

# Общие сведения о Java

Разработчики: Патрик Нотон, Джеймс Гослинг и Майк Шеридан, Крис Варт, Эд Франк.

Язык **Oak** (дуб) с 1991г., в 1995 году переименован в Java.

Причина создания Java – потребность в независимом от платформы языке для разработки ПО, встраиваемого в различные бытовые электронные устройства (для использования на различных процессорах) .

**Java** - объектно-ориентированный язык программирования (Sun Microsystems), унаследовавший синтаксис основных операторов от C, объектно-ориентированные свойства от C++.

На данный момент технология **Java** принадлежит компании **Oracle**.

Дата создания Java - **23 мая 1995 г.**

## **Виды Java – программ:**

1. Приложение – обычная Java программа;
2. Апплет – особый вид Java программы, загружается через Интернет и выполняется браузером на компьютере пользователя;
3. Сервлет – Java программа, выполняется на сервере.

Документация по Java **API** (application programming interface) в Интернете:

[java.sun.com](http://java.sun.com) = [www.oracle.com/technetwork/java/index.html](http://www.oracle.com/technetwork/java/index.html).

## **Классификация платформ Java:**

1. JavaSE – Java Standart Edition;
2. JavaEE – Java Enterprise Edition;
3. JavaME – Java Micro Edition.

Что требуется установить для разработки java – программ:

**JDK** – Java Development Kit – средство разработчика.

JDK содержит утилиты:

1. **javac** – компилятор;
2. **java** – интерпретатор;
3. **appletviewer** – программа просмотра апплетов;
4. **jar** – утилита для создания jar-архивов;
5. **javadoc** – утилита для создания документации по проекту в стиле Java.

Для запуска java-программ требуется установить  
(без разработки):

**JRE** – Java Runtime Environment – среда выполнения java-программ ([java.com](http://java.com)).

## Свойства языка Java:

1. Архитектурно-независимый (класс скомпилирован один раз, работает везде);
2. Объектно-ориентированный;
3. Синтаксис основных операторов унаследован от C/C++;
4. Строгая типизация (8 примитивных типов, каждый тип строго определен, каждая переменная обладает типом, каждое выражение имеет тип, несоответствия типов являются ошибками, которые должны быть исправлены до завершения компиляции );
5. Компилируемый и интерпретируемый язык( компиляция java программы в байт-код, далее **JVM** (Java Virtual Machine) – виртуальная машина Java интерпретирует байт-код);

6. **Garbage collection** - механизм сборки мусора (нет оператора **delete**);
7. Поддержка многопоточной архитектуры;
8. Все переменные, кроме примитивных типов, являются объектами.
9. Выделение памяти всегда с помощью оператора **new**.
10. Безопасность: нет указателей ; сборка мусора; строгая типизация; программы выполняются JVM (JVM работает в своем адресном пространстве).
11. Нет перегрузки операций, нет операции «взятия адреса» **&**, нет указателей, нет операции разыменования **\***, нет множественного наследования.

Недостаток?

## Литература

1. **П. Нотон, Г. Шилдт JAVA 2. 2007.**
2. *Е.А. Аксёнова, А.А. Лазутина, А.В. Соколов* Введение в ООП в среде Java. 2006.
3. *Кёрк Скотт* Введение в язык программирования Java. 2005.
4. *Вязовик Н.А.* Программирование на Java: курс лекций для студентов. 2003.
5. *Герберт Шилдт* Искусство программирования на Java. Издательский дом "Вильямс". 2005.
6. *Брюс Эккель* Философия Java. Библиотека программиста. 2001.
7. *Хорстман К.С., Корнелл Г.* Библиотека профессионала. Java 2. Том 1 Основы, Том 2 Тонкости программирования. 2004.
8. **Герберт Шилдт Полный справочник по Java Java SE 6 Edition 7-е издание.2007.**
9. **Герберт Шилдт Java. Полное руководство. 8-е издание. 2012.**<sub>6</sub>

Пример программы – приложения:

```
import java.util.*;  
class HelloWorld {  
    public static void main (String args [ ]) {  
        System.out.println ("Hello, World!");    }    }
```

Имя файла должно совпадать с именем класса:

**HelloWorld.java**

Компиляция:

**javac HelloWorld.java**

Результат компиляции:

**HelloWorld.class**

Запуск:

**java HelloWorld**

Результат

**Hello, World!**

Пример программы – апплета:

```
import java.awt.*;  
import java.applet.*;  
  
public class Hello extends Applet {  
    public void paint(Graphics g) {  
        g.drawString("Hello, World!", 20, 20); } }
```

Имя:	<b>Hello.java</b>
Компиляция:	<b>javac Hello.java</b>
Результат компиляции:	<b>Hello.class</b>
Для запуска: файл	<b>Hello_applet.html</b>
Код файла Hello_applet.html:	

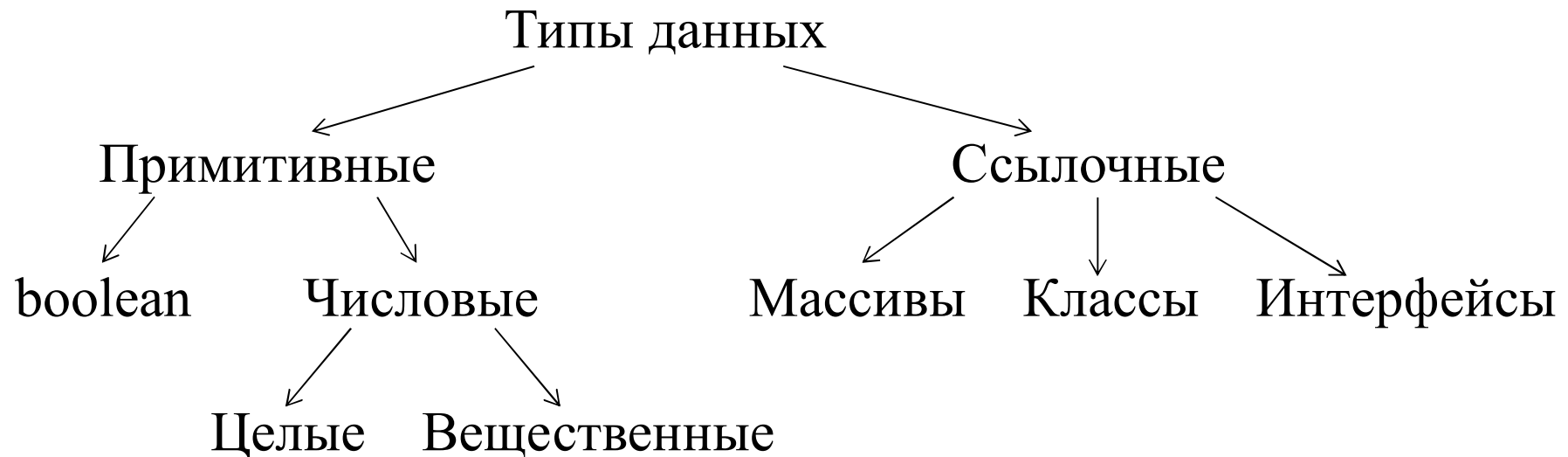
```
<APPLET CODE="Hello" width=200 height=40> </APPLET>
```

На каппе: **Hello.class** и **Hello\_applet.html** поместить в папку **public\_html** , в браузере зайти на свою страницу и запустить html-файл.



Java – строго типизированный язык (любая переменная и любое выражение имеют известный тип в момент компиляции).

1. Целый                      **byte (1) short(2) int(4) long(8) char(2)**
2. Вещественный    **float (4) double(8)**
3. Булевский            **boolean (true, false)**



2+'ж'=2+1046=1048

2+"ж"= "2ж" (строка)

Автоматическое преобразование типов в java выполняется в случае выполнения 2х условий:

- оба типа совместимы;
- длина целевого типа больше длины исходного типа.

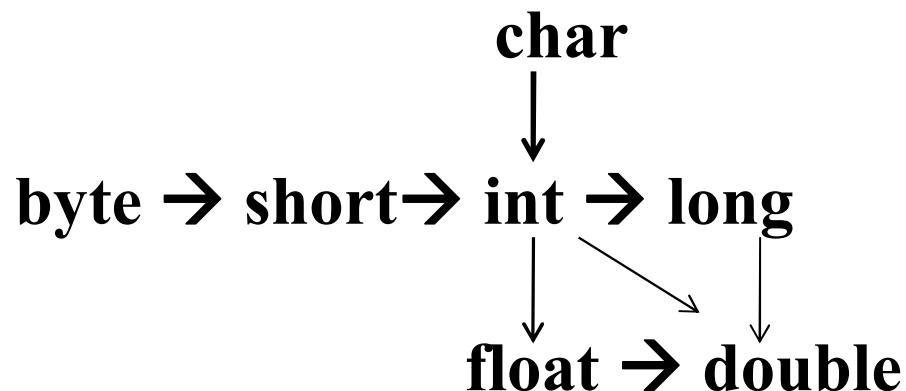
**byte → int → long**

**int → float**

**long → double**

В типы **char** и **boolean** автоматическое преобразование не выполняется.

**Схема повышения типа в выражениях:**



## Правила повышения типа в выражениях:

Тип всех значений **byte**, **short**, **char** всегда повышается до **int**.

Если один операнд имеет тип **long**, то тип всего выражения повышается до **long**.

Если один операнд имеет тип **float**, то тип выражения повышается до **float**.

Если один из операндов имеет тип **double**, то тип всего выражения повышается до **double**.

Нельзя преобразовывать переменную примитивного типа в ссылку. Над ссылками нельзя выполнять арифметические операции.

Привидение несовместимых типов – явное преобразование .

```
int a;  
byte b;  
b=(byte) a;
```

Автоматическое повышение типа в выражениях:

```
byte a =40;  
byte b = 50;  
byte c = 100;  
int d = a*b/c;    // тип short или byte операндов в выражениях  
                  // всегда повышен до int , если есть long, то до long
```

```
byte b = 50;  
byte d = b*2;    ???
```

```
byte b = 50;  
byte d = (byte)(b*2);
```

Математические функции : библиотека **java.lang.Math**

Класс **Math**

**Math.sqrt(x)**

**Math.sin(x)**

...

**Math.log(x)**

**Math.pow(x,a)**

...

**Math.PI**

**Math.E**

**Math.random()** ( от 0 до1)

**int R = (int)(Math.random()\*n) ;    n=10,100,1000 ...**

Строки – это объекты, но константы (менять нельзя, всегда будет создан новый объект).

```
String E = “ ”;
```

```
String A = “Hello”;
```

```
String MES = A + E + ”World”; // конкатенация
```

```
double ans, ddd=2.2;
```

```
...
```

```
System.out.println(“Ответ = ” + ans + ”, где ddd = ” + ddd);
```

Массивы

```
double [ ] a,b; //две ссылки a и b на массивы типа double
```

```
double a[ ], b; // a – ссылка на массив, b – переменная
```

```
a=new double[5];
```

```
double[ ] a ={0.1, 0.2, 0.3};
```

```
a=new double{0.1, 0.2, 0.3};
```

**a.length** – длина массива

**char [ ][ ] c = new char [5][4] ;**

**char c[ ][ ] = new char [3][ ];**

**c[0]=new char [1];**                      **c[0].length=1**

**c[1]=new char [2];**                      **c[1].length=2**

**c[2]=new char [3];**                      **c[2].length=3**

**int ABC [ ][ ][ ] = new int [3][4][5]**