Сафонов Владимир Олегович

JAVA-ТЕХНОЛОГИЯ: ИСТОРИЯ, СОСТОЯНИЕ И ПЕРСПЕКТИВЫ

Впервые о Святославе Сергеевиче я узнал в начале 70-х годов, когда учился на мат-мехе на первом курсе на отделении механики. Мой сокурсник Александр Шишлов (ныне – депутат Государственной Думы) с восторгом рассказал мне, что в его группе, на отделении кибернетики, практические занятия по программированию ведет замечательный ученый, чл.-корр. АН СССР, лауреат Ленинской премии С.С. Лавров. Узнав о нем, я со второго курса перешел на отделение кибернетики и впервые встретился с С.С. Лавровым, когда сдавал ему экзамен по теории множеств и математической логике. Помню даже первый вопрос, который я ему задал на втором курсе, – о семантике аксиомы присваивания в исчислении программ Р. Флойда – Ч. Хоара.

Когда на четвертом курсе необходимо было выбрать себе научного руководителя, у меня не было сомнений – только Святослав Сергеевич (хотя было несколько интересных предложений от других известных специалистов). Тем более, что он предложил нам интереснейшие темы по реализации новых языков программирования для нового отечественного вычислительного комплекса ВК ВС-1 (впоследствии переименованного в МВК «Эльбрус»). Шесть студентов нашей кафедры математического обеспечения ЭВМ, в том числе и я, начали работать под руководством С.С.Лаврова над реализацией языка АБВ, который был разработан самим Святославом Сергеевичем (я и еще двое студентов), и языка Паскаль (еще трое студентов). Святослав Сергеевич организовал очень интересные семинары для нашей группы по компиляторам и ВК ВС-1. Это была настоящая школа. Очень многое из того, что нам тогда рассказал Святослав Сергеевич, стало впоследствии для нас не только основой наших специальных знаний, но и руководством к действию. С.С. Лавров был достаточно строг, беспощадно и иронически критиковал любые наши недостатки. Но это только укрепляло наше искреннее уважение и, без преувеличения, любовь к нему.

Особенно ярко педагогический, научный и организаторский талант С.С. Лаврова проявлялся на семинарах с участием известных зарубежных специалистов, на заседаниях кафедры, на которых в то время выступали с интересными докладами многие известные программисты, а также в его умении анализировать и конструктивно критиковать написанные его учениками диссертации и статьи. Из его метких замечаний и рекомендаций можно было бы составить не одну книгу. Кто ж еще, кроме Святослава Сергеевича, мог сделать 57 (!) замечаний к диссертации? Это была тоже настоящая наука и настоящая школа. С.С. Лавров учил четкости и лаконичности формулировок, не терпел пустой фразеологии, «формализации ради формализации», псевдонаучной «липы».

Я на всю жизнь благодарен Святославу Сергеевичу за его замечательную науку и желаю ему в славный день 80-летнего юбилея крепкого здоровья, новых творческих успехов (а Святослав Сергеевич и по сей день активно работает, выпуская все новые книги и статьи), хорошего настроения и много новых способных учеников.

История программирования до середины 90-х годов еще не знала такой популярности и такого «бума», который вызвало появление и развитие Java-технологии [1].

Вот лишь некоторые факты:

– Java-технология преподается более чем в 70% университетов мира;

– число Java-программистов уже сейчас сравнимо с числом программистов на языках C и C++;

– в мире ежедневно выпускается не менее 60 тысяч мобильных телефонов, программное обеспечение которых полностью написано на Java;

– число скачиваний (downloads) с Web-страниц, посвященных Java [2], новых версий инструментального комплекса Java Development Kit (JDK) составляет несколько миллионов.

В чем же причина такого успеха?

Безусловно, значительную роль в этом сыграла широкая рекламная кампания, развернутая вокруг языка Java и Java-технологии ее фирмой-разработчиком Sun Microsystems.

Однако есть и ряд несомненных достоинств Java-технологии, очень быстро выдвинувших ее в один ряд с такими языками и системами программирования, как C, C++, Turbo Pascal, Delphi:

– удобство, привлекательность, компактность, легкость изучения, с одной стороны, и невиданная доселе мощность и гибкость, с другой стороны, языка Java. Этот язык, в отличие, например, от C++ или PL/1, изучить почти столь же просто, как и Паскаль. Это можно сделать, например, прочитав книгу [1]. В ней можно найти большое число наглядных примеров программ на Java, так что позволим себе в рамках данной небольшой статьи обойтись без очередного «Hello, world»;

– энциклопедическая полнота Java-библиотек (Java API), охватывающих практически все области развития программирования и потребности программистов – от создания простых интерактивных программ, базирующихся на графическом пользовательском интерфейсе (GUI), до Интернет-программирования, создания встроенного программного обеспечения электронных устройств, компьютерной трехмерной графики, искусственного интеллекта, разработки компиляторов и многих других классов программ;

...широкая рекламная кампания, развернутая вокруг языка Java...

– ориентация Java на программирование для Интернет и с использованием Интернет. Эти качества сделали Java поистине «всемирным» языком программирования;

– независимость от аппаратной платформы как программ на Java, так и промежуточного кода – *байткода*, в который эти программы транслируют Java-компиляторы. Как провозгласила Sun, «write once, run everywhere» – напишите один раз и исполняйте везде. Правда, острословы-программисты, экспериментировавшие с первыми версиями JDK, сразу же переиначили этот рекламный лозунг как «write once, hang everywhere» – написал один раз, а «виснет» везде. Однако это нисколько не ослабило интереса к Java-технологии. Ведь и язык C++ острословы от программирования в самом начале его появления окрестили «C– –» («Си-минус-минус»), но это не помешало ему, как впоследствии и языку Java, войти в золотой фонд современного программирования.

1. ИСТОРИЯ

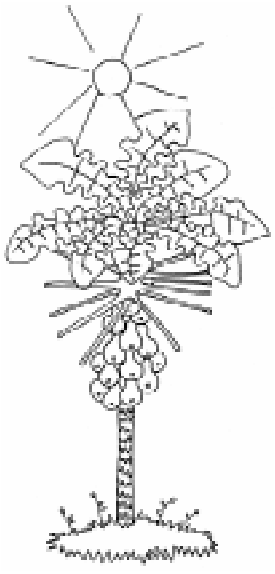
В 1991 г. выдающийся специалист фирмы Sun Microsystems доктор Джеймс Гослинг с небольшой группой коллег разработал и реализовал новый объектно-ориентированный язык Oak («дуб»), предназ-

...доктор Джеймс Гослинг ... разработал и реализовал новый объектно-ориентированный язык Oak («дуб»),

наченный для разработки программного обеспечения встроенных систем. Эту дату и можно считать датой появления Java-технологии, так как язык Oak стал впоследствии, в 1995 г., основой языка Java.

В 1995 г. фирма Sun выпустила первую версию JDK (JDK 1.0), которую сразу же сделала бесплатно доступной через Интернет. Эта традиция сохраняется и по сей день для всех новых версий JDK.

Появление принципиально новых, да еще и бесплатно доступных, технологий, инструментального комплекса и языка программирования сразу же привлекло внимание очень многих программистов. Однако было в Java и то, что вызвало, мягко говоря, удивление. Как такой мощный объектно-ориентированный язык после почти 50-летнего развития эффективных методов компиляции и оптимизации программ реализован путем компиляции в платформно-независимый байт-код – разновидность постфиксной записи – с последующей *интерпретацией* байт-кода?! Конечно, очень удобно один и тот же *класс-файл* (файл промежуточного кода) использовать и на Windows-машине, и на рабочей станции SPARC, но где же эффективность?! До этого уже давно, с середины 70-х годов, была известна система Pascal-P – переносимая реализация языка Паскаль проф. Никлауса Вирта, основанная на схожих принципах. Но хорошо известно, что интерпретация программы, пусть даже и в промежуточном коде, по сравнению с ее предварительной компиляцией в объектный код, может снизить скорость ее исполнения в 100 и более раз.



... фирма Sun выпустила первую версию JDK...

Откровенно говоря, критики были во многом правы. И, учтя все эти критические замечания, фирма Sun в следующей версии JDK – 1.1, выпущенной в 1997 г., – усовершенствовала схему реализации Java. Теперь *виртуальная машина Java (JVM)* – интерпретатор языка Java - не просто интерпретировала байт-код, но *динамически (во время исполнения) компилировала* каждый впервые вызываемый метод в объектный код целевой платформы. Такая схема получила название *Just-in-Time-компиляции (JIT),* что в буквальном переводе означает «компиляция как раз вовремя», или «компиляция по необходимости». Кстати, первый JIT-компилятор Java-байткода, включенный в JDK 1.1, был разработан не самой фирмой Sun, а фирмой Symantec.

Использование JIT-компиляторов позволило ускорить выполнение Java-программ в несколько раз, но этого, безусловно, было недостаточно. И тогда фирма Sun разработала и встроила в следующую версию JDK (1.2) уникальную по сложности систему динамической компиляции, оптимизации и *профилирования* (анализа свойств во время исполнения) Java-программ – *HotSpot* (полное название – *HotSpot Performance Engine*). Эта система, запускаемая по умолчанию во всех новых версиях JDK, динамически, то есть во время исполнения, анализирует свойства Java-программы – собирает статистику вызовов методов, использования памяти и т. д. – и на основе результатов этого анализа либо выполняет JIT-компиляцию с

оптимизацией для наиболее часто используемых методов, либо, наоборот, для редко вызываемых методов выполняет *деоптимизацию*, переключаясь обратно на их представление в виде байт-кода.

Применение HotSpot позволило ускорить выполнение программ на Java еще в несколько раз, так что она стала сравнима со скоростью выполнения программ, предварительно откомпилированных в объектный код целевой платформы. Однако на этом пути предстоит еще очень многое изобрести, реализовать и усовершенствовать. По признанию специалистов Microsoft, JIT-компиляция сейчас находится еще «в колыбели». Заметим, что JIT-компиляция является одной из основ новейшей разработки фирмы Microsoft – платформы .NET («dot-net») [3], в которой использованы и развиты многие идеи и методы Java-технологии в более общей, многоязыковой и метаязыковой, форме. Но .NET – это тема для отдельного разговора, заслуживающая самого серьезного анализа и изучения.

...“компиляция как раз вовремя”, или “компиляция по необходимости”...

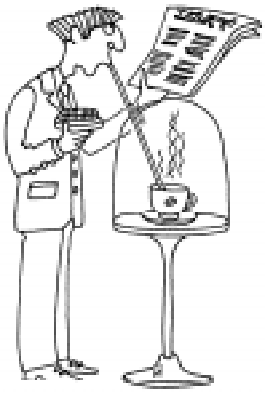
Существенным дополнением к языку Java и базовым средствам JDK стала разработанная в 1996 г. технология компонентного программирования на языке Java – Java Beans. Слово «beans» в буквальном переводе означает «зерна», а сам язык Java, по признанию его авторов, получил свое название от сорта кофе. «Кофейные» метафоры использованы и в логотипе Java (стилизованная чашка кофе, из которой идет пар) и в названии утилиты-архиватора jar (официально – сокращении от Java Archive): «jar» означает «кувшин», а «Java beans in a jar» – «зерна кофе Java в кувшине». Web-браузер, разработанный полностью на языке Java фирмой Sun, получил также «кофейное» название – HotJava (буквально – горячий кофе Java).

Технология Java Beans, средства поддержки которой были реализованы уже в 1997 г. в JDK 1.1, продолжает традицию технологий разработки программ из готовых многократно используемых компонент, начатую фирмами Microsoft (технологии COM и COM+), Borland (язык и система Delphi) и Высшей технической школой (ETH) Цюриха (язык Oberon и его развитие Active Oberon). Из готовых Java Beans-компонент, к которым Вы можете добавить и созданные Вами beans-компоненты, с помощью инструмента сборки Java-программ, например, инструментального комплекса BDK (Beans Development Kit) можно легко спроектировать и собрать готовую программу на Java, управляемую событиями и имеющую визуальное представление в виде комбинации элементов *графического пользовательского интерфейса* (*GUI*) – кнопок, меню, окон, рисунков и т.п., отображающих каждую bean-компоненту. Классический пример, который Вы можете легко создать сами с помощью BDK, - простая Java-программа с визуальным представлением в виде двух кнопок «Start» и «Stop» и фигурки жонглера, который начинает или прекращает жонглировать после нажатия соответствующей кнопки.

Начиная с JDK 1.2, в Java-технологию как неотъемлемая часть вошла система Swing – удобный гибкий инструментарий для создания GUI. Ранее в JDK для этой

Из готовых Java Beans-компонент... с помощью инструмента сборки Java-программ... можно легко спроектировать и собрать готовую программу на Java

цели использовался пакет *AWT (Abstract Windowing Toolkit)*, который хотя и обеспечивал независимость от платформы, но был недостаточно удобен и медленно работал. Существенной особенностью AWT является возможность вставки графических подкомпонент без указания их конкретных координат, с помощью так называемых *менеджеров компоновки (Layout Managers)*, то есть набора типовых способов размещения графических подкомпонент внутри компоненты-хозяина. В дополнение к этому в системе Swing, которая была разработана «поверх» AWT полностью на Java, появилась уникальная возможность изменять *стиль визуализации (Look-and-Feel)* динамически, без редактирования и перекомпиляции Java-программы, управляющей GUI. Кроме того, в Swing, по сравнению с AWT, появилось много дополнительных удобных GUI-компонент, а также средства поддержки создания редакторов текстовой информации, представленной как в виде простого текста, так и в виде текстов на наиболее популярных в настоящее время языках разметки гипертекста, используемых для организации Web-страниц и обмена информацией в Web, HTML и XML.



...защита от вирусов, хакеров и других печально известных явлений.

Если к этому добавить, что в Java имеются очень простые в использовании, но мощные и удобные средства сетевого программирования (пакет java.net), позволяющие «выйти в Интернет» и получить необходимую информацию с помощью программы всего в несколько строк исходного текста, без использования каких-либо браузеров, то становится понятным, почему же Java столь привлекательна для программистов.

Для Интернет-программирования на Java с использованием браузеров и Web-серверов используются две популярные разновидности Java-программ, воплощающие обе стороны общей классической клиент-серверной схемы использования Web:

– *апплеты (applets)* – Java-программы специального вида, связанные с HTML-страницами, предназначенные для их анимации и управления ими в ходе просмотра Web-страниц с помощью браузеров, загружаемые через Web и исполняемые на машинах клиентов;

– *сервлеты (servlets)* – Java-программы, предназначенные для реализации Web-серверов и, соответственно, исполняемые на компьютерах-серверах.

Наверняка многих заинтересует, как же обеспечивается в Java-технологии защита от вирусов, хакеров и других печально известных явлений, сопутствующих широкому распространению сетей и Интернет. Ответ на этот вопрос не столь краток и выходит за рамки данной статьи. Интересующихся отсылаю к работам [1, 2, 4].

2. СОВРЕМЕННОЕ СОСТОЯНИЕ

В настоящее время базовые средства Java-технологии (официальное название с 1998 г. – «платформа Java 2»), поддерживаемые и развиваемые фирмой Sun, подразделяются на ряд *изданий (editions)*, каждое из которых поддерживает разработку программ на языке Java для определенного класса задач:

• Java 2 Standard Edition (J2SE) – это наиболее широко используемое издание Java, предназначенное для широкого круга программистов, использующих персональные компьютеры (рабочие станции) наиболее распространенных в мире архитектур – Intel (x86) с операционной системой Windows или Linux, Macintosh с операционной системой MacOS, SPARC с операционной системой Solaris. Именно это издание Java используется миллионами программистов во всем мире. Его основные части – компилятор с Java (javac); виртуальная машина Java, включающая JIT-компилятор и систему HotSpot; реализация Java-библиотек (Java API); ряд Java-утилит, включая уже известную Вам утилиту *jar*, а также утилиту *appletviewer* для исполнения и отладки апплетов без использования браузера.

Программное обеспечение всех новых моделей ... мобильных телефонов написано на Java.

• Java 2 Micro Edition (J2ME, что следует произносить как «Java to me» – «Java для меня») – это «микро-издание» Java, предназначенное для создания на Java программного обеспечения широкого класса столь популярных ныне микроэлектронных устройств – мобильных телефонов, электронных органайзеров, «интеллектуальных» карт (smart cards) и др. Коммерческий успех Java-технологии в настоящее время обеспечивается, главным образом, благодаря именно этому изданию Java, которое приобретается и используется многими ведущими фирмами-производителями микроэлектроники – Motorola (США), Nokia (Финляндия), DoCoMo (Япония) и многими другими. Программное обеспечение всех новых моделей выпускаемых ими мобильных телефонов написано на Java. Трудно поверить, что Java можно «вложить» в миниатюрный мобильный телефон, или органайзер, например, в PalmPilot фирмы 3COM, или телефон Nokia. Однако это именно так. За счет чего это обеспечивается? Разумеется, за счет введения и учета жестких ограничений – на объем памяти, на размер и состав Java-библиотек и даже самого языка Java. Однако при этих разумных ограничениях авторам Java-технологии удалось реализовать на языке Си «мини-виртуальную машину» Java – KVM, а также достаточно широкое подмножество Java API, включающее специальные библиотеки для поддержки создания программного обеспечения мобильных телефонов – MIDP (Mobile Intelligent Device Profile), интеллектуальных карт – JavaCard, и другие.

• Java 2 Enterprise Edition (J2EE) – издание Java, предназначенное для создания больших и сложных распределенных программ – *решений (solutions)*, управляющих работой предприятий, организаций, банков, сложных систем взаимосвязанных электронных устройств и др. Концептуальной и инструментальной основой J2EE являются *Enterprise Java Beans (EJB)* – расширение технологии Java Beans для компонентного программирования решений. Подобные сложные системы включают распределенные базы данных, содержащие информацию о сотрудниках, банковских счетах и т. д.; *бизнес-логику (business logic)* – программные компоненты, реализующие деловые расчеты (например, зачисление средств на банковский счет или подведение баланса); компоненты, управляющие работой всей сети, обеспечивающие передачу информации, ее защиту и т.п. Все это оказывается удобным программировать на Java с использованием J2EE. В качестве интерфейса с базами данных, для генерации и обработки результатов SQL-запросов, в J2EE используется *JDBC (Java DataBase Connectivity)*, реализованный в виде пакета java.sql.

3. ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ

Введение в Java-технологию было бы неполным без хотя бы краткого обсуждения существующих проблем.

В настоящее время фирма Sun обеспечивает сопровождение и развитие базовых средств Java-технологии – языка Java (который развивается очень медленно), компилятора Java, JVM, Java API (который, напротив, разрастается гигантскими темпами, что делает все более и более сложным его сопровождение и своевременное исправление ошибок), JDK, Forte for Java – интегрированной среды для работы на Java, являющейся развитием системы NetBeans (последняя создана небольшой одноименной фирмой из Чехии, в настоящее время являющейся частью фирмы Sun).

Всего этого, разумеется, явно недостаточно. Не хватает удобных (как универсальных, так и специализированных) интегрированных сред для программирования на Java, Java-библиотек и многого другого.

В связи с этим, практически с момента появления Java целый ряд крупных

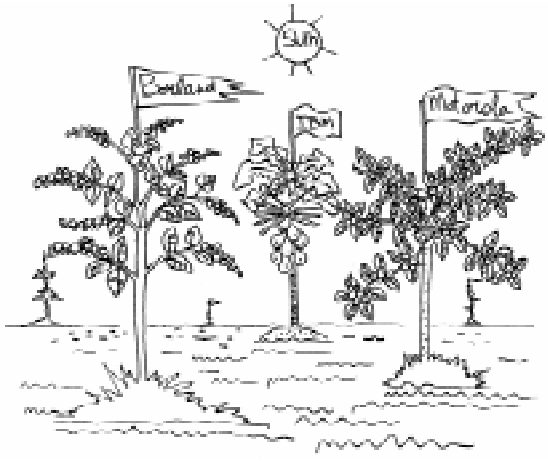
фирм, наряду с Sun, активно занимается развитием инструментов Java-технологии. Следует признать, что в значительной степени именно благодаря их разработкам Java-технология продолжает оставаться столь популярной.

Прежде всего следует упомянуть фирму Borland, известную своими замечательными интегрированными средами – Turbo Pascal, Delphi и др. Именно фирма Borland (а не фирма Sun) создала, по признанию многих специалистов, наиболее удобную интегрированную среду для работы на Java – Borland JBuilder, которую я и рекомендую использовать.

Активно поддерживает и развивает Java-технологию самый мощный гигант компьютерной индустрии – фирма IBM, которая, как и Borland, имеет собственную реализацию Java и использует ее в своих программных продуктах на собственных аппаратных платформах в самом широком круге приложений.

О поддержке и развитии Java-технологии фирмой Motorola уже говорилось.

По информации, полученной на недавней международной (проводимой ежегодно) конференции JavaOne, организуемой фирмой Sun, в настоящее время разработано не менее 30 (!) интегрированных сред для работы на Java. Из них в России известны и используются практически лишь Borland JBuilder и Forte for Java.



...целый ряд крупных фирм ... активно занимается развитием инструментов Java-технологии.

Как же взаимодействуют фирмы, разрабатывающие собственные реализации Java, с фирмой Sun? Каждая такая фирма покупает у Sun лицензию на право создания собственной реализации Java, приложением к которой является набор *сертификационных* программных средств, разработанных Sun и предназначенных для тестирования реализаций Java на соответствие спецификациям языка Java, JVM и Java API. Этот программный инструмент получил название *JCK (Java Compatibility Kit)* и содержит несколько десятков тысяч (!) тестов. Только в случае, если *все* эти тесты проходят успешно, фирма-разработчик собственной реализации Java получает, согласно купленной у Sun лицензии, право использовать в названиях, сообщениях, элементах GUI и документации по своей реализации термин Java (который является зарегистрированным товарным знаком).

Следует признать, что еще более широкому распространению и развитию Java-технологии препятствует ряд обстоятельств, – прежде всего, отсутствие международных стандартов на элементы Java-технологии (работа по их созданию ведется, и можно надеяться, что вскоре они будут приняты), а также вынужденная ограниченность самого подхода, при котором жестко фиксируется как входной язык разработки (Java), так и промежуточный язык (Java-байткод). При необходимости дополнения Java-программы модулями, разработанными на других языках, последние оформляются в виде так называемых *платформно-зависимых методов* (*native methods),* что не вполне адекватно, так как эти модули могут быть написаны на языках Си или C++, имеющих международные стандарты и доступных на самых разных платформах. Выразим надежду, что авторы Java-технологии учтут и решат эти проблемы в ходе ее последующего развития.

В ближайшей перспективе Java-технологии – реализация *параметризованных компонент* (*generics*), что еще более усилит полиморфизм, имеющийся в Java, и даст возможность разрабатывать на Java программные компоненты в более общем виде (например, класс для обработки списков,

параметризованный типом элемента). Подобная идея далеко не нова. Впервые она была реализована еще в середине 1970-х годов в языке абстрактных типов данных CLU [5]. Аналогичная возможность, но в менее безопасной, с точки зрения возможных ошибок, форме (*templates*), имеется в языке C++. Экспериментальная реализация generics для Java уже существует. На сайте java.sun.com доступна информация об этом и электронная презентация фирмы Sun, демонстрирующая эти возможности.

В настоящее время налицо конкуренция двух наиболее популярных и современных платформ для разработки программ – платформы Java 2 и платформы Microsoft.NET. У первой – преимущество в возрасте (и, соответственно, более широком распространении); у второй, разработанной лишь в 2001 г., – в более универсальном, общем и широком подходе, обеспечивающем, в отличие от Java, многоязыковое программирование. Основным языком платформы Microsoft.NET является язык C#, который, несомненно, многое унаследовал от Java, но имеет и целый ряд весьма удобных дополнительных возможностей.

Подчеркнем, что *обе* платформы – и Java, и NET – в равной степени важны для всех программистов и заслуживают самого тщательного изучения. Глубокое знание обеих технологий и практическое владение ими является неотъемлемой частью современной программистской культуры и необходимой компонентой высшего профессионального образования системных программистов.

В связи с этим, на математико-механическом факультете Санкт-Петербургского государственного университета в течение более чем 5 лет преподается Java-технология [1]. Читается базовый курс проф. В.О. Сафонова по Java, неизменно привлекающий более 100 студентов с четырех естественных факультетов, проводится базовый спецсеминар по языку Java и Java API для третьего курса и более «продвинутый» спецсеминар по Java-технологии для пятого, выпускного, курса. Элементы языка Java используются даже в более ранних базовых курсах по информатике (1 – 2 курс) и представлению данных (2 курс), что, как выяснилось, вполне себя оправдало. Автор и реализатор этой идеи – доц. А.А. Кубенский.

Подчеркнем, что обе платформы ... в равной степени важны для всех программистов...

В течение двух лет на факультете также читается спецкурс и проводится спецсеминар по Microsoft.NET.

В заключение хочу призвать читателей к самому активному освоению и использованию Java-технологии. Вы имеете уникальную возможность принять участие во всемирном творческом сотрудничестве программистов-энтузиастов языка Java.

Литература.

1. *Сафонов В.О.* Введение в Java-технологию. СПб.: Наука, 2002.
2. Web-сайт Java-технологии фирмы Sun Microsystems: http://java.sun.com
3. *Платт Д.* Знакомство с Microsoft.NET. М.: Microsoft Press, 2001.
4. *Ноутон П., Шилдт Г.* Java 2. СПб.: BHV-Санкт-Петербург, 2000.
5. *Сафонов В.О.* Языки и методы программирования в системе Эльбрус. М.: Наука, 1989.

Сафонов Владимир Олегович,
доктор техн. наук,
профессор кафедры информатики
Санкт-Петербургского
университета.