www.performance-lab.ru

PerformanceLab
Обеспечение производительности ИТ

История языка Java

В начале был C++ и попытка его расширения.

В середине 1991 г. был задуман язык Oak.

В конце 1992 года коллектив проекта выпустил «*7».

В 1995 году Sun покупает Oak.

Коммерческим названием становится «Java».

2000-2005гг. появляются приложения для телефонов.

В 2005 появляются сервера.

Демонстрация «*7»: https://www.youtube.com/watch?v=1CsTH9S79qI

> Мы хотели разработать систему, которая позволяла бы создавать большую распределенную разнородную сеть из бытовых электронных устройств, способных взаимодействовать между собой.
>
> *Джеймс Гослинг*

Основные идеи

Ключевые идеи:

Написано один раз – работает везде

Встроенный сборщик мусора

Безопасность исполнения

Результат:

Кроссплатформенность

Сборщик мусора устраняет утечки памяти

Сборщик мусора всегда работает в ненужное время.

PerformanceLab
Обеспечение производительности ИТ

Точка входа в приложение

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

- Класс должен быть публичным
- Метод должен иметь называться «main» и иметь указанную сигнатуру.

Jar архив

Jar – архив (zip), с упакованными внутри скомпилированными файлами.

Должен содержать манифест.

Команда для сборки: `jar cfe ep.jar EntryPoint EntryPoint.class`

Варианты запуска:

```
java –jar ep.jar
java –classpath ep.jar EntryPoint
java –classpath lib.jar:ep.jar EntryPoint
```

Нужные определения

JVM – Виртуальная машина Java
JIT – Just-in-time Compiler – компилятор

Стек – это область оперативной памяти, которая создаётся для каждого потока. Он работает в порядке LIFO (Last In, First Out), то есть последний добавленный в стек кусок памяти будет первым в очереди на вывод из стека.

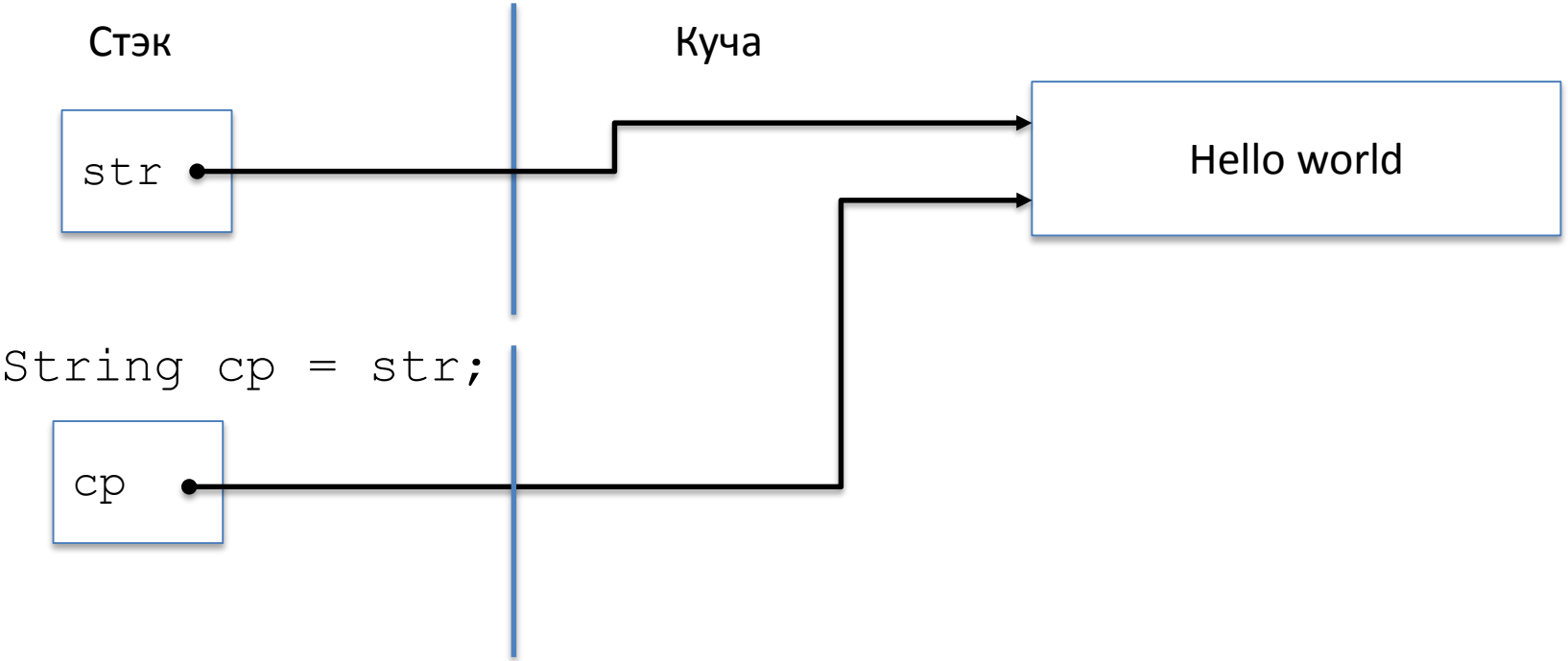
Куча — это хранилище памяти, также расположенное в ОЗУ, которое допускает динамическое выделение памяти и не работает по принципу стека: это просто склад для ваших переменных. Когда вы выделяете в куче участок памяти для хранения переменной, к ней можно обратиться не только в потоке, но и во всем приложении.

Примитивные типы

Тип	Длина	Диапазон
boolean	1	[0, 1] или [true, false]
byte	8	[-128, 127] или [-27, 27-1]
short	16	[-32768, 32767] или [-215, 215-1]
char	16	['\u0000', '\uffff'] или [0, 65535]
int	32	[-2147483648, 2147483647] или [-2^{31}, $2^{31}-1$]
long	64	[-9223372036854775808, 9223372036854775807] или [-263, 263-1]
float	32	[1.4e-45, 3.4028235e38]
double	64	[4.9e-324, 1.79769313486231573e308]

Ссылочные типы

```
String str = "Hello World";
```



Логический тип

```
boolean testTrue = true;
boolean testFalse = false;
boolean testEquals = 10 == 10;
boolean testInequality = 1 < 100;
```

Логические операции:

```
!var //не
var1 && var2 //и
var2 || var2 //или
var1 ^ var2 //исключающее или
```

```
var &= expr;
var |= expr;
var ^=expr
```

Целочисленные типы

```
int decimal = 99;
int octal = 0755;
int hex = 0x2F;
int bin = 0b100;
int tm = 10_000_000;
long tb = 10_000_000_000L;
```

Арифметические операции

```
int sum = a + b;
int diff = a - b;
int mult = a * b;
int div = a / b; //!!!
int mod = a % b;
int inc = a++; // ++a
int dec = a--; // --a
```

Переполнение:

```
byte b = 127; // 01111111
b++; // 10000000 == -128
```

Тип char

```
char literal = 'c';
char esc = '\n';
char quote = '\'';
char hex= '\u03A9';
```

Вещественные типы

Тип	Размер	Знак	Мантисса	Экспонента
float	32	1	23	8
double	64	1	52	11

$$A = +m \cdot 2^e$$

```
double simple = -1.234;
double exp = -1.234e-2;
double hex = 0x1.Fp10;
float f = 1.234f;
double d = 4d;
```

```
double f = 0.0;
for (int i=1; i <= 10; i++)
{
    f += 0.1;
}
// 0.9999999999999989
```

Вещественные типы

- Целочисленное деление на 0 генерирует исключение, в то время как результатом деления на 0 чисел с плавающей точкой является бесконечность (или `NaN` в случае деления `0.0/0`).
- `Double.NEGATIVE_INFINITY` и `Double.POSITIVE_INFINITY`, равны -1.0 / 0.0 и 1.0 / 0.0
- `Double.MIN_VALUE` на самом деле не самое маленькое число, которое можно записать в double, а число максимально близкое к 0.
- Самым маленьким значением, которое вы можете сохранить в double является "–`Double.MAX_VALUE`".
- Метод `Math.nextUp()` возвращает следующее число с плавающей точкой.

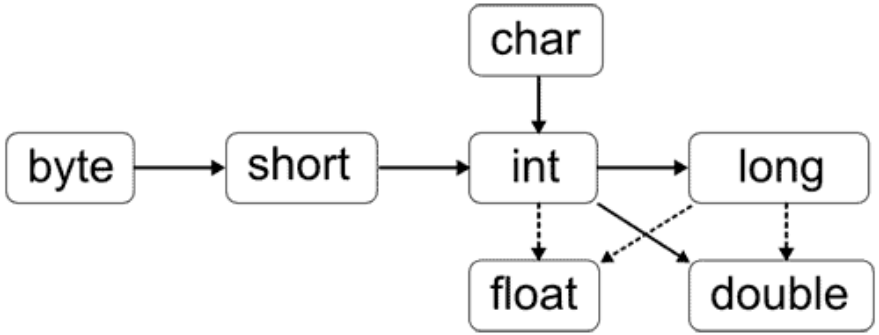
Пакет Math

```
Math.sin();
Math.sqrt();
Math.ceil();
Math.abs();
Math.max();
```

```
BigInteger.valueOf("2")
BigDecimal.valueOf("1")
```

PerformanceLab
Обеспечение производительности ИТ

Преобразование типов



Мораль: Никогда не используйте типы меньше int без существенной на то необходимости

```
double dv = 1d + 1f;
float fv= 1f + 1;
long lv = 1L – '0';

byte a = 1;
byte b = -1;
int c = a + b;
a += 3; //(byte)(a + 3)
b >>>= 7;//(byte)(b >>> 7) == -1
```

Массивы

Массив – ссылочный тип, конечная упорядоченная последовательность элементов в памяти. Может нести в себе любой из примитивных типов либо ссылку на объект в куче.

Массивы

```
int[] arrayOfInt;
arrayOfInt = new int[100]; // Default value: 0
int[] arrayOfString = new String[1]; // null
int[] arrayOfBool = new bool[0];  // false
int num[] = {1, 2, 3, 4, 5};
int matrix[][] = new int[2][2];
int matrix2[][] = {{1, 2}, {3, 4, 5}};

static int printFirstFalue(int... values) {
    System.out.println(values[0]);
}
```

PerformanceLab
Обеспечение производительности ИТ

Сравнение массивов

```
int[] a = {1, 2, 3};
int[] b = {1, 2, 3};
a == b; //false
a.equals(b); //false
Arrays.equals(a, b); //true
Arrays.deepEquals(a, b); //true
```

Полезные функции:

```
Arrays.sort(a);
Arrays.sort(a, Collections.reverseOrder());
Arrays.toString(a);
```

Строки

```
String str = "new string";
String empty = "";
char[] charSequence = {'n', 'e', 'w'};
String fromArray = new String(charSequence);
String zeroChar="\u0000";
```

Полезные методы:

```
int length = str.length();
char firstChar = str.charAt(0);
str.contains("new");
str.substring(0, 6);
```

Строки

Конкатенация:

```
String str1 = "a";
String str2 = "b";
String result = str1 + str2;
StringBuinder sb = new StringBuilder();
sb.append(str1);
sb.append(str2);
String resultsb = sb.toString();
```

Сравнение:

```
str1.equals(str2); str1.equalsIgnoreCase(str2);
```

Условный оператор

```
if (boolVar) {
    //do something
} else if {
    //another action
} else {

}

boolvar ? actionTrue() : actionFalse();
```

PerformanceLab
Обеспечение производительности ИТ

Условный оператор

```
switch (someDigit) {//char, String, Enum, int, !long
    case 0:
        //do something
        break;
    case 1:
        //do something
        break;
    default:
        //do something
}
```

Циклы

```
while (boolExpr) {
    //do something
}

do {
    //do something
} while (boolExpr)

for (int i=0; i<10; i++) {
    //do something
}
```

```
for (String arg : args) {
    //do something
    break;
    continue;
}
outer:
while (boolExpr) {
    //do something
    while (boolExpr2) {
        break outer;
    }
}
```

PerformanceLab

Обеспечение производительности ИТ

Спасибо за внимание