*.
- И наконец, в *главе 20* мы рассмотрим возможность вывода текста надписей на различных языках в зависимости от настроек локали, научимся отображать окно с заставкой на время запуска приложения, а также создадим дистрибутив для запуска приложения, включающий только необходимые модули и компоненты.

Чтобы уменьшить объем книги, основная часть кода примеров вынесена в отдельные проекты, которые размещены в электронном архиве. Электронный архив можно загрузить с FTP-сервера издательства «БХВ-Петербург» по ссылке: **ftp://ftp.bhv.ru/9785977540728.zip** или со страницы книги на сайте **www.bhv.ru** (см. *приложение*).

Каждый проект состоит из двух файлов:

- JAR-архив — содержит скомпилированные файлы с классами (Java 11), файл манифеста, необходимые ресурсы (например, изображения), а также файлы с исходным кодом классов. Чтобы посмотреть исходный код примеров, достаточно распаковать JAR-архив с помощью любого архиватора. Исходный код дополнительно содержит фрагменты, которые закомментированы. Эти фрагменты являются либо альтернативным способом написания кода, либо ограничивают количество данных, выводимых в консоль. Запустить примеры можно двойным щелчком левой кнопкой мыши на значке JAR-архива при условии создания ассоциации запуска с файлом runJAR11.bat (см. *разд. 1.3*). Такой способ запуска не отображает окно консоли, поэтому отладочные данные, выводимые в консоль, будут недоступны;
- BAT-файл — содержит код для запуска JAR-архива в Windows с отображением окна консоли, в которое будет выводиться отладочная информация. Запустить файл можно двойным щелчком левой кнопкой мыши на значке BAT-файла. Предварительно нужно указать в файле path_to_JRE.txt полный путь к JRE, а в файле path_to_JavaFX.txt — полный путь до библиотеки JavaFX. В составе пути следует избегать пробелов и русских букв, а в файлах должна быть только одна строка без символов перевода строки, иначе получите ошибку.

Желаю приятного изучения и надеюсь, что книга поможет вам реализовать как самые простые, так и самые сложные приложения.

ГЛАВА 1

Первые шаги

Надеюсь, что вы владеете основами языка Java и знаете, как запустить командную строку. Если это не так, то вначале следует изучить язык Java, например, по моей книге «Основы Java»[1]. Материала настолько много, что у меня нет возможности объяснять здесь, что такое класс, метод, поле и т. д. Мы будем изучать библиотеку JavaFX и не станем тратить объем книги на основы языка.

1.1. Установка OpenJDK

Для изучения возможностей библиотеки JavaFX нужно установить на компьютер Java 11. Мы воспользуемся не Oracle JDK, а свободной реализацией JDK, которая имеет название OpenJDK. OpenJDK представляет собой комплект разработчика Java Development Kit (сокращенно JDK), включающий компилятор javac.exe, стандартную библиотеку, исполнительную среду Java Runtime Environment (сокращенно JRE) и различные утилиты. Для загрузки дистрибутива переходим на страницу **http://jdk.java.net/11/** и скачиваем архив openjdk-<Версия>_windows-x64_bin.zip. Далее в корне диска C: создаем каталог OpenJDK и копируем в него каталог jdk-11 из архива. Путь до компилятора должен быть таким: C:\OpenJDK\jdk-11\bin\javac.exe.

Далее необходимо добавить путь к каталогу C:\OpenJDK\jdk-11\bin в системную переменную PATH. Если вы устанавливали предыдущие версии Java, то нужно дополнительно удалить путь C:\ProgramData\Oracle\Java\javapath, иначе будет запускаться другая версия Java.

Чтобы изменить системную переменную, переходим в **Параметры | Панель управления | Система и безопасность | Система | Дополнительные параметры системы**. В результате откроется окно **Свойства системы**. На вкладке **Дополнительно** нажимаем кнопку **Переменные среды**. В списке **Системные переменные** открывшегося окна выделяем строку с переменной **Path** и нажимаем кнопку **Изменить**. В открывшемся окне изменяем значение в поле **Значение переменной** — для

[1] Прохоренок Н. А. Основы Java. — СПб.: БХВ-Петербург, 2017. — 704 с.: ил. (см. **http://www.bhv.ru/books/book.php?id=198051**).

этого переходим в конец существующей строки, ставим точку с запятой, а затем вводим путь к каталогу C:\OpenJDK\jdk-11\bin:

```
<Текущее значение>;C:\OpenJDK\jdk-11\bin
```

> ***Внимание!***
>
> Случайно не удалите существующее значение переменной PATH, иначе другие приложения перестанут запускаться.

Помимо добавления в переменную PATH пути к каталогу C:\OpenJDK\jdk-11\bin, во избежание проблем с настройками различных редакторов, необходимо прописать переменную окружения JAVA_HOME и присвоить ей путь к каталогу с установленным JDK. Делается это также в окне **Переменные среды**, но в списке переменных для пользователя, хотя вы можете добавить ее и в список системных переменных. Нажимаем в этом окне кнопку **Создать**, в поле **Имя переменной** вводим значение JAVA_HOME, а в поле **Значение переменной** — C:\OpenJDK\jdk-11. Сохраняем все изменения.

Запускаем командную строку и проверяем установку переменной JAVA_HOME:

```
C:\book>set JAVA_HOME
JAVA_HOME=C:\OpenJDK\jdk-11
```

Чтобы проверить правильность установки Java 11, выполняем следующие команды:

```
java --version
javac --version
```

Результат должен быть примерно следующим:

```
C:\book>java --version
openjdk 11 2018-09-25
OpenJDK Runtime Environment 18.9 (build 11+28)
OpenJDK 64-Bit Server VM 18.9 (build 11+28, mixed mode)

C:\book>javac --version
javac 11
```

> ***Примечание***
>
> Если вы получили в результате другую версию Java или компилятора, то вначале перезапустите командную строку и повторите проверку. Если версия не изменилась, то, скорее всего, в переменной PATH прописан путь к предыдущей версии Java. Путь к старой версии Java нужно удалить.

1.2. Установка библиотеки JavaFX

Для начала создадим следующую структуру каталогов:

```
book\
    src\
    bin\
    modules\
```

```
JavaSE\
   eclipse\
   JavaFX\
      projects\
```

Каталоги book и JavaSE лучше разместить в корне какого-либо диска. В моем случае это будет диск C:, следовательно, пути к содержимому каталогов: C:\book и C:\JavaSE. Можно создать каталог и в любом другом месте, но в пути не должно быть русских букв и пробелов — только английские буквы, цифры, тире и подчеркивание. Остальных символов лучше избегать, если не хотите проблем с компиляцией и запуском программ.

Кстати, рекомендация не использовать русские буквы относится и к имени пользователя. Многие программы сохраняют настройки в каталоге C:\Users\<Имя пользователя>. В результате в пути окажутся русские буквы, которые могут стать причиной множества проблем.

В каталоге book мы станем размещать наши тестовые программы. Вложенный каталог src мы будем использовать для файлов с исходным кодом, каталог bin — для скомпилированных файлов, а каталог modules — для модульных JAR-архивов.

Внутри каталога JavaSE у нас созданы два вложенных каталога:

- eclipse — в этот каталог мы установим редактор Eclipse;
- JavaFX — в этот каталог мы установим библиотеку JavaFX и программу Scene Builder, а во вложенном каталоге projects станем сохранять проекты из редактора Eclipse.

Как уже отмечалось ранее, библиотека JavaFX входит в состав стандартной библиотеки, начиная с Java 8 и заканчивая Java 10. При использовании указанных версий Java дополнительно устанавливать JavaFX не нужно. Начиная с Java 11, библиотека JavaFX не входит в состав стандартной библиотеки, и ее приходится устанавливать дополнительно в виде комплекта модулей, называемого JavaFX SDK. Официальный сайт библиотеки JavaFX: **https://openjfx.io/**. Документацию к библиотеке JavaFX можно найти на странице: **https://openjfx.io/javadoc/<Версия>/**.

Для загрузки библиотеки JavaFX 12 переходим на страницу **https://gluonhq.com/products/javafx/** и из раздела **JavaFX Windows SDK** скачиваем архив openjfx-12_windows-x64_bin-sdk.zip. Распаковываем этот архив в каталог C:\JavaSE\JavaFX, а затем переименовываем каталог javafx-sdk-12 в 12, чтобы сделать путь к библиотеке более коротким, т. к. нам придется его указывать каждый раз в качестве пути поиска модулей при компиляции и запуске программ. Путь до модуля `javafx.base` должен быть таким: C:\JavaSE\JavaFX\12\lib\javafx.base.jar.

Для создания дистрибутива нам нужно еще скачать архив с файлами в формате `jmod`. Со страницы **https://gluonhq.com/products/javafx/** из раздела **JavaFX Windows jmods** скачиваем архив openjfx-12_windows-x64_bin-jmods.zip. Распаковываем архив и переименовываем каталог с модулями в jmods, а затем копируем его в каталог C:\JavaSE\JavaFX\12. Путь до модуля `javafx.base` должен быть таким: C:\JavaSE\JavaFX\12\jmods\javafx.base.jmod. Процесс создания дистрибутива мы рассмотрим в *разд. 20.4*.

JavaFX SDK состоит из трех каталогов:

- bin — содержит библиотеки динамической компоновки (DLL), необходимые для запуска JavaFX-приложений;
- legal — содержит различную информацию о модулях, в частности лицензии;
- lib — содержит модули, файл с настройками и файл src.zip с исходным кодом модулей. Путь к этому каталогу нужно прописывать в качестве пути поиска модулей в опции `--module-path` (или `-p`) при компиляции и запуске JavaFX-приложений.

Библиотека JavaFX состоит из следующих модулей (расположены в каталоге C:\JavaSE\JavaFX\12\lib):

- `javafx.base` — базовый модуль, от которого зависят все остальные модули библиотеки JavaFX. Зависимость от других модулей:

```
C:\book>jdeps --module-path C:\JavaSE\JavaFX\12\lib -s
              -m javafx.base
javafx.base -> java.base
javafx.base -> java.desktop
```

- `javafx.graphics` — содержит пакеты с классами, позволяющими создать объект приложения, различные окна, управлять размещением компонентов внутри сцены, работать с изображениями, графикой, анимацией, эффектами, трансформациями и пр. Зависимость от других модулей:

```
C:\book>jdeps --module-path C:\JavaSE\JavaFX\12\lib -s
              -m javafx.graphics
javafx.graphics -> java.base
javafx.graphics -> java.desktop
javafx.graphics -> java.xml
javafx.graphics -> javafx.base
javafx.graphics -> jdk.unsupported
```

- `javafx.controls` — содержит пакеты с классами, позволяющими создать различные компоненты (например, надпись, кнопку, график и т. д.). Зависимость от других модулей:

```
C:\book>jdeps --module-path C:\JavaSE\JavaFX\12\lib -s
              -m javafx.controls
javafx.controls -> java.base
javafx.controls -> javafx.base
javafx.controls -> javafx.graphics
```

- `javafx.media` — позволяет работать с аудио и видео. Зависимость от других модулей:

```
C:\book>jdeps --module-path C:\JavaSE\JavaFX\12\lib -s
              -m javafx.media
javafx.media -> java.base
javafx.media -> javafx.base
javafx.media -> javafx.graphics
```

- `javafx.web` — содержит классы, предназначенные для работы с Интернетом. Зависимость от других модулей:

```
C:\book>jdeps --module-path C:\JavaSE\JavaFX\12\lib -s
              -m javafx.web
javafx.web -> java.base
javafx.web -> java.desktop
javafx.web -> java.xml
javafx.web -> javafx.base
javafx.web -> javafx.controls
javafx.web -> javafx.graphics
javafx.web -> javafx.media
javafx.web -> jdk.jsobject
javafx.web -> jdk.xml.dom
```

- `javafx.fxml` — позволяет использовать язык разметки FXML. Зависимость от других модулей:

```
C:\book>jdeps --module-path C:\JavaSE\JavaFX\12\lib -s
              -m javafx.fxml
javafx.fxml -> java.base
javafx.fxml -> java.scripting
javafx.fxml -> java.xml
javafx.fxml -> javafx.base
javafx.fxml -> javafx.graphics
```

При работе с этим модулем следует учитывать, что используется рефлексия. Если вы создаете модульное приложение, то в файле module-info.java нужно объявить пакет с классом открытым (с помощью ключевого слова `opens`), иначе получите исключение:

```
opens <Пакет>;
```

Если вы не хотите открывать пакет для всех, то после ключевого слова `to` нужно указать для каких модулей пакет является открытым:

```
opens <Пакет> to <Модули через запятую>;
```

- `javafx.swing` — позволяет встроить компоненты Swing внутрь JavaFX-приложения и наоборот. Зависимость от других модулей:

```
C:\book>jdeps --module-path C:\JavaSE\JavaFX\12\lib -s
              -m javafx.swing
javafx.swing -> java.base
javafx.swing -> java.datatransfer
javafx.swing -> java.desktop
javafx.swing -> javafx.base
javafx.swing -> javafx.graphics
javafx.swing -> jdk.unsupported.desktop
```

1.3. Создание и запуск JavaFX-приложения из командной строки

Давайте попробуем создать наше первое JavaFX-приложение для запуска из командной строки. Вначале создаем следующую структуру каталогов: C:\book\src\TestJavaFX\application. Затем добавляем в каталог application файл Main.java с кодом из листинга 1.1.

Для создания файла с программой можно воспользоваться любым текстовым редактором, например редактором Notepad++. Файл с программой должен быть сохранен в кодировке UTF-8. Мы будем пользоваться именно этой кодировкой, т. к. она позволяет хранить любые символы. Работая с редактором Notepad++, убедитесь, что в меню **Кодировки** установлен флажок **Кодировать в UTF-8 (без BOM)**. Если это не так, то в меню **Кодировки** следует выбрать пункт **Преобразовать в UTF-8 без BOM**.

Листинг 1.1. Первое JavaFX-приложение

```
package application;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.text.Font;
import javafx.stage.Stage;

public class Main extends Application {

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

   @Override
   public void start(Stage stage) throws Exception {
      BorderPane root = new BorderPane();
      Label text = new Label("Привет, мир!");
      text.setFont(new Font(32));
      root.setCenter(text);
      Scene scene = new Scene(root, 400, 150);

      stage.setTitle("Заголовок окна");
      stage.setScene(scene);
      stage.show();
   }
}
```

Скомпилируем этот файл в командной строке (команда состоит не из трех строк, как показано здесь, а из одной общей длинной строки):

```
C:\book>javac -encoding utf-8 --module-path C:\JavaSE\JavaFX\12\lib
              --add-modules javafx.controls -d C:\book\bin\TestJavaFX
              C:\book\src\TestJavaFX\application\*.java
```

В результате компиляции в каталоге C:\book\bin\TestJavaFX\application будет создан файл Main.class с байт-кодом. Попробуем запустить наше первое JavaFX-приложение из командной строки:

```
C:\book>java --module-path C:\JavaSE\JavaFX\12\lib
             --add-modules javafx.controls
             -cp C:\book\bin\TestJavaFX application.Main
```

В результате отобразится окно, показанное на рис. 1.1. Окно можно перемещать по экрану, изменять его размеры, взявшись мышью за границы, а также свернуть или развернуть его с помощью соответствующих кнопок в заголовке окна. Для закрытия окна следует воспользоваться кнопкой **Закрыть** в заголовке окна, кнопкой **Закрыть** в меню окна (щелкните левой кнопкой мыши на значке в заголовке окна, чтобы отобразить меню) или нажать комбинацию клавиш <Alt>+<F4>.

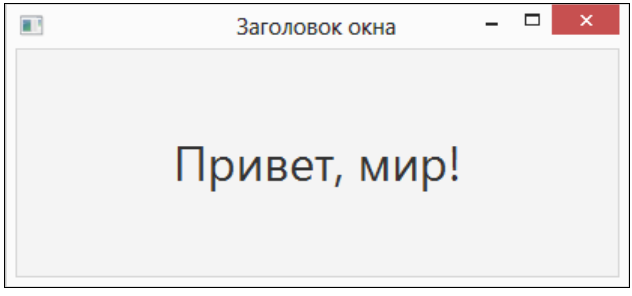

Рис. 1.1. Результат выполнения программы из листинга 1.1

В приведенных командах мы указали следующие опции:

- `-encoding` — задает кодировку исходных файлов;
- `--module-path` — указывает путь поиска модулей;
- `--add-modules` — добавляет модули. Для компиляции и запуска нашего приложения нужны модули `java.base`, `javafx.base`, `javafx.graphics` и `javafx.controls`. Так как модуль `javafx.controls` зависит от всех остальных модулей, достаточно добавить только этот модуль. Вместо указания модулей через запятую можно задать значения `ALL-DEFAULT`, `ALL-SYSTEM` и `ALL-MODULE-PATH`;
- `-cp` — задает путь поиска классов;
- `-d` — указывает каталог, в который будут записаны скомпилированные файлы.

Теперь создадим исполняемый JAR-архив:

```
C:\book>jar cvfe C:\book\TestJavaFX.jar application.Main
            -C C:\book\bin\TestJavaFX\ .
```

```
added manifest
adding: application/(in = 0) (out= 0)(stored 0%)
adding: application/Main.class(in = 1036) (out= 644)(deflated 37%)
```

В результате в каталоге C:\book будет создан файл TestJavaFX.jar. Вначале попробуем запустить его из командной строки:

```
C:\book>java --module-path C:\JavaSE\JavaFX\12\lib
             --add-modules javafx.controls -jar C:\book\TestJavaFX.jar
```

Как видно из приведенных примеров, всегда нужно указывать путь к библиотеке JavaFX и добавлять нужные модули. Если мы сейчас попробуем запустить программу двойным щелчком мыши на значке JAR-архива, то ничего не получится, т. к. путь поиска модулей не указан, и модули не добавлены. Давайте сделаем так, чтобы исполняемые JAR-архивы с JavaFX-приложениями запускались путем двойного щелчка на значке архива. Для этого сначала в каталоге C:\OpenJDK\jdk-11 создаем файл runJAR11.bat (кодировка файла должна быть windows-866) со следующим содержимым (вторая команда состоит из одной длинной строки, а не из двух строк, как здесь показано):

```
@echo off
start C:\OpenJDK\jdk-11\bin\javaw.exe --module-path
C:\JavaSE\JavaFX\12\lib --add-modules ALL-MODULE-PATH -jar %1
```

Далее щелкаем правой кнопкой мыши на значке JAR-архива TestJavaFX.jar и из контекстного меню выбираем пункт **Открыть с помощью** (или **Открыть с помощью | Выбрать программу**). В открывшемся окне устанавливаем флажок **Использовать это приложение для всех файлов jar**. Переходим по ссылке **Дополнительно | Найти другое приложение на этом компьютере** и выбираем программу runJAR11.bat. Теперь при двойном щелчке на значке JAR-архива откроется черное окно, которое сразу закроется, а затем запустится приложение из JAR-архива. Приложение должно быть с оконным интерфейсом, например, как в листинге 1.1, иначе результат выполнения вы не увидите.

После этих действий можно запустить JavaFX-приложение с помощью двойного щелчка мышью на значке JAR-архива TestJavaFX.jar. В результате откроется окно, показанное на рис. 1.1.

Вместо обычного приложения мы можем сделать модульное приложение. Для этого в каталоге C:\book\src\TestJavaFX создаем файл module-info.java со следующим содержимым:

```
module TestJavaFX {
   exports application;
   requires javafx.controls;
}
```

Скомпилируем программу и создадим модульный JAR-архив:

```
C:\book>javac -encoding utf-8 --module-path C:\JavaSE\JavaFX\12\lib
              -d C:\book\bin\TestJavaFX
              C:\book\src\TestJavaFX\module-info.java
              C:\book\src\TestJavaFX\application\*.java
```

```
C:\book>jar cvf C:\book\modules\TestJavaFX.jar
            -C C:\book\bin\TestJavaFX\ .
added manifest
added module-info: module-info.class
adding: application/(in = 0) (out= 0)(stored 0%)
adding: application/Main.class(in = 1036) (out= 644)(deflated 37%)
```

В результате в каталоге C:\book\modules будет создан модульный JAR-архив TestJavaFX.jar. Запустим его из командной строки:

```
C:\book>java --module-path C:\JavaSE\JavaFX\12\lib;C:\book\modules
             --module TestJavaFX/application.Main
```

В этой команде мы добавили в путь поиска модулей и каталог с JavaFX-модулями, и каталог с нашим модулем через точку с запятой. Обратите внимание: при компиляции и запуске модуля не нужно указывать опцию `--add-modules`, т. к. все зависимости от других модулей указываются в файле module-info.

Еще раз обращаю ваше внимание на то, что при использовании FXML в файле module-info нужно указывать инструкцию `opens`, а не `exports`, т. к. используется рефлексия:

```
module TestJavaFX {
   opens application;
   requires javafx.fxml;
   requires javafx.controls;
}
```

В этом примере пакет `application` открыт для всех модулей. Чтобы доступ был только для некоторых модулей, следует указать эти модули через запятую после ключевого слова `to`:

```
module TestJavaFX {
   opens application to javafx.fxml, javafx.graphics, javafx.controls;
   requires javafx.fxml;
   requires javafx.controls;
}
```

1.4. Установка и настройка редактора Eclipse

Набирать текст программы в обычном текстовом редакторе, а затем компилировать программу и запускать ее в командной строке, не очень удобно. Поэтому в дальнейшем мы воспользуемся специализированным редактором Eclipse, в котором подсвечивается код программы, выводятся подсказки и справочная информация, а также имеется возможность скомпилировать программу и сразу запустить ее нажатием всего одной кнопки.

Для загрузки редактора Eclipse переходим на страницу **http://www.eclipse.org/downloads/packages/** и скачиваем архив с программой из раздела **Eclipse IDE for Java Developers**. Распаковываем скачанный архив и копируем каталог eclipse с фай-

лами редактора в каталог C:\JavaSE. Редактор не нуждается в установке, поэтому просто переходим в каталог C:\JavaSE\eclipse и запускаем файл eclipse.exe.

Eclipse хорошо работает на Java 11, но некоторые дополнительные плагины могут работать некорректно. Если после установки плагина редактор не запускается, то нужно явным образом указать путь к более старой версии Java в настройках редактора. Чтобы узнать предпочтительную для редактора версию Java, открываем файл C:\JavaSE\eclipse\eclipse.ini и смотрим значение опции `requiredJavaVersion`. В моем случае опция имеет такое значение:

```
-Dosgi.requiredJavaVersion=1.8
```

Значение `1.8` означает, что нужна Java 8, а мы используем Java 11. Устанавливаем на компьютер требуемую версию Java и указываем путь к ней с помощью опции `-vm`. Самый простой способ: создать ярлык для eclipse.exe, отобразить **Свойства** и в поле **Объект** указать такое значение (замените фрагмент `<Путь к JRE>` реальным значением):

```
C:\JavaSE\eclipse\eclipse.exe -vm "<Путь к JRE>\bin\server\jvm.dll"
```

Пример:

```
C:\JavaSE\eclipse\eclipse.exe
-vm "C:\Program Files\Java\jre1.8.0_77\bin\server\jvm.dll"
```

> ***Примечание***
>
> Все способы указания значения подробно описаны на странице **http://wiki.eclipse.org/Eclipse.ini**.

При запуске редактор попросит указать каталог с рабочим пространством (рис. 1.2). Указываем C:\JavaSE\JavaFX\projects и нажимаем кнопку **Launch**.

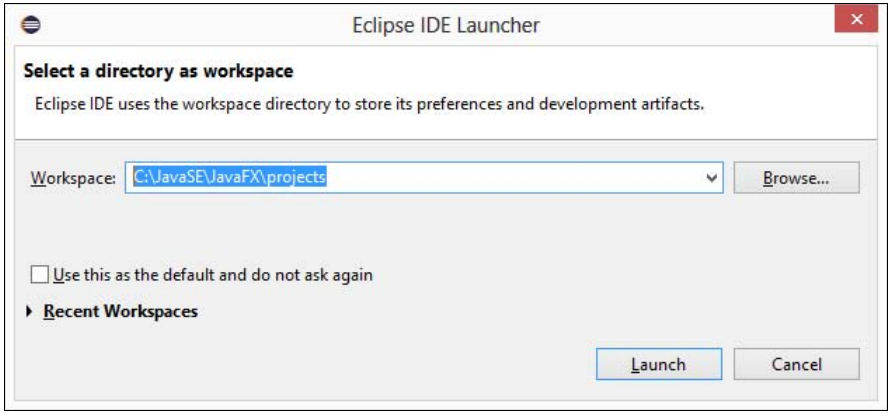

Рис. 1.2. Указание каталога с рабочим пространством при запуске редактора

Редактор Eclipse использует встроенный компилятор, а не внешний компилятор из каталога с установленным JDK. Причем доступны компиляторы практически всех версий. Для выбора версии компилятора в меню **Window** выбираем пункт

Preferences. В открывшемся окне переходим на вкладку **Java | Compiler** (рис. 1.3) и выбираем нужную версию из списка **Compiler compliance level**. Мы будем использовать Java 11, поэтому в этом списке должен быть выбран пункт **11**.

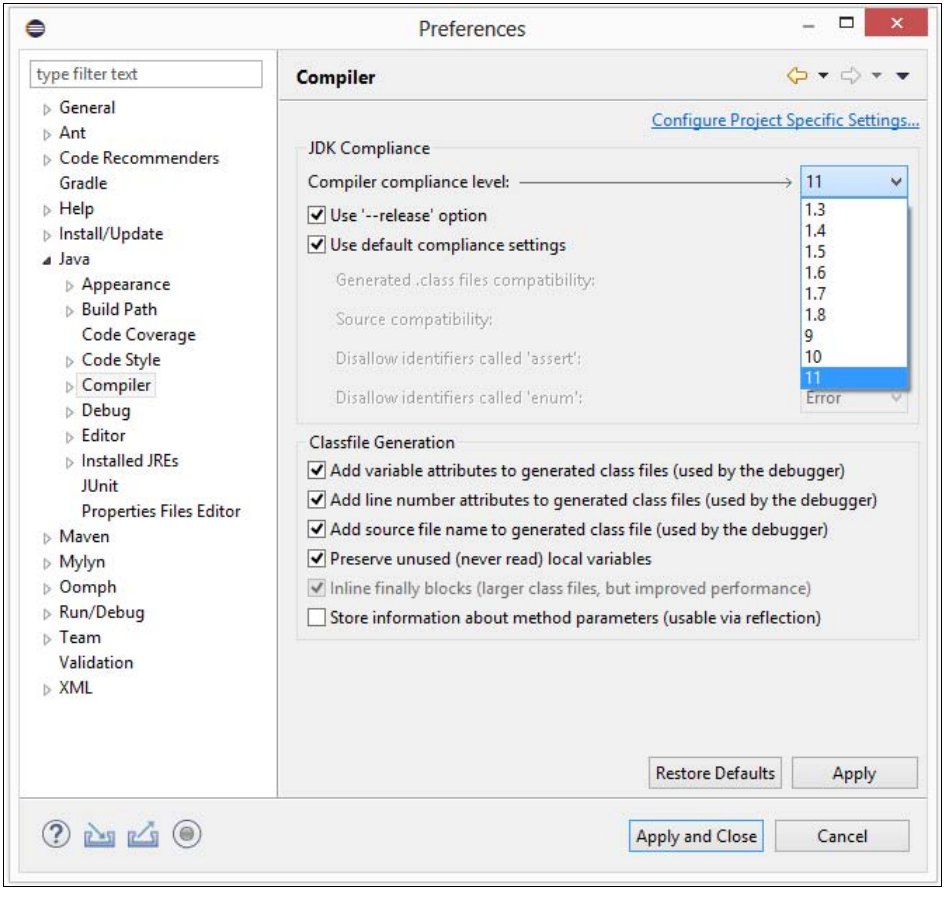

Рис. 1.3. Окно **Preferences**: вкладка **Compiler**

Если в списке **Compiler compliance level** нет пункта **11**, то в меню **Help** выбираем пункт **Eclipse Marketplace**. В открывшемся окне в поле **Find** вводим `Java 11` и нажимаем клавишу <Enter>. В списке находим пункт **Java 11 support for Eclipse** и нажимаем кнопку **Install**. После установки перезапускаем редактор, чтобы изменения вступили в силу.

Теперь добавим Java 11 в настройки редактора. Для этого в меню **Window** выбираем пункт **Preferences**. В открывшемся окне переходим на вкладку **Java | Installed JREs** (рис. 1.4). Если в списке нет пункта **jdk-11**, то нажимаем кнопку **Search** и указываем каталог C:\OpenJDK. Все найденные версии отобразятся в списке. Устанавливаем флажок **jdk-11** и сохраняем настройки.

Редактор позволяет получить справку при наведении указателя мыши на какое-либо название. Это будет работать при активном подключении к Интернету или

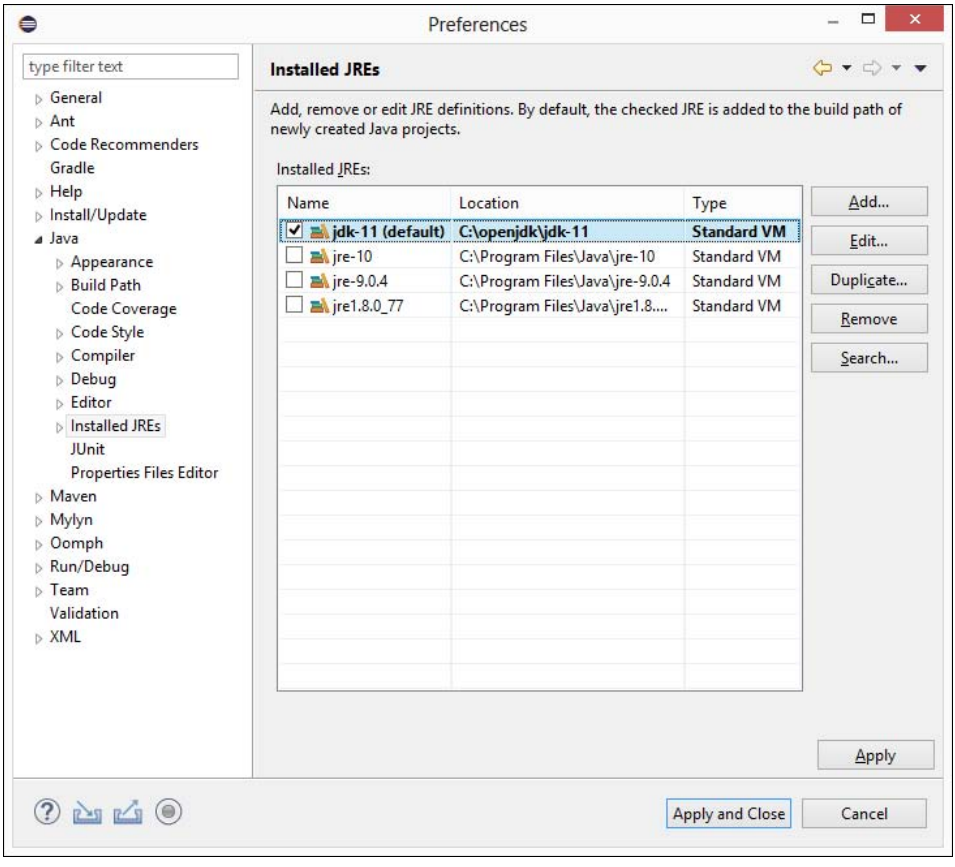

Рис. 1.4. Окно **Preferences**: вкладка **Installed JREs**

при подключении архива src.zip с исходным кодом (расположен в каталоге C:\OpenJDK\jdk-11\lib). В моем случае редактор подключил архив с исходным кодом автоматически (рис. 1.5), поэтому документацию скачивать отдельно не нужно, она входит в состав исходного кода.

Работать мы будем с кодировкой UTF-8, поэтому давайте настроим кодировку файлов по умолчанию. Для этого в меню **Window** выбираем пункт **Preferences**. В открывшемся окне переходим на вкладку **General | Workspace** (рис. 1.6). В группе **Text file encoding** устанавливаем флажок **Other** и из списка выбираем кодировку UTF-8. Сохраняем изменения. Если необходимо изменить кодировку уже открытого файла, в меню **Edit** выбираем пункт **Set Encoding**.

По умолчанию редактор вместо пробелов вставляет символы табуляции. Нас это не устраивает. Давайте изменим настройку форматирования кода. Для этого в меню **Window** выбираем пункт **Preferences**. В открывшемся окне переходим на вкладку **Java | Code Style | Formatter** (рис. 1.7). Нажимаем кнопку **New**. В открывшемся окне (рис. 1.8) в поле **Profile name** вводим название стиля, например `MyStyle`, а из списка выбираем пункт **Eclipse [built-in]**. Нажимаем кнопку **OK**. Откроется окно, в котором можно изменить настройки нашего стиля. На вкладке **Indentation**

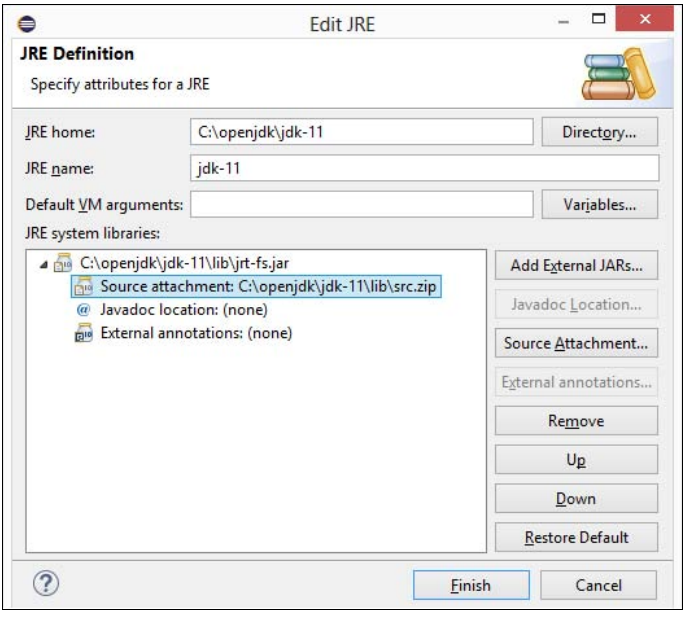

Рис. 1.5. Подключение архива с исходным кодом в окне **Edit JRE**

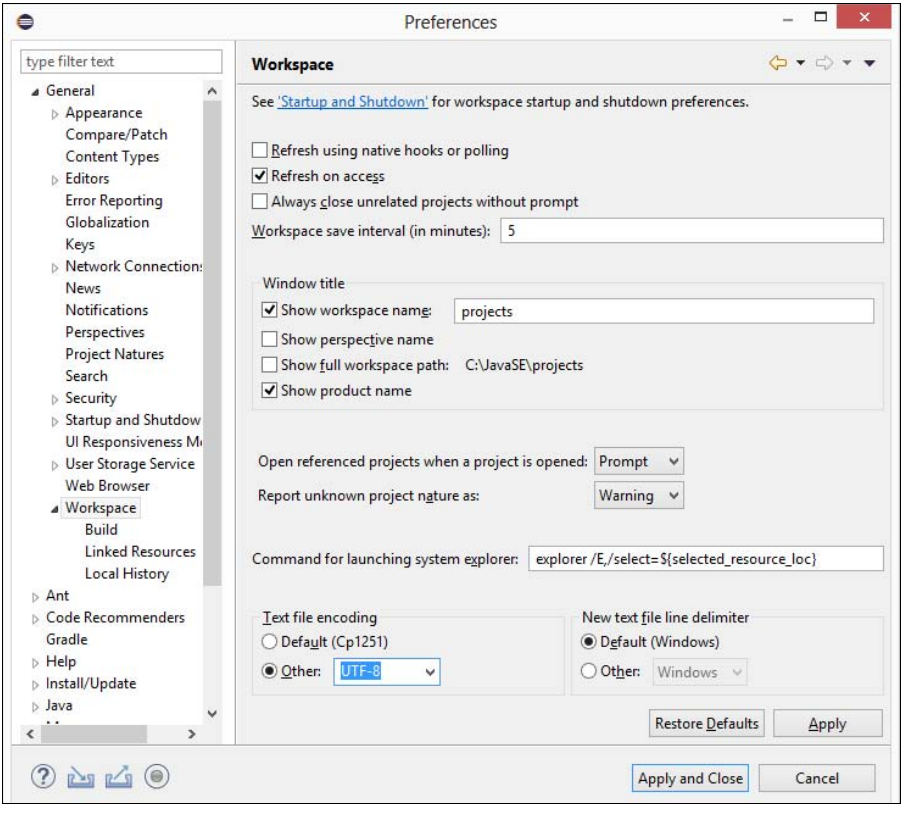

Рис. 1.6. Указание кодировки файлов

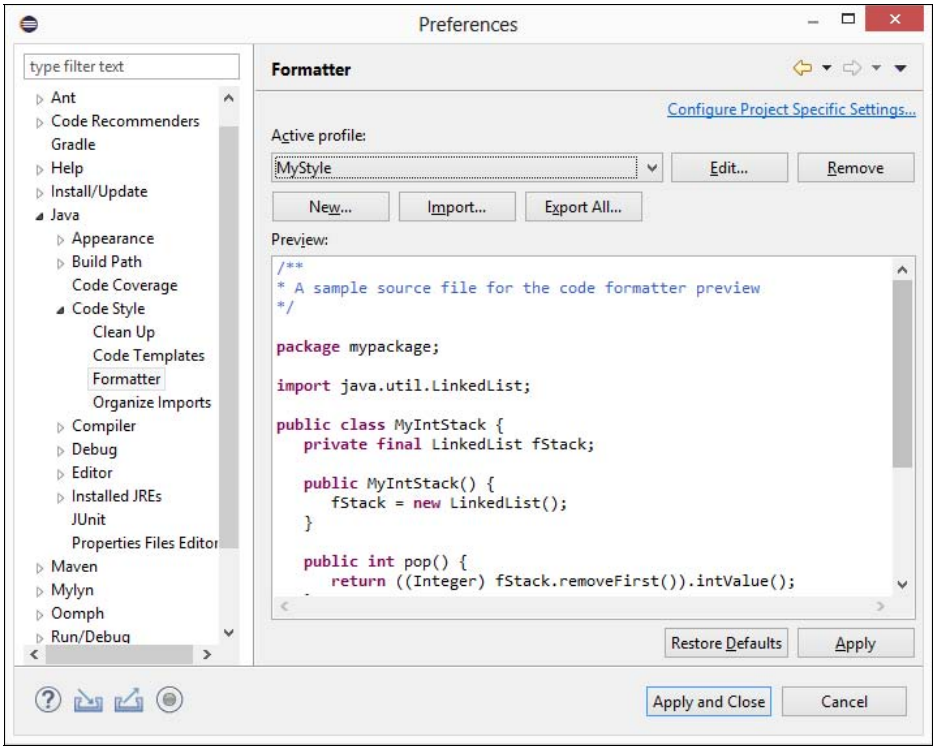

Рис. 1.7. Окно **Preferences**: вкладка **Formatter**

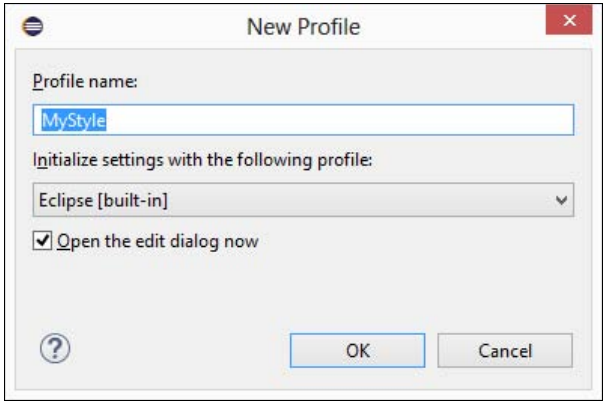

Рис. 1.8. Окно **New Profile**

из списка **Tab policy** выбираем пункт **Spaces only**, а в поля **Indentation size** и **Tab size** вводим число 3. Сохраняем все изменения.

Если необходимо изменить размер шрифта, то в меню **Window** выбираем пункт **Preferences**. В открывшемся окне переходим на вкладку **General | Appearance | Colors and Fonts** (рис. 1.9). Из списка выбираем пункт **Java | Java Editor Text Font** и нажимаем кнопку **Edit**.

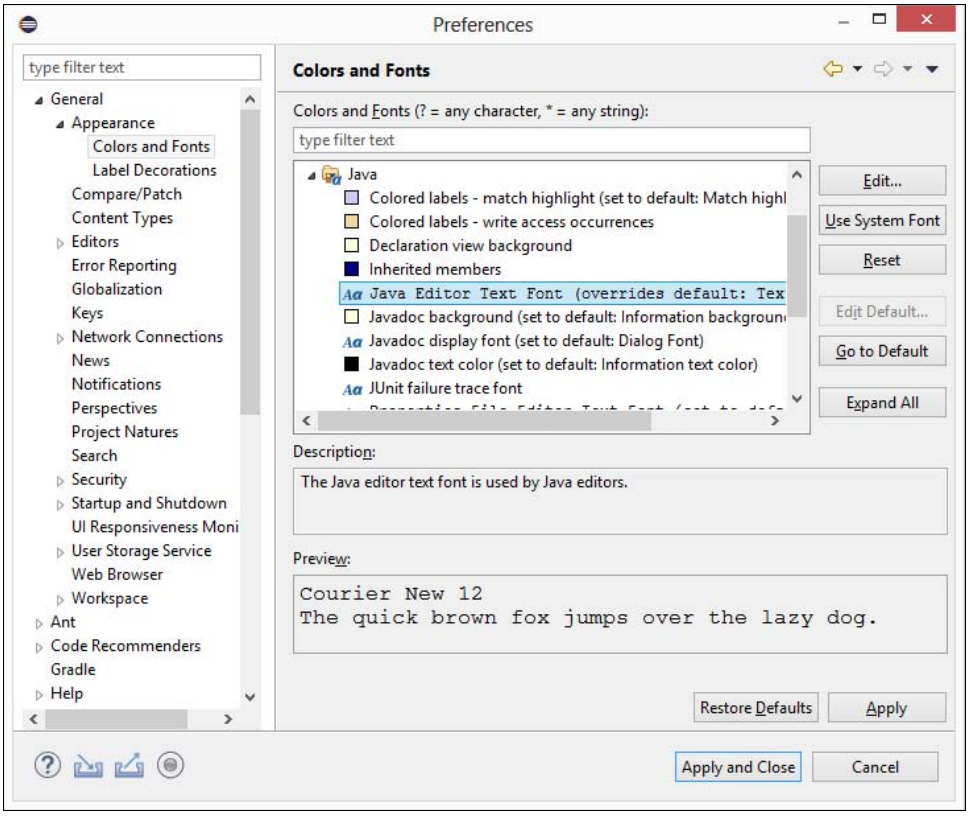

Рис. 1.9. Окно **Preferences**: вкладка **Colors and Fonts**

Если после названия объекта вставить точку, то редактор отобразит список методов и свойств. Если при этом набирать первые буквы, то содержимое списка сократится. Достаточно выбрать метод из списка, и его название будет вставлено в код. Каким бы длинным ни было название метода, мы можем его вставить в код без ошибок в его названии и очень быстро.

Редактор позволяет также закончить частично набранное слово. Для этого достаточно нажать комбинацию клавиш <Ctrl>+<Пробел>. В результате редактор автоматически закончит слово или предложит список с возможными вариантами. Причем этот способ отобразит еще и названия шаблонов кода. При выборе шаблона будет вставлено не просто слово, а целый фрагмент кода. Например, введите слово `sysout` и нажмите комбинацию клавиш <Ctrl>+<Пробел>. В результате будет вставлен следующий код:

```
System.out.println();
```

Это очень удобно и экономит много времени. Чтобы увидеть все доступные шаблоны, в меню **Window** выбираем пункт **Preferences**. В открывшемся окне переходим на вкладку **Java | Editor | Templates** (рис. 1.10). С помощью этой вкладки можно не только посмотреть все шаблоны, но и отредактировать их, а также создать свой собственный шаблон.

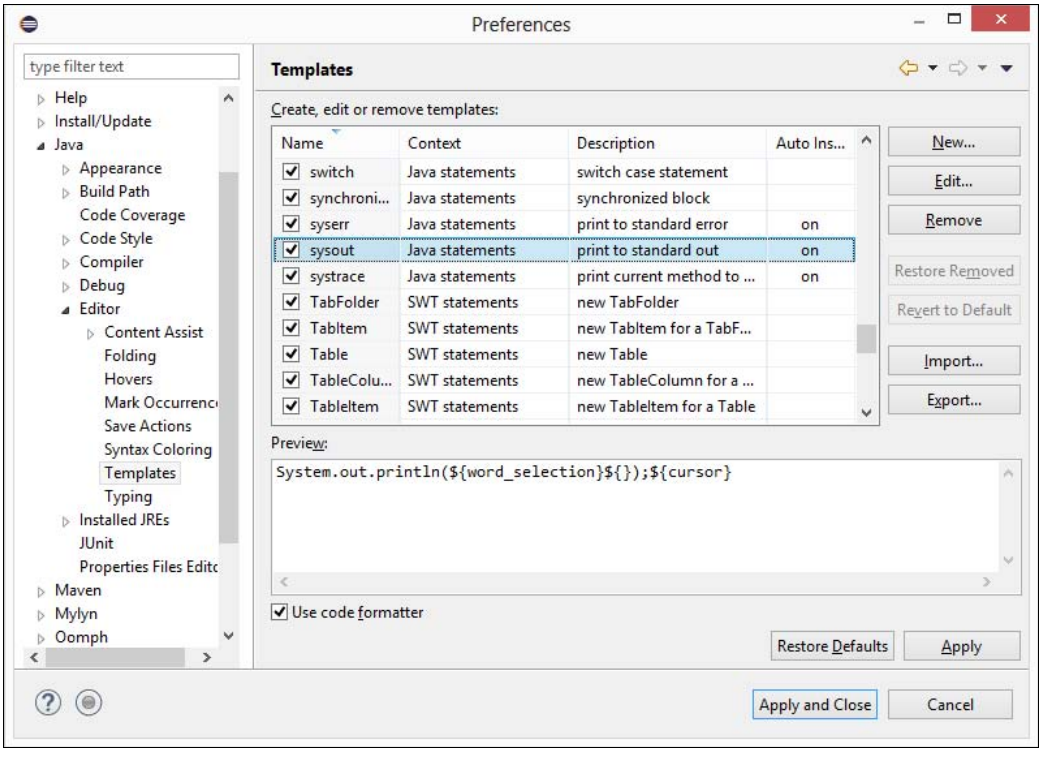

Рис. 1.10. Окно **Preferences**: вкладка **Templates**

1.5. Установка модуля *e(fx)clipse*

Чтобы создавать полноценные JavaFX-приложения с поддержкой FXML и CSS, нужно установить модуль для редактора Eclipse под названием `e(fx)clipse`. Для установки модуля (потребуется подключение к Интернету) в меню **Help** выбираем пункт **Eclipse Marketplace**. В открывшемся окне (рис. 1.11) в поле для поиска вводим `e(fx)clipse`. Нажимаем клавишу <Enter>. Находим пункт в списке и нажимаем кнопку **Install**. Отобразится окно (рис. 1.12), в котором будут указаны устанавливаемые компоненты. Соглашаемся с лицензионным соглашением и нажимаем кнопку **Finish**. После установки перезапускаем редактор Eclipse, чтобы изменения вступили в силу.

> ***Примечание***
>
> Подробную информацию о модуле `e(fx)clipse` можно получить на странице **http://www.eclipse.org/efxclipse/index.html**.

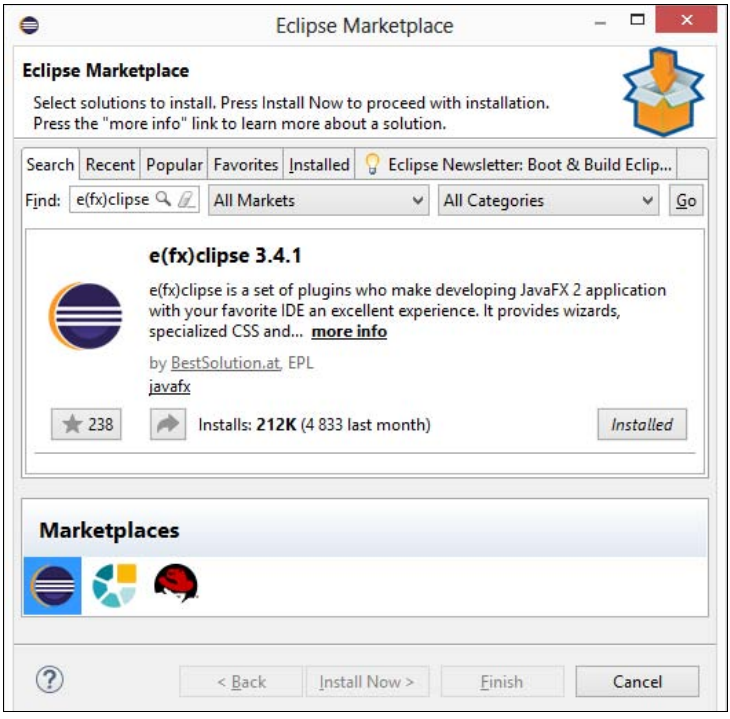

Рис. 1.11. Окно **Eclipse Marketplace**

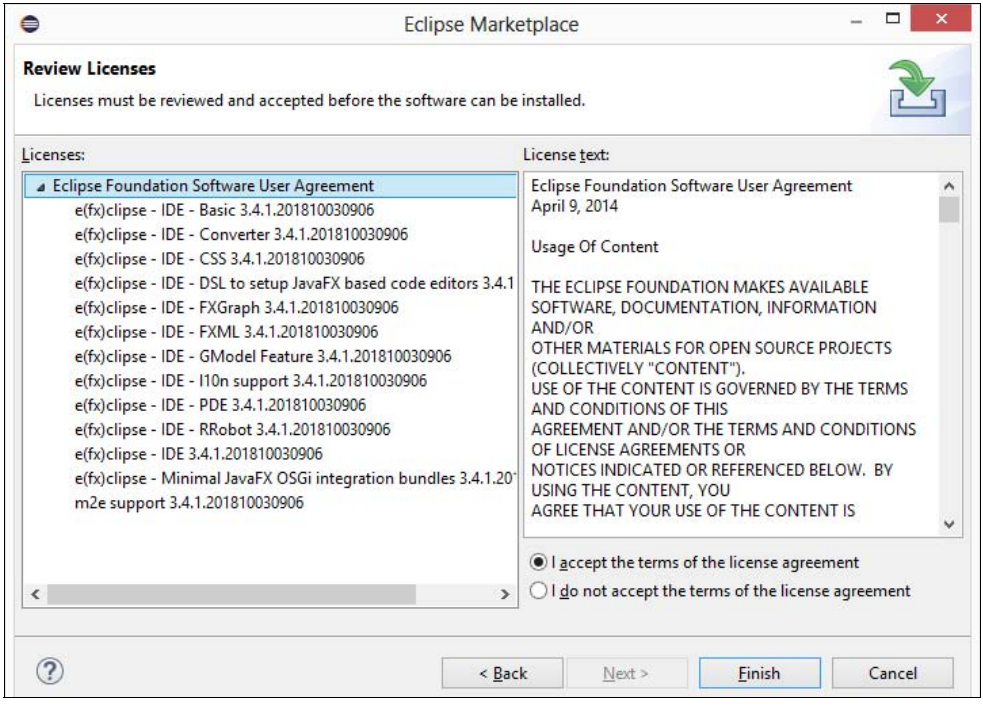

Рис. 1.12. Список устанавливаемых компонентов модуля `e(fx)clipse`

1.6. Установка программы Scene Builder

Если вы ранее пользовались Visual Studio или Delphi, то вспомните, что размещение компонентов на форме производили с помощью мыши. Щелкали левой кнопкой мыши на соответствующей кнопке на панели инструментов и перетаскивали компонент на форму. Далее с помощью инспектора свойств производили настройку значений некоторых свойств, а остальные свойства получали значения по умолчанию. При этом весь код генерировался автоматически. Произвести аналогичную операцию в JavaFX позволяет программа Scene Builder.

Скачать дистрибутив программы Scene Builder можно со страницы **https://gluonhq.com/products/scene-builder/**. Находим раздел **Download Scene Builder for Java 10** и нажимаем кнопку **Download** у пункта **Windows Installer**. Скачиваем программу установки и запускаем ее. В открывшемся окне (рис. 1.13) соглашаемся с лицензией и нажимаем кнопку **Next**. В следующем окне (рис. 1.14) указываем путь к каталогу C:\JavaSE\JavaFX\SceneBuilder и нажимаем кнопку **Next** для начала установки.

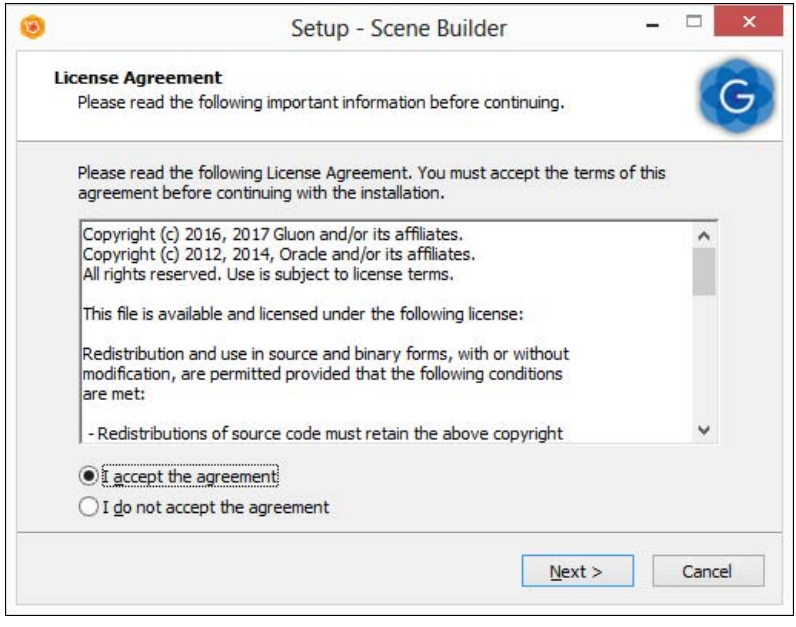

Рис. 1.13. Установка программы Scene Builder: соглашаемся с лицензией

Для запуска программы используется файл SceneBuilder.exe, расположенный в каталоге C:\JavaSE\JavaFX\SceneBuilder. Главное окно программы Scene Builder показано на рис. 1.15.

Теперь подключим программу Scene Builder к модулю `e(fx)clipse`. В редакторе Eclipse в меню **Window** выбираем пункт **Preferences**. В открывшемся окне (рис. 1.16) из списка слева выбираем пункт **JavaFX** и на вкладке справа нажимаем кнопку **Browse**. Находим файл C:\JavaSE\JavaFX\SceneBuilder\SceneBuilder.exe. В итоге

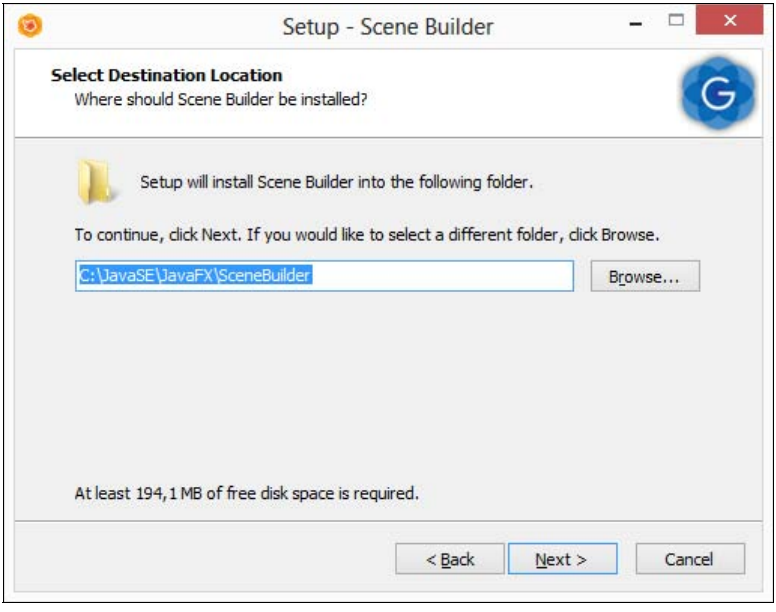

Рис. 1.14. Установка программы Scene Builder:
указываем путь к каталогу с программой

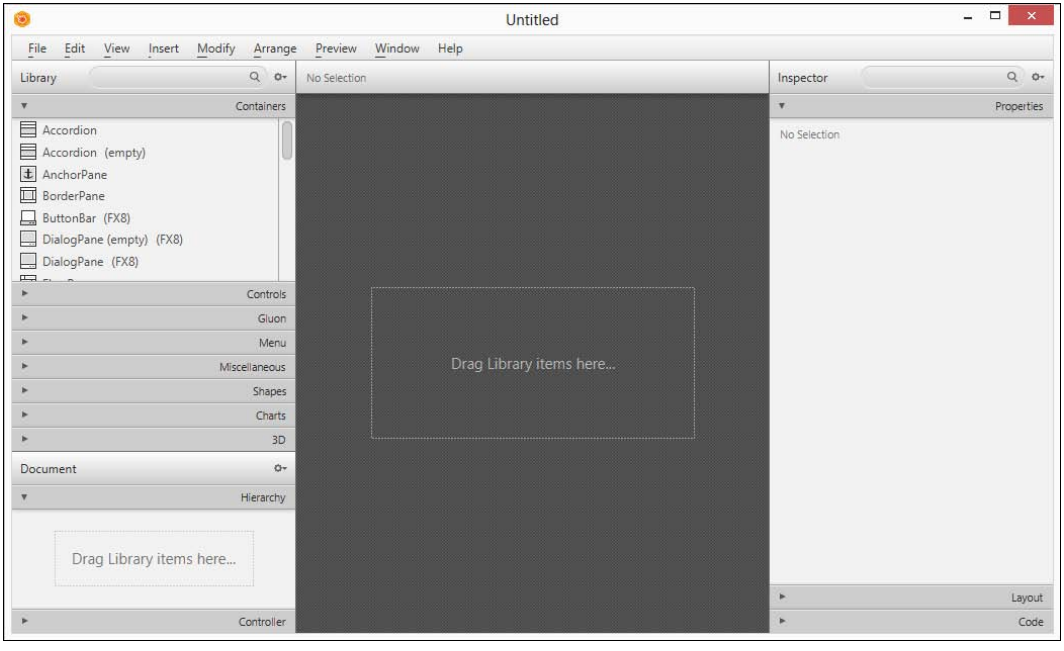

Рис. 1.15. Главное окно программы Scene Builder

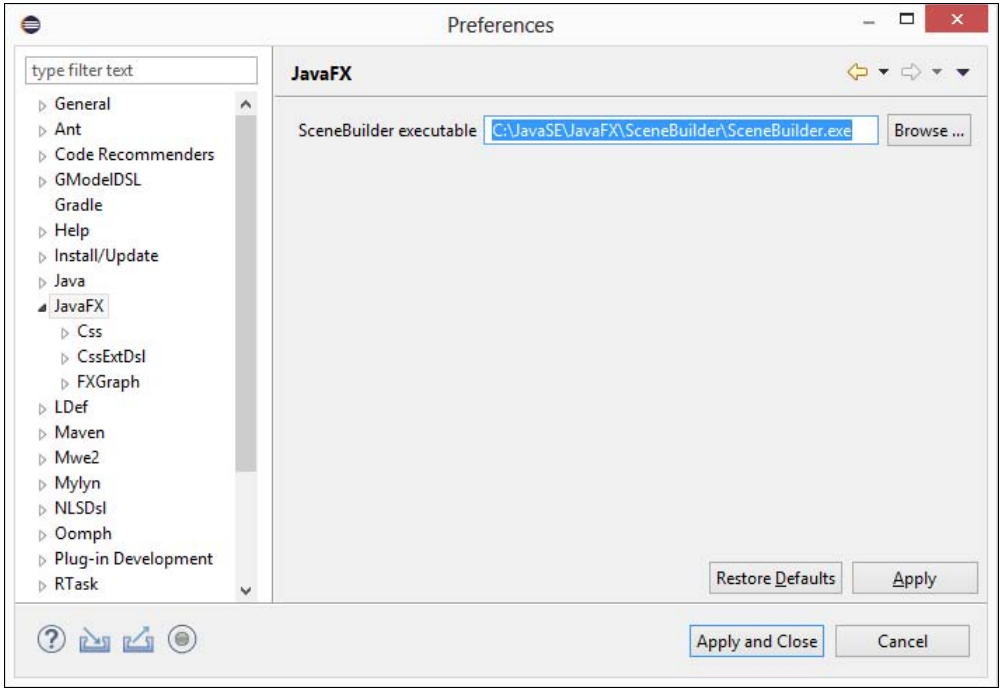

Рис. 1.16. Подключение программы Scene Builder к модулю `e(fx)clipse`

путь к программе отобразится в поле **SceneBuilder executable**. Сохраняем изменения.

1.7. Создание JavaFX-приложения в Eclipse. Способ 1

Создать JavaFX-приложение можно несколькими способами. Первый способ совпадает с созданием обычного Java-приложения. Давайте попробуем создать Java-приложение в редакторе Eclipse. Вначале создадим проект. Для этого в меню **File** выбираем пункт **New | Java Project**. В открывшемся окне (рис. 1.17) в поле **Project name** вводим `FX_CH01_01_Start`, выбираем версию JRE и нажимаем кнопку **Finish**. Если редактор предложит создать файл module-info, то отказываемся. Далее создадим пакет. Для этого в меню **File** выбираем пункт **New | Package**. В открывшемся окне (рис. 1.18) в поле **Name** вводим `application` и нажимаем кнопку **Finish**. Теперь можно добавить класс в пакет. Для этого в меню **File** выбираем пункт **New | Class**. В открывшемся окне (рис. 1.19) в поле **Name** вводим `Main`, в поле **Package** — `application`, а затем нажимаем кнопку **Finish**.

В результате в каталоге C:\JavaSE\JavaFX\projects\FX_CH01_01_Start\src\application будет создан файл Main.java и открыт на редактирование на отдельной вкладке. Добавляем в этот файл код из листинга 1.1.

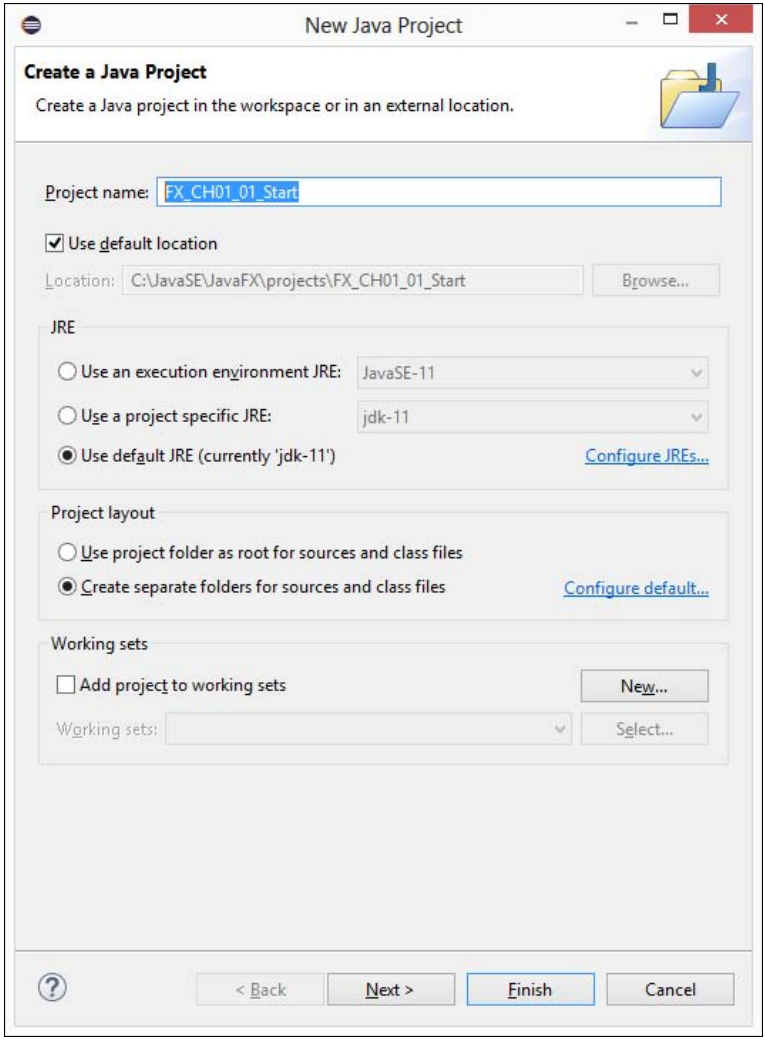

Рис. 1.17. Создание проекта

После вставки кода редактор подчеркнет красной волнистой линией все классы и пакеты, входящие в состав библиотеки JavaFX, т. к. пути к библиотеке редактор не знает. Чтобы подключить библиотеку к проекту, в окне **Package Explorer** щелкаем правой кнопкой мыши на названии проекта и из контекстного меню (рис. 1.20) выбираем пункт **Properties**. Запомните этот способ отображения свойств проекта, т. к. в дальнейшем мы будем просто говорить «отобразите свойства проекта» или «в свойствах проекта» без явного указания на то, как это сделать.

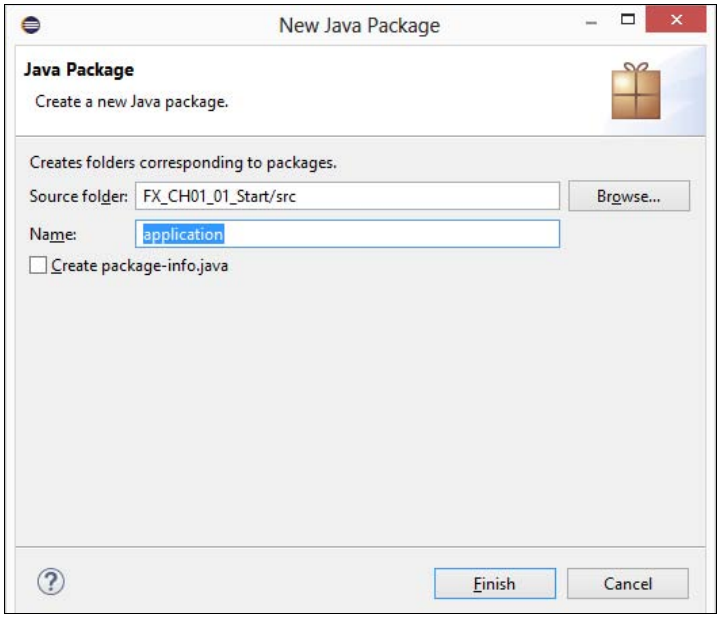

Рис. 1.18. Создание пакета

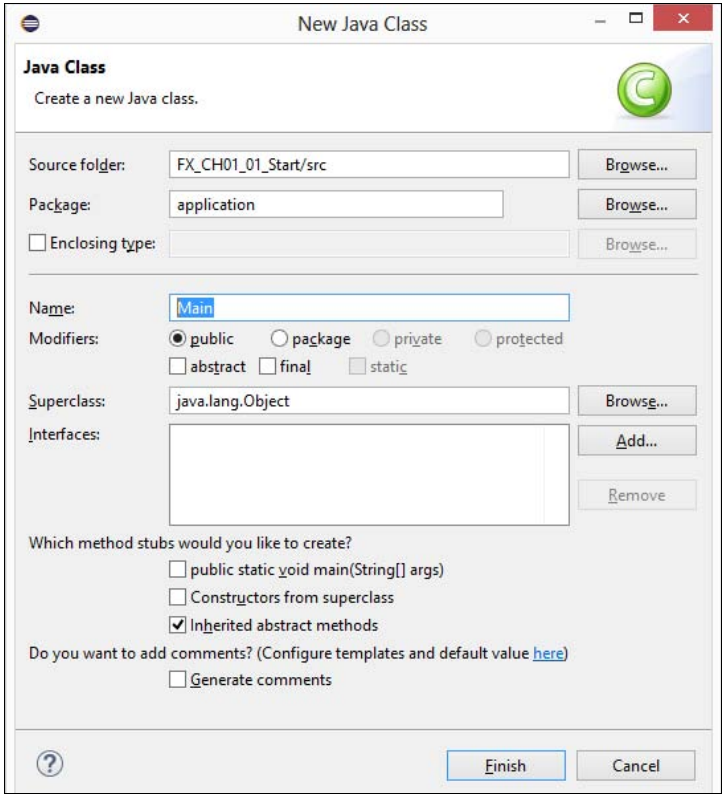

Рис. 1.19. Создание класса

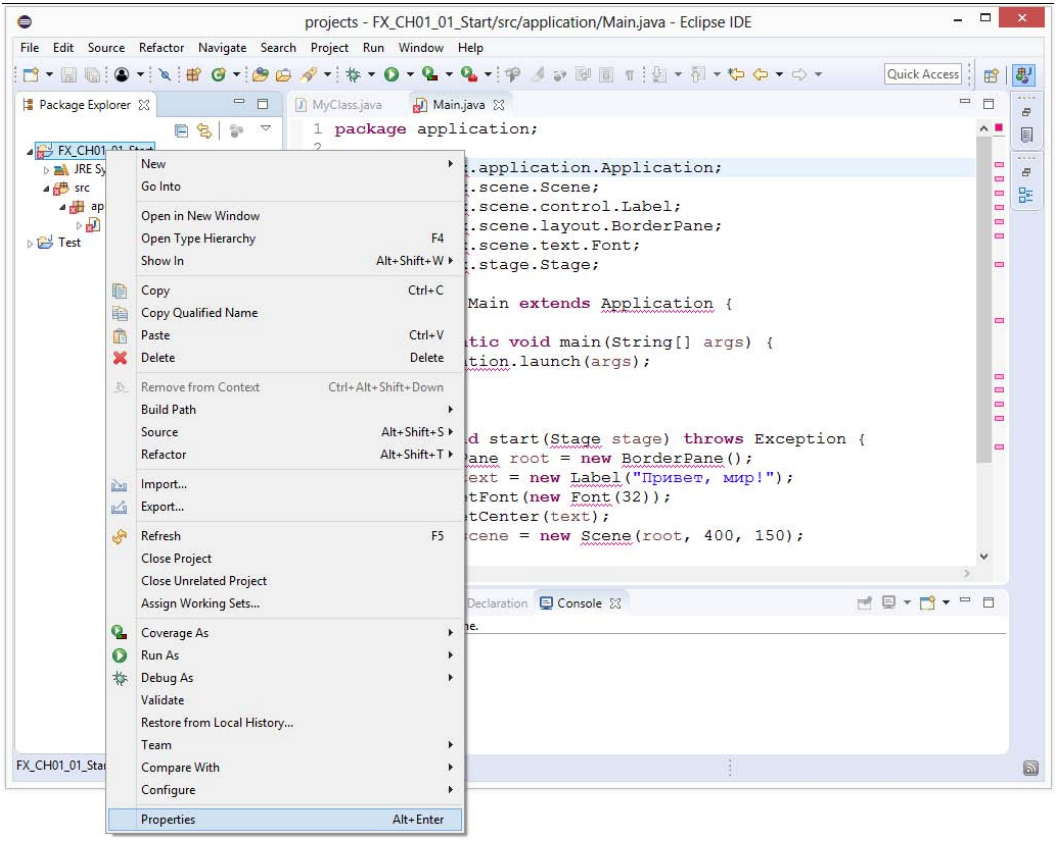

Рис. 1.20. Контекстное меню проекта

В результате откроется окно **Properties for FX_CH01_01_Start** (рис. 1.21). На вкладке **Resource** можно указать кодировку файлов проекта, а также символ-разделитель строк. По умолчанию эти два параметра наследуются из основных настроек редактора. Советую всегда явным образом задавать эти два параметра, иначе при смене кодировки по умолчанию редактор автоматически преобразует кодировку файлов проекта, что может привести к потере данных.

Для подключения библиотеки JavaFX в списке слева выбираем пункт **Java Build Path**. Далее отображаем вкладку **Libraries** и выделяем пункт **Modulepath**. Нажимаем кнопку **Add External JARs**. В открывшемся окне (рис. 1.22) переходим в каталог C:\JavaSE\JavaFX\12\lib, выделяем все модули (или только нужные для проекта) и нажимаем кнопку **Открыть**. Выбранные модули будут добавлены в список **Modulepath**. Чтобы иметь возможность получать справку при наведении указателя мыши на метод, класс и другие элементы, нужно подключить файл C:\JavaSE\JavaFX\12\lib\src.zip к каждому модулю (рис. 1.23). Нажимаем кнопку **Apple and Close** для сохранения свойств проекта. Все подчеркивания в коде должны исчезнуть, а при наведении указателя мыши на любой класс должна отобразиться справочная информация. Структура проекта с подключенными модулями в окне **Package Explorer** показана на рис. 1.24.

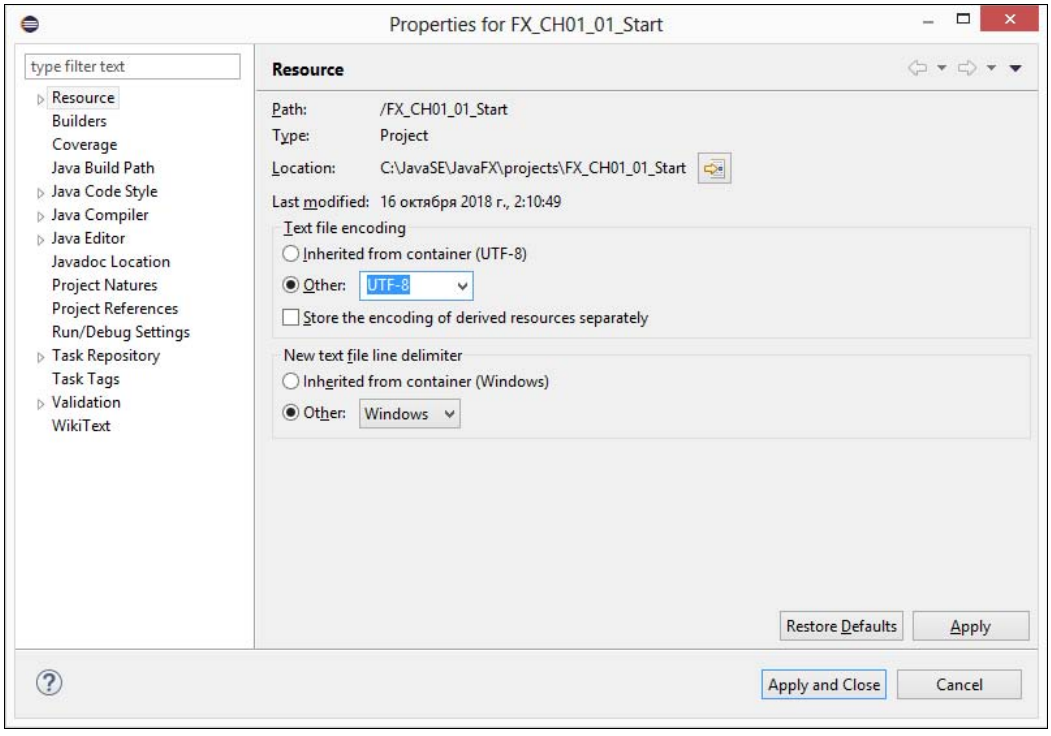

Рис. 1.21. Свойства проекта

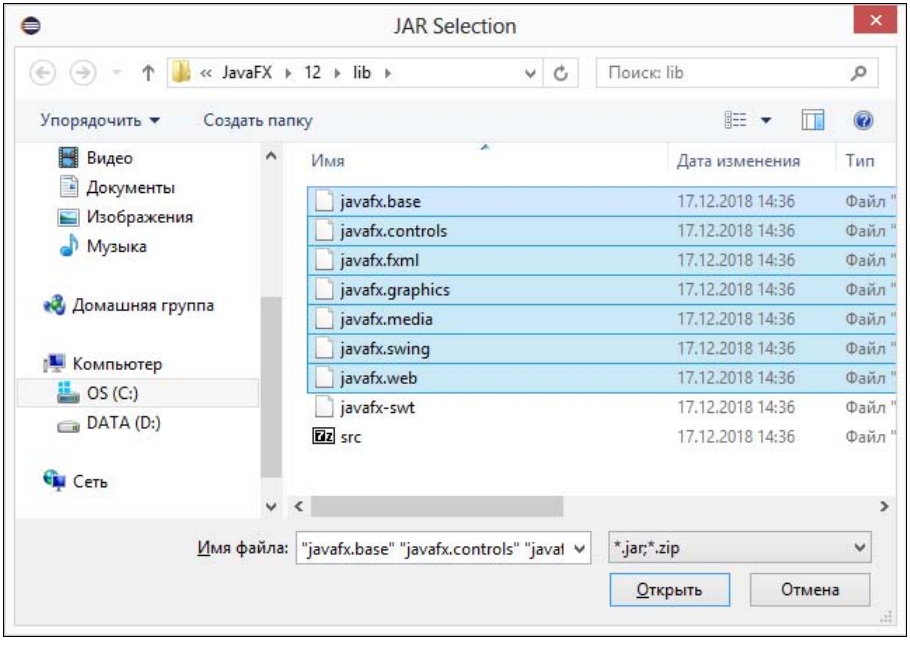

Рис. 1.22. Выбор подключаемых модулей

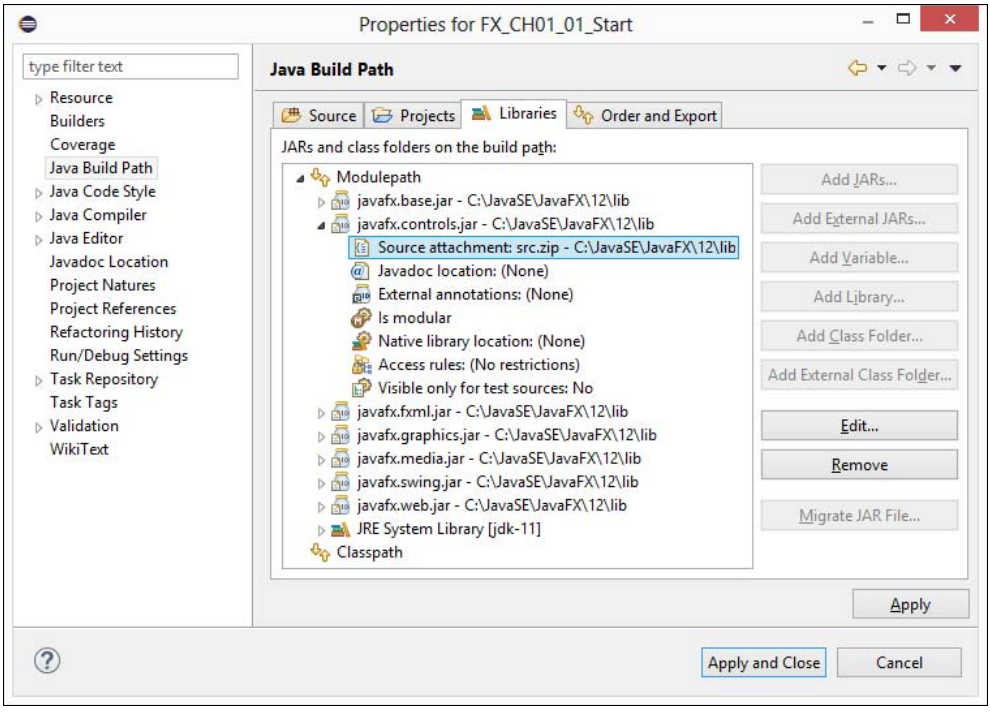

Рис. 1.23. Подключенные модули в списке **Modulepath**

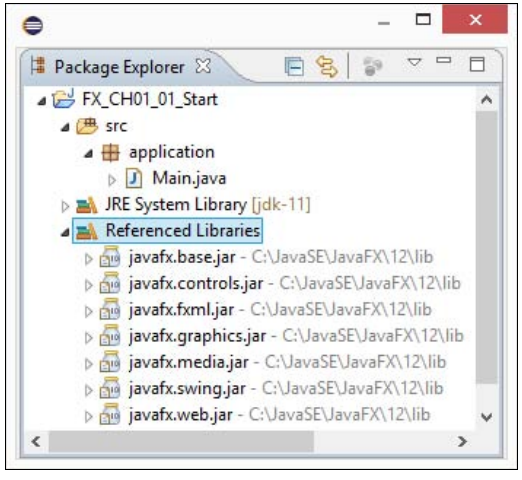

Рис. 1.24. Структура проекта с подключенными модулями в окне **Package Explorer**

Запуск JavaFX-приложения осуществляется точно так же, как и запуск обычного Java-приложения: в меню **Run** выбираем пункт **Run**, или нажимаем комбинацию клавиш <Ctrl>+<F11>, или нажимаем кнопку на панели инструментов. Однако сейчас после запуска мы получим следующее сообщение об ошибке:

```
Error: Could not find or load main class application.Main
Caused by: java.lang.NoClassDefFoundError: javafx/application/Application
```

На самом деле программа успешно скомпилировалась. Если не верите, то посмотрите содержимое каталога C:\JavaSE\JavaFX\projects\FX_CH01_01_Start\bin\application и убедитесь, что файл с байт-кодом успешно создан. Мы даже можем запустить его из командной строки:

```
C:\Users\Unicross>cd C:\JavaSE\JavaFX\projects\FX_CH01_01_Start\bin

C:\JavaSE\JavaFX\projects\FX_CH01_01_Start\bin>
          java --module-path C:\JavaSE\JavaFX\12\lib
               --add-modules javafx.controls application.Main
```

В результате отобразится окно, показанное на рис. 1.1. Как видите, все прекрасно работает из командной строки. Но чтобы программа запускалась из редактора, нужно либо добавить в проект файл module-info и прописать в нем используемые модули, либо в настройках запуска добавить опцию `--add-modules`. Давайте попробуем оба эти способа.

Начнем с создания модульного приложения. Чтобы добавить файл module-info в проект, в окне **Package Explorer** щелкаем правой кнопкой мыши на названии проекта и из контекстного меню выбираем пункт **Configure | Create module-info.java**. В открывшемся окне (рис. 1.25) в поле **Module name** вводим название модуля, например `mymodule`, и нажимаем кнопку **Create**. Файл module-info.java будет добавлен в пакет по умолчанию и станет доступен для редактирования на отдельной вкладке. В файл module-info.java редактор вставит следующий код:

```
module mymodule {
   exports application;

   requires javafx.base;
   requires javafx.controls;
   requires javafx.graphics;
}
```

Как видите, все зависимости от модулей редактор добавил самостоятельно. Зная зависимости модуля `javafx.controls`, мы можем сократить этот код и дополнительно определить транзитивную зависимость (иначе редактор будет подчеркивать класс `Stage` желтой волнистой линией и предлагать добавить перед классом аннотацию `@SuppressWarnings("exports")`):

```
module mymodule {
   exports application;
   requires transitive javafx.controls;
}
```

Если мы сейчас запустим приложение, то получим окно, показанное на рис. 1.1.

Теперь рассмотрим второй способ, которым будем пользоваться в дальнейших примерах. Сначала удаляем файл module-info.java из проекта. Для этого в окне **Package Explorer** щелкаем правой кнопкой мыши на названии файла и из контекстного меню выбираем пункт **Delete**. В открывшемся окне подтверждаем удаление

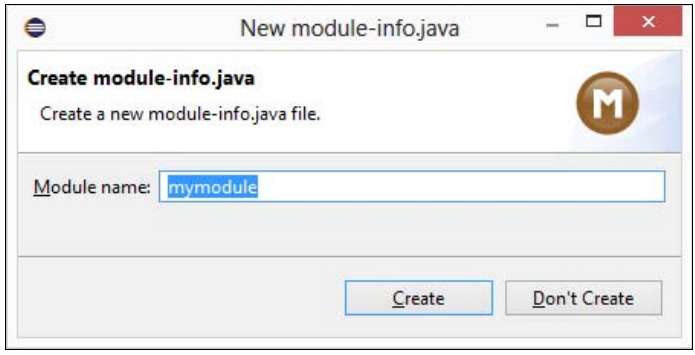

Рис. 1.25. Создание модуля

файла. Далее нам нужно добавить опцию `--add-modules` в командную строку для запуска приложения. Для этого открываем свойства проекта и в списке слева выбираем пункт **Run/Debug Settings** (рис. 1.26). Выделяем пункт с названием конфигурации запуска и нажимаем кнопку **Edit**. В открывшемся окне (рис. 1.27) переходим на вкладку **Arguments** и в поле **VM arguments** вводим следующую команду:

```
--add-modules ALL-MODULE-PATH
```

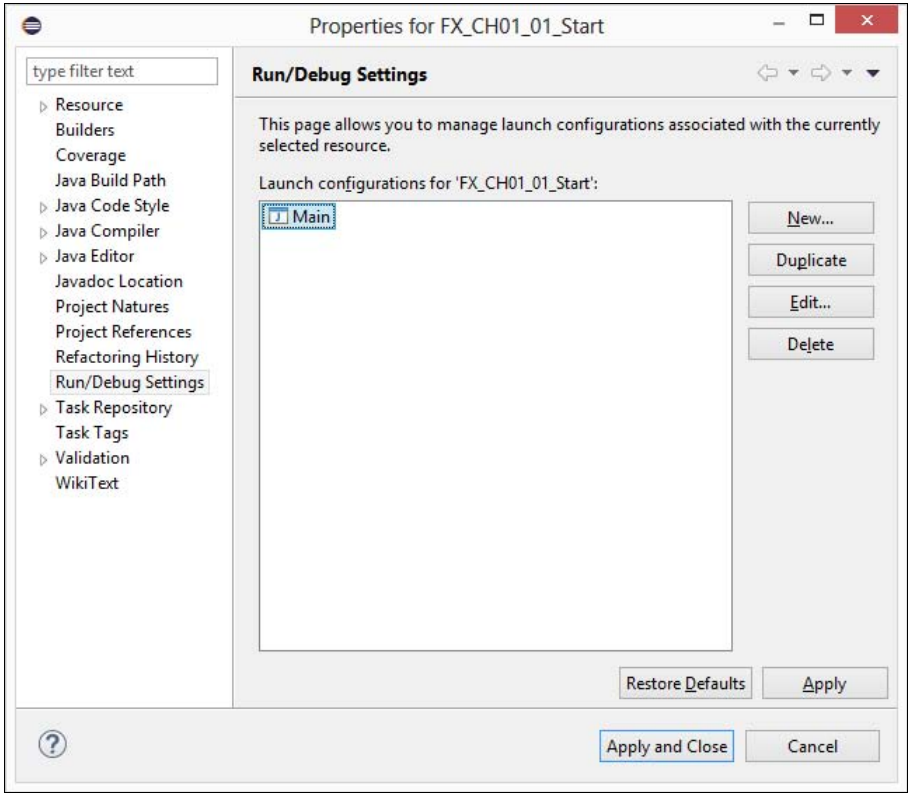

Рис. 1.26. Свойства проекта: вкладка **Run/Debug Settings**