ТИМУР МАШНИН

bhv®

СОВРЕМЕННЫЕ Java ТЕХНОЛОГИИ НА ПРАКТИКЕ

- ОБЗОР JAVA-ТЕХНОЛОГИЙ И ОБЛАСТЕЙ ИХ ПРАКТИЧЕСКОГО ПРИМЕНЕНИЯ
- БИБЛИОТЕКИ, СПЕЦИФИКАЦИИ, СЕРВИСЫ
- АРХИТЕКТУРА ПЛАТФОРМ JAVA SE, JAVA ME И JAVA EE
- РАЗРАБОТКА АППЛЕТОВ, НАСТОЛЬНЫХ, МОДУЛЬНЫХ И РАСПРЕДЕЛЕННЫХ JAVA-ПРИЛОЖЕНИЙ

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

+CD

Тимур Машнин

Современные Java технологии на практике

Санкт-Петербург
«БХВ-Петербург»
2010

УДК 681.3.06
ББК 32.973.26-018.2
М38

Машнин Т. С.

М38 Современные Java-технологии на практике. — СПб.: БХВ-Петербург, 2010. — 560 с.: ил. + CD-ROM — (Профессиональное программирование)

ISBN 978-5-9775-0561-1

Рассмотрено создание широкого круга Java-приложений с помощью современных Java-технологий и среды разработки NetBeans. Подробно рассмотрена архитектура платформ Java SE, Java ME и Java EE. Показано создание апплетов с использованием графических библиотек AWT и Swing, настольных приложений на основе платформы Swing Application Framework, а также расширяемых Java-приложений с использованием библиотек ServiceLoader API, Lookup и др. для платформы Java SE. Рассмотрено создание мобильных приложений на основе конфигурации CLDC и профиля MIDP для платформы Java ME. Показано применение технологий Java Servlet, JavaServer Pages, JavaServer Faces, Web-сервисов, Enterprise JavaBeans и др. при программировании для платформы Java EE. Материал книги сопровождается большим количеством примеров с подробным анализом исходных кодов. На компакт-диске находятся проекты примеров приложений.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.05.10.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 45,15.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0561-1

Оглавление

Введение

Что такое технология Java?

Начнем с самого понятия технологии программирования. Можно сказать, что *технология программирования* — это совокупность методов и инструментов, позволяющих создавать программное обеспечение. Технологии программирования могут иметь различный уровень применения. В процессе разработки программного обеспечения могут применяться технологии, решающие как конкретные задачи, так и технологии, являющиеся платформой для создания частей приложения или всего приложения. Поэтому, как правило, для создания программного обеспечения применяется целый набор различных технологий.

Применительно к Java, *технология Java* — это язык программирования Java и платформа Java, разрабатываемые компанией Sun Microsystems, Inc. Помимо компании Sun разработкой продуктов Java занимается сообщество Java, объединяющее как компании, так и независимых разработчиков.

Язык программирования Java представляет собой объектно-ориентированный язык программирования, имеющий синтаксис, близкий к синтаксису языка C++. Отличия языка Java от языка C++ обусловлены самим происхождением этих языков программирования. Язык C++ является расширением языка C, который создавался как язык системного программирования. Язык Java, в свою очередь, создавался для решения задач сетевого программирования и является самостоятельным языком программирования. Главные отличия языка Java от языка C++ — это более строгая типизация, ограничения работы с памятью, автоматическая сборка мусора.

Понятно, что для создания программного обеспечения наличие одного языка программирования недостаточно. Для компилируемых языков нужны инструменты, компилирующие исходный код в машинный, исполняемый опера-

ционной системой компьютера. Для интерпретируемых языков программирования необходимы интерпретаторы, выполняющие исходный код в операционной системе. В случае языка Java, реализация платформы Java как раз и обеспечивает выполнение Java-кода в операционной системе компьютера. Таким образом, для того чтобы Java-приложение могло быть запущено, необходима реализация платформы Java.

Я упомянул реализацию платформы Java. Что это такое? *Платформа Java* состоит из виртуальной машины Java Virtual Machine (JVM) и библиотек интерфейса программирования Java Application Programming Interface (API). Для всех распространенных операционных систем существуют свои виртуальные машины JVM, тем самым реализуется принцип "Write Once, Run Anywhere" ("Написанное однажды, работает везде"). *Реализация платформы Java* — это конкретная реализация JVM для конкретной операционной системы плюс библиотеки Java API.

На самом деле, компанией Sun для выполнения Java-приложений предоставляется набор сред выполнения *Java Runtime Environment (JRE)*, охватывающий все распространенные операционные системы. Виртуальная машина JVM составляет основную часть среды выполнения JRE. Помимо JVM, JRE содержит базовые библиотеки API, необходимые для выполнения Java-приложений, а также дополнительные инструменты, включая Java Plug-in для запуска апплетов в браузере и Java Web Start для развертывания Java-приложений через Интернет.

Компанией Sun Microsystems, Inc. также предоставляется минимальный комплект разработки Java-приложений *Java Development Kit (JDK)*, состоящий из набора инструментов, включая компилятор в байт-код `javac`, документации, примеров и среды выполнения JRE.

Язык программирования Java является одновременно и интерпретируемым, и компилируемым. Причина этого кроется в устройстве виртуальной машины JVM.

Виртуальная машина JVM — это набор специальных программ, созданных для конкретной операционной системы. Точкой входа в виртуальную машину JVM является программа `java`, запускающая Java-приложение. Приложения, написанные на языке Java, представляют собой текстовые файлы с расширением java. Чтобы JVM выполнила Java-приложение, приложение должно быть откомпилировано в специальный двоичный формат — *байт-код*. Откомпилированное Java-приложение состоит из файлов с расширением class, которые могут быть упакованы в архивный исполняемый файл с расширением jar. При запуске Java-приложения на вход JVM подается байт-код Java-приложения, а также байт-код используемых приложением библиотек Java API.

Виртуальная машина JVM может выполнять приложения, написанные и на других языках программирования — Ruby, PHP, JavaScript, Python и др., при этом приложения также должны быть откомпилированы в байт-код.

В процессе обработки байт-кода виртуальная машина JVM производит его интерпретацию, т. е. выполняет команды, содержащиеся в байт-коде, или использует компилятор Just-in-time compilation (JIT), который транслирует байт-код в машинный код непосредственно во время выполнения Java-приложения, и тем самым увеличивает скорость обработки байт-кода.

Таким образом, язык Java является компилируемым, потому что необходима компиляция исходного кода в промежуточный по отношению к машинному байт-код, и интерпретируемым, потому что байт-код не может быть исполнен самой операционной системой компьютера, а должен интерпретироваться.

Платформа Java содержит два типа JVM:

- *Java HotSpot Client VM (Client VM).* Вызывается опцией `-client` инструмента `java` и обеспечивает быстрый запуск и потребление небольшого объема оперативной памяти;
- *Java HotSpot Server VM (Server VM).* Вызывается опцией `-server` инструмента `java` и обеспечивает максимальную скорость выполнения приложения.

Для обеих JVM технология Java HotSpot оптимизирует обработку байт-кода, распределение памяти, сборку мусора и управление потоками.

Технология Java — это общее понятие, на самом деле обозначающее широкий спектр Java-технологий. Для технологий Java существует определенный стандарт. Каждая Java-технология состоит из спецификации, стандартной реализации *Reference Implementation (RI)*, набора автоматических тестов *Technology Compatibility Kit (TCK)* на соответствие реализаций спецификации и, как правило, оптимизированной реализации.

> ***Примечание***
>
> Для того чтобы упорядочить процесс разработки спецификаций технологии Java, в 1998 г. был организован формальный процесс разработки *Java Community Process (JCP)*, состоящий из запроса на разработку спецификации, который должен быть одобрен Исполнительным Комитетом, варианта, разрабатываемого группой экспертов и обсуждаемого сообществом, варианта для создания эталонной реализации Reference Implementation (RI) и набора автоматических тестов Technology Compatibility Kit (TCK), финальной реализации и поддержки спецификации.

Архитектура технологии Java

Среда выполнения JRE и комплект разработки JDK являются основными продуктами платформы Java Platform, Standard Edition (Java SE). Подробно структуру JRE и JDK можно посмотреть в *приложении 1*.

Как уже было сказано, платформа Java содержит библиотеки интерфейса программирования Java API. Для чего они предназначены и какую роль они играют?

Библиотеки Java API — это готовые классы и интерфейсы, обеспечивающие для создаваемых Java-приложений общую функциональность. С библиотеками Java API программисту не нужно самому реализовывать ввод/вывод, сетевое соединение, создавать стандартные графические компоненты для интерфейса пользователя и многое другое. Все это уже предоставлено технологией Java.

Интерфейс программирования платформы Java SE представляет собой следующий набор библиотек:

- ❒ базовые библиотеки платформы;
- ❒ интегрированные библиотеки платформы;
- ❒ библиотеки интерфейса пользователя.

Подробная структура Java API платформы Java SE представлена в *приложении 2*.

Платформа Java SE является основой для всех остальных платформ технологии Java. Все вместе Java-платформы обеспечивают применение технологии Java к широкому диапазону устройств — от смарт-карт, встроенных и мобильных устройств до серверов и суперкомпьютеров.

Технология Java представлена следующими платформами:

- ❒ *Java Platform, Standard Edition (Java SE)* — предоставляет среду выполнения и набор технологий и библиотек API для создания и запуска серверных и настольных приложений, апплетов и является основой для остальных платформ. Кроссплатформенность обеспечивается наличием сред выполнения для различных операционных систем. Платформа Java SE включает в себя следующие компоненты: среду выполнения Java Runtime Environment (JRE) и комплект разработчика приложений Java Development Kit (JDK);
- ❒ *Java SE for Embedded* — предназначена для встроенных систем, таких как интеллектуальные маршрутизаторы и коммутаторы, профессиональные принтеры и др. Платформа Java SE for Embedded обеспечивает ту же функциональность, что и платформа Java SE, дополнительно добавляя поддержку платформ, специфических для встроенных систем, оптимиза-

цию использования памяти, а также предоставляя уменьшенную среду выполнения Small Footprint JRE и опцию Headless для устройств, не имеющих дисплея, мыши или клавиатуры;

- *Sun Java Real-Time System (Java RTS)* — коммерческое расширение платформы Java SE, позволяющее увеличить временной контроль над выполнением Java-приложений, используемых в таких областях, как авиакосмическая промышленность, телекоммуникации, научные исследования и др. Платформа Java RTS предоставляет дополнительно новые потоки реального времени, диспетчеризацию и синхронизацию, схему управления памятью, механизмы асинхронной коммуникации, установку времени с высоким разрешением, прямой доступ к физической памяти;
- *Java Platform, Micro Edition (Java ME)* — содержит набор сред выполнения и библиотек API, предназначенный для встроенных и мобильных устройств, таких как мобильные телефоны и карманные персональные компьютеры;
- *Java Card* — позволяет создавать и запускать небольшие приложения (аплеты Java Card) в смарт-картах и других устройствах с очень ограниченными ресурсами, таких как SIM-карты мобильных телефонов, банковские карточки, карты доступа цифрового телевидения и др.;
- *Java Platform, Enterprise Edition (Java EE)* — является расширением платформы Java SE и добавляет библиотеки, позволяющие создавать распределенные, многоуровневые серверные Java-приложения.

Как разрабатываются приложения Java?

Применение грамотной методологии при разработке приложения необходимо для сокращения времени на устранение ошибок и добавление новой функциональности. В случае непродуманного проекта необходимость в видоизменении или устранении недостатков приложения может привести к закрытию проекта и созданию нового. Поэтому важно придерживаться основных этапов разработки приложений, которые мы рассмотрим далее.

Основные этапы разработки приложений:

1. **Определение задач.** В самом начале необходимо определить перечень задач и функций, которые должно выполнять приложение. Список функций должен разделяться на две части — основные функции приложения и функции, которые могут быть добавлены в будущем.
2. **Составление последовательности выполнения действий.** Следующим этапом является разработка алгоритма работы приложения. На этом этапе задачи логически группируются и располагаются в порядке их выполнения.

3. **Определение необходимых ресурсов.** Для каждой задачи необходимо проанализировать данные, которые могут понадобиться для ее выполнения. Данные могут быть внешними или вычисляемыми, изменяемыми или не изменяемыми. Кроме того, важно определить область видимости данных.
4. **Разработка макета пользовательского интерфейса.** Наиболее важной частью приложения является пользовательский интерфейс, с помощью которого пользователь взаимодействует с программой, поэтому очень важно правильно разработать функциональность интерфейса, т. к. именно он определяет привлекательность приложения для конечного потребителя.
5. **Выбор технологий.** Технология Java в свою очередь состоит из огромного набора различных технологий. Выполнение каждой задачи приложения может быть реализовано с помощью одной или нескольких Java-технологий. На данном этапе нужно определить перечень библиотек API, которые будут использоваться при разработке приложения.
6. **Выбор среды разработки.** Существует огромное количество инструментов разработки Java-приложений — как бесплатных, так и коммерческих. На этом этапе необходимо выбрать оптимальную для данного проекта среду разработки приложений.
7. **Создание приложения.** Опираясь на все предыдущие этапы, разрабатывается приложение, решающее все требуемые задачи, обладающее дружественным интерфейсом пользователя и возможностью будущего расширения. Данный этап включает в себя отладку приложения и устранение ошибок.
8. **Тестирование.** Для обеспечения надежности и корректности работы приложения необходимо его тестирование при различных условиях и вводимых данных. В процессе тестирования происходит видоизменение приложения и устранение его недостатков. Тестирование приложений — это отдельная технология со своей методологией и инструментами.
9. **Определение способов распространения приложения.** На этом этапе определяется, как и с помощью каких технологий созданное приложение будет поставляться конечному пользователю и разворачиваться на клиентском компьютере.

Обзор сред разработки Eclipse и NetBeans

Для написания исходного кода приложения хватит простейшего редактора типа Блокнот. Для отладки и компиляции достаточно командной строки. Однако интегрированная среда разработки (IDE) предоставляет массу преимуществ разработчику программного обеспечения, представляя собой полный

набор инструментов для создания приложений. Как правило, данный набор включает в себя текстовый редактор, средства автоматизации сборки и отладки, компилятор/интерпретатор, интегрированные средства контроля версий, конструктор графического интерфейса пользователя, браузер классов, инспектор объектов, средства управления проектами, настраиваемый интерфейс пользователя, средства навигации, шаблоны, средства тестирования и многое другое.

В настоящее время из бесплатных IDE наиболее популярными являются Eclipse и NetBeans.

Примечание

Проект Eclipse изначально создавался как технология, запатентованная фирмой Object Technology International (OTI), которая являлась дочерней компанией корпорации IBM. Целью проекта было создание интегрированной среды для разработки Java-приложений. В 2001 г. IBM открыла исходный код и сформировала консорциум Eclipse, задачей которого было дальнейшее развитие проекта. В 2004 г. консорциум Eclipse был преобразован в некоммерческую организацию Eclipse Foundation, обслуживающую сообщество разработчиков открытого исходного кода и являющуюся координатором разработок Eclipse.

Проект Eclipse представляет собой набор расширяемых за счет встраиваемых плагинов (модулей) проектов с открытым исходным кодом, построенных на основе технологии Equinox OSGi (Open Source Gateway Initiative). Equinox — это название одной из реализаций стандарта OSGi, который определяет связывание удаленных узлов, состыковывая различные платформы и слои архитектуры. Equinox OSGi определяет модель динамически подключаемых модулей Eclipse и является модульной технологией, позволяющей создавать приложения, реализующиеся в наборе компонентов *(bundles)*, которые используют общие сервисы и инфраструктуру, при этом модули Java-классов загружаются по требованию, т. е. динамически. Каждый модуль имеет свои файлы манифеста *(manifest)*, содержащие информацию о его соединениях с другими модулями. При этом модуль может объявить любое количество именованных точек расширения *(extension point)* и любое количество расширений *(extension)* к одной или более точкам других модулей.

Подробно структуру проекта Eclipse можно посмотреть в *приложении 3*.

Примечание

Проект NetBeans начался в 1996 г. как студенческий в Карловом университете г. Праги. Целью проекта было создание 100% Java Delphi-подобной среды Java-разработки. В 1997 г. командой разработчиков была основана компания NetBeans для продвижения коммерческих версий среды разработки. В 2000 г. компания Sun Microsystems приобрела NetBeans и открыла исходные коды проекта, организовав сообщество NetBeans. В настоящее время свыше 500 тыс. разработчиков более чем из 130 стран и большое количество фирм составляют сообщество NetBeans.

Проект NetBeans состоит из:

- платформы NetBeans;
- среды разработки NetBeans IDE.

Платформа NetBeans — модульная и расширяемая за счет интеграции дополнительных модулей база, на основе которой возможно создание насыщенных клиентских приложений, использующих графическую библиотеку Swing. Платформа NetBeans является фундаментом для самой среды разработки NetBeans IDE и предоставляет общие для всех настольных приложений возможности, обеспеченные соответствующими API.

Возможности, предоставляемые платформой NetBeans:

1. **Модульная архитектура разрабатываемого приложения.** Платформа NetBeans предоставляет возможность организации динамической загрузки модулей в созданных на ее основе приложениях.
2. **Управление графическим интерфейсом пользователя.** Платформа обеспечивает оконную систему, которая упрощает управление составными компонентами внутри главного окна, такими как окна, меню, панели инструментов и др., общими для всех настольных приложений, а также систему управления действиями, связанными с этими компонентами. Платформа позволяет добавлять, удалять или изменять меню, панели инструментов, "горячие" клавиши, используя файл layer.xml, являющийся неотъемлемой частью каждого модуля. При изменении позиций, размеров, состояния открытия или закрытия окон приложения платформа сохраняет информацию до следующего запуска приложения.
3. **Управление данными.** Window System API, Visual Library API и Explorer & Property Sheet API обеспечивают богатый набор графических элементов пользовательского интерфейса для представления и манипулирования данными.
4. **Управление установками.** С помощью Options Settings API и NbPreferences API возможно управление сохранением и восстановлением установок.
5. **Графический редактор.** Применяя NetBeans IDE как комплект средств разработки (SDK) для платформы NetBeans, возможно использование GUI Builder для графического представления данных. Кроме того, Visual Library API позволяет использовать вместе комплексное моделирование и графическое представление.
6. **Редактор.** Приложения, созданные на основе платформы NetBeans, могут использовать NetBeans Editor независимо от NetBeans Platform.
7. **Мастер.** Платформа обеспечивает генерацию общего кода согласно шаблонам и установку структуры ресурсов проекта.

8. **Управление хранением данных.** Платформа обеспечивает для модулей доступ к файлам данных, расположенным как локально, так и удаленно на FTP-сервере, в CVS-хранилище, в виде XML-файла или в базе данных, с помощью класса `FileObjects`.
9. **Механизм автообновления**, обеспечивающий динамическое обновление установленного приложения. Возможность использования приложений, созданных на основе платформы NetBeans, как технологии Java Web Start для распространения и обновления, так и Plugin Manager для обновления или добавления новой функциональности приложению через Интернет.

Перечень основных библиотек NetBeans API можно посмотреть в *приложении 4*.

Для создания приложения на базе платформы NetBeans достаточно написать модуль, реализующий специфические задачи приложения, и связать его с модулями платформы. В результате получится кроссплатформенное приложение с гибкой структурой.

Основное отличие платформы NetBeans от платформы Eclipse состоит в том, что NetBeans использует платформно-независимую графическую библиотеку Swing, созданную на языке программирования Java, в то время как Eclipse использует графическую библиотеку SWT, тесно интегрированную с конкретной операционной системой. Модульная архитектура Eclipse основана на технологии OSGi, в то время как NetBeans для построения модульной системы использует NetBeans API.

NetBeans IDE — это модульная, кроссплатформенная, с открытым исходным кодом, интегрированная среда разработки, предназначенная для создания, компилирования, развертывания и отладки настольных, корпоративных, Web- и мобильных приложений.

Среда разработки NetBeans IDE является насыщенным клиентским приложением, созданным на основе платформы NetBeans с использованием языка программирования Java. Поддерживает разработку не только на языке Java, но и на C/C++, Ruby и PHP, имеет встроенную поддержку платформ Java SE, Java EE, Java ME и большой выбор дополнительных модулей.

Базовая среда NetBeans IDE включает в себя следующие средства.

- *Основные компоненты среды разработки*, такие как: Apache Ant — инструмент автоматизации сборки приложения; окна проектов, файлов, сервисов, навигатора; настраиваемый интерфейс пользователя; настраиваемое форматирование кода; настраиваемая оконная система; меню навигации; группировка проектов; встроенные шаблоны программирования и примеры приложений; сборка и запуск проекта; окна стандартного вывода и списка задач; совместное использование проектов; возможность импорта проектов и др.

- *Редактор исходного кода* с поддержкой Java, XML, DTD, CSS, HTML, ERB, RHTML, JSP, Javadoc, JavaScript, PHP, Groovy, Ruby, C/C++, а также таких функций, как: интеллектуальное дополнение кода; выделение кода с учетом синтаксического и семантического анализа; выделение событий, ошибок, предупреждений, подсказок, вариантов быстрого ввода; автоматическое компилирование при сохранении и развертывание при изменении в целом для проекта; автоматическая генерация кода; свертывание исходного кода; настраиваемое форматирование кода; средства улучшения (refactoring) кода; всплывающие окна документации; возможность расширения редактора с поддержкой любого языка и др.
- *Инструменты контроля версий и сотрудничества разработчиков* включают в себя: интегрированные средства, которые обеспечивают автоматическое распознавание существующих каталогов под контролем версий, а также обновление, просмотр, сравнение и переименование файлов, фиксацию изменений; возможность выбора между системами контроля версий: CVS, Subversion и Mercurial; модуль программного средства Rational ClearCase фирмы IBM — системы конфигурационного управления программным проектом; возможность обзора состояния и истории файлов каталога с помощью окна **Versioning**; средство просмотра различий на основе строк с выделением строк условными цветами; модуль Developer Collaboration, который обеспечивает совместное использование проектов в режиме реального времени по сети.
- *Поддержку платформы Java SE* (Java Standard Edition), обеспечивающую: разработку на языке программирования Java с поддержкой JDK; модульное тестирование с использованием библиотеки JUnit; менеджер библиотек и шаблонов, доступный из элемента меню **Tools** среды разработки; окно иерархии вызовов — щелкнув правой кнопкой мыши на строке исходного кода и выбрав в появившемся меню пункт **Call Hierarchy**, можно увидеть появившееся окно, в котором будут отображаться отношения между методами классов; инструменты Javadoc Auto Comment — для генерации комментариев, Javadoc Analyzer — для определения существующих комментариев; автоматическое определение корневого каталога документации и исходного кода; поддержку технологии Java Beans; визуальный конструктор графического пользовательского интерфейса Visual Swing GUI Builder; интегрированный профайлер — инструмент для анализа производительности центрального процессора (CPU), памяти и потоков выполнения с целью оптимизации быстродействия приложения и использования памяти; интегрированный отладчик с поддержкой нескольких языков программирования с возможностью установки точек останова в исходном коде, осуществлением пошагового выполнения кода, многосеансовой и многопоточной отладкой, вычислением выражений.

- *Поддержку платформы Java EE* (Java Enterprise Edition), включая:
  - разработку Web-приложений с использованием технологий и языков программирования Ajax, CSS, JavaScript, PHP, апплетов, Java Web Start (JNLP), JavaServer Faces (JSF) и JavaServer Pages (JSP);
  - интеграцию с базами данных, которая обеспечивает пользовательский интерфейс приложений баз данных;
  - разработку серверных Java-приложений с использованием стандартов J2EE 1.3, J2EE 1.4 и EE Java 5 (включая аннотации), интерфейсов Java Persistence API, Java Servlet API, технологий JavaServer Pages, Enterprise JavaBeans, платформ Hibernate, Spring, серверов приложений Sun Java System Application Server, GlassFish, Apache Tomcat, JBoss, BEA WebLogic, IBM WebSphere, Sailfin и др.;
  - разработку Web-служб на базе SOAP и RESTful с использованием стандартов Web-служб JAX-WS 2.1, JAX-RS (JSR 311), JAX-RPC (JSR 101), а также разработку мобильных Web-служб Java ME (JSR 172).
- *Поддержку платформы Mobility* (Java Micro Edition), которая обеспечивает разработку приложений для мобильных телефонов, карманных компьютеров, телеприставок, для встраиваемых систем, включая: поддержку профилей Mobile Information Device Profile (MIDP) 1.0, 2.0, 2.1 (MSA), Connected Limited Device Configuration (CLDC) 1.0 и 1.1, Connected Device Configuration (CDC); визуальную разработку графических интерфейсов с помощью Visual Mobile Designer (VMD); визуальный редактор мобильных игр Mobile Game Builder для интерфейса MIDP 2.0 Game API; поддержку графики SVG (JSR 226); использование сценариев компоновки Apache Ant и тестирования JMUnit и др.
- *Разработку на языках программирования* Ruby, C/C++, PHP, Python, Groovy, языке UML, языке сценариев JavaFX.
- *Разработку приложений на базе сервис-ориентированной архитектуры* (SOA) с использованием Web-служб, проектировщиков BPEL и XSLT, редактора сборки служб для составных приложений, средства и редактора схем XML, графического редактора для языка описания Web-служб (Web Services Description Language, WSDL).
- *Разработку насыщенных клиентских приложений* на базе платформы NetBeans.
- *Инструмент Plugin Manager*, который объединяет Update Center и Module Manager и позволяет подключать дополнительные модули, доступные на портале Plugin Portal официального сайта NetBeans (**http://plugins.netbeans.org/PluginPortal/**).

И Eclipse и NetBeans представляют собой увлекательные миры, погрузившись в которые каждый найдет для себя много полезного и интересного.

В данной книге мы рассмотрим использование среды разработки NetBeans для создания Java-приложений.

Установка необходимого программного обеспечения

Мы будем рассматривать использование среды NetBeans в операционной системе Windows XP/Vista.

Для начала работы нам потребуется установить комплект разработки JDK (Java™ Platform, Standard Edition Development Kit), а также интегрированную среду разработки NetBeans.

Для установки JDK:

1. Скачайте бесплатно дистрибутив JDK (**http://java.sun.com/javase/downloads/**).
2. Загруженный файл будет представлять собой приложение вида jdk-6ux-windows-i586-p.exe. Запустите это приложение.
3. Откроется окно установщика JDK (рис. B1).
4. Дальше следуйте инструкциям. В результате получится два каталога: jre1.6.0_0x и jdk1.6.0_0x, содержащие Standard Edition Runtime Environment (JRE) и Standard Edition Development Kit (JDK).

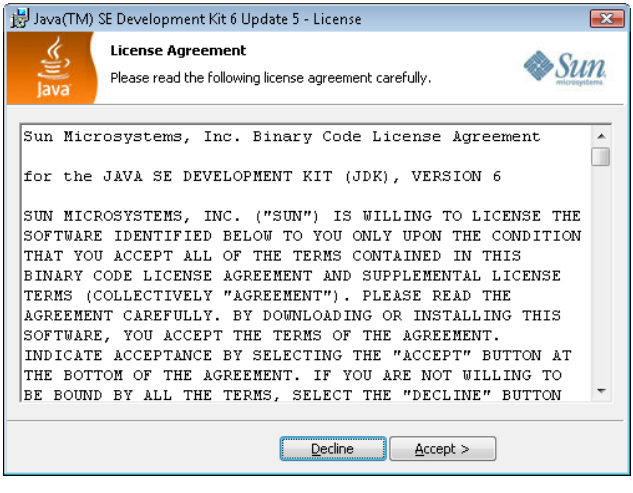

Рис. B1. Окно установщика JDK

Для установки среды разработки NetBeans:

1. Загрузите с сайта по адресу **http://www.netbeans.org** интегрированную среду разработки NetBeans.
2. Загруженный файл будет представлять собой приложение вида netbeans-x.x-ml-windows.exe. Запустите это приложение.
3. Откроется окно инсталлятора среды NetBeans (рис. В2).
4. Следуйте инструкциям. В результате появятся каталоги NetBeans и glassfish-vx, содержащие сервер приложений GlassFish.

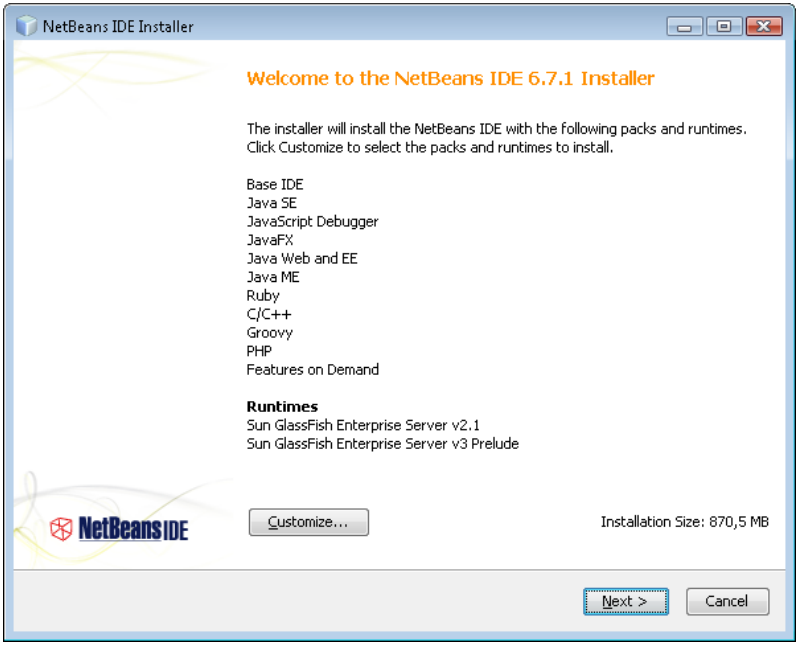

Рис. В2. Окно установщика NetBeans

ЧАСТЬ I

Платформа Java SE

Глава 1	Создание апплетов с использованием графической библиотеки AWT (Abstract Window Toolkit)
Глава 2	Создание апплетов с использованием графической библиотеки Swing
Глава 3	Создание настольных приложений на базе платформы Swing Application Framework
Глава 4	Создание расширяемых Java-приложений

Общая структура платформы Java SE была рассмотрена во *введении*. Еще раз можно повторить, что платформа Java SE является основой для всех остальных платформ технологии Java и предназначена для создания апплетов и настольных приложений. Платформа Java SE содержит набор технологий, которые описываются соответствующими спецификациями. Перечень спецификаций платформы Java SE и их общее применение можно посмотреть в *приложении 5*.

Создаваемые на основе платформы Java SE апплеты представляют собой программы, написанные на языке Java и работающие в среде браузера, который загружает их и запускает виртуальную машину JVM для их выполнения.

В отличие от настольных приложений, апплеты являются управляемыми программными компонентами. Это означает, что апплет не может быть запущен, как настольное приложение, одной только виртуальной машиной JVM. Для его работы необходим браузер, который распознает теги `<APPLET>` или `<OBJECT>` и `<EMBED>`, включающие апплет в HTML-страницу. Главный класс апплета должен быть подклассом класса `java.applet.Applet`, при этом класс `Applet` служит интерфейсом между апплетом и браузером. Жизненным циклом апплета управляет компонент Java Plug-in среды выполнения JRE.

Апплеты могут быть двух типов: имеющие электронную подпись и сертификат и не имеющие сертификата. Не имеющие сертификата апплеты, или апплеты, сертификат которых не принял конечный пользователь, работают в "песочнице" — данное понятие обозначает специальный механизм системы безопасности, ограничивающий доступ работающего приложения к компьютеру пользователя. Апплеты, сертификат которых принят пользователем, работают вне "песочницы", в частности, они способны осуществлять доступ к файловой системе пользователя.

Область применения апплетов достаточно широка. Апплеты используются в качестве элементов, украшающих Web-страничку, создавая продвинутую анимацию, служат интерактивной рекламой, представляют интерактивные игры. Однако назначение апплетов гораздо шире, чем использование в Web-дизайне. При создании приложений уровня предприятия с помощью апплетов создается графический интерфейс пользователя *"тонкого"* клиента.

Настольные приложения платформы Java SE — это независимые Java-приложения, которые выполняются виртуальной машиной JVM, при этом точкой входа в приложение является главный класс приложения, содержащий статический метод `main`. С помощью технологии Java можно создать практически любое настольное приложение с областью применения, ограниченной только фантазией разработчика. Это различные проигрыватели, навигаторы, читалки, переводчики, почтовые клиенты и многое другое. В приложениях уровня предприятия настольное приложение представляет *"толстого"* клиента.

ГЛАВА 1

Создание апплетов с использованием графической библиотеки AWT (Abstract Window Toolkit)

Обзор графической библиотеки AWT

Любое приложение, требующее взаимодействия с пользователем, должно иметь интерфейс пользователя. От интерфейса пользователя зависит привлекательность и удобство работы с программой. Интерфейс пользователя может быть как низкоуровневым, в виде командной строки, так и иметь разнообразные графические компоненты — кнопки, переключатели, меню, поля ввода и другие элементы. Технология Java предлагает набор библиотек классов и интерфейсов, на основе которых можно построить эффективный графический интерфейс пользователя, имеющий необходимый внешний вид и поведение для создания комфортной среды конечному потребителю.

Самой первой графической Java-библиотекой была создана библиотека AWT. Она была включена в первую версию JDK 1.0. Затем библиотека AWT была дополнена библиотекой Java 2D API, расширяющей возможности работы с двухмерной графикой и изображениями. Так как технология Java является платформно-независимой, то соответственно и графическая Java-библиотека должна быть платформно-независимой. Сам по себе язык Java не обладает возможностями низкоуровневого взаимодействия с конкретной операционной системой, обеспечивающими передачу информации от мыши или клавиатуры приложению и вывод пикселов на экран. Поэтому библиотека AWT была создана так, что каждый AWT-компонент имеет своего двойника `peer` — интерфейс, обеспечивающий взаимодействие с конкретной операционной системой. Таким образом, переносимость графической библиотеки AWT обусловлена наличием реализации пакета `java.awt.peer` для конкретной

операционной системы. Вследствие этого, AWT-компоненты называют *тяжеловесными*.

Все AWT-компоненты, кроме элементов меню, представлены подклассами класса `java.awt.Component`. Для элементов меню суперклассом является класс `java.awt.MenuComponent`. Архитектура AWT устроена таким образом, что компоненты размещаются в контейнерах (суперкласс `java.awt.Container`) с помощью менеджеров компоновки — классов, реализующих интерфейс `java.awt.LayoutManager`.

Для настольных приложений корневое окно графического интерфейса пользователя представляет контейнер `java.awt.Window`, который в свою очередь должен содержать окно `java.awt.Frame` с заголовком и границами или диалоговое окно `java.awt.Dialog`, также имеющее заголовок и границы. AWT-компоненты добавляются в панель `java.awt.Panel` — контейнер, который может содержать как компоненты, так и другие панели.

Для апплетов класс `java.applet.Applet`, расширяющий класс `java.awt.Panel`, является корневым контейнером для всех графических компонентов.

Помимо графических компонентов, библиотека AWT содержит классы и интерфейсы, позволяющие обрабатывать различные типы событий, генерируемые AWT-компонентами. Суперклассом, представляющим все AWT-события, является класс `java.awt.AWTEvent`. Для обработки событий компонента необходимо создать класс-слушатель, реализующий интерфейс `java.awt.event.ActionListener`, и присоединить его к данному компоненту.

Кроме пакетов `java.awt` и `java.awt.event` библиотека AWT включает в себя пакеты:

- `java.awt.color` используется для создания цвета;
- `java.awt.datatransfer` применяется для передачи данных внутри приложения и между приложениями;
- `java.awt.dnd` реализует технологию drag-and-drop;
- `java.awt.font` обеспечивает поддержку шрифтов;
- `java.awt.geom` реализует двухмерную геометрию;
- `java.awt.im` обеспечивает поддержку нестандартных методов ввода текста;
- `java.awt.image` используется для создания и редактирования графических изображений;
- `java.awt.print` обеспечивает поддержку печати.

Так как AWT-компоненты основываются на peer-объектах, то использование библиотеки AWT является потоково-безопасной (thread safe), поэтому не нужно беспокоиться о том, в каком потоке обновляется состояние графиче-

ского интерфейса. Однако беспорядочное использование потоков может замедлять работу AWT-интерфейса.

Обобщая вышесказанное, можно сказать, что графическая библиотека AWT представляет собой промежуточный уровень между операционной системой и Java-кодом приложения, скрывая все низкоуровневые операции, связанные с построением графического интерфейса пользователя. Такое прямое взаимодействие с конкретной операционной системой является и основным недостатком AWT, т. к. графический интерфейс, созданный на основе AWT, в операционной системе Windows выглядит как Windows-подобный, а в операционной системе Mac OS X — как Mac-подобный.

Применение AWT и сравнение с другими графическими Java-библиотеками

Графическая библиотека AWT — это стандарт платформы Java SE, однако набор графических компонентов, предлагаемый AWT, ограничен. В частности, отсутствуют такие компоненты, как таблицы, строка состояния, переключатель (radio button) и др. Кроме того, внешний вид и поведение графического AWT-интерфейса пользователя зависят от операционной системы, в которой он отображается.

Все это послужило причиной создания графической библиотеки Swing. Библиотека Swing расширяет AWT и создана полностью на языке Java, поэтому Swing-компоненты называются *легковесными*, за исключением четырех компонентов верхнего уровня: `JWindow`, `JFrame`, `JDialog` и `JApplet`, которые являются прямыми наследниками тяжеловесных AWT-компонентов. Библиотека Swing предлагает гораздо более широкий набор компонентов и менеджеров компоновки по сравнению с библиотекой AWT. Кроме того, т. к. Swing создана полностью на Java, то решается проблема одинакового отображения и поведения интерфейса пользователя в различных операционных системах.

Помимо всего прочего, библиотека Swing расширяет модель обработки событий AWT-библиотекой соответствующих классов и интерфейсов. Однако при этом использование библиотеки Swing не является потоково-безопасной, поэтому Java-код, изменяющий состояние интерфейса и обрабатывающий события, должен выполняться в специальном потоке *Event Dispatch Thread (EDT)*.

Так же как и AWT, библиотека Swing — стандарт платформы Java SE.

Альтернативой AWT и Swing служит библиотека SWT (Standard Widget Toolkit) и набор расширенных возможностей JFace проекта Eclipse, не являющий-

ся стандартом Java SE. Это означает, что их использование требует включения в CLASSPATH Java-приложения.

Примечание

CLASSPATH — переменная среды выполнения, содержащая информацию о расположении Java-файлов. Данная информация используется инструментами javac и java при компиляции и выполнении Java-кода соответственно.

Концепция SWT близка концепции AWT. Компоненты SWT также взаимодействуют с операционной системой с помощью интерфейсов peer. Отличие заключается в том, как данное взаимодействие происходит. В SWT интерфейсы peer выполняют функцию оболочек для графических библиотек конкретных операционных систем. Поэтому приложение, созданное с использованием библиотеки SWT, становится полностью совместимым с конкретной операционной системой, тем самым нарушается принцип "Write Once, Run Anywhere". Однако использование графических возможностей самой операционной системы, вместо создания собственной графики, делает применение библиотеки SWT более эффективным. По ассортименту графических компонентов, менеджеров компоновки и модели обработки событий SWT сравнима со Swing. Так же как и Swing, библиотека SWT не является потоковобезопасной.

В платформу Java SE также включена графическая библиотека Java 3D, которая, правда, не поставляется вместе с комплектом разработки JDK, а требует отдельной загрузки и инсталляции. Библиотека Java 3D позволяет создавать 3D-апплеты и Java-приложения, использующие трехмерную графику. С Java 3D можно эффективно конструировать виртуальные миры, создавая отдельные графические элементы и затем соединяя их в древовидные структуры — *scene graph*. Библиотека Java 3D — это результат синтеза лучших идей, взятых из таких технологий, как Direct3D, OpenGL, QuickDraw3D и XGL.

Казалось бы, при таком выборе графических Java-библиотек библиотека AWT должна потерять свою актуальность. Однако если нет необходимости в широком ассортименте графических компонентов, если требуется работа в основном с двухмерной графикой и изображениями, использование библиотеки AWT удобно. Кроме того, библиотека AWT является частью платформы Java ME и используется для создания приложений, работающих в устройствах с ограниченными возможностями — КПК и телевизионных приставках.

Использование AWT на примере создания апплета-игры "Звездные войны"

В качестве примера рассмотрим создание апплета, представляющего игру "Звездные войны".

ПРИМЕЧАНИЕ

Файлы с кодами апплета-игры "Звездные войны" расположены на прилагаемом к книге компакт-диске в папке Часть_I\Глава_1\StarWar.

ОПИСАНИЕ ИГРЫ

Космический корабль летит в звездном пространстве и поражает огнем лазера появляющихся врагов. За каждого пораженного врага начисляются очки. У корабля есть определенное количество жизней, которое уменьшается при столкновении с врагом. Игра прекращается при обнулении количества жизней космического корабля.

В самом начале создадим в простейшем графическом редакторе Paint, поставляемом с операционной системой Windows, графические изображения, необходимые для работы апплета:

- космический корабль — ship.gif;
- пламя двигателя корабля — fire.gif;
- огонь лазера пушки корабля — laser.gif;
- враги — enemy.gif;
- взрыв при столкновении или при поражении врага — explosion.gif.

Далее в среде разработки NetBeans создаем новый проект:

1. В главном меню среды разработки (верхняя строка) выберем **File | New Project**. Появится диалоговое окно **New Project** (рис. 1.1), в котором выберем **Java Class Library**.
2. В строке **Project Name** введем название проекта StarWar (рис. 1.2).
3. Нажмем кнопку **Finish**.
4. В окне **Projects** среды NetBeans щелкнем правой кнопкой мыши на названии проекта StarWar. В появившемся контекстном меню выбираем **New | Other**.
5. В появившемся окне **New File** выберем **Java | Applet** (рис. 1.3). В этом же окне есть пункт шаблон **JApplet**, который отличается от шаблона **Applet** тем, что использует класс `JApplet`, являющийся расширением класса `Applet` и использующий графическую библиотеку Swing, в то время как класс `Applet` создан на основе графической библиотеки AWT.
6. Далее нажмем кнопку **Next** и введем имя класса и пакета (рис. 1.4).
7. Нажмем кнопку **Finish** и попадем в окно редактора исходного кода среды NetBeans (рис. 1.5).

При создании проекта среда NetBeans сгенерировала код, являющийся основой нашего апплета. Для общего понимания проанализируем этот код, представленный в окне редактора (листинг 1.1).