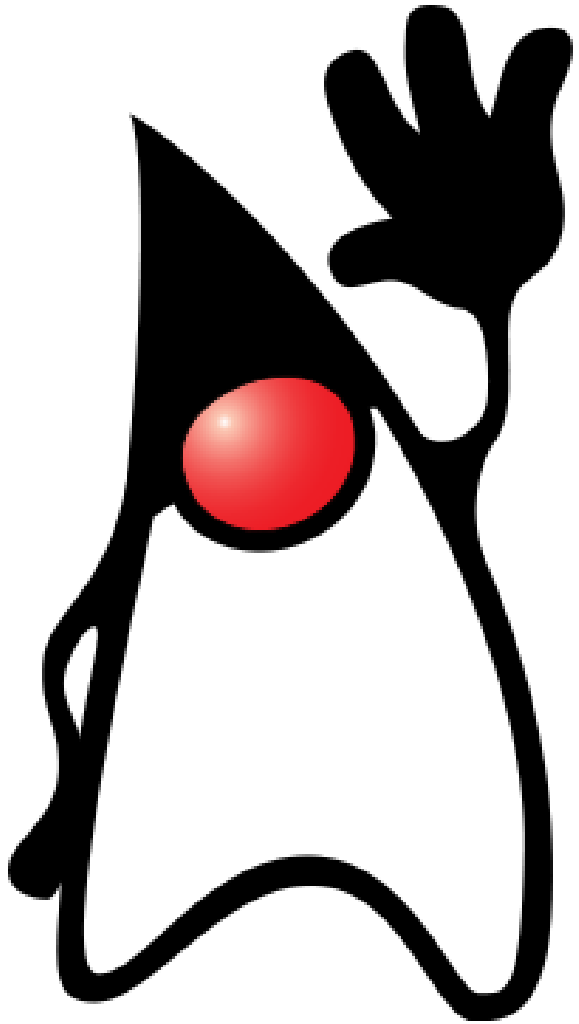
# Введение в Java

Алексей Владыкин

8 сентября 2014

1. Знакомьтесь: Java
2. История и эволюция
3. Особенности
4. Hello World
5. Инструменты разработчика


Java™



# Почему стоит изучать Java?

- Один из самых популярных и востребованных языков программирования. Много лет в верхних строчках индекса TIOBE
- На Java пишут:
  - высоконагруженные системы (Одноклассники)
  - корпоративные приложения (Confluence, JIRA)
  - настольные приложения (Minecraft!)
  - программы и игры для телефонов, в том числе под Android
  - апплеты для смарт-карт
  - интерактивный контент для Blu-ray
- Язык развивается и совершенствуется

- Java — это не только ~~ценный мех~~ язык программирования, но и. . .

- Обширная стандартная библиотека
- Сторонние библиотеки и фреймворки
- Инструменты разработки (сборка, тестирование)
- Методология ООП, паттерны проектирования
- Платформа для альтернативных языков (Clojure, Groovy, JRuby, Jython, Kotlin, Scala)

# Что мы будем изучать?

- Java Standard Edition (SE)
  (а есть еще Micro Edition, Enterprise Edition, JavaCard, Android)
- Реализация от Sun/Oracle
  (а есть еще Oracle JRockit, IBM J9, Azul Zing, Apache Harmony)
- Несколько самых распространенных сторонних библиотек и инструментов


James Gosling

1991 внутренний проект Sun Microsystems по созданию платформы для разработки встраиваемых систем — Green Project;
вместо C++ решили создать новый язык, названный Oak

1991 внутренний проект Sun Microsystems по созданию платформы для разработки встраиваемых систем — Green Project;
вместо C++ решили создать новый язык, названный Oak

1992 первое демонстрационное устройство на новой платформе — PDA Star7

1991 внутренний проект Sun Microsystems по созданию платформы для разработки встраиваемых систем — Green Project;
вместо C++ решили создать новый язык, названный Oak

1992 первое демонстрационное устройство на новой платформе — PDA Star7

1993 попытка занять нишу TB-приставок для кабельного телевидения

1991 внутренний проект Sun Microsystems по созданию платформы для разработки встраиваемых систем — Green Project;
вместо C++ решили создать новый язык, названный Oak

1992 первое демонстрационное устройство на новой платформе — PDA Star7

1993 попытка занять нишу TB-приставок для кабельного телевидения

1994 фокус на разработке интерактивных приложений (апплетов) для веб-страниц;
язык переименован в Java

1991 внутренний проект Sun Microsystems по созданию платформы для разработки встраиваемых систем — Green Project;
вместо C++ решили создать новый язык, названный Oak

1992 первое демонстрационное устройство на новой платформе — PDA Star7

1993 попытка занять нишу TB-приставок для кабельного телевидения

1994 фокус на разработке интерактивных приложений (апплетов) для веб-страниц;
язык переименован в Java

1996 Java Development Kit 1.0

1996 Java Development Kit 1.0

1996 Java Development Kit 1.0

1997 JDK 1.1, JIT-компиляция

1996 Java Development Kit 1.0

1997 JDK 1.1, JIT-компиляция

1998 J2SE 1.2, «Java 2», разделение на ME/SE/EE

1996 Java Development Kit 1.0

1997 JDK 1.1, JIT-компиляция

1998 J2SE 1.2, «Java 2», разделение на ME/SE/EE

2000 J2SE 1.3

1996 Java Development Kit 1.0

1997 JDK 1.1, JIT-компиляция

1998 J2SE 1.2, «Java 2», разделение на ME/SE/EE

2000 J2SE 1.3

2002 J2SE 1.4

1996 Java Development Kit 1.0

1997 JDK 1.1, JIT-компиляция

1998 J2SE 1.2, «Java 2», разделение на ME/SE/EE

2000 J2SE 1.3

2002 J2SE 1.4

2004 J2SE 5.0, изменение нумерации

1996 Java Development Kit 1.0

1997 JDK 1.1, JIT-компиляция

1998 J2SE 1.2, «Java 2», разделение на ME/SE/EE

2000 J2SE 1.3

2002 J2SE 1.4

2004 J2SE 5.0, изменение нумерации

2006 Java SE 6, уход от понятия «Java 2»

1996 Java Development Kit 1.0

1997 JDK 1.1, JIT-компиляция

1998 J2SE 1.2, «Java 2», разделение на ME/SE/EE

2000 J2SE 1.3

2002 J2SE 1.4

2004 J2SE 5.0, изменение нумерации

2006 Java SE 6, уход от понятия «Java 2»

2011 Java SE 7

1996 Java Development Kit 1.0

1997 JDK 1.1, JIT-компиляция

1998 J2SE 1.2, «Java 2», разделение на ME/SE/EE

2000 J2SE 1.3

2002 J2SE 1.4

2004 J2SE 5.0, изменение нумерации

2006 Java SE 6, уход от понятия «Java 2»

2011 Java SE 7

2014 Java SE 8


KEEP
CALM
AND
LOVE
JAVA

# Виртуальная машина и байткод

- Традиционный подход:
  исходный код $\rightarrow$ машинный код $\rightarrow$ процессор
  - программа работает только на той платформе, под которую она скомпилирована

# Виртуальная машина и байткод

- Традиционный подход:
  исходный код → машинный код → процессор
  - программа работает только на той платформе, под которую она скомпилирована
- Подход Java:
  исходный код → байткод виртуальной машины → виртуальная машина → процессор
  - программа работает на любой платформе, где есть виртуальная машина Java
  - “Write once, run anywhere!”

# Виртуальная машина и байткод

- Как быстро работает виртуальная машина?

# Виртуальная машина и байткод

- Как быстро работает виртуальная машина?
- Интерпретация байткода на порядок (10–20 раз) медленнее исполнения аналогичного машинного кода...

# Виртуальная машина и байткод

- Как быстро работает виртуальная машина?
- Интерпретация байткода на порядок (10–20 раз) медленнее исполнения аналогичного машинного кода...
- но есть Just-In-Time компиляция
  - виртуальная машина компилирует байткод в машинный код
  - используется с JDK 1.1

# Виртуальная машина и байткод

- Как быстро работает виртуальная машина?
- Интерпретация байткода на порядок (10–20 раз) медленнее исполнения аналогичного машинного кода...
- но есть Just-In-Time компиляция
  - виртуальная машина компилирует байткод в машинный код
  - используется с JDK 1.1
- а также HotSpot
  - адаптивный оптимизирующий JIT-компилятор
  - используется с JDK 1.3

# Виртуальная машина и байткод

- Как быстро работает виртуальная машина?

- Интерпретация байткода на порядок (10–20 раз) медленнее исполнения аналогичного машинного кода...

- но есть Just-In-Time компиляция
  - виртуальная машина компилирует байткод в машинный код
  - используется с JDK 1.1
- а также HotSpot
  - адаптивный оптимизирующий JIT-компилятор
  - используется с JDK 1.3
- в результате Java 8 всего в 1.5–2 раза медленнее C, а в некоторых тестах не хуже или даже быстрее!

# Сборка мусора

- Подход C/C++:
  выделил память $\rightarrow$ поработал $\rightarrow$ освободил память
  - всё управление памятью в руках программиста

# Сборка мусора

- Подход C/C++:
  выделил память $\rightarrow$ поработал $\rightarrow$ освободил память
  - всё управление памятью в руках программиста
- Подход Java:
  выделил память $\rightarrow$ поработал $\rightarrow$ молодец
  - виртуальная машина считает ссылки на каждый объект
  - освобождает память, когда ссылок больше нет

# Безопасность

- Верификация байткода
  - некорректный байткод будет отвергнут перед исполнением
- Автоматическое управление памятью
  - нет арифметики указателей
  - невозможно испортить память
- Встроенный механизм управления правами
  - можно запустить код в «песочнице» без доступа к файлам, к сети, без возможности создавать потоки и т. п.

# Многопоточное и распределенное программирование

- Многопоточность
  - встроенная поддержка потоков
  - богатая библиотека примитивов синхронизации
- Распределенность
  - встроенные сетевые возможности
  - пересылка данных и объектов по сети
  - работа с удаленными объектами (RMI)


HELLO
WORLD

# HelloWorld.java

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

- Java Coding Conventions
- Demo

# Среды разработки

- Eclipse
- IntelliJ IDEA
- NetBeans

- Подсветка синтаксиса
- Автодополнение, гиперссылки
- Рефакторинг
- Интерактивный отладчик

# JRE/JDK

- Java Runtime Environment (JRE)
  виртуальная машина и стандартная библиотека классов для запуска скомпилированных программ

- Java Development Kit (JDK)
  набор инструментов для разработчиков,
  включает в себя JRE

# javac

- Java Compiler
- Компилирует исходный код (`*.java`) в байткод (`*.class`)

- `javac MyClass.java YetAnotherClass.java`
- `javac -d classes MyClass.java`
- `javac -classpath library.jar -d classes MyClass.java`
- `javac -version`

# Отступление: о classpath

- Все используемые классы должны быть доступны в classpath
- По умолчанию содержит текущую директорию «.» и классы стандартной библиотеки

- Задается как список директорий и/или JAR-файлов
- Разделитель списка
  - «:» в Unix/Linux/Mac OS X
  - «;» в Windows

# jar

- Java Archive Tool
- Создает и распаковывает JAR-файлы

- `jar cf library.jar -C classes_dir .`
- `jar cfm library.jar manifest.mf -C classes_dir .`
- `jar cfe library.jar MyMainClass -C classes_dir .`
- `jar tf library.jar`
- `jar xf library.jar`

# Отступление: о MANIFEST.MF

- Любой JAR-файл содержит `META-INF/MANIFEST.MF`
- Пример:

```
Manifest-Version: 1.0
Created-By: 1.8.0_05 (Oracle Corporation)
```

- `Main-Class` — имя класса с методом `main`
- `Class-Path` — список необходимых JAR'ов, через пробел

# java

- Java Virtual Machine
- Исполняет байткод
- Главный класс должен иметь метод
  `public static void main(String[] args)`

- `java MyClass`
- `java -classpath classes_dir;library.jar MyClass`
- `java -jar library_with_main_class.jar`
- `java -version`

# Инструменты для сборки

- Ant — `build.xml`
- Gradle — `build.gradle`
- Maven — `pom.xml`

- Автоматизация процесса сборки
  (компиляция, запуск тестов, генерация документации и т.п.)
- Независимость от среды разработки
- Возможность сборки в командной строке

# Maven

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>ru.compscicenter.java2014</groupId>
  <artifactId>hello-world</artifactId>
  <version>1.0</version>

</project>
```

- mvn clean install
- Demo

# Что сегодня узнали

- Что такое Java и с чем её едят
- В чем особенности Java и отличия от C/C++
- Как написать, собрать и запустить программу на Java