

**Java**

# ВВЕДЕНИЕ В JAVA

СПБАУ 02.09.2015

# ИСТОРИЯ И ЭВОЛЮЦИЯ JAVA

ВВЕДЕНИЕ В JAVA

# ИСТОРИЯ И ЭВОЛЮЦИЯ 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 ТВ-приставок для кабельного телевидения
- 1994 фокус на разработке интерактивных приложений (апплетов) для веб-страниц;  
язык переименован в Java
- 1996 Java Development Kit 1.0

# СОСТАВ ПЛАТФОРМЫ

- Виртуальная машина Java (JVM)
  - The Java Virtual Machine Specification
- Стандартная библиотека
  - Соответствует версии платформы
- Компилятор Java
  - The Java Language Specification (JLS)

# ИСТОРИЯ И ЭВОЛЮЦИЯ JAVA

1996 Java Development Kit 1.0

JLS 1.0, JVM 1.0, минимальная стандартная библиотека

1997 JDK 1.1, JLS 2.0, ...для кофеварок...

1998 J2SE 1.2, «Java 2», разделение на SE/EE, JVM 2.0

2000 J2SE 1.3, выделена ME

2002 J2SE 1.4

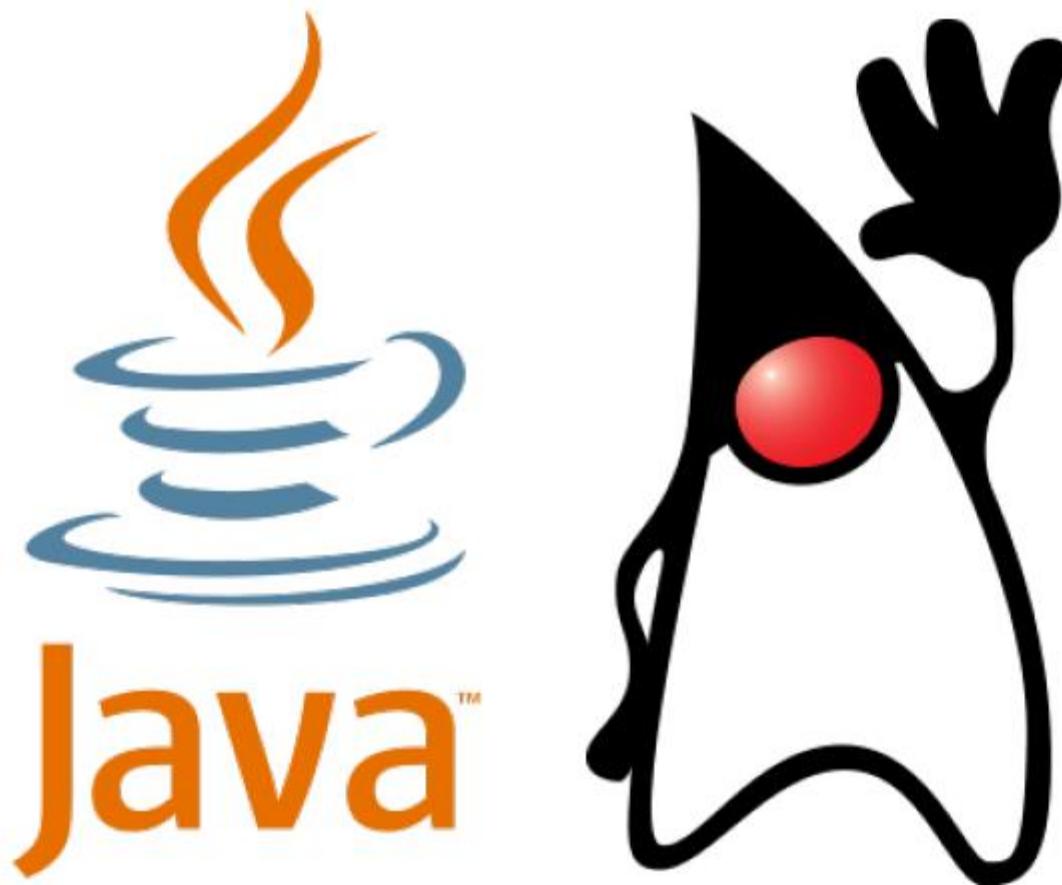
2004 J2SE 5.0, изменение нумерации, JLS 3.0

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

2011 Java SE 7

2014 Java SE 8





Java

# ПОЧЕМУ JAVA?

ВВЕДЕНИЕ В JAVA

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     char quote[] = "
6
7
8
9
10    Язык С позволяет выстрелить себе в ногу; с C++ это
11    сделать сложнее, но если вам это удастся, вы лишитесь
12
13    всей ноги разом.
14
15
16
17
18    ";
19
20    /* @ Бъярне Строуструп, разработчик C++ */
21
22    return 0;
23 }
```

# WHY DO JAVA PROGRAMMERS WEAR GLASSES?



## THEY CAN'T SEE SHARP.

# ОСОБЕННОСТИ JAVA

ВВЕДЕНИЕ В JAVA

# ВИРТУАЛЬНАЯ МАШИНА И БАЙТКОД

- Подход C/C++:

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

# 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 1.2
- а также HotSpot (основная виртуальная машина Java для настольных компьютеров и серверов, выпускаемая корпорацией Oracle)
  - адаптивный оптимизирующий JIT-компилятор
  - используется с JDK 1.3
- в результате Java 7 всего в 1.5–2 раза медленнее С, а в некоторых тестах не хуже или даже быстрее!

# СБОРКА МУСОРА

- Подход C/C++:

**выделил память → поработал → освободил память**  
всё управление памятью в руках программиста

- Подход Java:

**выделил память → поработал → забыл**  
виртуальная машина считает ссылки на объекты  
освобождает память, когда ссылок больше нет



**Это не правда! :)**

# 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 ПРОГРАММИРОВАНИЕ

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

# РАЗНОВИДНОСТИ JAVA

ВВЕДЕНИЕ В JAVA

# JRE/JDK

- Java Runtime Environment (JRE)
  - виртуальная машина и стандартная библиотека классов для запуска скомпилированных программ
- Java Development Kit (JDK)
  - набор инструментов для разработчиков (компилятор), включает в себя JRE
- Документация
  - на JVM
  - на Стандартную библиотеку

# РЕДАКЦИИ JAVA

- Standard Edition (SE)
- Micro Edition (ME)
  - подмножество SE + специфические библиотеки
- Enterprise Edition (EE)
  - SE + дополнительные библиотеки и возможности
- Java Card
  - сильно урезанная версия SE,  
изменения в виртуальной машине

# РЕАЛИЗАЦИИ JAVA

- Oracle Java
  - <http://java.oracle.com/>
- OpenJDK
  - <http://openjdk.java.net/>
- IcedTea
  - <http://icedtea.classpath.org/>
- IBM J9
- еще несколько десятков

# АЛЬТЕРНАТИВНЫЕ ЯЗЫКИ

На Java машине можно запускать программы на любом языке, компилируемом в байт-код JVM

- Groovy, Kotlin, Scala
- Clojure

Диалект LISP

- JRuby  
Реализация Ruby на JVM
- Jython  
Реализация Python на JVM

# ЯЗЫК JAVA

ВВЕДЕНИЕ В JAVA

# HelloWorld.java

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

- Java Coding Conventions
  - <http://www.oracle.com/technetwork/java/codeconv-138413.html>

# HelloWorldComments.java

```
/**  
 * Created by AntonK on 03.09.15.  
 * @author Anton Kuznetsov  
 */  
public class HelloWorldComments {  
    /* Example program */  
    public static void main(String[] args) {  
        // Make some magic  
        System.out.println("Hello, flower");  
    }  
}
```

# HelloUsers.java

```
/*
 * @author Anton Kuznetsov
 */
public class HelloUsers {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Use: java HelloUsers <user1>...");
        }
        System.out.printf("Hello");
        for (int i = 0; i < args.length; i++) {
            System.out.printf(", %s", args[i]);
        }
        System.out.println("!");
    }
}
```

# HelloWorld2.java

```
public class HelloWorld2 {  
    public static void main(String[] args) {  
        for (String str : args) {  
            System.out.println("Hello, " + str);  
        }  
    }  
}
```

- Java Coding Conventions
  - <http://www.oracle.com/technetwork/java/codeconv-138413.html>

# ИНСТРУМЕНТЫ JAVA

ВВЕДЕНИЕ В JAVA

# 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`

# CLASSPATH

- Все используемые классы должны быть доступны в путях, содержащихся в classpath
- Всегда содержит классы стандартной библиотеки (jre/lib/rt.jar)
- По умолчанию содержит текущую директорию «.»
- Задается как список директорий и/или JAR-файлов
- Разделитель списка
  - « : » в Unix/Linux/Mac OS X
  - « ; » в Windows

# java

- Java Virtual Machine
- Исполняет байткод
- **Главный класс должен иметь метод**  
**public static void main(String [] args)**  
**Этот метод – точка входа в программу**
- java MyClass
- java -classpath classes\_dir;library.jar MyClass
- java -jar library\_with\_main\_class.jar
- java -version

# ПРИМЕРЫ

ВВЕДЕНИЕ В JAVA

# Parser.java

```
package ru.spbau.kuznetsov.test01.parser;

public class Parser {
    public int parse (String str) {
        int result = 0;
        for (int i = 0; i < str.length(); i++) {
            result = result*10 + str.charAt(i)-'0';

        }
        return result;
    }

}
```

# Parser.java

```
package ru.spbau.kuznetsov.test01.parser;

public class Parser {
    public int parse (String str) {
        int result = 0;
        for (int i = 0; i < str.length(); i++) {
            result = result*10 + str.charAt(i)-'0';
        }
        return result;
    }
}
```

К каждому  
методу нужно  
писать  
модификатор  
доступа

Название  
пакета. В  
чем-то  
сходство с  
namespace



# AdvancedParser.java

```
package ru.spbau.kuznetsov.test01.parser;

public class AdvancedParser extends Parser {
    private int num;

    public AdvancedParser (int num) {
        this.num = num;
    }
    public int parse (String str) {
        int result = super.parse(str);
        return result*num;
    }
}
```

# AdvancedParser.java

```
package ru.spbau.kuznetsov.test01.parser;

public class AdvancedParser extends Parser {
    private int num;
    public AdvancedParser (int num) {
        this.num = num;
    }
    public int parse (String str) {
        int result = super.parse(str);
        return result*num;
    }
}
```

private поле

private int num;

public AdvancedParser (int num) {

    this.num = num;

}

public int parse (String str) {

    int result = super.parse(str);

    return result\*num;

}

}

Наследование.  
Множественного  
наследования  
нет.

конструктор

переопределение  
метода

вызов  
родительской  
реализации  
метода

```
package ru.spbau.kuznetsov.test01;

//import ru.spbau.kuznetsov.test01.parser.*;
import ru.spbau.kuznetsov.test01.parser.Parser;
import ru.spbau.kuznetsov.test01.parser.AdvancedParser;

public class ParserTest {
    public static void main(String[] args) {
        //ru.spbau.kuznetsov.test01.parser.Parser p =
new ru.spbau.kuznetsov.test01.parser.AdvancedParser(3);
        Parser p = new AdvancedParser(3);

        for (int i = 0; i < args.length; i++) {
            System.out.println(
                p.parse(args[i])+1);
        }
    }
}
```

```
package ru.spbau.kuznetsov.test01;
```

для использования  
классов из других  
пакетов их  
необходимо  
импортировать

```
//import ru.spbau.kuznetsov.test01.parser.*;  
import ru.spbau.kuznetsov.test01.parser.Parser;  
import ru.spbau.kuznetsov.test01.parser.AdvancedParser;
```

```
public class ParserTest {
```

или использовать  
полное имя

```
    public static void main(String[] args) {
```

```
        //ru.spbau.kuznetsov.test01.parser.Parser p =  
new ru.spbau.kuznetsov.test01.parser.AdvancedParser(3);
```

```
        Parser p = new AdvancedParser(3);
```

создание  
экземпляра класса

```
        for (int i = 0; i < args.length; i++) {
```

```
            System.out.println(
```

```
                p.parse(args[i])+1);
```

```
}
```

```
}
```

```
}
```