Java — Lurkmore

Анонимус!
Возможно, по данной тематике ты искал статью: JavaScript.

ACHTUNG! Опасно для моска!

Министерство здравоохранения Луркмора предупреждает: вдумчивое чтение нижеследующего текста способно нанести непоправимый ущерб рассудку. Вас предупреждали.

Эта статья про язык программирования и платформу от Солнца и Прорицателя; Про другие значения см. Жаба.

«Ява? Это когда вместо «Кобол наносит ответный удар» пишут «удар.нанестиОтвет (новый Кобол ())» — вот это Ява. »

— Программисты Lisp

«Saying that Java is nice because it works on all OS's is like saying that anal sex is nice because it works on all genders. »

— Программисты .NET

Жа́ба (moon.) — искажённое название языка программирования Java, наследника C++, откуда выпилили почти все беззнаковые типы данных и указатели. Определённой эмоциональной окраски не несёт, но, ввиду плохого отношения к Java значительной части населения ЛОРа, чаще употребляется в уничижительных выражениях: жабокодер, жабобыдлокодер и т. д. — это значение было популяризовано Луговским. Алсо, жабонебыдлокодеры имеют зарплату, на которую можно купить не только хлеб, воду и сало, как похапекодерам. А некоторым хватает ещё и на чёрную икру, Феррари и 25-летний виски.

Наиболее заметная особенность языка — крайне низкая скорость работы, связанная, в основном, с некоторыми особенностями национального программирования.

Серьёзный бизнес

История


Естественная среда обитания

«Посадить двуногого автомата, можно нескольких, набирать счета вручную. И никакой жабы не надо. »

— Программисты 1С

«3 billion devices не могут ошибаться. »

— Oracle

Есть версия, что миф о «тормознутости» Java еще связан с тем, что в доисторические времена в браузерах были апплеты, которые адски тормозили. Открытие странички с Java-апплетом вызывало подвисание браузера на минуту-другую, а в строке состояния появлялась надпись: «Приложеньице загружается…» — что не могло не радовать пользователей. Особенно цинично это выглядело, когда причиной тормозов был апплет с анимированной кнопкой «e-mail» размером 100 на 50 пикселей. Были даже спецпрограммы, чтобы генерировать такие апплеты для говносайтов.


То, чего не было.

Реклама тех лет преподносила жаббу как волшебную таблетку, как серебряную пулю, как избавление от страданий: «Life is too short to spend rewriting code». Обещалось также, что исходные коды станут собственностью разработчиков, они смогут продавать их, получая отчисления, да такие, шо все бабы будут теч. Корпоративный коммунизм не наступил, а исходные коды апплетов просуществовали ровно два года, с 1997 по 1999, после чего жабба уползла в мир Энтерпрайзных Серверов, а те немногие, кто по молодости верил в рекламные бредни, херачили интернет-магазины на решётках да на Пыхе по 8 часов в день 40 часов в неделю: в нирвану их так и не взяли.

В мире тяжёлых серверов жаба мутировала настолько, что изменились даже классы для работы с датой и временем, а некоторые методы в них исчезли. Кроме того, в доисторические времена у жаба-машин не было JIT-компиляции, и они выполняли байт-коды последовательно, команда за командой. Но теперь, при обращении к участку кода, который был вызван уже с десяток тысяч раз (как известно, 99% всего времени выполняется 0,1% кода), его стали компилировать в реальный машинный код. В итоге, жабба теперь ещё сильней тормозит при запуске… но зато потом уже работает очень быстро. В том числе и на новых процессорах, под которые JIT создаёт максимально оптимизированный код (обычные

компиляторы — как правило создают один код, который совместим с большинством существующих процессоров).

Достоинства

Попадание в энтерпрайз имело и хорошие стороны. Прежде всего:

- Долгий цикл поддержки: хеллоу ворлды на всех новых версиях обратно совместимы вплоть до java 1.0.2 (да, это камень в сторону .NET). Для программ сложнее обязательно всплывет десяток деталей реализации например.
- Полная кроссплатформенность: работает на Windows, Linux, Mac OS X, BSD, Solaris, AIX, HP-UX, QNX, канувшим в Лету Irix, OS/2, а также совсем безумные реализации типа NanoVM для микроконтроллеров — подо всё есть совместимая реализация JDK, причём правильно написанная программа будет под всем этим работать одинаково. Естественно доступный функционал будет пересечением возможностей выбранных платформ. Вряд ли получится поиграть в написанного на JavaFX сапера на стиральной машине без долгого допиливания.
- Наличие макро-языков: NetRexx, Clojure, Groovy, Scala (функциональный язык широкого назначения), Kotlin, Ceylon и таких библиотек, как Quercus, позволяющих повторно использовать код, написанный ранее на скриптовых языках.
- Безопасность: такие вещи, как SQL Injection (разумеется, если не использовать SQL со строковыми параметрами), падения в корку при неправильной работе с памятью и ряд других «казалось бы мелочей» — в жаббе отсутствуют. Какой ценой та безопасность была достигнута — отдельный вопрос, но в мире корпораций (и в мире бумажных денег) это *никого не интересует*.

Жабба версий 1.2 и далее заточена на написание и поддержку сетевых приложений, которые по технической сложности могут превосходить самотрансформирующуюся шатл-субмарину с минимальными затратами на отладку и каждодневное латание дыр. Достигнуто это за счёт таких её особенностей, как:

- Принудительное ООП. В правильных (то есть выпрямленных суровой необходимостью) руках, управляемых средней тупости головой (этими качествами обладают большинство прогеров на западе), ООП приводит к большей модульности кода с меньшим числом концептуально лишних зависимостей.
- Программы пишутся не под настоящий процессор и исполняются не на физической машине, а в виртуальной, которая заточена на поддержку именно таких программ, что разом устраняет тонны матана, брэйнфака и задротства из проектной документации и списков требований к работникам (но не надо думать, что их остается так уж мало).
- Существует поверие, что на SPARC-серверах Sun/Oracle с установленной солярой java-байткод исполняется аппаратно и даже опережает в производительности C/C++ чуть более, чем в два раза.
- Отсутствие указателей, адресной арифметики и беззнаковых целых, что исключает как класс примерно 93,7438935621% всех ошибок и уж точно ВСЕ самые сложные и дорогостоящие из них (ошибки общего дизайна продукта здесь не к месту, так как такими вещами рядовые кодеры не занимаются). ЧСХ, на практике оказывается, что никакой нужды в адресной арифметике нет.
- Тысячи их!

Этим всем жабба бессовестно и нагло лишила программинг романтики и обеспечила возможность добиваться сносного результата в приемлемые сроки даже с кудрявыми руками и не очень мощным мозгом. Чем, конечно же, вызвала неприятие Ъ гиков.

Жабба — язык для создания серверных приложений. Никто не пишет на ней тру-GUI-приложений на стандартных библиотеках AWT/SWING и нет в этом особой необходимости. Зато ещё как пишут распределённые вычислительные системы, разворачивают большие интернет-магазины, порталы и прочие вещи, на которых зарабатывают деньги. А всё потому, что это надёжно, просто и безопасно.

А ещё

А главное, и, пожалуй, единственное реальное достоинство — востребованность на рынке труда, и, как следствие, оплачиваемость этого самого труда, особенно в международных корпорациях: при весьма низком пороге вхождения за 2-3 года опыта вполне можно получить зарплату на 50% выше средней в IT-индустрии, причём востребованность работодателем не падает уже которые годы (говоря проще, берут любой неадекват, часто с минимальным опытом, лишь бы умел набивать хоть как-то работающие строки кода). Как нетрудно догадаться — это достоинство запросто перекрывает и тормоза, и отсутствие метапрограммирования и этих ваших беззнаковых целых. Впрочем, индусы получают не больше PHP'шников, а для более-менее приличной зарплаты нужно выучить OVER 9000 фреймворков.

НЕНАВИСТЬ!

Основными претензиями в сторону чудо-языка были, есть и будут:

- Не слишком вменяемые стандартные библиотеки, тянущиеся ещё с 1-й версии. На самом деле, это проблема всего ООП, где ставят существительные перед глаголами и где описание предметной области важнее, чем описание действий в ней.
- До сих пор указатели (только теперь они называются «ссылки на объекты»), до сих пор глобальные переменные (только здесь они были названы «поля в объектах»), что по прежнему вызывает зависимости в коде и затрудняет отладку — но надо учесть, что это не Java-машина создаёт несколько ссылок на одни и те же данные или циклические ссылки между ними, а кое-кто другой.
- Отсутствие нормального метапрограммирования. Функций высшего порядка не было (сейчас уже есть, в 8-й версии запилили лямбды и функциональные интерфейсы): вместо них — па́ттерны. В версии 1.5 запилили обобщения и аннотации, но они не решают всех проблем. И количество ключевых слов в языке всё растёт и растёт, а выразительность языка — отнюдь нет. Впрочем, метапрограммирование — штука для тех еще месье и на практике требуется в исчезающе малой доли процента случаев, да и там при желании можно обойтись автогенерацией кода.
- Сборщик мусора — в силу своей тупоголовости на машинах с недостаточным количеством этого вашего ОЗУ может запросто начать собирать мусор в swap-е, вызывая эпические тормоза и задержки; в системах с многими гигабайтами ОЗУ в силу изначальной дефективности модели сборки мусора — также может на несколько десятков (!) секунд отправить весь сервер в задумчивость, собирая мусор среди миллиардов объектов, тем самым вызывая вполне себе неиллюзорные отказы запросов клиентов по time-out; этот же сборщик мусора эпизодически вызывает весьма неиллюзорные залипания на несколько секунд сред Eclipse, Idea, Netbeans (все останавливается, код набивать нельзя, пока мусор не соберется), правда Java кодерам пофиг, они привыкли. Этот же гребанный сборщик мусора является причиной «залипания» и «дерганности» интерфеса Android: так как в андроиде всё, что отображается и нажимается на экране — всё пропускается через Java-код, то внезапные активации сборщика мусора могут запросто приостановить на доли секунд обработку действий пользователя и отображение результатов. Справедливости ради стоит отметить, что

этой фигней со сборщиком мусора страдают и PHP, и Python/Ruby, только там эта проблема обычно стоит не так остро, так как большинство web-приложений на этих языках работает не в режиме демона.
- Необходимость тонкой оптимизации, — как кода, так и настроек JVM, — для получения приличной скорости работы. Да, она возможна, но часто нетривиальна (флажки вроде «noasyncgc») и нереальна (см. выше про гигабайты heap).

Ещё не слишком обоснованные претензии:

- Жаба тормозит (анонимусом проверено, что зачастую over 9000 денег и целого кластера буржуйских серверов не хватает, чтобы стабильно держать несколько серверных приложений).
- Жаба хавает 9000+ МБайт памяти (волшебная JIT компиляция и утечки. Решается это просто: одна программа на Жабе — один сервер, но, будем честными с собой, в особых положениях планет солнечной системы, космическое излучение может парализовать всю работу). Для сравнения — Visual Studio успешно работает на слабом ноутбуке с 4 ГБ памяти, для JetBrains IDEA нужно не менее 16 ГБ.
- Жаба кривая (особенно когда её держат кривые руки).
- Жаба убогая. Да, в ней действительно нет прямой работы с памятью, что можно обойти через sun.misc.Unsafe, но это не соответствует идеологии языка.
- Жаба ооочень медленно развивается, в то время как другие языки обрастают новыми фичами чуть ли не каждый день. Ныне любители фич могут невозбранно перейти на Kotlin.

Алсо, Oracle, нынешний владелец торговой марки, делает только то, что Он хочет. Было бы странно верить в то, что Он любит всех нас и всех нас спасёт. С другой стороны, у все того же Oracle куча проектов на жабе, так что корпорация кровно заинтересована в дальнейшем развитии языка. Истина, как всегда…

О недостатках есть длинная бумага, которую никто не будет читать. А по мнению Никлауса Вирта, Java и JVM были слизаны с его языка Oberon чуть менее чем полностью и испорчены сишным синтаксисом. Честно говоря, у Вирта есть на то основания, но разве это кого-то волнует?…

Связанные мемы

В основном мемы связаны со скоростью работы программ на Java, а также с тем, что это индустриальный язык программирования, точнее, с тем, что на нём пишет большое количество быдлокодеров-индусов. Типичные примеры:

- «*Жаба не тормозит*», «просто gc невовремя запустился», «просто всё остальное слишком быстро работает» и тому подобное по вкусу.
- *«Томми»*, *«убей себя как Томми»*. Применяется в основном к быдложабокодерам. Мем, связанный с тем, что на одной из гонок машин-роботов имени DARPA одна из машин по имени Томми, софт для которой был написан на Java, сошла с дистанции в связи со сбоем чипа, что привело к её убиению об стену. Что интересно, неполадка была чисто аппаратной, но всем похуй, а попадание Томми в стену фанатами Java было радостно воспринято со словами: «Вот видите, Java не тормозит!»
- *«Закат солнца вручную»*. Впарить корпоративному клиенту всемогущее, хоть и монстроватое поделие, которое вскоре потребует фирменного «железа» — очень мудрая стратегия, да.
- *«Серверная Java»*. Используется при обсуждении любых проблем с производительностью жабоприложений. Заявляется, что проблема в десктопной жабе, а с серверными настройками все нормально.

Не тормозит

> Java is high performance. By high performance we mean adequate. By adequate we mean slow.

В холиварах жабакодеры любят приводить в качестве главного аргумента того, что жаба не тормозит, линки на бенчмарки, в которых сравнивается скорость нативного кода и байт-кода. Обычно заголовки таких тестов выглядят в духе «Жаба обгоняет по производительности C++» и приводится код, где Java быстрее C/C++ (нужное подчеркнуть). Это происходит по нескольким причинам:

1. Такие бенчмарки примитивны: они работают только со скалярными типами (с такими, как int, long, double), память под которые выделяется на стеке, используются простые операции и, наконец, весь код находится только в одном классе.
2. Обычно такие тесты содержат много циклов, а замеры проводятся не во время первого запуска, а когда JIT-компилятор уже подставил на место байт-кода команды процессора. Значит, меряется скорость хорошо оптимизированного динамически скомпилированного кода. Да и garbage collector не успевает запуститься в таких программах.
3. Авторы бенчмарков зачастую поверхностно знают C/C++ и используют его в «жабьем» стиле, например используя кучу там, где она не нужна и ноют про необходимость free и delete.

В реальных же приложениях ситуация с быстродействием несколько иная:

1. Все объекты ссылочные и хранятся в куче, а значит, самые элементарные составные типы, такие, как Point, будут каждый раз создаваться в ней. А таких объектов сотни тысяч. Даже при хорошей оптимизации кучи, она медленнее стека, а стоимость алгоритма сборки мусора в общем случае — $O(n^2)$, то есть с ростом числа объектов в памяти затраты на сбор мусора растут нелинейно.
2. Большое потребление памяти: один только хелловорлд при запуске отъедает 10 МБайт (они забираются виртуальной машиной на внутренние нужды), а так как с каждым объектом связаны метаданные (хотя бы ссылка на таблицу методов класса), то какой-нибудь массив объектов класса RGBColor будет занимать в 4-5 раз больше памяти, чем даже хотя бы в C#[1]. На венде, которая свопит всё подряд, это приводит к известным результатам[2].
3. Динамическая природа языка: динамическое связывание кода подразумевает постоянные проверки во время каждого приведения типов (например, при работе с коллекциями). Обобщения здесь не дают ничего, так как они были введены в язык позже, и, для сохранения обратной совместимости, действуют только на уровне исходного кода, в байт-коде их нет.
4. Бросание исключений на каждый мелкий чих, чего язык не требует, но требует идеология и сторонние библиотеки, а выполняются исключения исключительно медленно. Оборотной стороной повышенной надежности является то, что многие программы в процессе работы генерируют over 9000 исключений[3], а исправлять подобный код разработчики не спешат.
5. Другая семантика языка: во избежание воспаления мозга от i++ + i++ были введены правила, устраняющие неоднозначности, что мешает компилятору заниматься оптимизацией. Так, например, на Java оператор вида x+=foo(); в отличие от Си требует запоминания предыдущего значения x перед вызовом foo()[4].
6. Garbage collector: хотя он проходится по памяти нечасто, на GUI приложениях это ощутимо. Алсо, для серверных

приложений с огромным хипом (овер 100G) настройка GC при помощи свичей превращается в ад. И всё равно он тупит.
7. Неотключаемая, если не заниматься извращениями вроде GCJ, проверка индексов при обращении к массивам.

С другой стороны, многие серверные приложения активно используют базы данных, так что большую часть времени занимает обработка запросов на стороне этой базы, так что хоть на чём пиши — разница в скорости будет небольшая. Более того, сервлет на Java может запросто обогнать какой-нибудь кривой cgi-скрипт на Си, потому что сервлету надо инициализироваться один раз, а cgi-скрипту может понадобится стучаться на базу при ответе на каждый запрос для получения каких-нибудь там настроек. Но к энтерпрайзу это как раз и не относится: потянуть тысячи объектов из базы через ORM, чтобы посчитать что-нибудь в цикле — это для них обычное дело.

Не стоит также забывать, что различных реализаций JVM много. Некоторые из них имеют узкую специализацию и умеют очень многое. Например, широко известный в узких кругах Zing от Azul позволяет в 1000+ раз уменьшить время (как единовременное, так и суммарное) блокировок мира на gc, ценой всего-навсего утроенного расхода памяти (минимум для работы 32 Гб), минимум 6 процессоров с поддержкой Intel-VT или AMD-V (только современные серверные модели) и примерно 14000 долларов за годовую лицензию на 1 сервер. Работает только под Linux и требует установки собственного модуля для ядра.

Многие причины тормозов жабы идут от работы с либами, ибо солнцевские пытаются обеспечить универсальность и безопасную среду выполнения. Ситуация усугубляется тем, что в стандартном наборе обычно имеется минимум две, а всяческих коллекций и вовсе зоопарк, библиотеки с одинаковой функциональностью, но различными механизмами. И если взять не ту, возможен былинный отказ. А выбирают их, как правило, «по указаниям технического директора», то есть — наугад.

Алсо, в соревнованиях по программированию алгоритмического толка ACM, авторы задач просто вынуждены удваивать/утраивать ограничение по времени или вовсе устанавливать специальные ограничения для жабы, чтобы каждый жабакодер мог сдать задачу. Для одинаковых лимитов времени часто бывало, что кривой алгоритм на жабе, не укладывающийся в рамки времени и/или памяти, будучи переписанным один-в-один на си, проходил. Обратные случаи неизвестны. Также, в ACM'е каждый жабокодер вынужден писать свой класс для чтения чисел из файла, так как поставляемый со стандартной библиотекой работает слишком медленно.

Выполнение в мире существительных

> «Энтырпрайз-говно не проектируется, а лепится студентотой по хуй-знает-кем-написанным техзаданиям, с целью быть выкинутым и переписанным через пол года после сдачи (хотя сдача энтырпрайз-говна происходит чисто номинально, конечно же). Для джавы это нормально, поэтому мы и не увидим ни одной годной программы, написанной на джаве. »
>
> — *Обсуждение:Программист*

Когда язык разрабатывался, простота кода была одним из самых важных его параметров и это легко увидеть! Например, так Ъ жабакодеры пишут факториал:

Код:

«Так надо»

```
package com.java.brainfuck.ported;

import java.lang.reflect.Method;
import java.math.BigInteger;
import java.util.Arrays;

public class Test {
    static Partial fac = new Partial() {
        public BigInteger fac(Partial partial, BigInteger n) throws Exception {
            return n.equals(BigInteger.ZERO) ? BigInteger.ONE : n.multiply((BigInteger)(partial.apply(n.subtract(BigInteger.ONE)).eval()));
        }
    };
    static Partial s = new Partial() {
        public Partial s(Partial x, Partial y, Object z) throws Exception {
            return x.apply(z, y.apply(z));
        }
    };
    static Partial k = new Partial() {
        public Object k(Object x, Object y) {
            return x;
        }
    };
    public static BigInteger factorial(BigInteger n) throws Exception {
        Partial y = s.apply(s, k, s.apply(k.apply(s.apply(s, s.apply(s.apply(s, k)))), k));
        return (BigInteger) y.apply(fac, n).eval();
    }
    public static void main(String[] args) throws Exception {
        for (int i = 0; i < 51; i++) {
            System.out.println("i=" + i + " fac=" + factorial(BigInteger.valueOf(i)));
        }
    }
}

class Partial {
    private Object[] parameters = new Object[0];

    public Partial apply(Object... objects) throws Exception {
        Partial p = this.getClass().newInstance();
        p.parameters = Arrays.copyOf(parameters, parameters.length + objects.length);
```

```
        System.arraycopy(objects, 0, p.parameters, parameters.length, objects.length);
        return p;
    }
    public Object eval() throws Exception {
        Method method = this.getClass().getDeclaredMethods()[0];
        Object[] current = Arrays.copyOfRange(parameters, 0, method.getParameterTypes().length);
        Object[] rest = Arrays.copyOfRange(parameters, method.getParameterTypes().length, parameters.length);
        Object result = method.invoke(this, current);
        if (rest.length > 0) {
            return ((Partial) result).apply(rest).eval();
        }
        return result;
    }
}
```

Ещё более Ъ вариант:

Код:

```
import java.math.BigInteger;

public class Factorial {

   public static BigInteger factorial(int i) {
        if (i < 0) {
            throw new IllegalArgumentException("Факториал от отрицательного числа");
        }
        BigInteger r = BigInteger.ONE;
        for (; i > 1; i--) {
            r = r.multiply(BigInteger.valueOf(i));
        }
        return r;
    }

    public static void main(String[] args) {
        System.out.println(factorial(30));
    }
}
```

Хотя можно и так:

Код:

```
class b {
        public static void main(String[] args) {
                int n = 5;
                System.out.println("fac:" + n + "!="  + f(n));
        }

        private static int f(int n) {
                switch (n) {
                        case 0:
                        case 1: return 1;
                        default: return f(n - 1) * n;
                }
        }
}
```

Код на Java интуитивно понятен:[5]

Код:

```
class Test1 {
        public static void main(String[] args) {
                Integer a = 127;
                Integer b = 127;
                Integer c = 128;
                Integer d = 128;
                System.out.println(a==b); // true
                System.out.println(c<=d); // true
                System.out.println(c>=d); // true
                System.out.println(c==d); // false

                int[] u = {2,3}, v = {2,3};
                // Неправильно
                System.out.println(u==v); // false
                System.out.println(u.equals(v)); // false
                // Правильно
                System.out.println(Arrays.equals(u, v)); // true
        }
}
```

Или:

```
       System.out.println(1+1+"=1+1");  //2=1+1
       System.out.println("1+1="+1+1);  //1+1=11
```

Фетишизм

Первоначально язык назывался Oak, что как бы намекает на дровосексуальность разработчиков. Потом у языка появился маскот в виде клизмы с глазом. Ничего напоминающего SICP или «Алгоритмы и структуры данных» Никлауса Вирта так и не было написано. И сам язык, и технология (со всеми библиотеками) были с самого начала рассчитаны на корпоративный

рынок, где читателям этих книг, мягко говоря, не место.

Также, были приложены усилия, чтобы Math.sin(Math.PI) возвращала не 1.2246063538223773e-16, а 1.2246467991473532E-16, что доставляет тем, кто хоть что-то понимает в машинной арифметике или метрологии. Похуй, что не полностью поддерживается стандарт IEEE 754, похуй, что нельзя иметь плавучку точнее, чем 53 бита, зато можно воспроизвести погрешности, свойственные технике, которая была на момент разработки языка.

Ведроид

Не так давно (2007 год) Google для реализации плана захвата мира сделал мобильную ОС для КПК и коммуникаторов, назвав её Android. Для исполнения приложений в ОС в Гугл написали свою виртуальную машину по имени Dalvik (начиная с версии 5.0 замененили на ART), оптимизировав её для низкого потребления памяти. Отличается она от солнечной тем, что основана на регистрах, а не на стеке. Причем изначально никакого JIT-компилятора в ней не было и хиленькие процессоры тогдашних смартфонов весело интерпретировали байт-код в реальном времени. Запилили JIT только в версии 2.2, в результате чего скорость выполнения «чистого» кода была увеличена в некоторых случаях до 100 раз, но счастье было омрачено тем, что для хранения скомпилированного исполняемого кода требуется дополнительная память, так что владельцы старых смартфонов как всегда пролетели как фанера над Парижем. Но даже наличие JIT-компилятора не спасало ведроид от адских тормозов в «числодробильных» алгоритмах, так что гугол, скрипя зубами, выпустил таки NDK, позволяющий писать адроидный софт не на жабе, а на кошерном C и C++. В последних версиях ведроида можно вообще отказаться от JIT и скомпилировать все программы раз и навсегда в native-код, что в целом показывает ущербность джавовского подхода к выполнению программ (особенно на «слабом» железе).

А ведь пророк ещё 4 года назад предупреждал:

> Разработчикам систем на GNU/Linux не следует использовать такие технологии как Java и Macromedia Flash, поскольку они не являются свободными
>
> — *Ричард Столлман*

ЛОР не остался в стороне:

> Ужасно. Оракл изящным и быстрым ударом убил джаву. Теперь с неё повалят все. Legioner
>
> Сволочной Oracle! Чтоб они сдохли! Ненависти моей нет конца. zloy_buratino

А разгадка одна — джависты часто любили пугать дотнетчиков, дескать MC подсадит вас на иглу, а там закрутит гайки и начнет доить, а получилось наоборот. Тем временем, альтернативная реализация дотнета — монодроид вполне позволяла писать под Андроид (и, кстати, под iOS, но небесплатно), так что Жабу могут выбросить и оттуда. Майкрософт на радостях гордо заявила, что препятствовать развитию технологий .NET на ведроиде не будет. Правда, монодроид издох, но плагин Mono for Android под Monodevelop позволяет делать то же самое.

J2ME

Жабу заталкивали куда только можно. И сия отрасль является, пожалуй, самой популярной версией жабы среди быдла, но не в плане программинга, а в плане юзинга, ибо круто вечером в подъезде показать пацанам, что у тебя на мобиле есть ещё игры, вроде «Трахни шлюху», которую ты бесплатно скачал, отправив смс на номер (а потом куда-то всё бабло ушло в минус, но это другая история).

Итак, мобильная жаба весьма доставляет тем, у кого не хватает денег или мозгов на смартфон, а хочется игр или софта. Отличается она всё той же мультиплатформенностью, хотя реализация ява-машины у разных контор разная, и несчастным разработчикам, создавая очередной хит на кроссплатформенном языке нужно выкладывать десяток версий для нокий новых, нокий старых, а также эриксонов, сименсов, моторолов и прочей хуиты. Именно из-за мультиплатформенности на яве игр не просто больше 9000, а больше 9000 в квадрате, а то и в кубе.

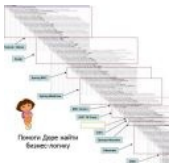
Стоит так же отметить систему безопасности явы на мобилках. Как правило, наиболее опасные функции (отправка смс, выход в инет и т. д.) требуют подтверждения пользователем каждый раз, что, по сути, сделало написание вирусов под эту платформу бессмысленным. Но нашлось куча долбоёбов, которые-таки пишут своих «троянов» и раскидывают их по инету. Самое интересное, что находится ещё больше долбоёбов, которые, запустив какую-нибудь игру или «антивирус касперского», получая запрос на отправку смс, тут же её подтверждают.

Эпичнейшим проектом на яве по праву можно назвать браузер Opera Mini. Им, как ни странно, очень многие пользуются даже на смартфонах, где своих браузеров со свистелками и перделками жопой ешь. А разгадка проста — неебическая экономия трафика + сотни модов, позволяющих творить чудеса.

Что касается скорости сей жабы: разработчики микропроцессора ARM добавили к нему сопроцессор, выполняющий байт-код.

В наше время устарел в связи с распространением дешёвых ведрофонов, которые умеют куда больше за куда меньшие деньги, и уже давно не поддерживается разработчиками или игроделами.

Галерея



Вебкомикс

Шило на мыло

Майка

Где-то в Англии…

Чистая жаба такая чистая (pure java is so pure)

Дьюк старается ради всех нас! Дьюк — против корпораций!

Пчёлы против мёда, а Дьюк — против цехов и гильдий

Facepalm

Видео

Lady Java
Lady Java
The "Java Life" Rap Music Video
The «Java Life» Rap Music Video

Java 4-ever
Java vs Microsoft .NET Trailer
Java RTS Inverted Pendulum Demo
JavaOne 2004
Система реального времени — маятник

Никита Джигурда — Я напишу тебя на Java
Джигурда тоже пишет на жабе

Алсо

- Также «Ява» — винрарная марка чешских байков. А ещё — марка сигарет. А ещё — остров на юг от Сингапура, на котором расположена столица Индонезии Джакарта. А ещё — чай. И кофе, вестимо.
- Яванский человек — ископаемый бизян, хомо эректус.
- Джавы — в звёздных войнах — раса дикарей на планете Татуин.
- Во вселенной людей Икс есть такой злодей Жаба — террорист и непонятая личность.
- Жаба — хорошо известное каждому лично жывтоне, приходящее к нему всякий раз, когда сосед покупает новую мебель/машину/квартиру.
- Кстати, при общении с разносчиками Жаббера следует уточнять, какую жабу вы имеете в виду; иначе случаются когнитивные диссонансы с обеих сторон.

Информация к размышлению

- Квака на жабе (к вопросу о тормозах в разделе выше)
- **b_mahno**/1727 Java, какой я её увидел десять лет назад.
- Опасности обучения на Java
- **mishkaegli**/491000 Java и иврит.

Ссылки

- Выполнение в мире существительных и её перевод.
- Энергонезависимое решение
- Java considered harmful

См. также

- Копипаста:Яваблядь

Примечания

1. ↑ Стоит отметить, что кодер на жабе без энтерпрайза головного мозга в данном случае будет хранить данные в виде массива байтов, как в низкоуровневых языках — но сама жаба делать этого не будет.
2. ↑ Потребление памяти программами на разных языках в shootout
3. ↑ Например на жабе можно легально суммировать массив так: try { for(int i=0;;i++) sum+=a[i]; } catch(Exception e) {}
4. ↑ См. также [1]
5. ↑ Здесь дело в автобоксинге — криво добавленном в версии 1.5 автоматическом преобразовании примитивных типов в обёртки. В версии 1.4 и ранее пришлось бы явно писать `Integer.valueOf(42)`. А объекты оператор == сравнивает по ссылке, а не по значению, в то время как >= автоматически распаковывает эти объекты `Integer` обратно в `int` неявным вызовом `intValue()` (Алсо, для Integers от −127 до 127 созданы статические объекты (для экономии памяти), поэтому a==b возвращает true). И так у них всё — в смысле всё тяжёлое историческое наследие. В этом вашем .NET подобных проблем не возникает (даже строки сравниваются по значению вместо явного вызова `Equals()`), хотя там есть свои тонкости с различиями между структурными и ссылочными объектами.

Языки программирования
++i + ++i 1C AJAX BrainFuck C Sharp C++ Dummy mode Erlang Forth FUBAR
God is real, unless explicitly declared as integer GOTO Haskell Ifconfig Java JavaScript LISP My other car
Oracle Pascal Perl PHP Prolog Pure C Python RegExp Reverse Engineering Ruby SAP SICP Tcl TeX
Хуzzy Анти-паттерн Ассемблер Быдлокодер Выстрелить себе в ногу Грязный хак Дискета ЕГГОГ
Индусский код Инжалид дежице Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помолясь
Проблема 2000 Программист Процент эс Рекурсия Свистелки и перделки Спортивное программирование
СУБД Тестировщик Умение разбираться в чужом коде Фаза Луны Фортран Хакер Языки программирования

Software
12309 1C 3DS MAX 8-bit Ache666 Alt+F4 Android BonziBuddy BrainFuck BSOD C++
Chaos Constructions Cookies Copyright Ctrl+Alt+Del Denuvo DOS DRM Embrace, extend and extinguish
FL Studio Flash FreeBSD GIMP GNU Emacs Google Google Earth I2P Internet Explorer Java Lolifox
LovinGOD Low Orbit Ion Cannon Me MediaGet MenuetOS Microsoft Miranda Movie Maker MS Paint
Open source Opera PowerPoint PunkBuster QIP Quit ReactOS Rm -rf SAP SecuROM Sheep.exe Skype
StarForce Steam T9 Tor Vi Windows Windows 7 Windows Phone 7 Windows Phone 8 Windows Vista Wine
Winlogon.exe Wishmaster Word ^H ^W Автоответчик Антивирус Ассемблер Баг
Билл Гейтс и Стив Джобс Блокнот Бот Ботнет Браузер Варез Винлок Вирусная сцена Генерал Фейлор
Глюк Гуй Даунгрейд Демосцена Джоэл Спольски Донат Защита от дурака Звонилка Интернеты
Кевин Митник Китайские пингвины Костыль Красноглазики Леннарт Поттеринг Линуксоид
Линус Торвальдс Лог Ман Машинный перевод Мегапиксель

ЛОР
12309 128 bit Arch Common Sense Dimmu Borgir - 51k Fedora Generatorglukoff GIMP GNOME
GNOME vs. KDE GuttaLinux Java JB (ЛОР) Just for Fun K48 KDE Komintern Libastral Nixburg Perl
Shaman007 Sherak Sikon Silvy TeX TRUE-DEATH-PRIMITIVE-LINUX-MITOLL Алексей Бабушкин
Анальное рабство Аппрув Апстена Вдоль Великий Исход Модераторов Вендекапец Вещества Вилфред
Ганс Рейзер Грегоре Денис Попов Донской табак Ждём ебилдов Закопайте обратно КЛБ Костыль
Леннарт Поттеринг Летающие коровы Лолкс ЛОР Луговский Лузирс Машина времени Метанация
Михаил (ЛОР) Насиловать труп Не нужен Не работай под рутом Ничего не поделаешь, это Flash Нытик-тред
Плазма не падает Программа из одной строчки на Perl Распечатать лицензию на Линукс Решето Рут Саныч
Скриншот с ЛОР Слака Слешдот-эффект Специалисты по всему Телепаты в отпуске Только закончил собирать
УМВР Шрифты — говно Электролит из соплей девственницы

w:Java en.w:Java ae:Java