Технология Java

курс

Программирование для мобильных платформ

Что такое Java?

✓Остров Ява в Малайском архипелаге, территория Индонезии.
✓Сорт кофе, который любят пить создатели Java
✓А если серьезно, то ответить на этот вопрос трудно, потому что границы Java, и без того размытые, все время расширяются.

*Сначала Java (**официальный день рождения технологии Java — 23 мая 1995 г.**) предназначалась для программирования бытовых электронных устройств, таких как сотовые телефоны и другие мобильные устройства. Потом Java стала применяться для программирования браузеров — появились апплеты.*

Что такое Java?

- ✓ Затем оказалось, что на Java можно создавать полноценные приложения. Их графические элементы стали оформлять в виде компонентов — появились JavaBeans, с которыми Java вошла в мир распределенных систем и промежуточного программного обеспечения, тесно связавшись с технологией CORBA.
- ✓ Остался один шаг до программирования серверов — этот шаг был сделан — появились сервлеты (servlets), страницы JSP (JavaServer Pages) и EJB (Enterprise JavaBeans). Серверы должны взаимодействовать с базами данных — появились драйверы JDBC. Взаимодействие оказалось удачным, и многие системы управления базами данных и даже операционные системы включили Java в свое ядро, например Oracle, Linux, MacOS X, AIX (UNIX-подобная операционная система компании IBM).
- ✓ Что еще не охвачено? Назовите и через полгода услышите, что Java уже вовсю применяется и там. Из-за этой размытости самого понятия его описывают таким же размытым словом — **технология**.

Что такое Java?

- ***Java*** *— сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, с помощью виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года.*[*)]
- *Язык Java активно используется для создания мобильных приложений под операционную систему Android. Разработку приложений можно вести в среде* ***Android Studio***, ***NetBeans***, *в среде* ***Eclipse***, *используя при этом плагин Android Development Tools (ADT), или в* ***IntelliJ IDEA***. *Версия JDK при этом должна быть 5.0 или выше.*
- *8 декабря 2014 года Android Studio признана компанией Google официальной средой разработки под ОС Android.*

[*)] Цитаты из Википедии

Что включает технология Java?

- ✓Язык программирования
- ✓Среда разработки
- ✓Прикладная среда
- ✓Среда развертывания
- ✓Он похож по синтаксису на C ++.
- ✓ Используется для разработки как апплетов, так и прикладных программ.

Что такое технология Java?

Это также:

- ✓ Отсутствие адресной арифметики;
- ✓ Строгая типизация
- ✓ Отсутствие множественного наследования;
- ✓ Встроенная обработка исключений;
- ✓ Сборщик мусора
- ✓ Полная объектная ориентированность, т.е. структурная единица программы- класс – весь код Java-программы должен находиться внутри одного или нескольких классов.

Что такое технология Java?

Это также:

- ✓ Кроссплатформенность
- ✓ Многопоточность
- ✓ Безопасность
- ✓ Встроенная структура коллекций
- ✓ Удобство разработки

Что такое технология Java?

- ✓Кроссплатформенность

Кроссплатформеность

Осуществляется за счет JVM.

- ✓ Исходный модуль, написанный на Java компилируется сразу в машинные команды, но не команды какого-то конкретного процессора, а в команды так называемой виртуальной машины Java (Java Virtual Machine, JVM).
- ✓ Виртуальная машина Java — это совокупность команд вместе с системой их выполнения. JVM полностью стековая, так что не требуется сложная адресация ячеек памяти и большое количество регистров. Поэтому команды JVM короткие, большинство из них имеет длину 1 байт, отчего команды JVM называют байт-кодами (bytecodes), хотя имеются команды длиной 2 и 3 байта. Согласно статистическим исследованиям средняя длина команды составляет 1,8 байта.
- ✓ Полное описание команд и всей архитектуры JVM содержится в спецификации виртуальной машины Java (Virtual Machine Specification, VMS).

Что такое технология Java?

Это также:

- ✓Кроссплатформенность
- ✓Многопоточность

Многопоточность

Java предоставляет средства создания приложений с множеством одновременно активных потоков.

- Большинству современных сетевых приложений обычно необходимо осуществлять несколько действий одновременно. В Java реализован механизм поддержки легковесных процессов-потоков (нитей).
- Многопоточность Java предоставляет средства создания приложений с множеством одновременно активных потоков. Для эффективной работы с потоками в Java реализован механизм семафоров и средств синхронизации потоков: библиотека языка предоставляет класс Thread, а система выполнения предоставляет средства диспетчеризации и средства, реализующие семафоры.
- Важно, что работа параллельных потоков с высокоуровневыми системными библиотеками Java не вызовет конфликтов: функции, предоставляемые библиотеками, доступны любым выполняющимся потокам.

Что такое технология Java?

Это также:

- ✓Кроссплатформенность
- ✓Многопоточность
- ✓ Безопасность

Безопасность

Пока разрабатывался новый язык появился второй фактор, который впоследствии стал играть ключевую роль в продвижении языка Java. Началось бурное развитие Internet вообще и WWW в частности.

Поскольку Internet – это распределенная система, состоящая из компьютеров различной архитектуры, работающих под управлением различных ОС, проблема переносимости программ между платформами встала очень остро. Пользователи Internet хотели бы, чтобы скачанная ими программа выполнялась у них на компьютере независимо от типа CPU и ОС. Кроме того, возникла проблема обеспечения безопасности загружаемых из глобальной сети программ. Такая программа может содержать вирус или троян, который нарушит работу компьютера или получит несанкционированный доступ к частной информации пользователя.

Безопасность

Oracle утверждает, что безопасность Java является одним из основных приоритетов компании. Из-за того, что за последнее время Java не один раз подвергалась критике, связанной с безопасностью этой среды, а также по причине распространения атак, эксплуатирующих уязвимости в ней, Oracle решила усилить безопасность в сфере выпуска исправлений, управления цифровыми сертификатами, работы с неподписанными апплетами и возможностей централизованного управления для крупных корпоративных пользователей и т. д.

Безопасность

Некоторые ситуации, предотвращение которых было изначально заложено в систему безопасности Java.

- ✓Намеренное переполнение стека выполняемой программы — один из распространенных способов атаки.
- ✓Повреждение участков памяти, находящихся за пределами пространства, выделенного процессу.
- ✓Несанкционированное чтение файлов и их модификация.
- ✓Со временем в язык были добавлены новые средства защиты. В версии 1.1 в Java появилось понятие классов с цифровой подписью: пользуясь таким классом, вы получаете сведения об авторе. Если вы доверяете автору, то можете предоставить этому классу все необходимые привилегии.

Что такое технология Java?

Это также:

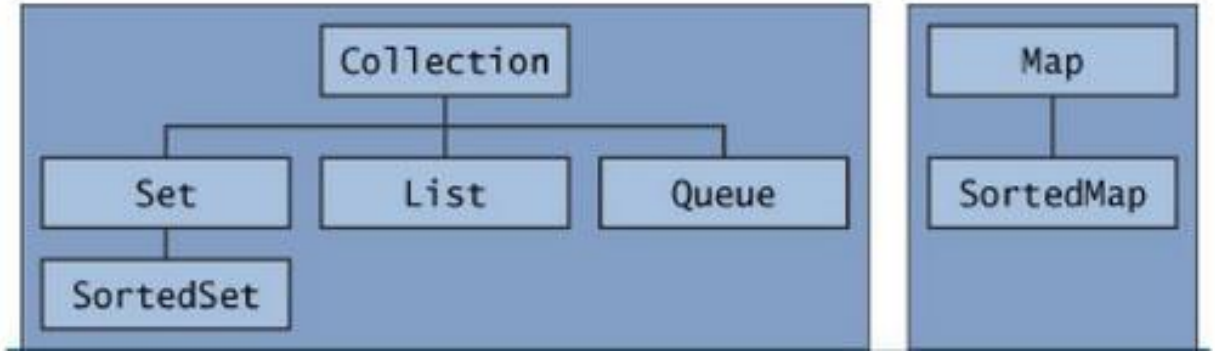
- ✓Кроссплатформенность
- ✓Многопоточность
- ✓ Безопасность
- ✓ Встроенная структура коллекций

Встроенная структура коллекций

Структура коллекций (collections framework) Java стандартизирует способ, с помощью которого программы хранят и обрабатывают структуры данных.

Структура коллекций

Интерфейсы **Реализации** **Алгоритмы**



Встроенная структура коллекций

Преимущества использования структуры коллекций:

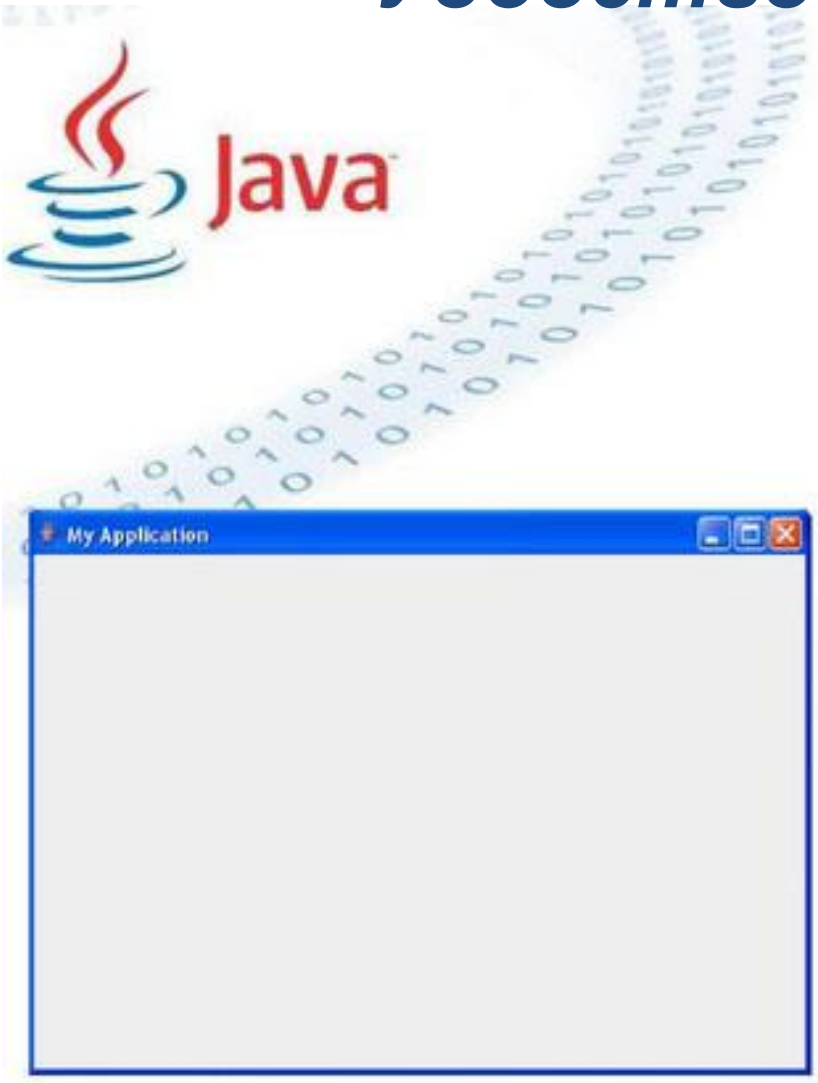
1. **Избавление от рутинных операций по кодированию стандартных структур данных и алгоритмов**
2. **Высокая эффективность реализации**
3. **Универсальность и простота изучения (различные типы коллекций работают похожим друг на друга образом и с высокой степенью способности к взаимодействию)**
4. **Расширяемость**
5. **Параметризация**

Что такое технология Java?

Это также:

- ✓Кроссплатформенность
- ✓Многопоточность
- ✓ Безопасность
- ✓ Встроенная структура коллекций
- ✓ Удобство разработки

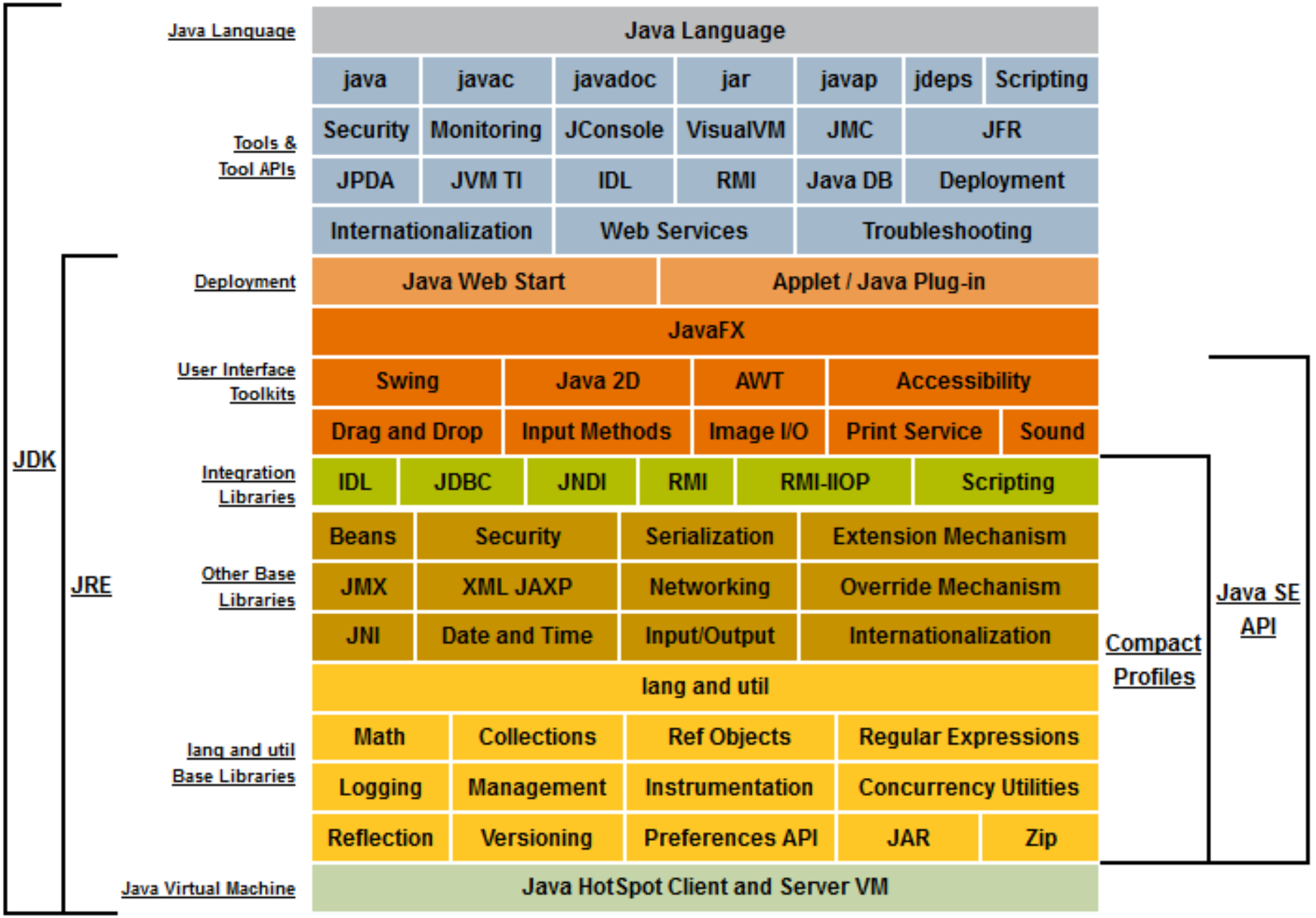
Удобство разработки GUI



В состав Java входят 2 библиотеки, предназначенных для разработки GUI:

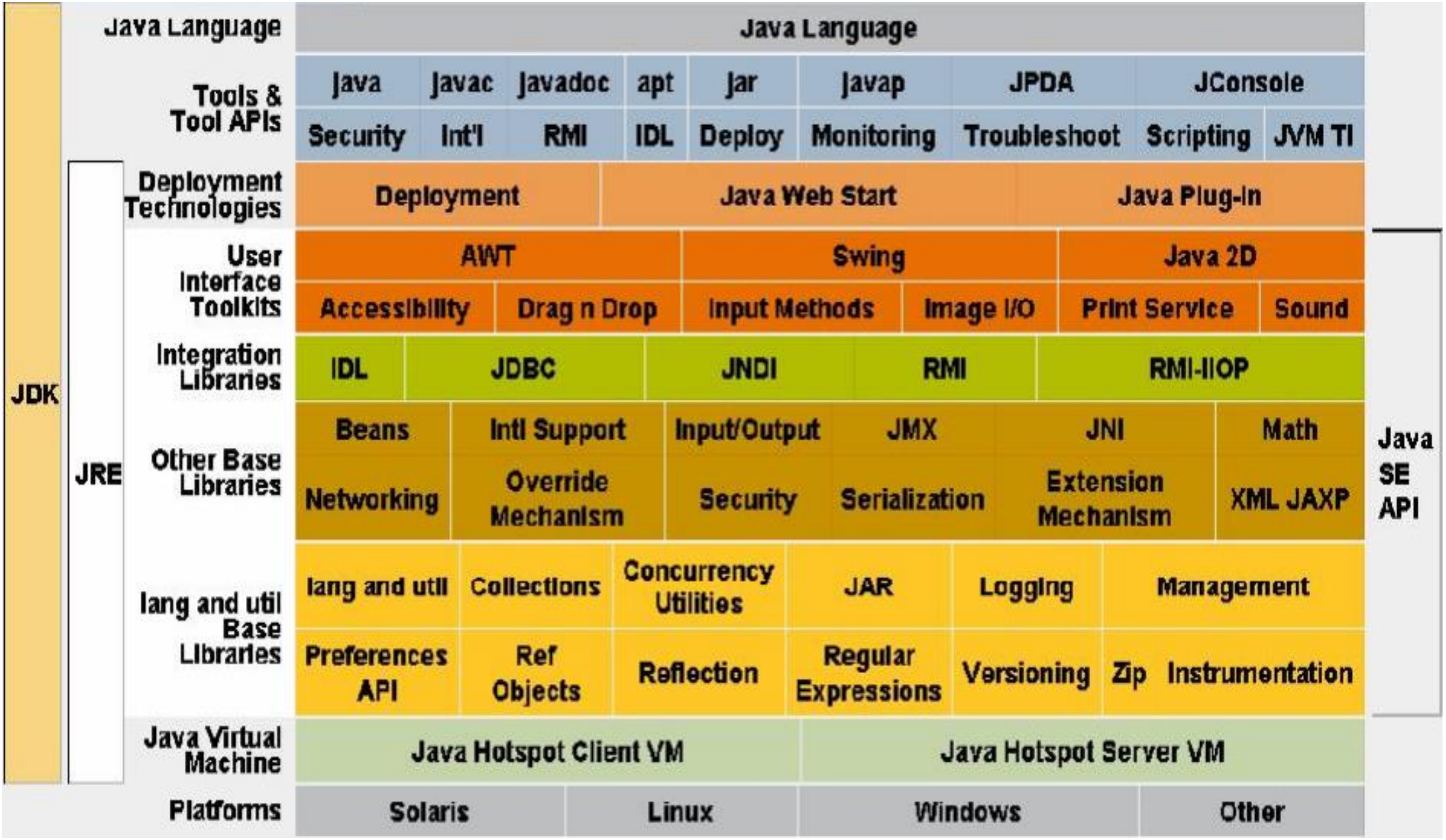
- AWT (Abstract Window Toolkit) – платформно-зависимая библиотека, вывод осуществляется через вызовы OS API
- Swing - платформно-незаивисимая библиотека, реализованная полностью на Java, через OS API выводится только окно, все остальное рисуется средствами Java

Технологии Java SE 8



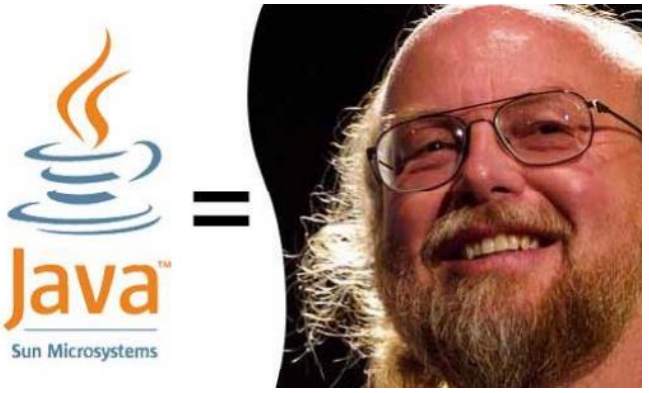
http://docs.oracle.com/javase/8/docs/index.html

Технологии Java SE 6



http://docs.oracle.com/javase/8/docs/index.html

История Java



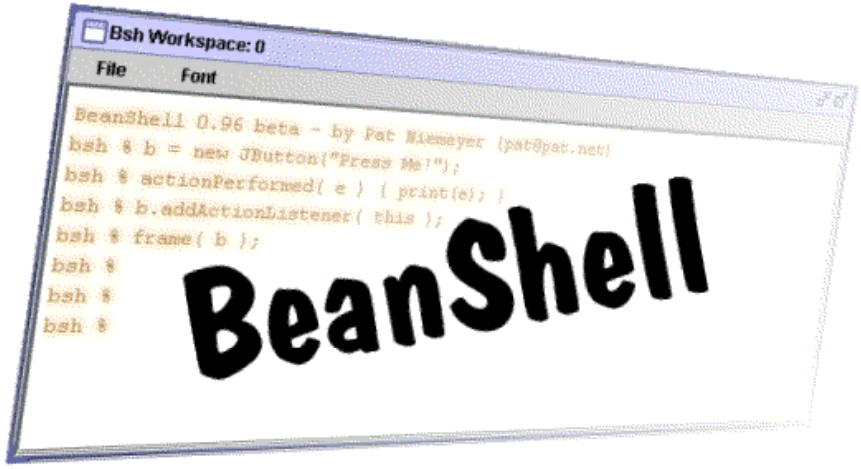
- Создан в 1991 году группой Джеймса Гослинга.
  Команда из шести человек (Джеймс Гослинг, Патрик Ноутон, Крис Варт, Эд Франк, Майк Шеридан)
- Первое название Oak.
  - Переименован в Java, ввиду того, что уже существовал язык программирования Oak.
- Причина создания.
  - Необходимость платформонезависимого языка для встраивания в бытовую технику.

История Java

- **Java** разработан компанией Sun Microsystems (в последующем приобретённой компанией Oracle).
- **5 декабря 1990 г.** "Группе была поставлена задача создать распределенную систему, которую можно было бы в качестве современной программной технологии продавать производителям бытовой электроники," - вспоминает Гослинг.
- Поскольку такие устройства не потребляют много энергии и не имеют больших микросхем памяти, язык должен был быть маленьким и генерировать очень компактные программы. Кроме того, поскольку разные производители могут выбирать разные центральные процессоры (Central Processor Unit— CPU), было важно не завязнуть в какой-то одной архитектуре компьютеров. Проект получил кодовое название "Green".
- 1991-1995г. – маркетинговые изыскания. Нотон сумел доказать необходимость сворачивания работ над телеприставками и концентрации усилий в области онлайновых служб.
- **23 мая 1995 г.** - Компания Sun официально представляет Java и HotJava на выставке SunWorld '95.

Интеграция с другими технологиями

- Ruby
- Python
- JavaScript
- Groovy
- Tcl
- PHP
- …



- http://www.jcp.org/en/jsr/detail?id=223
- http://www.beanshell.org/
- http://groovy.codehaus.org/
- http://tcljava.sourceforge.net/docs/website/index.html

Релизы

- .....
- 1.4.0 Merlin 2002/2/13
- 1.4.1 Hopper 2002/10/16
- 1.4.2 Mantis 2003/5/29
- 5.0 Java SE 5 2004/9/30
- Java SE 6 2006/12/15
- Java SE 7 2011
- Java SE 8 - март 2014 года
- Java SE 9 - сентябрь 2017 года
- Java SE 10 - март 2018 года
- Java SE 11 (LTS) - сентябрь 2018
- Java SE 12 - март 2018

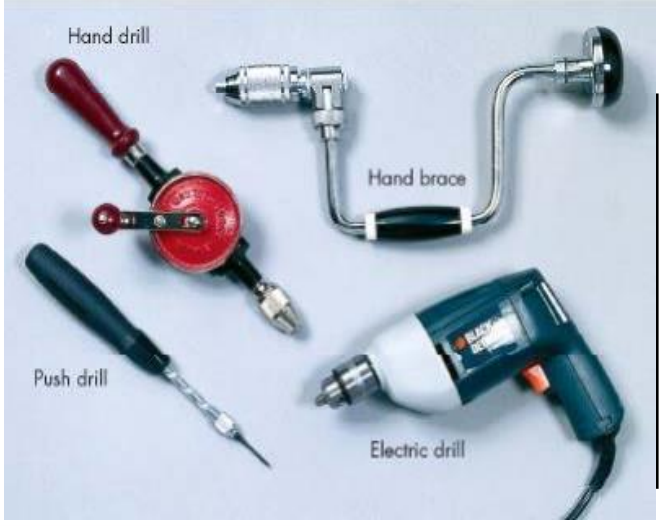
- Выпуск обновлений каждые 8-16 недель

Инструментальные средства

Большая часть инструментария для разработки Java-программ распространяется бесплатно!

1. Java Software Development Kit - Oracle
   Текущая версия Java(TM) SE Development Kit 7, 8 … 12
   http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. RAD – средства разработки:
   - Eclipse Project (open source project)
   - NetBeans (open source project)
   - IntelliJ IDEA (JetBrains)
   - JCreator Pro (Xinox Software)
   - Symantec Cafe (Symantec)
   - Visual J++ (Microsoft)
   - Together (TogetherSoft Corporation)



Итак,

Основными задачами технологии Java являются:

- ✓Предоставление простого в использовании языка путем избегания многих ошибок других языков
- ✓ Объектно-ориентированного языка
- ✓ Предоставление пользователям возможности создания упорядоченного и понятного кода
- ✓ Обеспечения интерпретируемой среды для:
  - ✓ Повышенной скорости разработки
  - ✓ Переносимости (кроссплатформенности) кода

Основные задачи технологии Java

- ✓ Позволяет пользователям запускать более одного потока действий
- ✓Подгружает классы динамически, то есть, тогда, когда они действительно необходимы
- ✓ Поддерживает смену программ динамически во время выполнения, путем подгрузки классов от различных источников
- ✓ Обеспечивает лучшую устойчивость и безопасность

Основные технологии Java

Следующие технологии призваны решить поставленные задачи:

- ✓ JVM – виртуальная машина Java
- ✓Сборщик мусора
- ✓JDK –Java Development Kit – набор программ и инструментов.
- ✓Java Runtime Environment (JRE)
- ✓JVM tool interface – инструментарий машины Java

Теперь подробнее обо всех компонентах технологии Java

Виртуальная машина Java

Виртуальная машина - абстрактное вычислительное устройство, которое может быть реализовано аппаратно или программно. Компиляция в набор команд виртуальной машины (байт-код) происходит почти так же, как и компиляция в набор команд микропроцессора.

Байт-код - машинно-независимый код низкого уровня, генерируемый транслятором и исполняемый интерпретатором. Трансляция в байт-код занимает промежуточное положение между компиляцией в машинный код и интерпретацией.

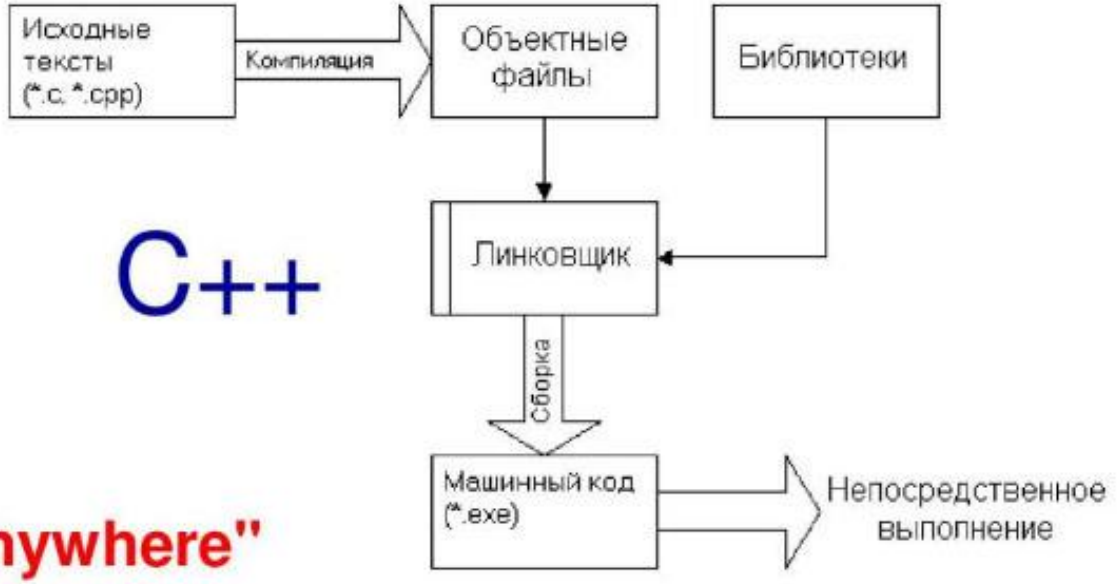
Архитектурная независимость и переносимость кода

Байт-код – архитектурно нейтральный, высокооптимизированный набор команд, предназначенных для выполнения специальной исполняющей системой.

JVM (Java Virtual Machine, виртуальная Java-машина) – исполняющая система, интерпретирующая байт-код

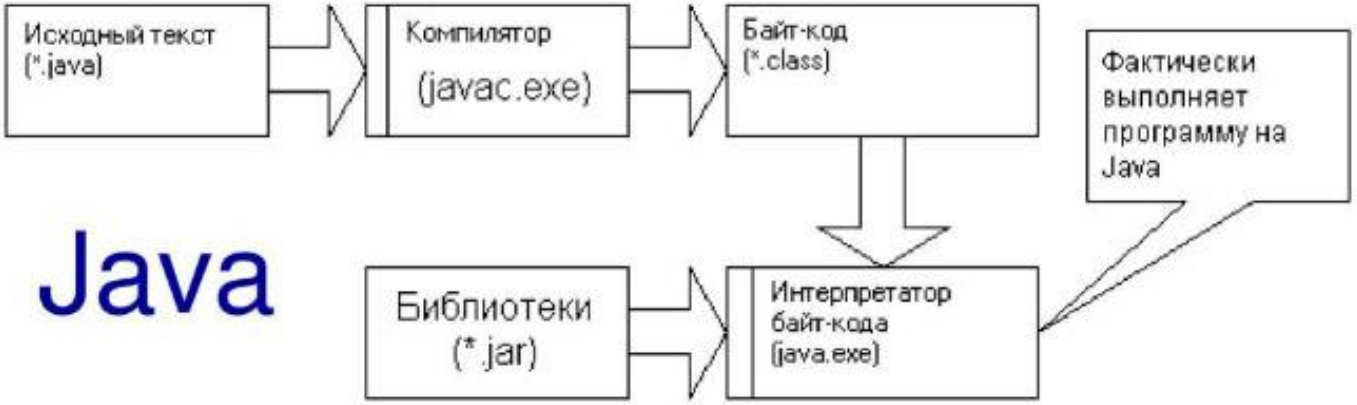
Архитектурная независимость и переносимость кода

Сравним:



C++

"Write Once, Run Anywhere"



Java

Виртуальная машина Java

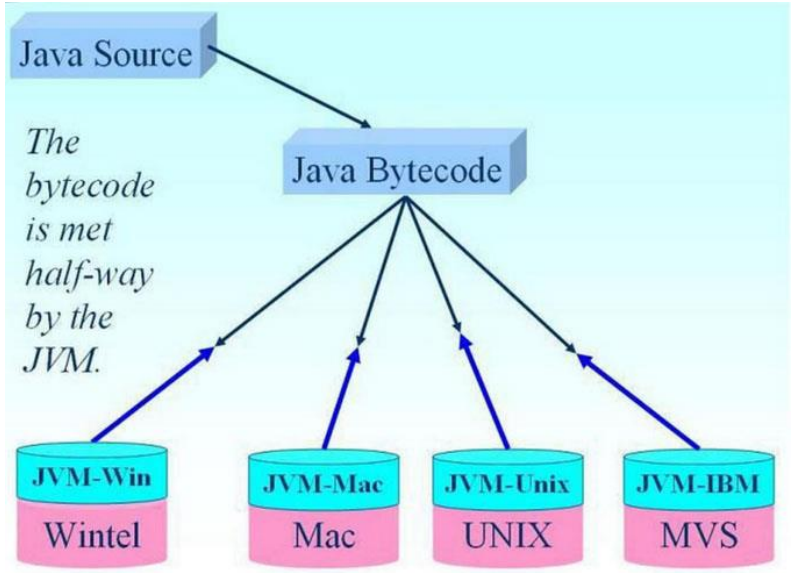
Java Virtual Machine (JVM) - основная часть исполняющей системы Java, т.н. Java Runtime Environment (JRE). Виртуальная машина Java интерпретирует и исполняет байт- код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac).

Использование одного байт-кода для многих платформ позволяет запускать приложения, написанные на Java, на разных платформах.

Основы Устройства JVM специфицированы в 1996 году.

JVM интерпретирует и исполняет байт-код Java. Байт-код может исполняться на любой JVM подходящей версии.

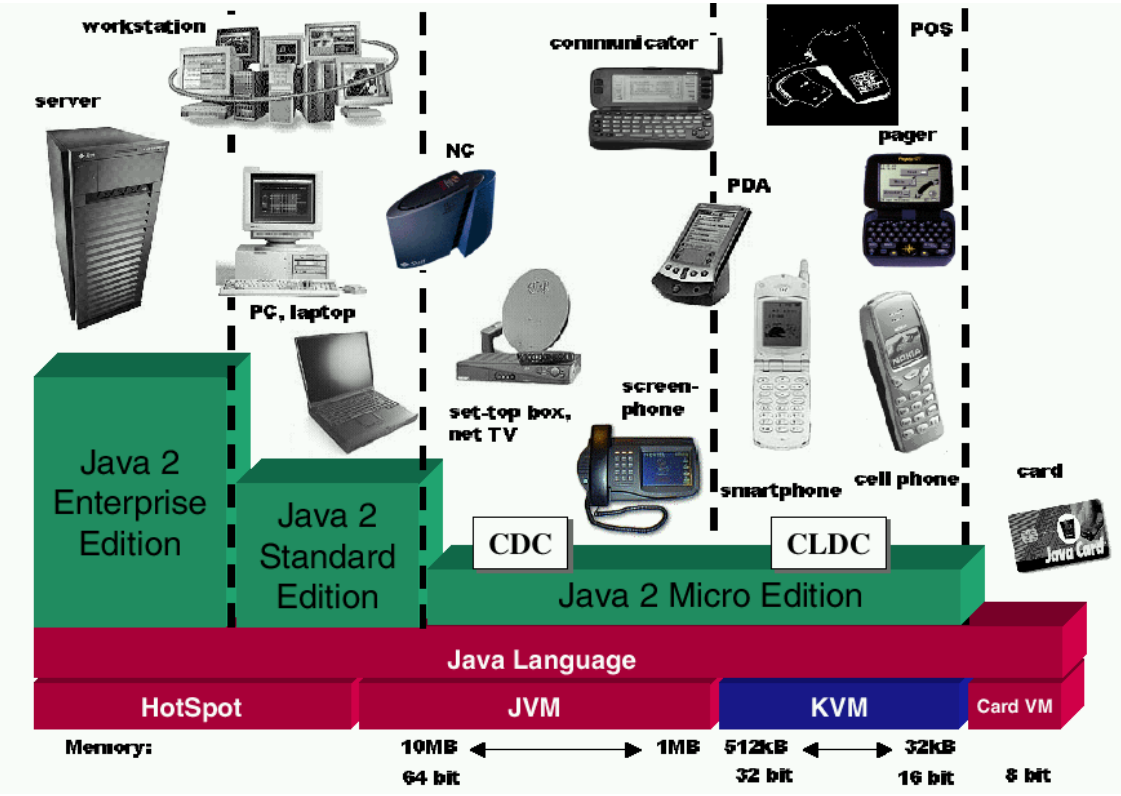
Для каждой аппаратной платформы используется своя JVM.



Разновидности JVM

JVM Sun Micro с открытым кодом

- KVM
  - http://www.cs.tut.fi/~taivalsa/j2me_cldc-1_1-fcs-src-b22-winunix 25_feb_2003.zip
- Squawk
  - https://squawk.dev.java.net/
- Monty
  - https://phoneme.dev.java.net/
- CVM
  - https://phonemeadvanced.dev.java.net/
- HotSpot/OpenJDK
  - http://openjdk.java.net/
- Maxine
  - https://maxine.dev.java.net/



Sun microsystems

Monty

- a.k.a. CLDC Optimized Implementation
- В настоящее время самая распространенная JVM на мобильных телефонах
- Написана на C++ и Java
  - ≈ 25K строк Java
  - ≈ 168K строк C++ (включая компилятор, ромизатор и генератор ассемблерного интерпрептатора)
  - 130K исполняемого кода ARM в минимальной конфигурации, 430K в максимальной
- Ассемблерный интерпретатор, динамическая компиляция
  - x86, ARM с разнообразными расширениями и сопроцессорами
- Целевая компиляция, ромизация, бинарная конверсия, многозадачность

29

Sun microsystems

OpenJDK

- Открытая реализация Java-платформы
  - Полная версия виртуальной машины
  - Полные версии библиотек
  - Поддержка динамических языков (Da Vinci Multi-language VM)
- Написана на C++ и Java
  - Код интерпретатора генерируется в памяти при старте виртуальной машины
- Динамическая компиляция
  - Многочисленные продвинутые оптимизации
  - Скорость сопоставима с оптимизированным C++
    http://blogs.azulsystems.com/cliff/2009/09/java-vs-c-performance-again.html
  - SPARC 32- и 64-bit, x86 64-bit

Пример: Байтовый код JVM

- Стек-ориентированный набор из ≈200 инструкций
- Аналогичен коду Smalltalk VM
  - В Java примитивные типы данных не являются объектами
- Четыре виртуальных регистра
  - ip, sp, fp, lp
- По одному стеку вызовов для каждого потока (thread)
- Стек операндов — часть секции активации
  - Компилятор в виртуальный код должен вычислить требуемую глубину стека операндов
  - Верификатор проверяет достаточность глубины стека операндов

Пример: Байтовый код JVM (2)

- Большинство инструкций однобайтовые
  - Операнды на вершине стека операндов
- Некоторые инструкции извлекают дополнительные 8- и 16-битовые операнды из потока инструкций
- Три инструкции переменной длины
  - Длина инструкции определяется операндами
  - lookupswitch, tableswitch, wide
- Несколько кодов зарезервировано для внутреннего использования VM

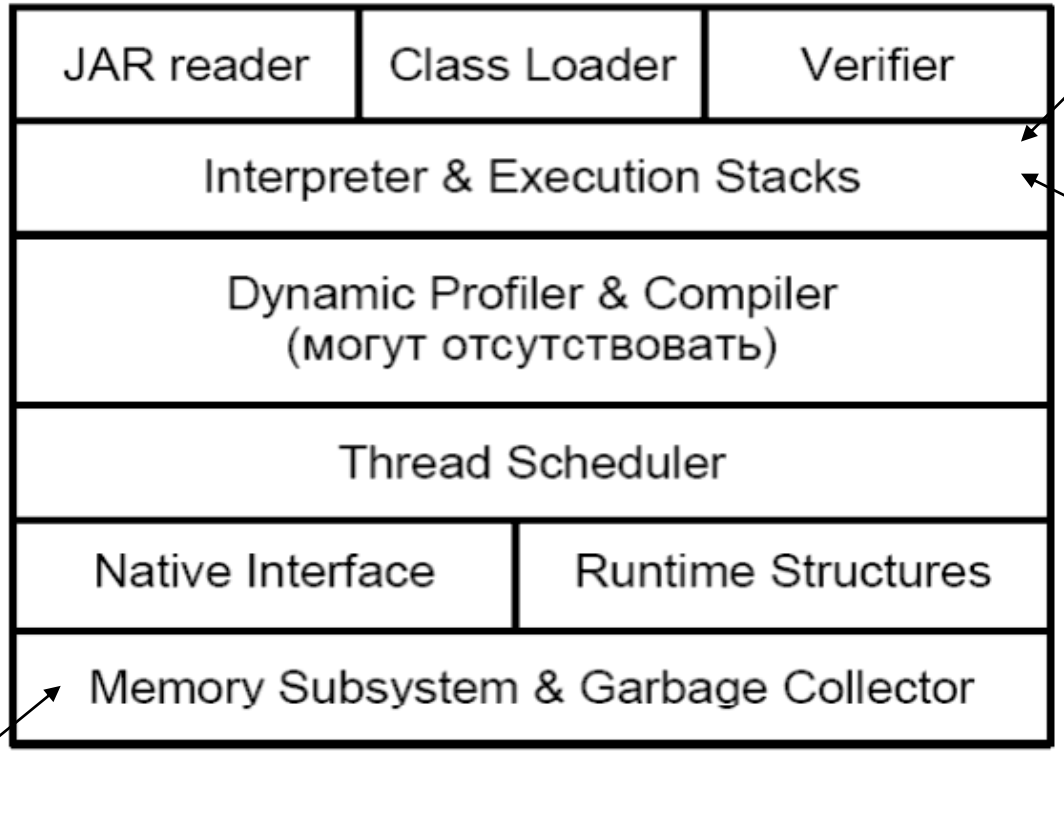
Основные составляющие Java подробнее

- ✓ JVM
- ✓ JRE
- ✓ JDK

Основные составляющие Java подробнее

- ✓ JVM

Компоненты JVM



Интерпретатор байт-кода

Стек для передачи параметров при вызове подпрограмм

Сборщик мусора (Garbage Collector, GC) – постоянно действующий код, освобождающий неиспользуемые куски памяти

Куча (heap) – нераспределенная область динамической памяти предназначенная для размещения объектов

Сборщик мусора - Garbage Collection

Выделенная память, которая больше не нужна должна быть освобождена. В других языках – очистка памяти – забота программиста.

Java предоставляет специальный системный поток для наблюдения за выделением памяти.

Задачи сборщика мусора:

- Автоматически фиксировать освобождение памяти, которая уже не будет использоваться;
- Освобождать память для дальнейшего использования в программах.

Загрузчик классов - Class Loader

✓ Загружает классы по необходимости для исполнения программы;

✓ Поддерживает классы в локальной системе в разных областях видимости (namespace);

✓ Предотвращает spoofing (получение доступа обманным путем).

Верификатор байт кода - Byte code Verifier

Проверяет, что:

- Код соответствует JVM спецификациям;
- Код не угрожает целостности системы;
- Код не является причиной переполнения либо потери значимости стека;
- Типы параметров корректны;
- В коде нет недопустимых преобразований типов данных.

Основные составляющие Java подробнее

- ✓ JVM
- ✓ JRE

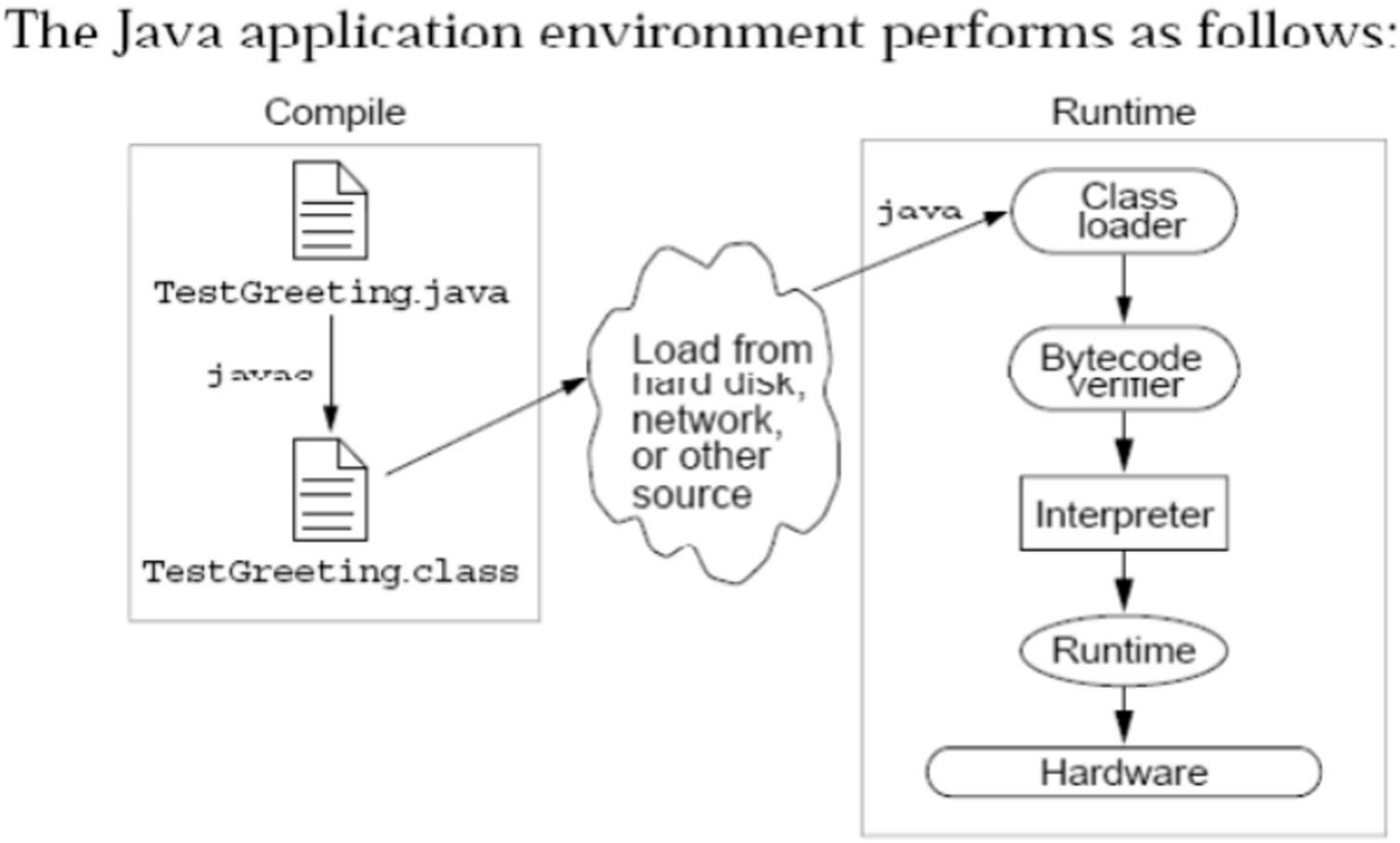
Исполнительная среда Java (JRE)

Java Runtime Environment (JRE) - исполнительная среда Java, в которой выполняются программы, написанные на этом языке. Среда состоит из JVM и библиотеки Java - классов. Это минимальная реализация JVM, необходимая для исполнения Java приложений, без компилятора и других средств разработки.

Именно JRE или его аналог других фирм используется в браузерах, умеющих выполнять программы на Java, операционных системах и системах управления базами данных. Установка JRE является необходимым и достаточным условием для выполнения Java программ. Для разработки программ JRE недостаточно - необходимо установить Java Development Kit (JDK), который может установить и JRE и дополнительные компоненты.

Исполнительная среда Java (JRE)

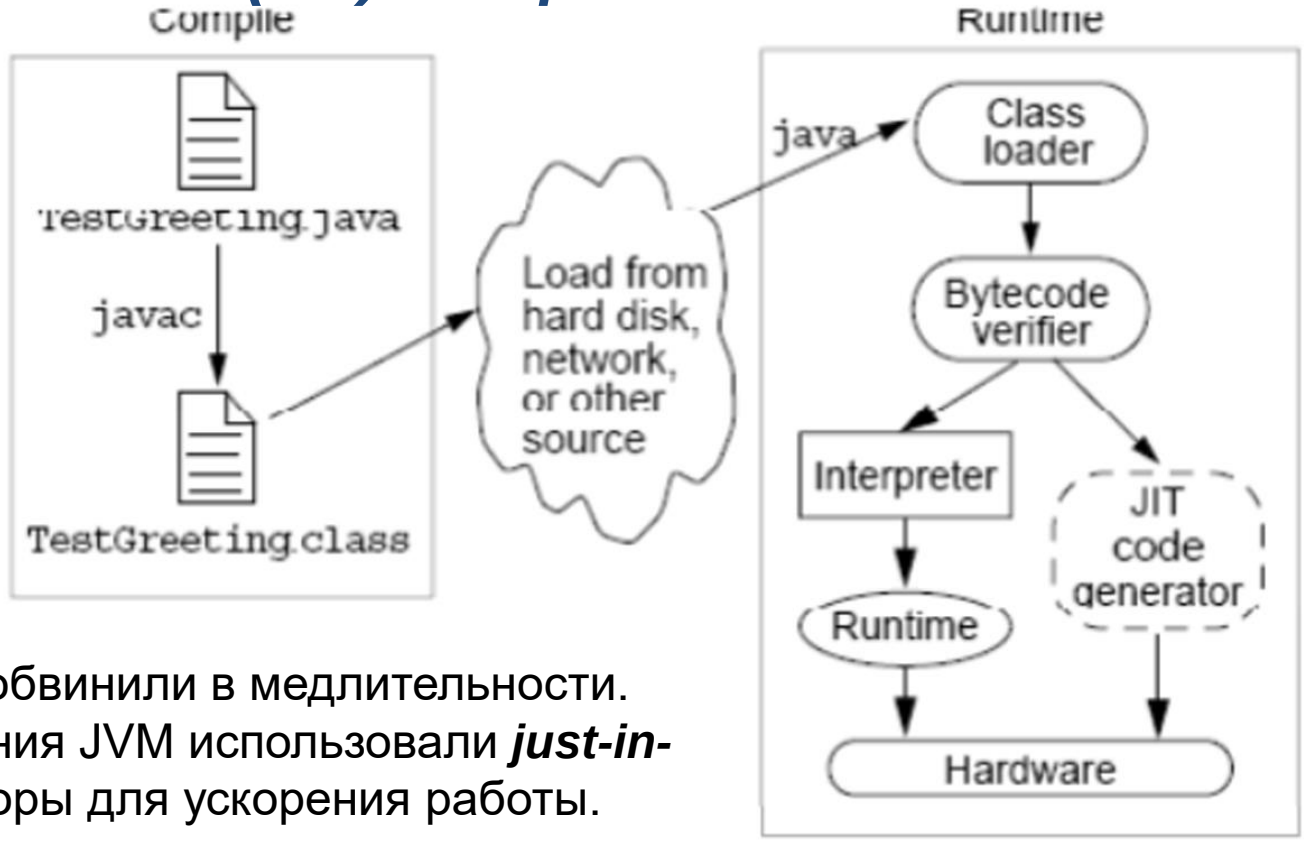
The Java Runtime Environment



Первые поколения JVM были полностью интерпретируемыми. JVM интерпретировала байт-код вместо компиляции его в машинный код и выполнения машинного кода.

Этот подход, естественно, не предлагал наилучшей возможной производительности, поскольку система больше времени тратила на выполнение интерпретатора, а не самой программы.

Operation of the JRE With a Just-In-Time (JIT) Compiler



Первые JVM сразу обвинили в медлительности. Следующие поколения JVM использовали ***just-in-time (JIT)*** компиляторы для ускорения работы.

Строго определенная JIT-виртуальная машина перед выполнением преобразовывает все байт-коды в машинный код, но делает это в неторопливом стиле: JIT компилирует code path только тогда, когда знает, что code path будет сейчас выполняться (отсюда и название ***just-in-time*** компиляция). Этот подход дает возможность программам стартовать быстрее, поскольку не требуется длительная фаза компиляции перед возможным началом исполнения кода.

Основные составляющие Java подробнее

- ✓ JVM
- ✓ JRE
- ✓ JDK

Java Development Kit (JDK)

Java Development Kit (JDK) Бесплатно распространяемый корпорацией Sun комплект разработчика приложений на языке Java, включающий в себя:

- ✓Sun компилятор Java (javac),
- ✓стандартные библиотеки классов Java,
- ✓примеры,
- ✓документацию,
- ✓различные утилиты, исполнительную среду Java (JRE).

Набор программ и классов JDK

JDK содержит:

- ✓ компилятор из исходного текста в байт-коды ***javac***;
- ✓ интерпретатор ***java***, содержащий реализацию ***JVM***;
- ✓ облегченный интерпретатор ***jre*** (в последних версиях отсутствует);
- ✓ программу просмотра апплетов ***appletviewer***, заменяющую браузер;
- ✓ отладчик ***jdb***;
- ✓ дизассемблер ***javap***;
- ✓ программу архивации и сжатия ***jar***;
- ✓ программу сбора и генерирования документации ***javadoc***;
- ✓ программу генерации заголовочных файлов языка C для создания «родных» методов ***javah***;
- ✓ программу добавления электронной подписи ***javakey***;
- ✓ программу ***native2ascii***, преобразующую бинарные файлы в текстовые;
- ✓ программы ***rmic*** и ***rmiregistry*** для работы с удаленными объектами;
- ✓ программу ***serialver***, определяющую номер версии класса;
- ✓ библиотеки и заголовочные файлы «родных» методов;
- ✓ библиотеку классов Java ***API*** (Application Programming Interface).

Управление памятью - сборка мусора

- Часто противопоставляется ручному управлению памятью
- Упрощает процесс программирования
- Предотвращает утечки памяти

– Но утечки могут происходить при невнимательном программировании

- Позволяет оптимизировать размещение объектов
- Система с реализованной сборкой мусора, как правило, менее производительна и более требовательна к ресурсам

Сборщик мусора JVM

- Определяет объекты, которые в будущем не будут использоваться
- Освобождает память, занятую такими объектами
- Все время находится в рабочем состоянии
- Влиять на работу сборщика мусора можно лишь косвенно:
  - Рекомендуя выполнять те или иные действия
  - Изменяя параметры запуска

Виды сборки мусора

- Отслеживающий метод – обход графа и определение достижимости объекта
- Использование поколений:
  - При срабатывании сборщика мусора к значению неудаленного объекта прибавляется единица – сколько сборок мусора он пережил
    - Объекты с одинаковыми такими значениями образуют поколение
    - Сборщик мусора с большей вероятностью удаляет экземпляры только некоторых поколений

Ограничения

- На один класс в константном пуле отводится максимум 65535 элементов
- Длина кода метода ограничена 65535 байтами
- Число слов аргументов в вызове метода ограничено 255
- Кол-во операций VM = 256 (можно провести аналогию с RISC- компьютером)

JAVA программа

HelloWorld.java

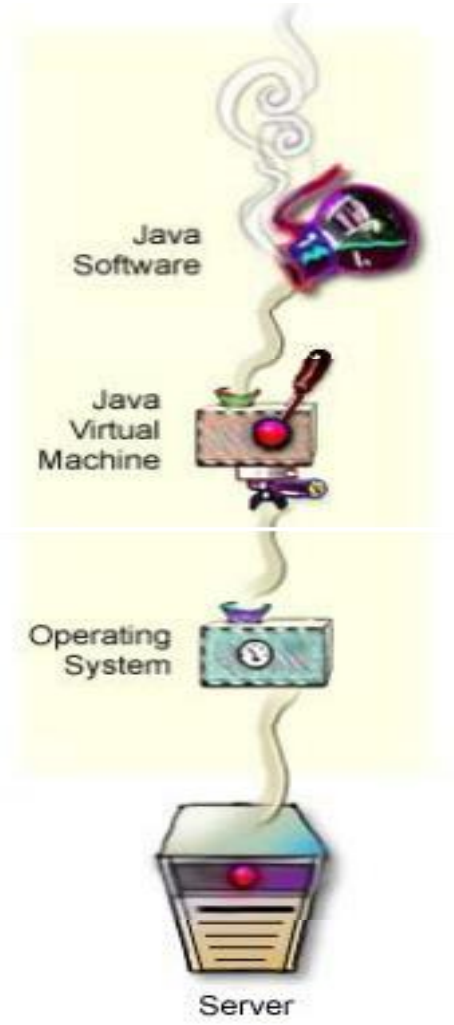
```
1
2  public class HelloWorld {
3
4      public static void main(String[] args) {
5
6          System.out.println("Hello World!");
7
8      }
9
10 }
11
```

файл программы называется точно так же как и класс который он содержит плюс расширение .java. Это обязательное условие, иначе файл просто не скомпилируется. То есть, в нашем случае, файл **HelloWorld.java** содержит класс **HelloWorld**.

Исполнение программы Java

При запуске приложения (класса) виртуальная машина выполняет следующие действия:

- ✓ Загрузку класса по его идентификатору загрузчиком классов (ClassLoader)
- ✓ Верификацию - проверку того, что структура класса верна (инструкции имеют корректные коды, методы имеют сигнатуры)
- ✓ Подготовку - присваивание значений по умолчанию статическим полям класса
- ✓ Инициализацию - выполнение статического инициализирующего блока и присваивание значений статическим полям



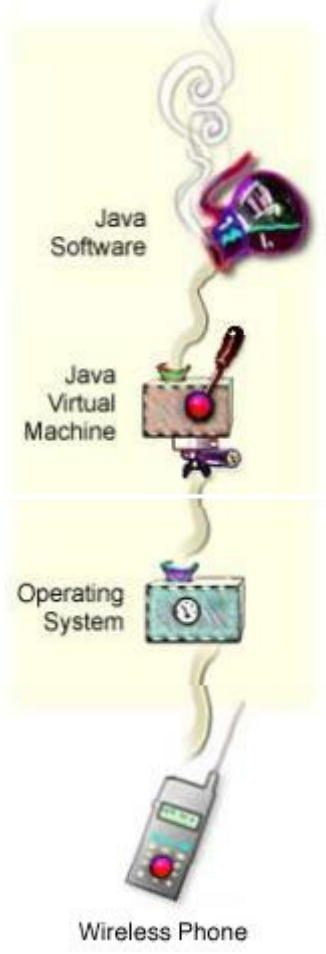
Выгрузка класса

- Класс может быть выгружен, если его загрузчик стал недоступен
- Системные классы не могут быть выгружены ввиду того, что загрузчик системных классов доступен всегда во время работы JVM все время находится в рабочем состоянии

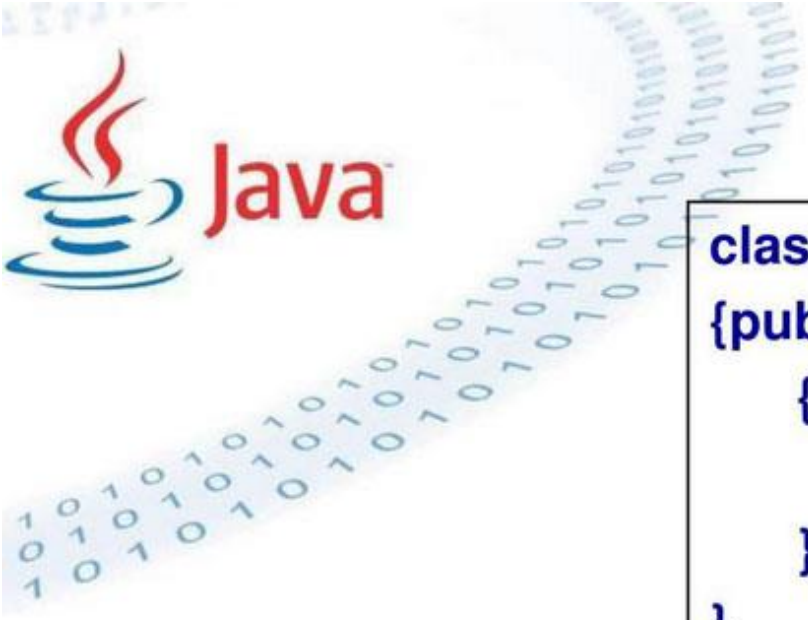
Прерывание работы JVM

JVM прекращает свою работу если выполнено одно из условий:

- ✓ Все потоки, не являющиеся демонами, завершены
- ✓ Один из потоков вызвал метод exit() и данная операция была разрешена менеджером безопасности



Пример программы



Первая программа на Java

```
class FirstProg
{public static void main(String args[ ])
    {
        System.out.println ("Hello, world");
    }
}
```

1. `>javac FirstProg.java` -> FirstProg.class

2. `>java FirstProg`

```
C:\Program Files\Java\jdk1.5.0_08\bin>java FirstProg
Hello, World!
```

Комментарии к программе

Рассмотрим некоторые характерные особенности языка Java на примере небольшой программы.

1. В Java исходный файл называется единицей компиляции ***(compilation unit).*** Это текстовый файл, содержащий одно или несколько определений классов, внутри которых должен находиться весь код программы. Компилятор Java требует, чтобы исходный файл имел расширение .java, а его **имя совпадало с именем одного из классов, описанных в этом исходном файле.**
2. Строка ***class FirstProg*** определяет новый класс с именем FirstProg.
3. Строка ***public static void main(String args[ ])*** – точка начала выполнения программы.

Комментарии к программе

4. Функций main() может быть несколько с разным набором входных параметров и разными возвращаемыми значениями, но стартовать программа всегда будет с той функции, у которой возвращаемое значение void, а набор входных параметров String args[]. Если такой функции не будет обнаружено, то программа успешно откомпилируется, но при попытке запуска возникнет исключение (run-time error) NoSuchMetod.

Комментарии к программе

5. System.out.println (“Hello, world”); - консольный вывод на экран. System – предопределенный класс, обеспечивающий доступ к системе, out – выходной поток, соединенный с консолью, println() – метод, выводящий на экран полученную им в качестве параметра строку и символ перевода строки (newline).
6. Аналогично C++, все инструкции в Java заканчиваются ; .

Для выполнения данной программы:

1. набрать текст и сохранить его в файле **FirstProg.java**
2. С помощью команды **javac FirstProg.java** откомпилировать данный файл. Будет создан файл байт-кода **FirstProg.class.**
3. Выполнить программу с помощью команды **java FirstProg**.

Sun Educational Services

A Simple Java Application

The TestGreeting.java Application

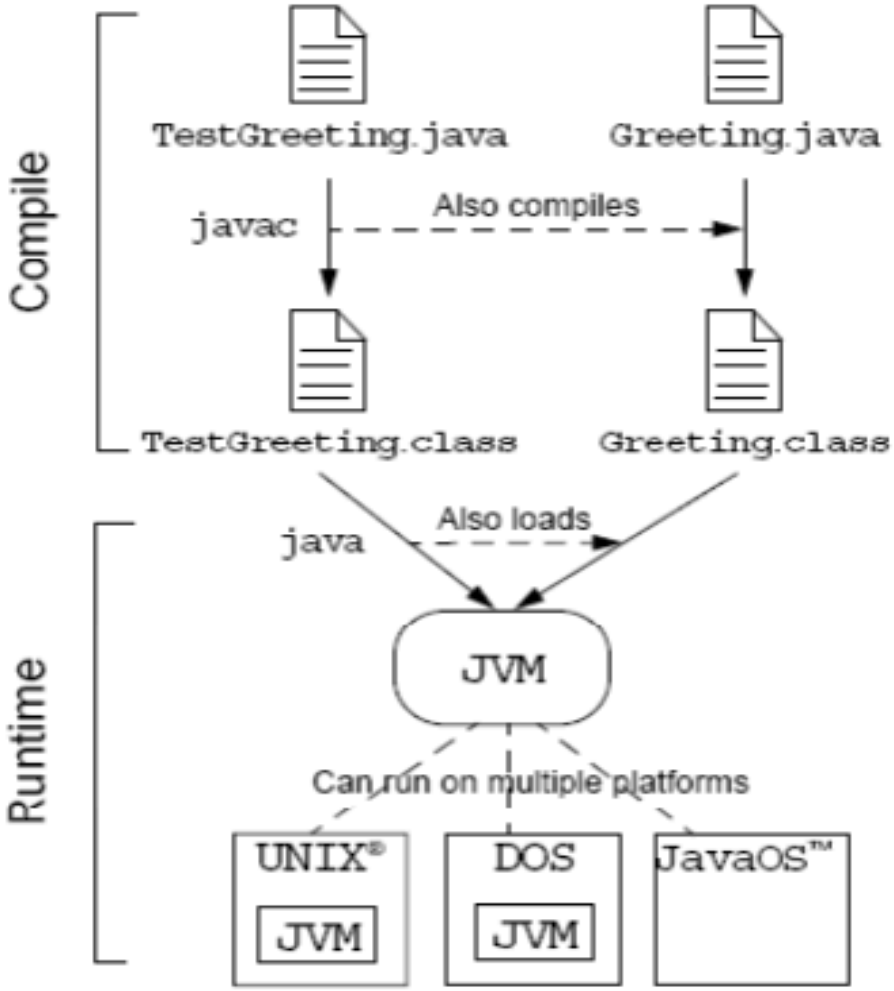
```
1   //
2   // Sample "Hello World" application
3   //
4   public class TestGreeting{
5     public static void main (String[] args) {
6       Greeting hello = new Greeting();
7       hello.greet();
8     }
9   }
```

The Greeting.java Class

```
1   public class Greeting {
2     public void greet() {
3       System.out.println("hi");
4     }
5   }
```

Sun Educational Services

Java Technology Runtime Environment



Литература и источники информации в сети Интернет

Учебный курс Intuit: http://www.intuit.ru/department/pl/javapl/

Официальное руководство Oracle: http://docs.oracle.com/javase/tutorial/

Книги:

1. Г.Шилдт. Java. Полное руководство. - М.: Вильямс, 2012.
2. Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала. - М.: Вильямс, 2010.
3. Б. Эккель. Философия Java. - Спб.:Питер, 2014.
4. Joshua Bloch. Effective Java: Second Edition. - Prentice Hall, 2008.
5. Роберт Лафоре. Структуры данных и алгоритмы в Java. - Спб.:Питер, 2013.
6. М.Гранд. Шаблоны проектирования в JAVA. Каталог популярных шаблонов проектирования, проиллюстрированных при помощи UML. - М.: Новое знание, 2004.

Параметры компиляции

Параметры компиляции можно изменять при помощи ключей компилятора javac:

-cp -classpath	указать путь, по которому можно найти классы, необходимые для компиляции (переменная CLASSPATH)
-bootclasspath	указать путь, по которому можно найти классы, необходимые для запуска JVM
-source	указать версию исходного кода
-target	указать версию JVM, для которой создается класс-файл
-version	вывести версию компилятора
-help	перечень разрешенных опций компилятора
-J	свойство, передаваемое в JVM. Виртуальная машина может изменять свое поведение в зависимости от переданных параметров.
-X	дополнительные опции

Информацию можно найти в презентации JDK_tools.pdf

Параметры запуска

Параметры исполнения можно изменять при помощи ключей интерпретатора, передаваемых java:

-cp(-classpath)	указание пути, по которому содержатся классы, необходимые для запуска
-D<name=val>	установка системного свойства
-X	расширенные параметры
-client/-server	выбор клиентской или серверной модификаций JVM
-d32	Выбор 32-битной модели данных
-d64	Выбор 64-битной модели данных

Исполняемые пакеты

- Классы можно объединять в jar-пакеты.
- Запуск программы из такого пакета происходит быстрее.
- Пакет может содержать ресурсы для приложения.
- Создание архива утилитой jar:

```
jar cf Hello.jar Hello.class
  GoodBye.class
```

Запуск приложения, запакованного в jar файл:

```
java –jar Hello.jar
```

Базовый инструментарий

- javac – компилятор языка Java
- java – интерпретатор байт-кода
- javah - создает заголовочные файлы
- javadoc - формирует стандартную документацию
- jar – создание дистрибутивов Java
- javap – дизассемблер
- apt – обработчик аннотаций
- Другие базовые инструменты (appletviewer, jdb, extcheck)

javadoc

- Генерирует документацию к программному интерфейсу приложения
- Предпосылки:
  - Для обеспечения поддержки код должен быть хорошо документирован
  - Найти нужную информацию непосредственно в коде не всегда просто
  - Нужен инструмент, собирающий разбросанные по коду комментарии и предоставляющий удобную навигацию по ним
- От разработчика требуется:
  - придерживаться несложных правил написания комментариев (чтобы Javadoc правильно их интерпретировал)
  - запускать утилиту Javadoc для создания и обновления документации

javadoc (продолжение)

По умолчанию Javadoc генерирует документацию для:

- Пакетов
- `Public` классов и интерфейсов
- `Public` и `protected` методов
- `Public` и `protected` полей
- При необходимости документировать `private` классы/методы/поля, можно указать ключ -private
- Внутри текста комментария можно использовать специальные тэги
- Стандартные теги HTML допустимы
- Запуск утилиты:

```
javadoc [options] [packagenames] [sourcefiles] [@files]
```

Jar (Java ARchive)

- Утилита для создания дистрибутивов Java программ
- Пример: `% jar cf myApp.jar *.class`
  - Все файлы текущей директории с расширением .class собираются в архив myApp.jar. При этом автоматически создается манифест, содержащий мета-информацию о приложении
- Использует алгоритм Zip
- Также можно сделать JAR файл самораспаковывающимся
  - Запакованное приложение можно запустить двойным щелчком
- JAR файлы могут быть подписаны автором

javap (дизассемблер)

- Разбирает класс-файл. Выводимая информация варьируется в зависимости от используемых опций
- По умолчанию javap выводит название пакета, а также protected и public поля и методы анализируемого класса
- Запуск утилиты из командной строки:

```
javap [ options ] class. . .
```

- Пример вывода информации о классе:

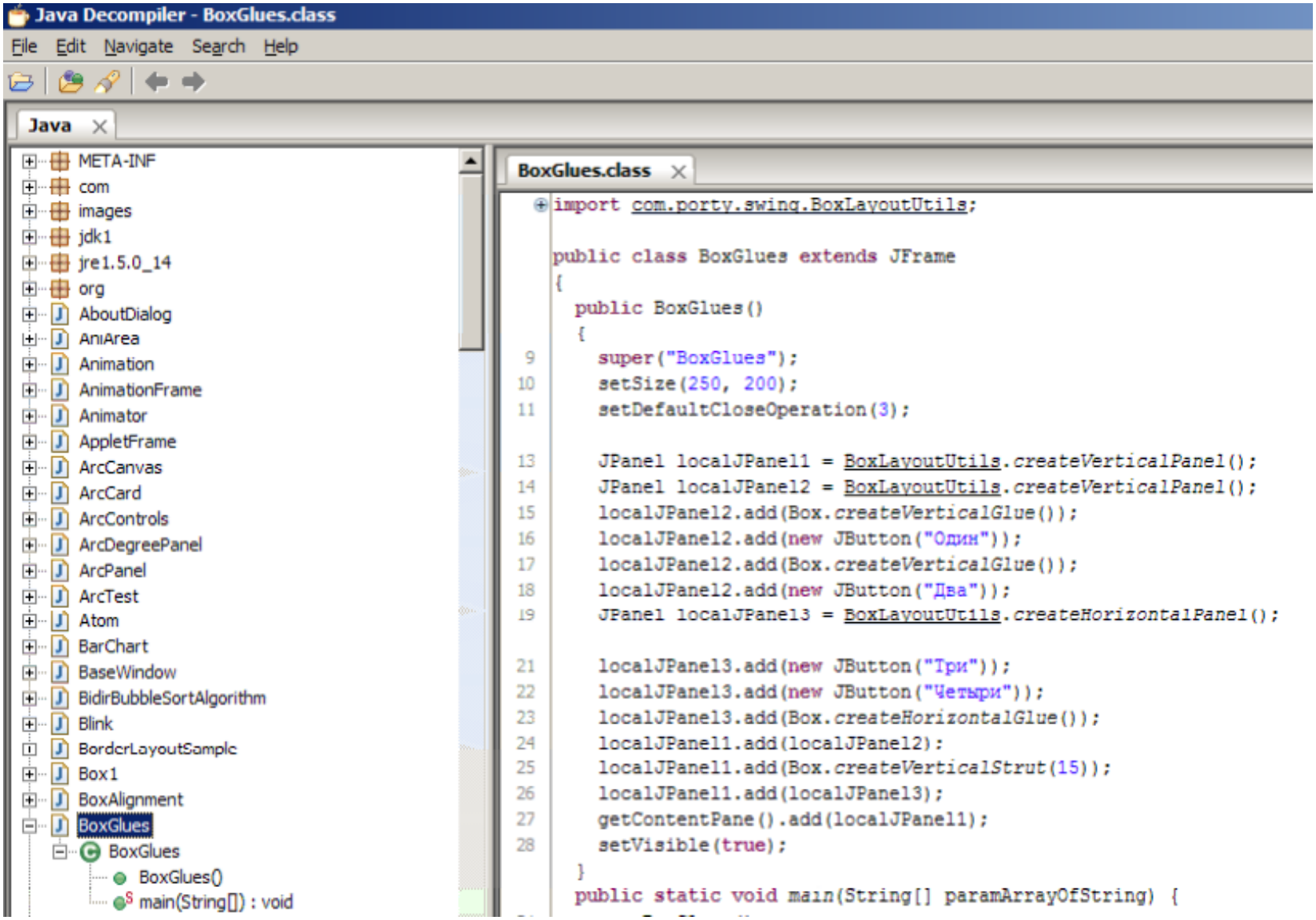
```
Compiled from DocFooter.java
public class DocFooter extends java.applet.Applet {
    java.lang.String date;
    java.lang.String email;
    public DocFooter();
    public void init();
    public void paint(java.awt.Graphics);
}
```

Пример вывода байт-кода класса

javap –c <classname> позволяет оценить структуру и сложность класс файла.

```
public ThreeD();
  Code:
   0:  aload_0
   1:  invokespecial#1; //Method
java/applet/Applet."<init>":()V
   4:  aload_0
   5:  iconst_1
   6:  putfield  #2; //Field painted:Z
   9:  aload_0
   10: fconst_1
   11: putfield  #3; //Field scalefudge:F
   14: aload_0
   15: new#4; //class Matrix3D
   18: dup
   19: invokespecial#5; //Method Matrix3D."<init>":()V
   22: putfield  #6; //Field amat:LMatrix3D;
   25: aload_0
   26: new#4; //class Matrix3D
   29: dup
   30: invokespecial#5; //Method Matrix3D."<init>":()V
   33: putfield  #7; //Field tmat:LMatrix3D;
```

Полезная утилита JD Java Decompiler



Взять JD можно здесь: http://java.decompiler.free.fr