

Платформа: история и окружение

Иван Игоревич Овчинников

GeekBrains. Java Core.

2022



Добро пожаловать на специализацию Java, лучшую из всех специализаций Java, что вы сможете найти.

Введение и знакомство (о себе)

Иван Овчинников. НПО ИТ, РКС, GB.

Используемые технологии

Много и с удовольствием C, C++, Java, Verilog. C# и Python даже не считаем.

Для GB более 20 потоков по разным направлениям, более двух тысяч студентов.



Здравствуйте, меня зовут Иван, я разработчик программного обеспечения в Российских космических системах, начальник группы программистов, автор и преподаватель нескольких курсов в направлении программирования на портале GeekBrains. Пара слов обо мне, почему я нахожусь здесь и планирую Вам что-то рассказывать про язык Java? Для этого придётся вынести за скобки мой опыт работы в цифровой схемотехнике и других языках, в том числе создание бортовой аппаратуры, которая прямо сейчас летает у нас над головами или продолжает доставлять другие полезные грузы в космос. С точки зрения языка Java, обо мне можно сказать, что я являюсь разработчиком нескольких отраслевых информационных систем поддержки единой наземной инфраструктуры российского космического агентства. На моём счету также участие в межотраслевых проектах по созданию единых баз данных космического применения.



Итак язык Java. Согласно википедии, Java — строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems. Разработка ведётся сообществом; язык и основные реализующие его технологии распространяются по лицензии GPL.

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года. Традиционно занимает высокие места в рейтингах популярности языков программирования (3-е место в рейтинге TIOBE (на август 2021)). На графике отлично видно, что с 2002 года язык уверенно держится в тройке популярных, значительную часть времени возглавляя его.

Но это всё - сухие факты, ничего не говорящие нам о том, чем руководствовались разработчики языка и что нам, программистам, с этим делать.

Существует, интересная версия происхождения названия языка, связанная с аллюзией на кофе машину как пример бытового устройства, для программирования которого изначально язык создавался. Кстати, в результате работы проекта мир увидел принципиально новое устройство, карманный пер-

сональный компьютер, который опередил своё время более чем на 10 лет, но из-за большой стоимости не смог произвести переворот в мире технологии и был забыт. Устройство Star7 не пользовалось популярностью, в отличие от языка программирования Java и его окружения. С середины 1990-х годов язык стал широко использоваться для написания клиентских приложений и серверного программного обеспечения. Тогда же некоторое распространение получила технология Java-апплетов — графических Java-приложений, встраиваемых в веб-страницы; с развитием возможностей динамических веб-страниц технология стала применяться редко и язык стал применяться для бэк-энда.

Почему Java?

Написано однажды, работает везде.

Иван Игоревич Овчинников (GB: Java) Платформа: история и окружение 2022 4/14

Итак Язык программирования Java занял свою нишу в системном и прикладном программировании из-за своей высокой скорости работы, и веб-программировании, из-за удобного сетевого окружения, ставшего основой для создания бэк-энда веб-сервисов. Java очевидно испытал влияние языков C, C++, Pascal и других, так что многое в этих языках достаточно досконально изучить один раз, чтобы иметь возможность понимать все связанные.

Поскольку язык изначально проектировался для множества разнообразных исполнителей с разными архитектурами процессоров и систем, было принято решение отделить исполнителя от решаемой прикладной задачи, так появилась виртуальная машина Java, речь о которой пойдёт немного позже. Основной смысл в том, что у языка Java появился девиз: Написано однажды, работает везде.

Базовый инструментарий

- ① Eclipse
- ② NetBeans
- ③ IntelliJ IDEA
- ④ BlueJ
- ⑤ Oracle JDeveloper
- ⑥ MyEclipse
- ⑦ Greenfoot
- ⑧ jGRASP
- ⑨ JCreator
- ⑩ DrJava



Мы привыкли к тому, что для программирования нужна некоторая среда, которая будет удобно подсказывать нам, что делать, как писать программу, запускать написанный код и помогать нам отлаживать его. Чаще всего это так, но я призываю вас не останавливаться на единственном инструменте, ведь если быть всегда сосредоточенным на отвёртке, вы никогда не узнаете, что люди придумали очень удобный шуруповёрт.

Я не планирую рекламировать ту или иную среду, скорее всего не буду даже много говорить о плюсах и минусах той или иной среды, просто покажу список и коротко расскажу о главных действующих лицах:

1. NetBeans - если коротко, то эта среда с нами из уважения к истории, она была первой, созданной для разработки на языке Java. Долгое время разработка этой среды не велась, но в последние несколько лет разработчики как будто бы вспомнили о ней и активно взялись за дело, сейчас есть версии под все популярные ОС, поддерживается весь популярный современный функционал, есть прикольные самобытные моменты, вроде расширенных клавиатурных макросов для написания функций с передаваемыми в них массивами;
2. Eclipse - проще, наверное, назвать язык или технологию, под которую нет плагина для этой весьма расширяемой и дополняемой среды программирования. Долгое время, вплоть до недавнего, а где-то и до сих пор, является корпоративным стандартом для написания Enterprise приложений и прочих сложных корпоративных приложений. Весьма гибкая среда и-за того, что разрабатывается Eclipse Foundation к ней можно подключить плагины, решающие почти любые задачи разработки. Из того что я видел своими глазами, Siemens использует Eclipse для разработки своих систем поддержки жизненного цикла изделий;
3. IntelliJ IDEA - стандарт де-факто примерно десять последних лет. Не уверен, можно ли назвать эту среду российской разработкой, но у компании, разрабатывающей эту среду, три русских учредителя. Да и сама эта компания началась именно с этой среды. В этой среде можно настроить почти любую деталь, что часто бывает очень удобно. Среда расширяется плагинами на почти все случаи жизни разработчика. Часто входит в экосистему JetBrains, развёрнутую на предприятии, поэтому тесно интегрируется в другие сферы деятельности компании. Существу-

ет как бесплатная, так и платная версия с расширенным функционалом, например, доступа к базам данных.

Отдельно стоит обговорить такую среду, как Android Studio, как мог заметить внимательный зритель, её нет в списке на слайде, но не потому, что в ней нельзя разрабатывать Java-приложения, а потому что если не разрабатывать в ней именно Android приложения, следует отказаться от её использования из-за некоторой избыточности. Среда снабжена значительным количеством надстроек и эмуляторов мобильных устройств, которые попросту будут вам мешать, если вы не используете их в своей профессиональной деятельности. С другой стороны, если ваша цель - это именно мобильная разработка под Android, то и выбора у вас особо нет, только Android Studio.

Что нужно скачать

- 1 JDK
- 2 среду

Иван Игоревич Овчинников (GB: Java) Платформа: история и окружение 2022 6/14

Для разработки на языке Java вам может понадобиться довольно много разного инструментария, тем более, что требования к инструментарию отличаются ещё и от команды к команде, вам может потребоваться средство для работы с БД, средства моделирования и описания систем, дополнительные средства документирования кода и решений. Но совершенно точно можно сказать, что без двух вещей обойтись не удастся: это иде, они же среды программирования и ждк, он же инструментарий разработчика на java. О средах программирования дополнительно только что поговорили, нужно выбрать какую-нибудь, и скачать. Скорее всего вы это сделали для прохождения более ранних курсов, сейчас сможете для себя выбрать инструмент более осознанно. В курсе, кроме этой лекции и следующего за ней семинара, будет использоваться комьюнити версия среды IntelliJ IDEA, можете для простоты выбрать её, что вы уже скорее всего и сделали. Наверняка, для прохождения предыдущих курсов вы уже скачали и установили JDK какой-нибудь хорошей новой версии, а сейчас пришла пора разобраться, что это, какие они бывают, откуда их брать и как их выбирать, мы же за осознанность как-никак.

Что нужно скачать

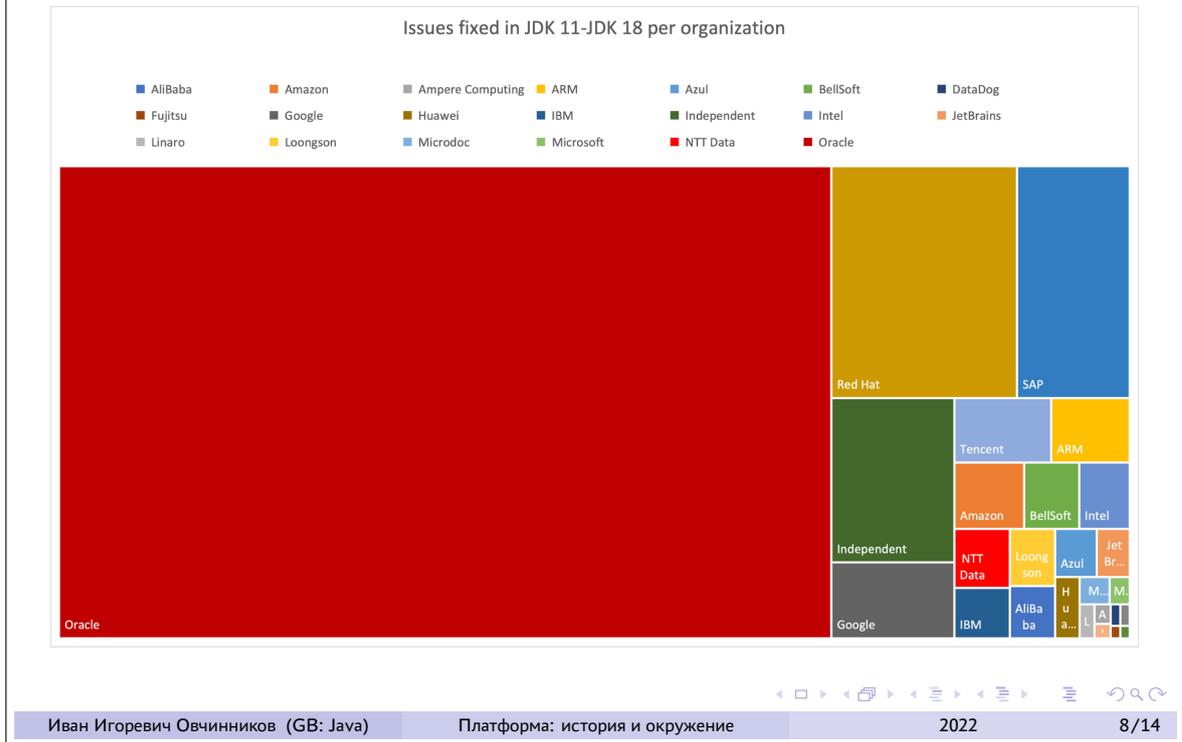
- ① Oracle JDK
- ② OpenJDK by Oracle
- ③ Liberica JDK
- ④ экзотические
 - GOST Java
 - AdoptOpenJDK
 - Red Hat OpenJDK
 - Azul Zulu
 - Amazon Corretto



Итак, вендоры и версии. JDK развивается, поэтому выходят новые версии, исправляющие предыдущие недочёты, добавляющие новую функциональность. Самая распространённая версия в современной российской действительности - восьмая, но есть очень серьёзный тренд на переход к 11 и 13й версиям, на более новые часто смотрят с опаской. Распространённых вендоров три - оракл, опен, либерика. Есть, конечно, компании, собирающие свой инструментарий из открытых источников самостоятельно, но их крайне мало. Про экзотических вендоров рассказывать смысла не много, важно помнить, что они есть и у них у всех разные лицензионные политики. Я в курсе буду использовать либерику, как делаю последние несколько лет, хотя довольно долгое время пользовался ораклом. Поскольку я работаю в около- государственной компании, не могу не упомянуть о ГОСТ джаве, которая часто всерьёз рассматривается в организациях, подобных моей, и об опенЖДК, который является основой для самостоятельных сборок инструментария.

Наиболее популярной, конечно, является оракл ждк, но, если мне будет позволено высказать персональное мнение, это больше дань традиции, нежели какая-то жизненная необходимость.

Кто делает для нас JDK



Если внимательно посмотреть на компании, внесшие вклад в развитие открытой части инструментария, начиная с ждк11 и до актуального на сегодня ждк18, конечно, максимально вкладывается оракл, но и беллсофт, производитель либерики, там будет в первой десятке. Повторюсь, я на курсе буду использовать либерику ждк11, благо для неё есть и докер-образ, если нужно как-то автоматизировать сборки вашего приложения, например; вы можете использовать любую другую не старше 8. Все примеры я буду приводить с учётом именно 8й версии языка, чтобы рассмотреть подходящий подавляющему большинству людей инструментарий. Если у вас уже установлена 11я версия от другого вендора - ничего менять не нужно, если установлена более новая версия, есть вероятность, что некоторые примеры будут отмечаться как чрезвычайно устаревшие, возможно, эти предупреждения на этапе обучения следует игнорировать. Для выбора вендора внимательно ознакомьтесь с лицензией, а для выбора версии языка руководствуйтесь здравым смыслом: большинство компаний в России и мире работают на 8, 11 или 13й версиях. Зачем учиться на какой-нибудь 18-й версии, если потом на работе вы не увидите половины каких-нибудь новых фиц и фактически, придётся переучиваться?

Когда нужно по-быстрому



The image shows the Jupyter logo, which consists of the word "jupyter" in a lowercase, sans-serif font, centered within a large orange circle. The circle is formed by two curved segments, one above and one below the text. Surrounding the central logo are numerous small, light gray icons representing various programming languages and technologies, including Python, JavaScript, PHP, Lua, VB, C#, F#, R, Spark, and others. The entire graphic is set against a light gray background.

Иван Игоревич Овчинников (GB: Java) Платформа: история и окружение 2022 9/14

Для решения некоторых несложных задач курса мы будем писать простые приложения, не содержащие ООП, сложных взаимосвязей и проверок, в этом случае нам понадобится Jupyter notebook с установленным ядром (kernel) IJava. Да, многие думают, что Jupyter ноутбуки - это только для языка пайтон или для скриптовых языков, но это не так. Архитектура юпитер ноутбука позволяет ему работать с любым ядром, главное, чтобы ядро умело корректно интерпретировать написанное в ячейке с кодом. Ядро IJava делает именно это - интерпретирует Java-код, используя установленный на компьютере разработчика JDK.

JDK и его друзья

TL;DR:

- JDK = JRE + инструменты разработчика;
- JRE = JVM + библиотеки классов;
- JVM = Native API + механизм исполнения + управление памятью.



Очень часто в интернете, при скачивании какого-то программного обеспечения, связанного с Java можно увидеть разные аббревиатуры: JRE, JVM, JDK и тому подобные, и чтобы в них не запутаться давайте коротко и быстро разберёмся что они значат, потому что я чувствую, как вам уже начинают надоедать все эти вступления.

Пойдём от самого масштабного и общего к самому маленькому и частному. Самая масштабная и общая аббревиатура - это JDK Java Development Kit - инструментарий разработчика на языке Java. Это обычный SDK который содержит в себе также всё необходимое для разработки именно на языке Java здесь есть куча других аббревиатур, компилятор, средства развёртывания, создания документации, итд они позволяют удобно писать программы на языке Java и абстрагируют программиста от необходимости описывать все вспомогательные процессы непосредственно на языке программирования то есть, грубо говоря это некие классы которые инкапсулирует сложное преобразование текста программ в простой интерфейс разработчика.

Всё, что разрабатывает разработчик, исполняется в специальной среде это среда является частью JDK и позволяет исполнять все программы написанные разработчиками на конечном в компьютере. Именно эта среда и устанавливается на компьютере пользователя чтобы пользователь мог запускать приложения, написанные на языке Java. Итак, самая необходимая вещь для любого пользователя это JRE. Это аббревиатура расшифровывается как Java Runtime Environment, то есть среда исполнения программ на языке Java. Она содержит в себе основной внутренний механизм который называется Java virtual machine и библиотеку из разных классов чтобы эта виртуальная машина хорошо работала умела показывать пользователю строки осуществлять ввод-вывод фреймворк коллекций и прочее, тут тоже есть куча интересных аббревиатур, которые мы так или иначе изучим, например, Java native interface, Java database connectivity и другие. Если копнуть ещё чуть глубже, можно и JRE разделить на JSE и JEE, но нас это деление пока не очень интересует, оно произойдёт естественным образом на более старших курсах. Пока что всё, что мы будем изучать, это JSE.

JVM и что в нём происходит

рис 1.2 (стр 4) конспекта



Иван Игоревич Овчинников (GB: Java)

Платформа: история и окружение

2022

11/14

Инструментарий разработчика мы будем так или иначе рассматривать весь оставшийся курс, части среды исполнения мы тоже будем всесторонне изучать на последующих уроках. Сейчас хотелось бы подробнее остановиться на виртуальной машине Java, поскольку понимание того, как она устроена, должно значительно облегчить для вас понимание процесса исполнения программы и, как следствие, процесс разработки этих самых программ. Виртуальная машина Java осуществляет загрузку классов программы в оперативную память, причём здесь имеется ввиду не оперативная память как аппаратная часть компьютера, а некая выделенная часть этой оперативной памяти, которой с нами поделилась операционная система. Также осуществляется управление памятью, а именно очистка и сборка мусора и непосредственное исполнение классов нашего приложения, путём компиляции методов из промежуточного байткода в непосредственные вызовы операционной системы или другого исполнителя, то есть грубо говоря преобразования классов Java в ассемблерный код конкретного компьютера. Этот процесс называется JIT компиляция. Существуют разные Реализация виртуальных машин (Wikipedia: Список виртуальных машин Java), даже экзотические, например мультязыковой интерпретатор GraalVM. По большей части виртуальные машины отличаются как раз этой частью, с Just In Time компиляцией, то есть с преобразованием методов в непосредственные машинные вызовы в реальном времени. Чем быстрее происходит эта JIT компиляция, тем, соответственно, быстрее работает приложение на виртуальной машине. JVM самостоятельно осуществляет сборку так называемого мусора, что значительно облегчает работу программиста по отслеживанию утечек памяти, но, на мой взгляд, не способствует должной концентрации внимания программиста на этом вопросе. Важно помнить, что в Java утечки памяти всё равно существуют, особенно при программировании многопоточных приложений.

Структура проекта

- простейшие (один файл)
- обычные (несколько пакетов)
- шаблонные (формируются сборщиками)
- скриптовые (jupyter notebook)



Ну, наконец-то проект. Проекты бывают разной сложности, сейчас поговорим о проектах без применения сборщиков, потому что сборщикам и соответствующим проектам у нас будет посвящено отдельное занятие. Все проекты по сложности структуры можно разделить на четыре основные категории: простейшие, обычные, шаблонные, скриптовые.

Простейший проект

Думаю, тут надо вживую покодить, благо немного

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

ivan@gb src > javac Main.java
ivan@gb src > java Main
Hello, world!



Начнём с простейшего проекта, который будет состоять из одного файла исходников. Естественно, вы спросите, что за чушь, мы же уже сотни раз писали «привет мир» и запускали его удобной зелёной стрелочкой в ИДЕ. Есть ответ. Мы же выясняем, что там внутри, под капотом этих удобных стрелочек и привычных процессов. Да и ради простейшего проекта запускать тяжеловесную среду программирования как-то неестественно что ли.

Jupyter Notebook

также вживую открыть юпитер ноутбук, показать, что возможно запускать код в скриптовом формате



Иван Игоревич Овчинников (GB: Java)

Платформа: история и окружение

2022

14/14

Пока далеко не ушли от простейших проектов, думаю, целесообразно будет проговорить о скриптовых возможностях Java. Поскольку язык является не только компилируемым, но и интерпретируемым благодаря работе JVM, его возможно использовать как скриптовый внутри юпитер ноутбука, подключив соответствующее ядро. Интересной особенностью юпитер ноутбука является разделение фронтенда и ядра, что даёт возможность заменять эти самые ядра (в терминах юпитера они называются кернелы). скачав и установив ядро PJava мы получаем шикарную возможность делать простые наброски логики, снабжая их комментариями в формате маркдаун. На самом деле можно делать и непростые наброски, но это выходит далеко за рамки нашего курса о языке. Если будет очень интересно, призываю вас самостоятельно почитать о возможностях ядра, благо на него очень хорошая документация с примерами.