

# Программирование на языке Java. Революция по имени Java

Картузов А.В.

Вообще опасно объявлять каждую новую технологию программирования революционной. Если вы поторопитесь подать свой голос за признание технологии, подобной той, которая реализована в языке Java, революционной — вас могут закидать тухлыми яйцами или занести в разряд пустозвонов, падких на модные новинки. Что же в таком случае делает тот или иной продукт революционным? Он не может быть только лишь компактней, быстрее и дешевле. Такой продукт должен изменить весь стиль работы, радикально упростив решение сложных проблем.

Создание языка Java — это действительно один из самых значительных шагов вперед в области разработки сред программирования за последние 20 лет. Язык HTML (Hypertext Markup Language — язык разметки гипертекста) был необходим для статического размещения страниц во “Всемирной паутине” WWW (World Wide Web). Язык Java потребовался для качественного скачка в создании интерактивных продуктов для сети Internet.

Три ключевых элемента объединились в технологии языка Java и сделали ее в корне отличной от всего, существующего на сегодняшний день.

- Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML.
- Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты.

- Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

Давайте поближе познакомимся со всеми этими тремя аспектами, но сначала – история создания.

## **1. История создания**

Язык Java зародился как часть проекта создания передового программного обеспечения (ПО) для различных бытовых приборов. Реализация проекта была начата на языке C++, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента—языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

Рождению языка Java предшествовала довольно интересная история. В 1990 году разработчик ПО компании *Sun Microsystems Патрик Нотон (Patrick Naughton)* понял, что ему надоело поддерживать сотни различных интерфейсов программ, используемых в компании, и сообщил исполнительному директору Sun Microsystems и своему другу *Скотту МакНили (Scott McNealy)* о своем намерении перейти работать в компанию NeXT. МакНили, в свою очередь, попросил Нотона составить список причин своего недовольства и выдвинуть такое решение проблем, как если бы он был Богом и мог исполнить все, что угодно.

Нотон, хотя и не рассчитывал на то, что кто-то обратит внимание на его письмо, все же изложил свои претензии, беспощадно раскритиковав недостатки Sun Microsystems, в частности, разрабатываемую в тот момент архитектуру ПО NeWS. К удивлению Нотона, его письмо возымело успех: оно было разослано всем ведущим инженерам Sun Microsystems, которые не замедлили откликнуться и высказать горячую поддержку своему коллеге и одобрение его взглядов на ситуацию в Sun Microsystems. Обращение вызвало одобрение и у высшего руководства компании, а именно, у *Билла*

*Джоя (Bill Joy)*, основателя Sun Microsystems, и *Джеймса Гослинга (James Gosling)*, начальника Нотона.

В тот день, когда Нотон должен был уйти из компании, было принято решение о создании команды ведущих разработчиков с тем, чтобы они делали что угодно, но создали нечто необыкновенное. Команда из шести человек приступила к разработке нового объектно-ориентированного языка программирования, который был назван *Oak* (дуб), в честь дерева, росшего под окном Гослинга.

Вскоре компания Sun Microsystems преобразовала команду Green в компанию First Person. Новая компания обладала интереснейшей концепцией, но не могла найти ей подходящего применения. После ряда неудач неожиданно ситуация для компании резко изменилась: был анонсирован браузер Mosaic—так родился World Wide Web, с которого началось бурное развитие Internet. Нотон предложил использовать Oak в создании Internet-приложений. Так Oak стал самостоятельным продуктом, вскоре был написан Oak-компилятор и Oak-браузер "*WebRunner*". В 1995 году компания Sun Microsystems приняла решение объявить о новом продукте, переименовав его в *Java* (единственное разумное объяснение названию—любовь программистов к кофе). Когда Java оказалась в руках Internet, стало необходимым запускать Java-апплеты—небольшие программы, загружаемые через Internet. WebRunner был переименован в *HotJava* и компания Netscape встала на поддержку Java-продуктов.

## **2. Апплеты Java**

Каждый апплет — это небольшая программа, динамически загружаемая по сети — точно так же, как картинка, звуковой файл или элемент мультимедиа. Главная особенность апплетов заключается в том, что они являются настоящими программами, а не очередным форматом файлов для хранения мультфильмов или какой-либо другой информации. Апплет не просто проигрывает один и тот же сценарий, а реагирует на действия пользователя и может динамически менять свое поведение.

Именно броские Web-страницы с анимацией привлекли большинство ранних приверженцев языка Java. Поскольку пользователи не сразу смогли полностью освоить наиболее революционные аспекты Java, этот язык часто сравнивался с другими технологиями для загрузки динамических изображений и простого взаимодействия с Web-клиентами. Компании, традиционно занимающиеся разработкой

мультимедиа-технологий, например, Adobe или Macromedia, утверждали, что их продукты предоставляют те же возможности, что и Java. По мнению Kaleida, Taligent и NeXT, их собственные объектно-ориентированные среды разработки были не менее революционны. Microsoft заявлял о победе технологии десятилетней давности. На самом деле ничто не может сравниться с тем, что вы откроете для себя, ощутив мощь программирования на языке Java.

Возможность задания любых уровней взаимодействия с пользователем существует лишь в том случае, когда используемая для разработки платформа предоставляет полнофункциональную среду программирования.

### **3. Революционный язык программирования**

Язык должен был воплощать следующие качества: простоту и мощь, безопасность, объектную ориентированность, надежность, интерактивность, архитектурную независимость, возможность интерпретации, высокую производительность и легкость в изучении. Даже если вы никогда не напишете ни одной строки на языке Java, знать о его возможностях весьма полезно, поскольку именно перечисленные выше свойства языка придают динамику страницам Всемирной паутины.

### **4. Простота и мощь**

После освоения основных понятий объектно-ориентированного программирования вы быстро научитесь программировать на Java. В наши дни существует много систем программирования, гордящихся тем, что в них одной и той же цели можно достичь десятком различных способов. В языке Java изобилие решений отсутствует — для решения задачи у вас будет совсем немного вариантов. Стремление к простоте зачастую приводило к созданию неэффективных и невыразительных языков типа командных интерпретаторов. Java к числу таких языков не относится — для Вас вся мощь ООП и библиотек классов.

### **5. Безопасность**

В популярной литературе наших дней, особенно если речь заходит об Internet, стало модной темой обсуждение вопросов безопасности. Люди уверены, что использование

Internet в коммерческой деятельности равносильно написанию номера своей кредитной карточки на стенке телефонной будки. Один из ключевых принципов разработки языка Java заключался в обеспечении защиты от несанкционированного доступа. Программы на Java не могут вызывать глобальные функции и получать доступ к произвольным системным ресурсам, что обеспечивает в Java уровень безопасности, недоступный для других языков.

## **6. Объектная ориентированность**

Забавно наблюдать, как многочисленные новые диалекты старых языков безапелляционно объявляются объектно-ориентированными. Поскольку при разработке языка отсутствовала тяжелая наследственность, для реализации объектов был избран удобный прагматичный подход. Разработчики Java старались выдержать разумный компромисс между моделью пуристов — “все является объектами”, и моделью хакеров — “уйди с моей дороги”. Объектная модель в Java проста и легко расширяется, в то же время, ради повышения производительности, числа и другие простые типы данных Java не являются объектами.

## **7. Надежность**

Java ограничивает вас в нескольких ключевых областях и таким образом способствует обнаружению ошибок на ранних стадиях разработки программы. В то же время в ней отсутствуют многие источники ошибок, свойственных другим языкам программирования (строгая типизация, например). Большинство используемых сегодня программ “отказывают” в одной из двух ситуаций: при выделении памяти, либо при возникновении исключительных ситуаций. В традиционных средах программирования распределение памяти является довольно нудным занятием — программисту приходится самому следить за всей используемой в программе памятью, не забывая освобождать ее по мере того, как потребность в ней отпадает. Зачастую программисты забывают освобождать захваченную ими память или, что еще хуже, освобождают ту память, которая все еще используется какой-либо частью программы. Исключительные ситуации в традиционных средах программирования часто возникают в таких, например, случаях, как деление на нуль или попытка открыть несуществующий файл, и их приходится обрабатывать с помощью неуклюжих и

нечитабельных конструкций (кроме Delphi). Java фактически снимает обе эти проблемы, используя сборщик мусора для освобождения незанятой памяти и встроенные объектно-ориентированные средства для обработки исключительных ситуаций.

## **8. Интерактивность**

Java создавалась как средство, которое должно удовлетворить насущную потребность в создании интерактивных сетевых программ. В Java реализовано несколько интересных решений, позволяющих писать код, который выполняет одновременно массу различных функций и не забывает при этом следить за тем, что и когда должно произойти. В языке Java для решения проблемы синхронизации процессов применен наиболее элегантный из всех когда-либо изобретенных методов, который позволяет конструировать прекрасные интерактивные системы. Простые в обращении изящные подпроцессы Java дают возможность реализации в программе конкретного поведения, не отвлекаясь при этом на встраивание глобальной циклической обработки событий.

## **9. Независимость от архитектуры ЭВМ**

Вопрос о долговечности и переносимости кода важнее религиозных войн между ПК и Макинтошами. Создатели Java наложили на язык и на среду времени выполнения несколько жестких требований, которые на деле, а не на словах позволяют, однажды написав, всегда запускать программу в любом месте и в любое время (где существует виртуальная Java-машина – браузеры на всех платформах, OS/2, Netware).

## **10. Интерпретация плюс высокая производительность**

Необычайная способность Java исполнять свой код на любой из поддерживаемых платформ достигается тем, что ее программы транслируются в некое промежуточное представление, называемое байт-кодом (bytecode). Байт-код, в свою очередь, может интерпретироваться в любой системе, в которой есть среда времени выполнения Java. Большинство ранних систем, в которых пытались обеспечить независимость от платформы, обладало огромным недостатком — потерей производительности (Basic, Perl). Несмотря на то, что в Java используется интерпретатор, байт-код легко

переводится непосредственно в “родные” машинные коды (Just In Time compilers) “на лету”. При этом достигается очень высокая производительность (Symantec JIT встроен в Netscape Navigator).

## **11. Простота изучения**

Язык Java, хотя и более сложный чем языки командных интерпретаторов, все же неизмеримо проще для изучения, чем другие другие языки программирования, например C++. Черты языка станут казаться вам естественным путем для решения тех или иных задач и будут способствовать отработке хорошего стиля программирования. Поскольку объектная модель в Java одновременно проста и выразительна, вы скоро освоитесь с объектно-ориентированным стилем создания программ.

## **12. Богатая объектная среда**

Среда Java — это нечто гораздо большее, чем просто язык программирования. В нее встроен набор ключевых классов, содержащих основные абстракции реального мира, с которым придется иметь дело вашим программам. Основой популярности Java являются встроенные классы-абстракции, сделавшие его языком, действительно независимым от платформы. Библиотеки, подобные MFC/COM, OWL, VCL, NeXTStep, Motif и OpenDoc прекрасно работают на своих платформах, однако сегодня главной платформой становится Internet.

В реализации Java 1.1.6 находится 23 пакета (в Java 1.0.2 их было 8), а количество классов – 503 (211). Сейчас проходит завершающую стадию бета-тестирования JDK 1.2. Для тех, кто собирается поддерживать Java 1.0 наряду с 1.1 (актуальность этой поддержки подтверждает Borland JBuilder 2.0 и другие продукты), специально выделены новшества 1.1 в [Приложение 2](#).

Имя пакета	Содержимое
java.applet	Классы для реализации апплетов
java.awt	Классы для работы с графикой, текстом, окнами и GUI
java.awt.datatransfer	Классы для обеспечения передачи информации (Copy/Paste)

## Программирование на языке Java. Революция по имени Java

java.awt.event	Классы и интерфейсы для обработки событий
java.awt.image	Классы для обработки изображений
java.awt.peer	GUI для обеспечения независимости от платформы
java.beans	API для модели компонентов JavaBeans
java.io	Классы для различных типов ввода-вывода
java.lang	Классы ядра языка (типы, работа со строками, тригонометрические функции, обработка исключений, легковесные процессы)
java.lang.reflect	Классы Reflection API
java.math	Классы для арифметических операций произвольной точности
java.net	Классы для работы в сети Интернет (сокеты, протоколы, URL)
java.rmi	Классы, связанные с RMI (удаленный вызов процедур)
java.rmi.dgc	Классы, связанные с RMI
java.rmi.registry	Классы, связанные с RMI
java.rmi.server	Классы, связанные с RMI
java.security	Классы для обеспечения безопасности
java.security.acl	Классы для обеспечения безопасности
java.security.interfaces	Классы для обеспечения безопасности
java.sql	
java.text	Классы для обеспечения многоязыковой поддержки
java.text.resources	Классы для обеспечения многоязыковой поддержки
java.util	Разнообразные полезные типы данных (стеки,

	словари, хэш-таблицы, даты, генератор случайных чисел)
java.util.zip	Классы для обеспечения архивации

**Таблица 1. Пакеты Java API**

Часть этих пакетов мы будем рассматривать очень подробно (это видно из оглавления), другая пойдет на самостоятельную проработку.