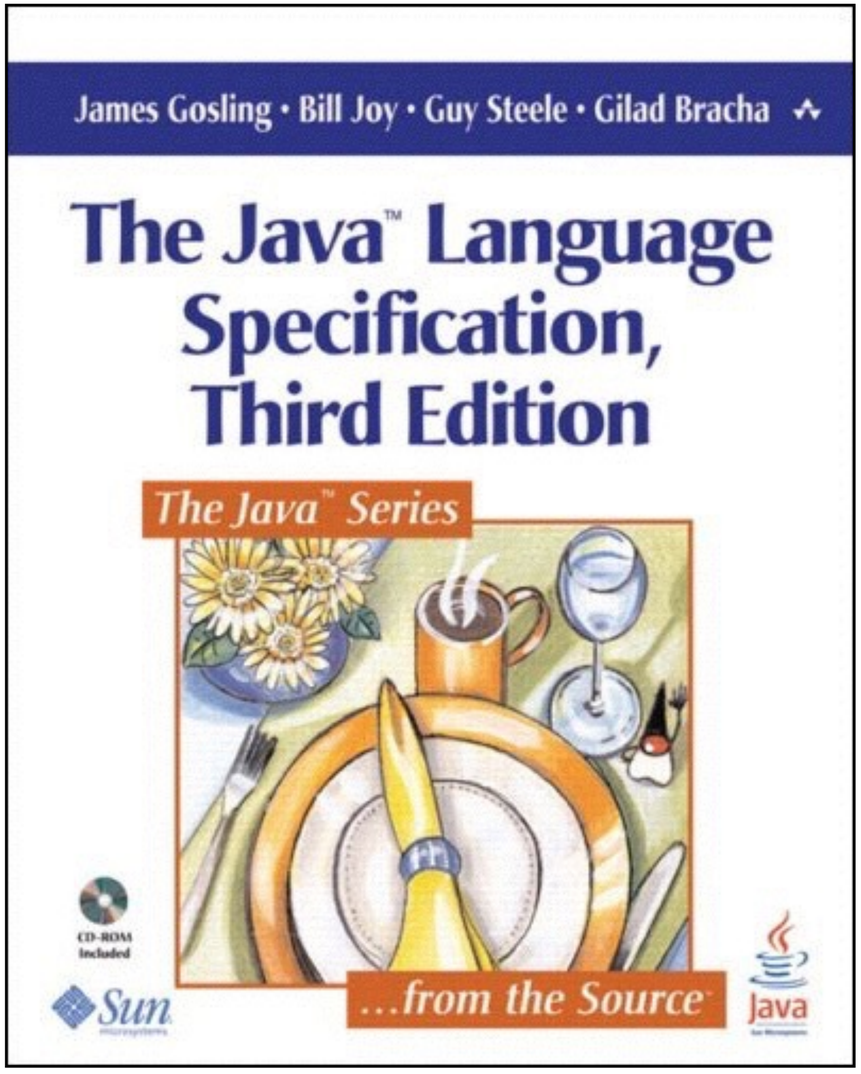
Байткод для любознательных

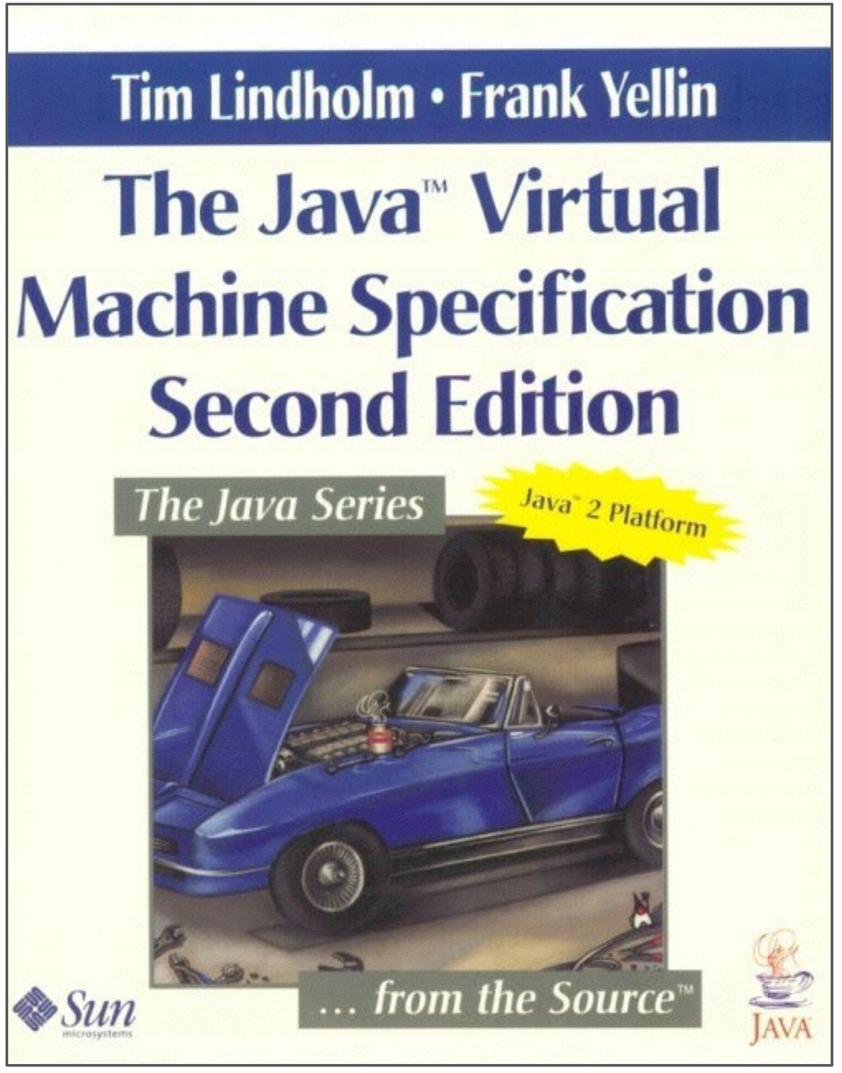
@antonarhipov

JPoint 2016

Антон Архипов

- ZeroTurnaround
- Product Manager
- Таллин, Эстония
- Инструменты для Java-разработчиков
- Программирую на Java с 2001 года


James Gosling • Bill Joy • Guy Steele • Gilad Bracha
The Java™ Language Specification, Third Edition
The Java™ Series
CD-ROM Included
...from the Source™
Sun microsystems
Java


Tim Lindholm • Frank Yellin
The Java™ Virtual Machine Specification Second Edition
The Java Series
Java™ 2 Platform
... from the Source™
Sun microsystems
JAVA

Зачем?

- Стоит знать свою платформу!
- Может быть вы хотите написать свой компилятор?
- Фреймворки (AOP, ORM)
- Всевозможные инструменты, например JRebel :)

- … ну или может просто скучно?

INTRO

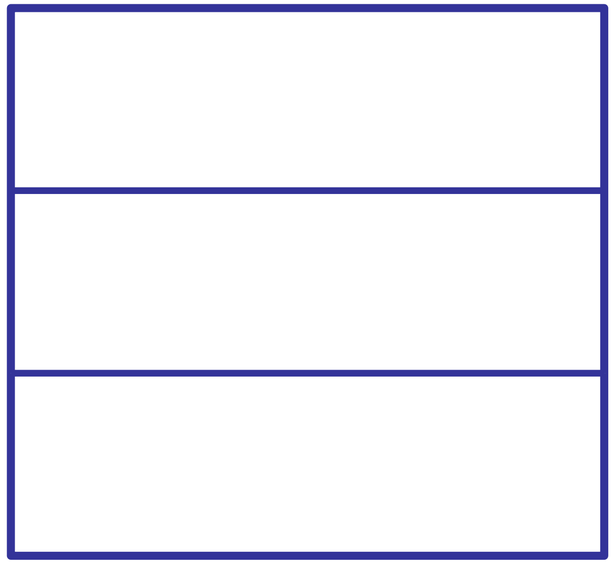
1 + 2

1 + 2

1 2 +

1 + 2

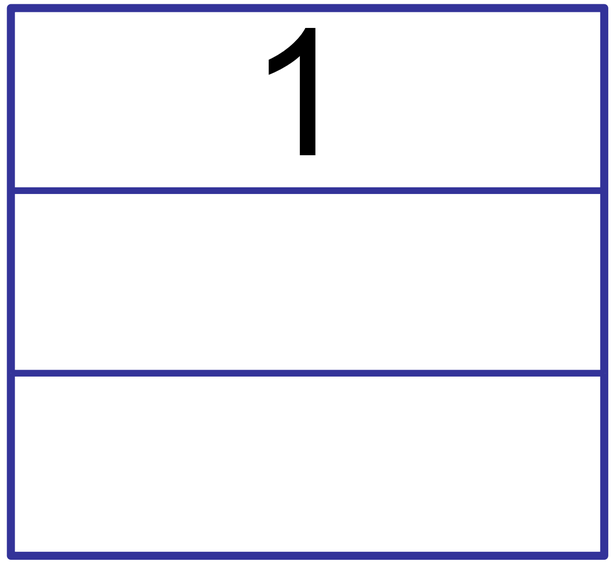
1 2 +



1 + 2

1 2 +

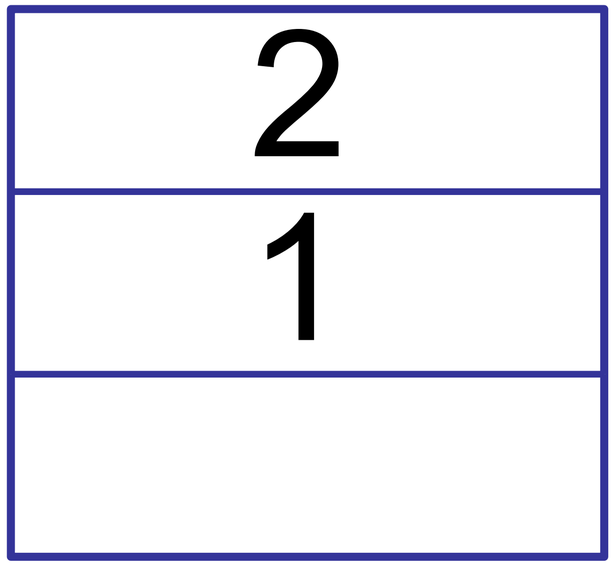
PUSH 1



1 + 2

1 2 +

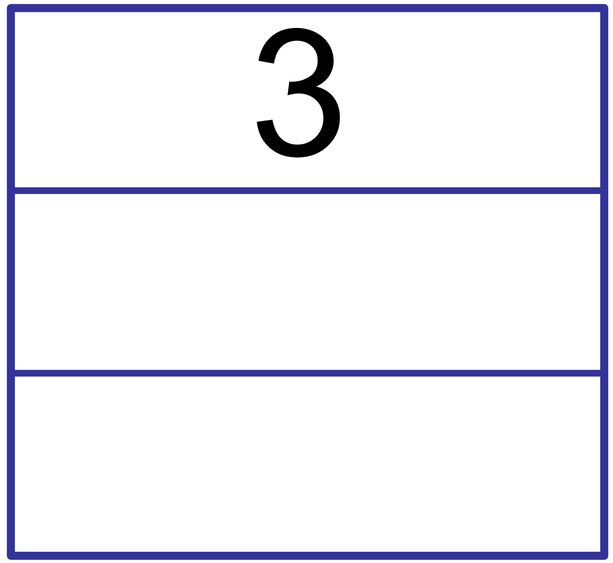
PUSH 1
PUSH 2



1 + 2

1 2 +

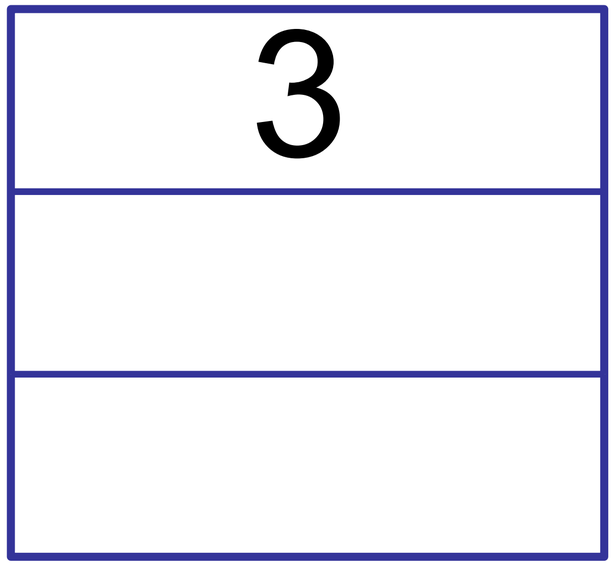
PUSH 1
PUSH 2
ADD



1 + 2

1 2 +

ICONST_1
ICONST_2
IADD



$$? = 1 + 2$$

ТАКСОНОМИЯ

Байт-код

- Одно-байтные инструкции
- 256 возможных вариантов
- Используется 200+

ТИП ОПЕРАЦИЯ

ТИП **ОПЕРАЦИЯ**

- <тип> ::= b, s, c, i, l, f, d, a

ТИП **ОПЕРАЦИЯ**

- <тип> ::= b, s, c, i, l, f, d, a
- ***константы (ldc, iconst_1)***

ТИП **ОПЕРАЦИЯ**

- <тип> ::= b, s, c, i, l, f, d, a
- ***константы (ldc, iconst_1)***
- локальные переменные и стек (**load**/**store**)
- Операции с массивами (**aload**, **astore**)
- Арифметика (**add**, **sub**, **mul**, **div**)
- Булевые/битовые операции (**iand**, **ixor**)
- Сравнения (**cmpg**, **cmpl**, **ifne**, **ifeq**)
- Преобразования (**l2d**, **i2l**)

Таксономия

Таксономия

Таксономия

Таксономия


Работа со стеком
Инструкции управления
Работа с объектами

Таксономия


Работа со стеком
Инструкции управления
Арифметика
Работа с объектами

Таксономия

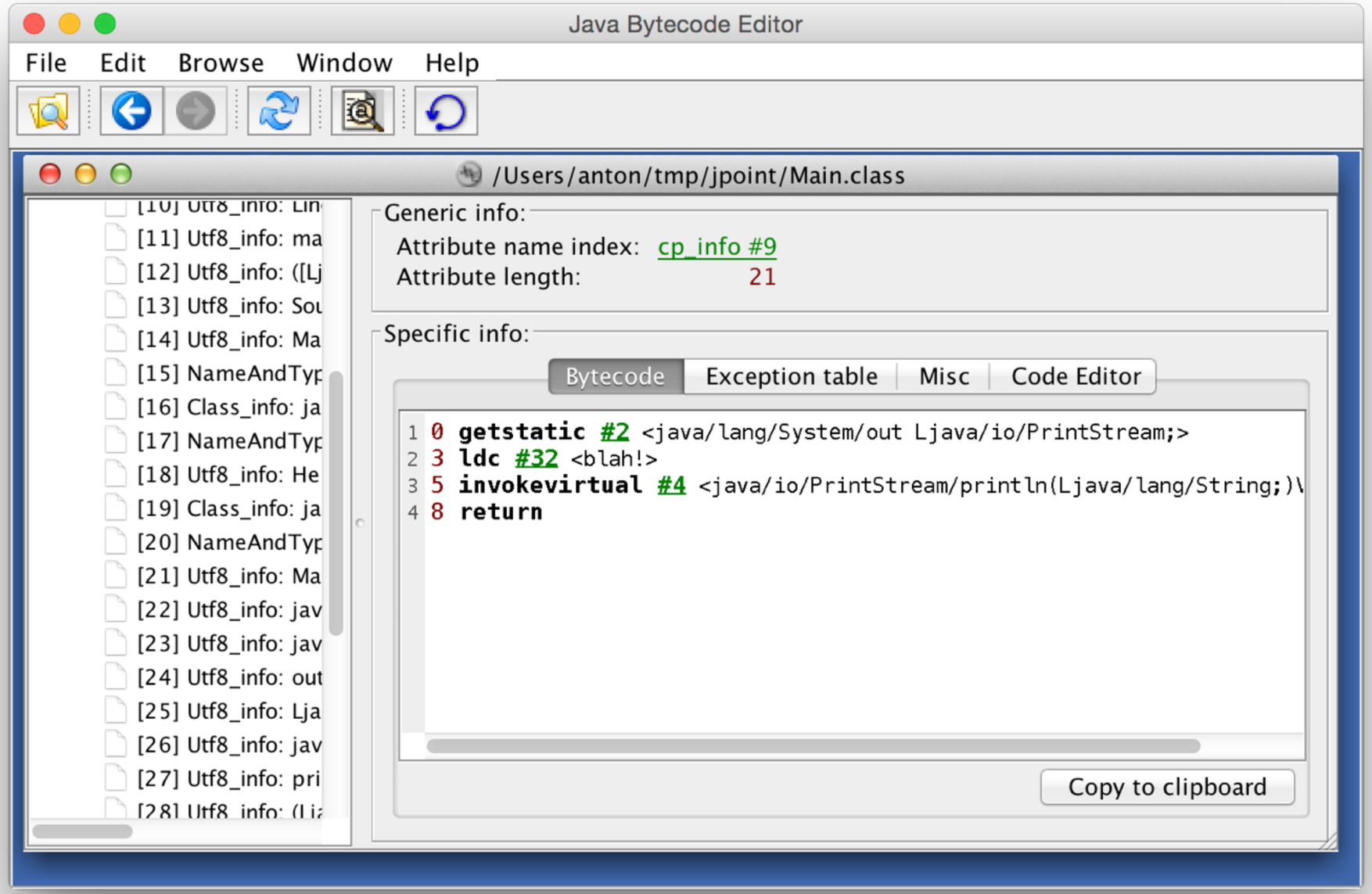

Работа со стеком
Инструкции управления
monitorenter
monitorexit
Арифметика
Работа с объектами

ИНСТРУМЕНТАРИЙ

javap

- Дизассемблер Java класс-файлов
- По-умолчанию показывает только структуру класса
  - Методы, супер-класс, интерфейсы, итд
- **-c** покажет байткод методов
- **-private** покажет все приватные поля и методы
- **-s** покажет сигнатуры
- **-l** покажет номера строк и таблицу локальных переменных

Java Bytecode Editor — http://set.ee/jbe/



ObjectWeb ASM

ASM - Home Page

Anton

asm.ow2.org

OW2 Consortium

Leading Open Source Middleware

Consortium | Activities | Projects | Forge | Events

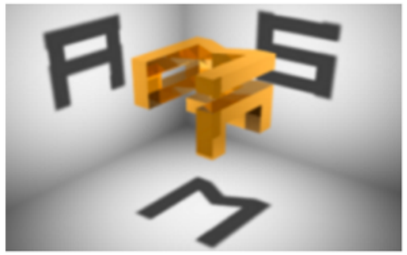
ASM

ASM
- Home
- Download
- Eclipse plugin
- User Guide
- Mailing Lists
- License
- History

ASMDEX
- Home
- Download
- Mailing Lists
- License
- History

Developers' Corner
- ObjectWeb Forge Site
- SVN Repository
- Issue Tracker
- Developer Guide

ASM



ASM is an all purpose Java bytecode manipulation and analysis framework. It can be used to modify existing classes or dynamically generate classes, directly in binary form. Provided common transformations and analysis algorithms allow to easily assemble custom complex transformations and code analysis tools.

ASM offer similar functionality as other bytecode frameworks, but it is focused on simplicity of use and performance.

Documentation

The best way to learn to use ASM is to write a Java source file that is equivalent to what you want to generate and then use the ASMifier mode of the Bytecode Outline plugin for Eclipse (or the ASMifier tool) to see the equivalent ASM code. If you want to implement a class transformer, write two Java source files (before and after transformation) and use the compare view of the plugin in ASMifier mode to compare the equivalent ASM code.

The following document provides an in-depth introduction to ASM 4.0, going from the basic features to advanced topics such as bytecode analysis.

ASM 4.0 A Java bytecode engineering library.

Tutorial for ASM 2.0.

Tutorial for ASM 1.5.x.

Tutorial for using J2SE 5.0 Annotations with ASM 1.5.x.

HELLO WORLD!

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}

```

```
1  public class Hello {
2
3    public static void main(String[] args) {
4      System.out.println("Hello, World!");
5    }
6
7  }
8
```

C:\work\geecon\classes>**javap Hello -c**

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return
```

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
     0:   aload_0
     1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
     4:   return
```

конструктор по-умолчанию

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
     0:   aload_0
     1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
     4:   return
```

выложить this на стек

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return
```

вызвать <init> для this

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return
```

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3:   ldc     #3; //String Hello, World!
   5:   invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
```

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
    0:  aload_0
    1:  invokespecial   #1; //Method java/lang/Object."<init>":()V
    4:  return

public static void main(java.lang.String[]);
  Code:
    0:  getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
    3:  ldc     #3; //String Hello, World!
    5:  invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
```

обратиться к статическому полю

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3:   ldc     #3; //String Hello, World!
   5:   invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
```

загрузить строковую константу в стек

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3:   ldc     #3; //String Hello, World!
   5:   invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
```

вызвать метод с параметром

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3:   ldc     #3; //String Hello, World!
   5:   invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
```

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c
Compiled from "Hello.java"
public class Hello extends java.lang.Object{
public Hello();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3:   ldc     #3; //String Hello, World!
   5:   invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
```

Что такое #1,#2, итд ?

```
1  public class Hello {
2
3    public static void main(String[] args) {
4      System.out.println("Hello, World!");
5    }
6
7  }
8
```

C:\work\geecon\classes>**javap Hello -c -verbose**

```
1  public class Hello {
2
3    public static void main(String[] args) {
4      System.out.println("Hello, World!");
5    }
6
7  }
8
```

C:\work\geecon\classes>**javap Hello -c -verbose**

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c -verbose
Compiled from "Hello.java"
public class Hello extends java.lang.Object
  SourceFile: "Hello.java"
  minor version: 0
  major version: 50
  Constant pool:
const #1 = Method       #6.#20; //  java/lang/Object."<init>":()V
const #2 = Field        #21.#22;        //  java/lang/System.out:Ljava/io/PrintStream;
const #3 = String       #23;    //  Hello, World!
const #4 = Method       #24.#25;        //  java/io/PrintStream.println:(Ljava/lang/String;)V
const #5 = class        #26;    //  Hello
const #6 = class        #27;    //  java/lang/Object
const #7 = Asciz        <init>;
const #8 = Asciz        ()V;
```

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c -verbose
Compiled from "Hello.java"
public class Hello extends java.lang.Object
  SourceFile: "Hello.java"
  minor version: 0
  major version: 50
  Constant pool:
const #1 = Method       #6.#20; //  java/lang/Object."<init>":()V
const #2 = Field        #21.#22;        //  java/lang/System.out:Ljava/io/PrintStream;
const #3 = String       #23;    //  Hello, World!
const #4 = Method       #24.#25;        //  java/io/PrintStream.println:(Ljava/lang/String;)V
const #5 = class        #26;    //  Hello
const #6 = class        #27;    //  java/lang/Object
const #7 = Asciz        <init>;
const #8 = Asciz        ()V;
```

```java
public class Hello {

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

```
C:\work\geecon\classes>javap Hello -c -verbose
...
public Hello();
  Code:
   Stack=1, Locals=1, Args_size=1
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return
  LineNumberTable:
   line 1: 0

  LocalVariableTable:
   Start  Length  Slot  Name   Signature
   0      5      0    this      LHello;
```

```
1  public class Hello {
2
3    public static void main(String[] args) {
4      System.out.println("Hello, World!");
5    }
6
7  }
8
```

```
C:\work\geecon\classes>javap Hello -c -verbose
...
public Hello();
  Code:
   Stack=1, Locals=1, Args_size=1
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return
  LineNumberTable:
   line 1: 0

  LocalVariableTable:
   Start  Length  Slot  Name   Signature
   0      5      0    this      LHello;
```

```java
public class Hello {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }

}
```

```
C:\work\geecon\classes>javap Hello -c -verbose
...
public Hello();
  Code:
    Stack=1, Locals=1, Args_size=1
    0:   aload_0
    1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
    4:   return
  LineNumberTable:
    line 1: 0

  LocalVariableTable:
    Start  Length  Slot  Name   Signature
    0      5       0     this      LHello;
```

```java
public class Hello {

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

```
C:\work\geecon\classes>javap Hello -c -verbose
...
public static void main(java.lang.String[]);
  Code:
    Stack=2, Locals=1, Args_size=1
    0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
    3:   ldc     #3; //String Hello, World!
    5:   invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
    8:   return
  LineNumberTable:
    line 4: 0
    line 5: 8
  LocalVariableTable:
    Start  Length  Slot  Name   Signature
    0      9       0     args       [Ljava/lang/String;
```

МОДЕЛЬ ВЫЧИСЛЕНИЙ

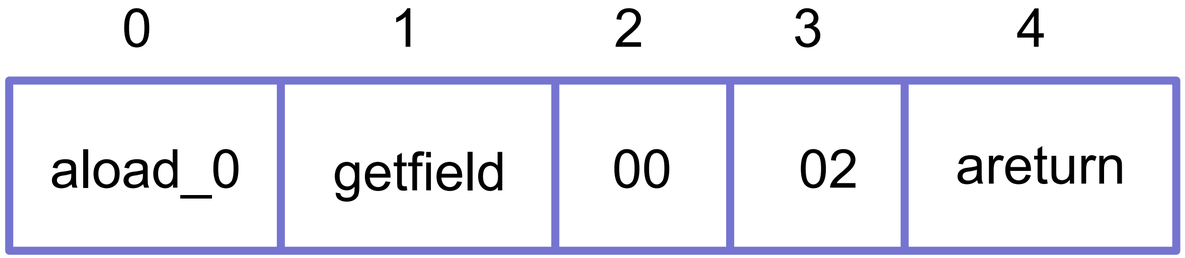
Стековая машина

- JVM работает со стеком
- У каждого потока есть стек
- Стек сохраняет “фреймы”
- Новый “фрейм” создаётся при вызове метода
- “Фрейм состоит из”:
  - Стек операций
  - Массив локальных переменных

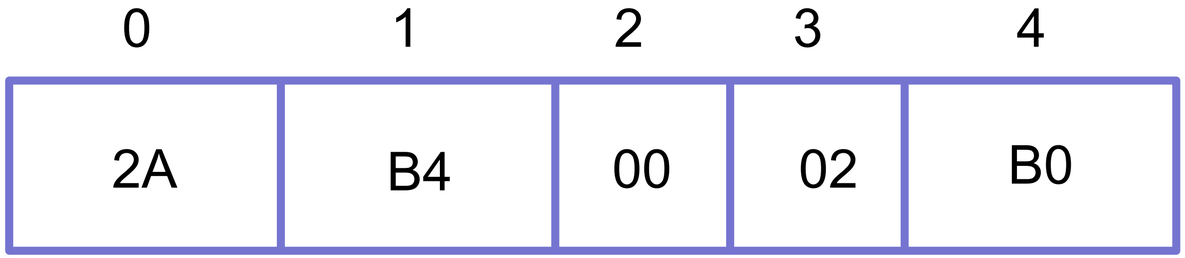

Локальные переменные
0
1
2
...
N
Стек операций
#1
Константы

```java
public class Get {

    String name;

    public String getName() {
        return name;
    }
}
```

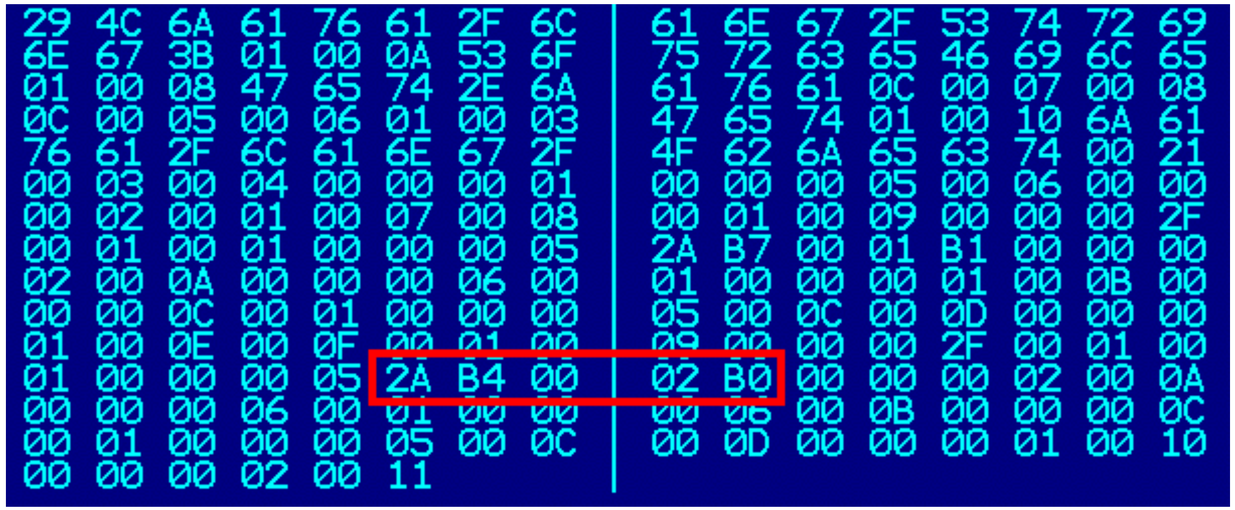
```
public java.lang.String getName();
  Code:
    Stack=1, Locals=1, Args_size=1
    0:   aload_0
    1:   getfield      #2; //Field name:Ljava/lang/String;
    4:   areturn
LocalVariableTable:
    Start  Length  Slot  Name   Signature
    0      5      0    this      LGet;
```




```
public java.lang.String getName();
  Code:
   Stack=1, Locals=1, Args_size=1
   0:   aload_0
   1:   getfield        #2; //Field name:Ljava/lang/String;
   4:   areturn
LocalVariableTable:
   Start  Length  Slot  Name   Signature
   0      5      0    this      LGet;
```




```
public java.lang.String getName();
  Code:
   Stack=1, Locals=1, Args_size=1
   0:   aload_0
   1:   getfield        #2; //Field name:Ljava/lang/String;
   4:   areturn
LocalVariableTable:
   Start  Length  Slot  Name   Signature
   0      5      0    this      LGet;
```



public java.lang.String getName();

Code:

Stack=1, Locals=1, Args_size=1

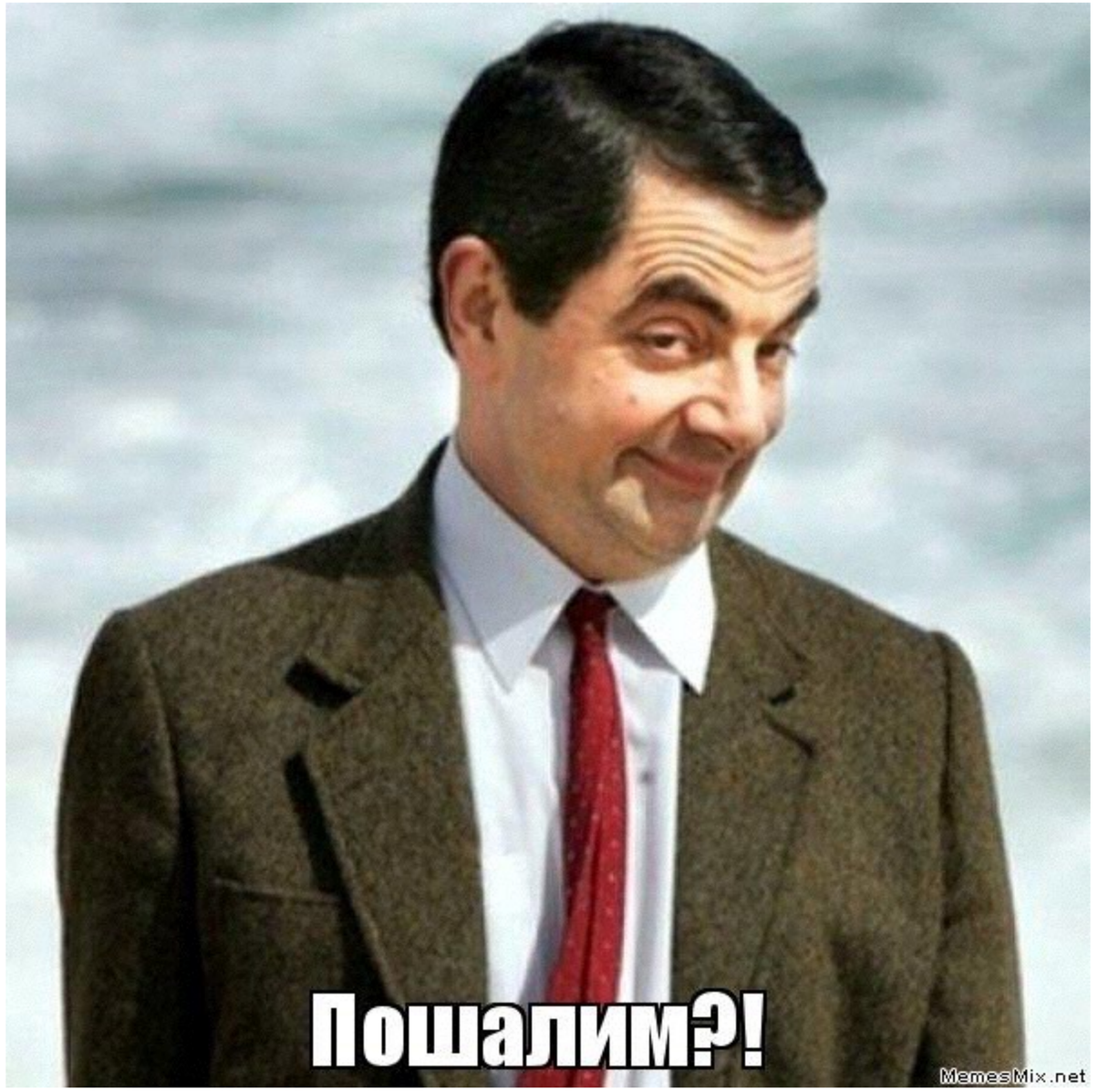
0: aload_0

1: getfield #2; //Field name:Ljava/lang/String;

4: areturn

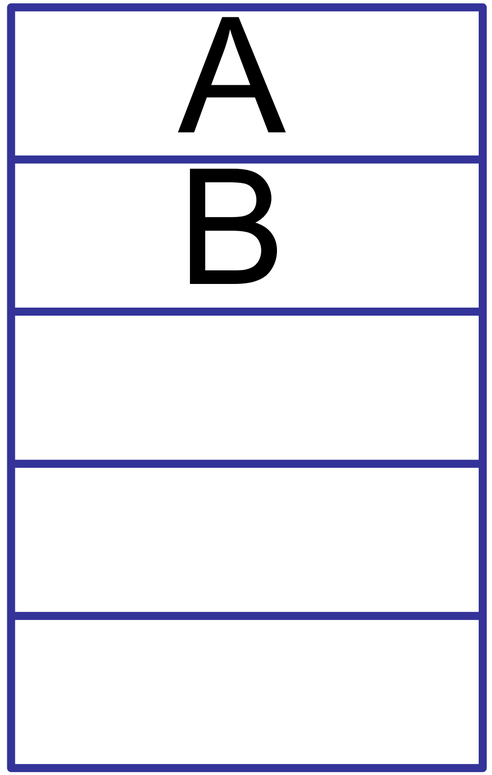
LocalVariableTable:

Start	Length	Slot	Name	Signature
0	5	0	this	LGet;


Пошалим?!
MemesMix.net

СТЕКОВЫЕ ОПЕРАЦИИ

dup
pop
swap
dup_x1
dup2_x1



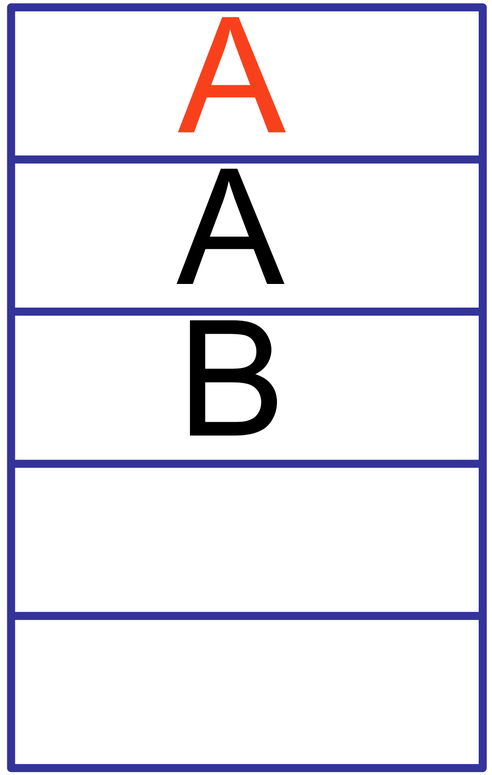
dup

pop

swap

dup_x1

dup2_x1



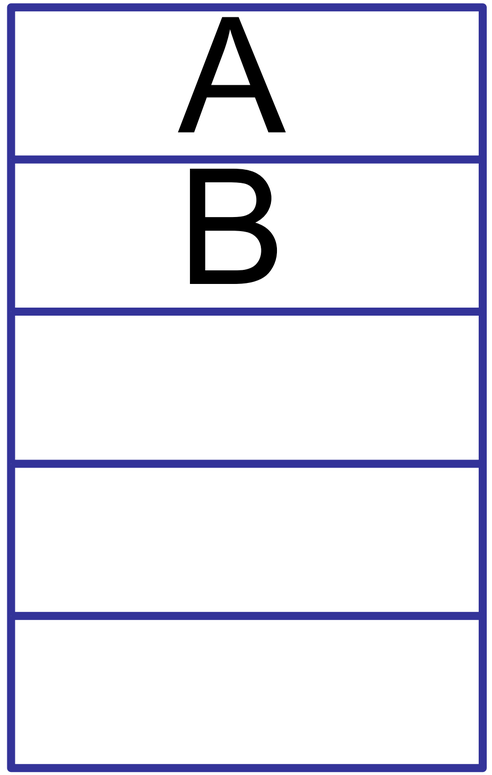
dup

pop

swap

dup_x1

dup2_x1



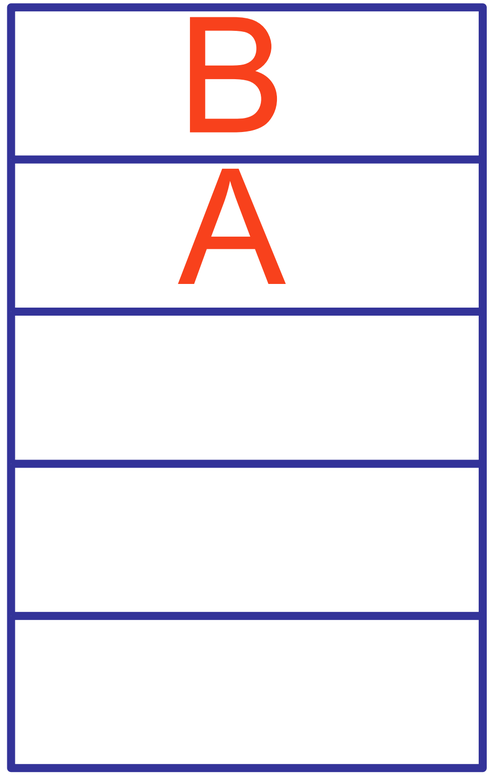
dup

pop

swap

dup_x1

dup2_x1



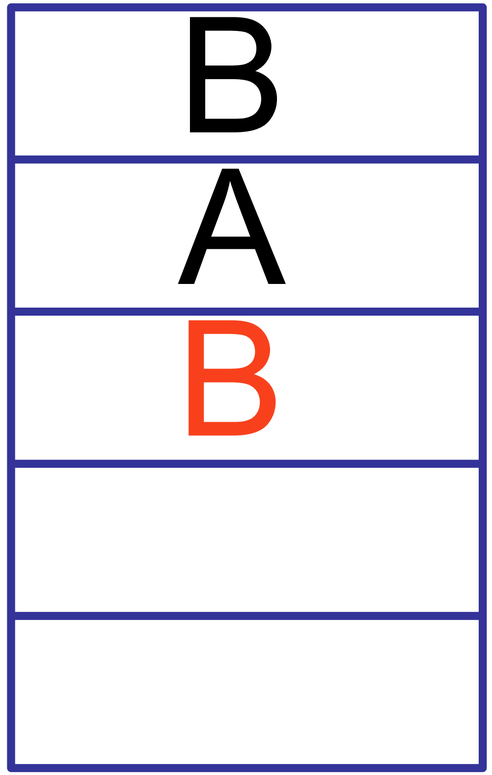
dup

pop

swap

dup_x1

dup2_x1



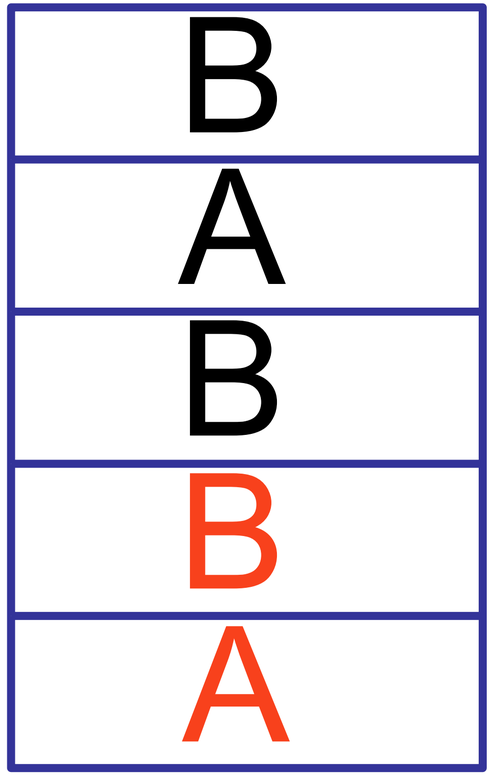
dup

pop

swap

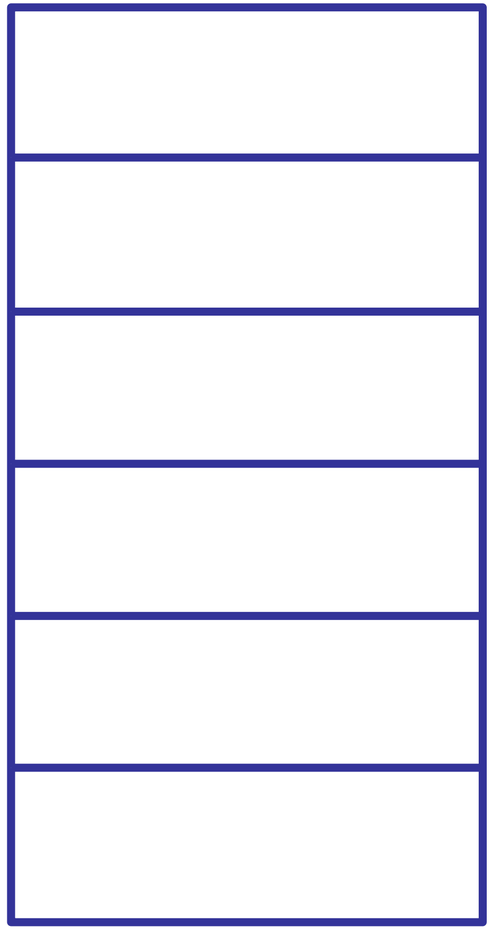
dup_x1

dup2_x1

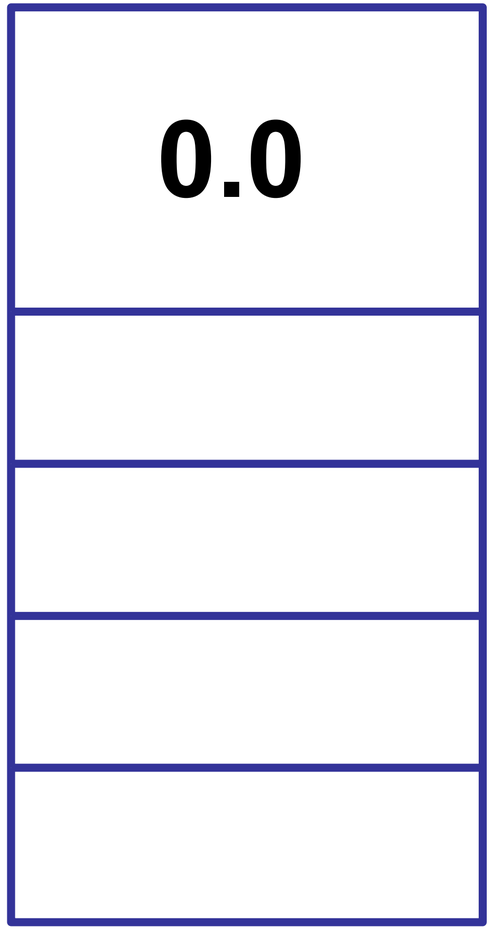


dup2_x2

Как поменять местами две переменные типа double?

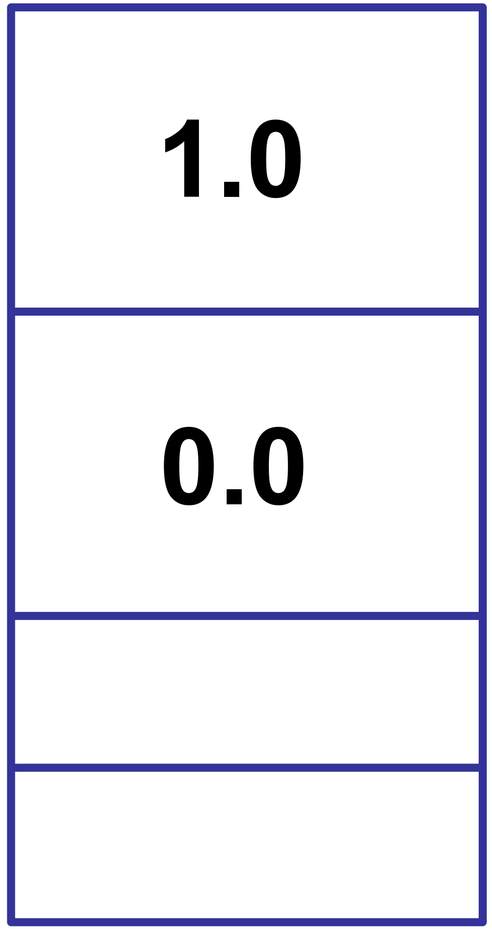


dconst_0

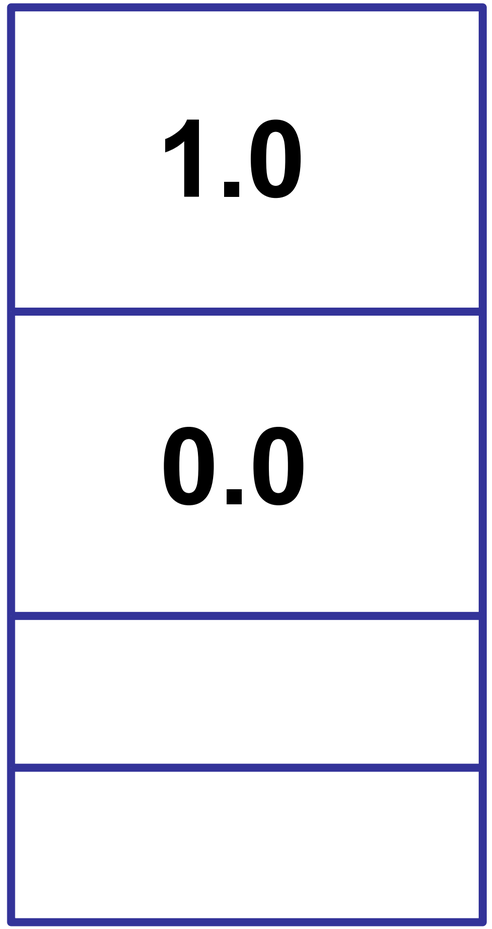


dconst_0

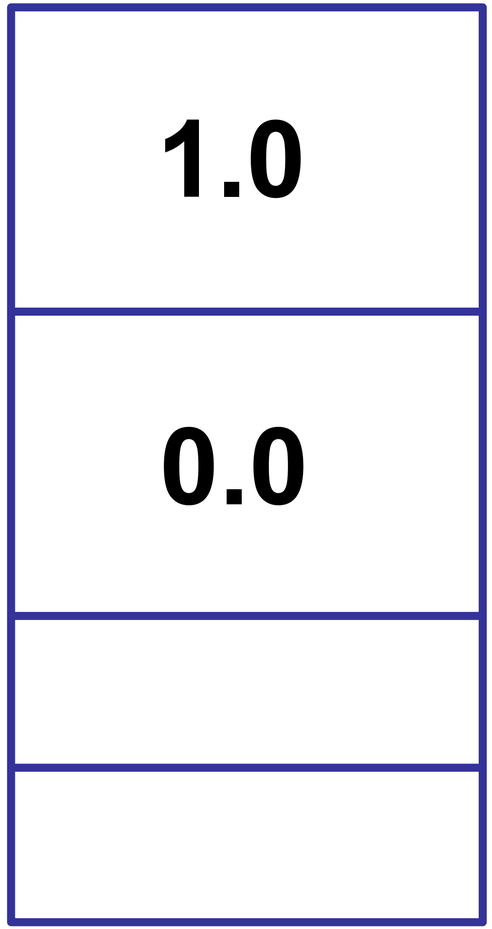
dconst_1



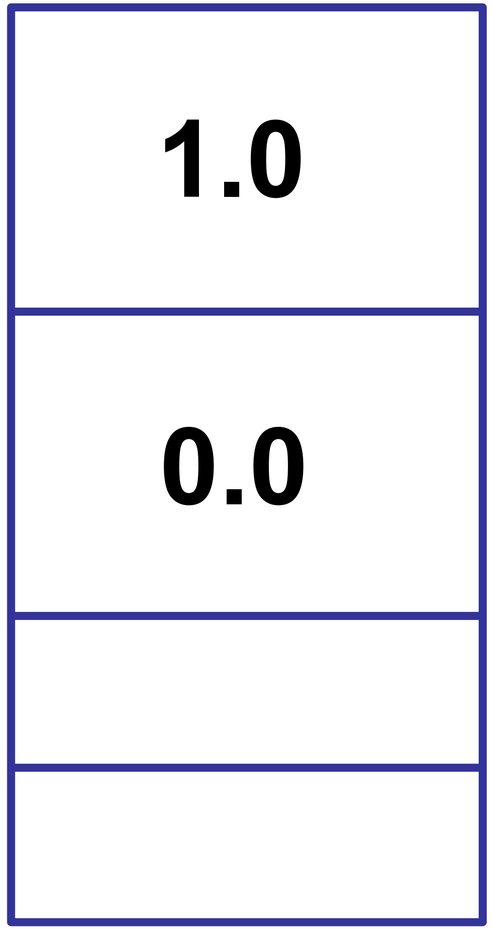
dconst_0
dconst_1
swap



dconst_0
dconst_1
swap ⟵ **нельзя!**

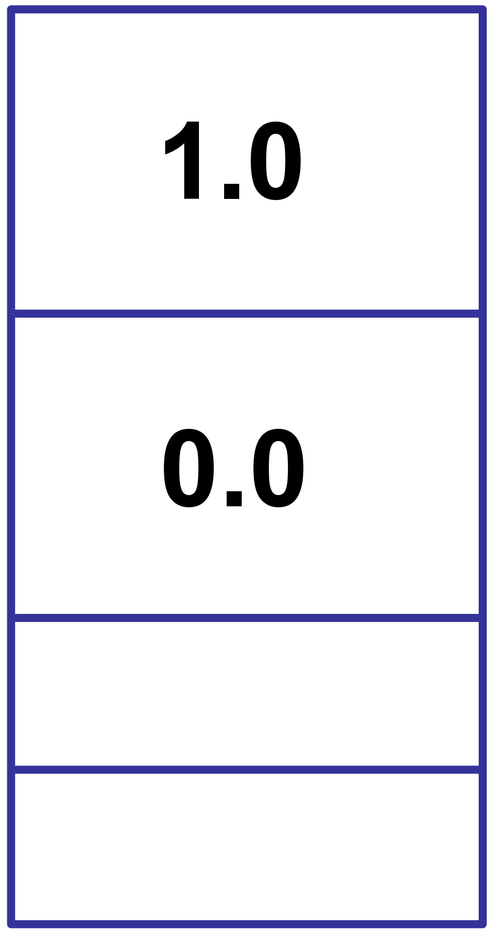


dconst_0
dconst_1
swap2

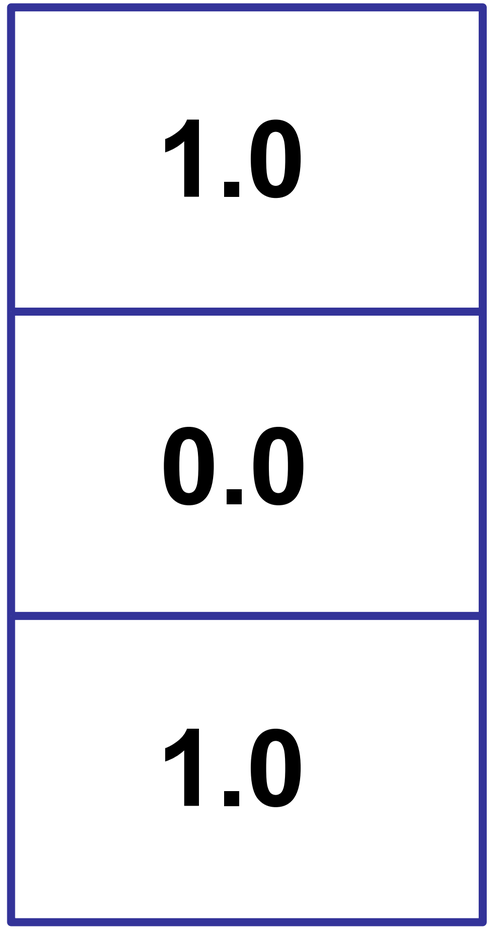


dconst_0
dconst_1
swap2

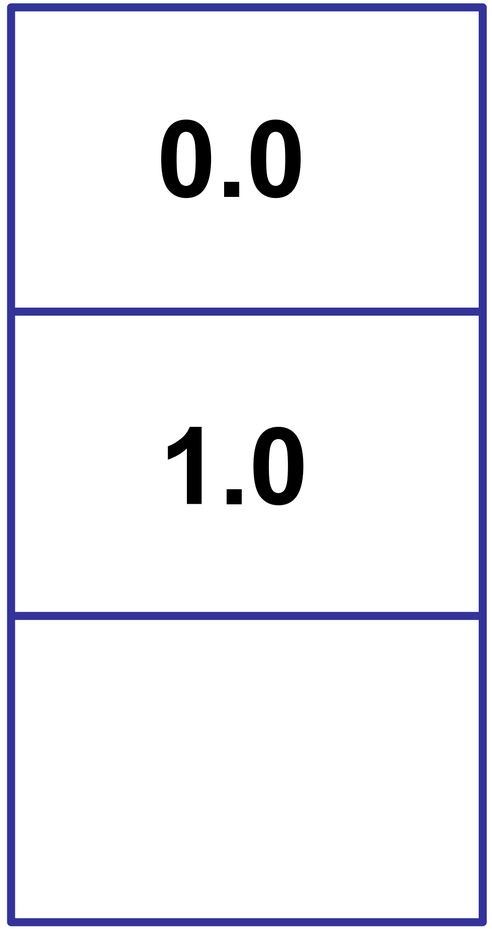
нет такой инструкции



dconst_0
dconst_1
dup2_x2



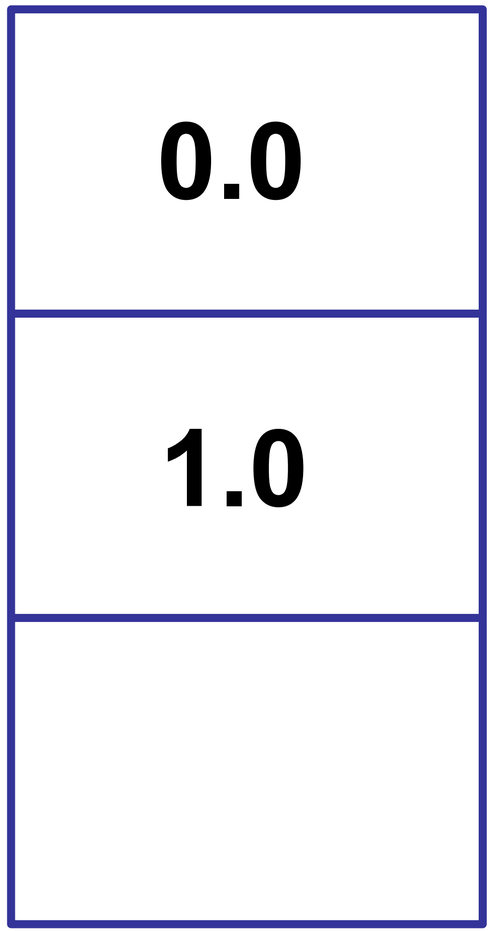
dconst_0
dconst_1
dup2_x2
pop2



dconst_0
dconst_1
dup2_x2
pop2

profit!

☺



ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ

Локальные переменные

```
public int calculate(int value){
    return value + 42;
}
```

Локальные переменные

```
public int calculate(int value){
    return value + 42;
}
```

```
public int calculate(int);
  Code:
    Stack=2, Locals=2, Args_size=2
    …

  LocalVariableTable:
    Start  Length  Slot  Name   Signature
        0       5     0  this   LLocalVariables;
        0       5     1  value  I
```

Локальные переменные

```
public int calculate(int value){
    return value + 42;
}
```

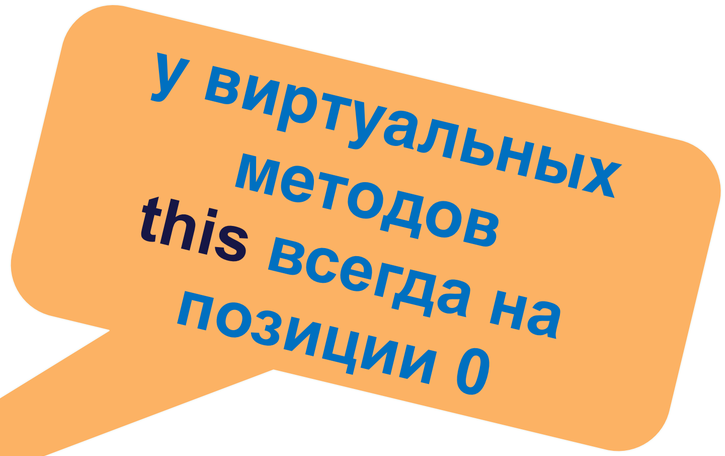
```
public int calculate(int);
  Code:
    Stack=2, Locals=2, Args_size=2
    …

  LocalVariableTable:
    Start  Length  Slot  Name   Signature
        0       5     0  this   LLocalVariables;
        0       5     1  value  I
```

Нумеровка элементов с нуля

Локальные переменные

```
public int calculate(int value){
    return value + 42;
}
```

```
public int calculate(int);
  Code:
    Stack=2, Locals=2, Args_size=2
    …

  LocalVariableTable:
    Start  Length  Slot  Name   Signature
        0       5     0  this   LLocalVariables;
        0       5     1  value  I
```



Локальные переменные

```
public int calculate(int value){
    return value + 42;
}
```

```
public int calculate(int);
  Code:
    Stack=2, Locals=2, Args_size=2
    ...

  LocalVariableTable:
    Start  Length  Slot  Name   Signature
        0       5     0  this   LLocalVariables;
        0       5     1  value  I
```

таблица сопоставления названия переменных к порядковому номеру

Локальные переменные

```
public int calculate(int value){
    return value + 42;
}
```

```
public int calculate(int);
  Code:
    Stack=2, Locals=2, Args_size=2
    …

  LocalVariableTable:
    Start  Length  Slot  Name   Signature
        0       5     0  this   LLocalVariables;
        0       5     1  value  I
```

размерность предопределена

Локальные переменные

переменная	значение
0	
1	
2	
3	
4	

```
ldc "Hello"
astore_0
iconst_1
astore_1
aload_0
```

Стек

глубина	значение
0	
1	
2	
3	
4	

Локальные переменные

переменная	значение
0	
1	
2	
3	
4	

```
ldc "Hello"
astore_0
iconst_1
astore_1
aload_0
```

Стек

глубина	значение
0	"Hello"
1	
2	
3	
4	

Локальные переменные

переменная	значение
0	"Hello"
1	
2	
3	
4	

```
ldc "Hello"
astore_0
iconst_1
astore_1
aload_0
```

Стек

глубина	значение
0	
1	
2	
3	
4	

Локальные переменные

переменная	значение
0	"Hello"
1	
2	
3	
4	

```
ldc "Hello"
astore_0
iconst_1
astore_1
aload_0
```

Стек

глубина	значение
0	1
1	
2	
3	
4	

Локальные переменные

переменная	значение
0	"Hello"
1	1
2	
3	
4	

```
ldc "Hello"
astore_0
iconst_1
astore_1
aload_0
```

Стек

глубина	значение
0	
1	
2	
3	
4	

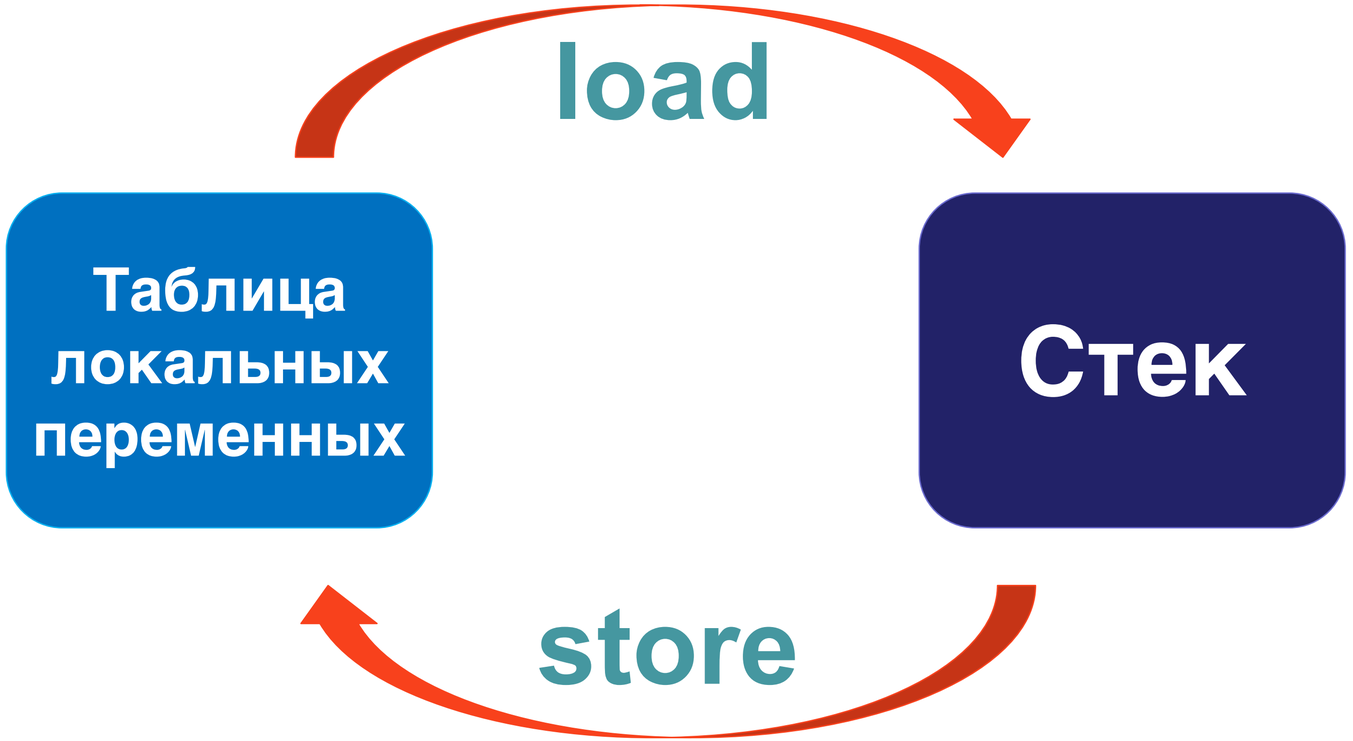
Локальные переменные

переменная	значение
0	"Hello"
1	1
2	
3	
4	

```
ldc "Hello"
astore_0
iconst_1
astore_1
aload_0
```

Стек

глубина	значение
0	"Hello"
1	
2	
3	
4	


load
Таблица локальных переменных
Стек
store

ОБЪЕКТЫ

Создание объектов

new

0xBB

<init>

инициализация экземпляра объекта

<clinit>

инициализация класса

static {}

```
1  public class Initializer {
2     static int a;
3     static int b;
4
5     static { a = 1; }
6     static { b = 2; }
7  }
```

static {};

```
  Code:
    0:   iconst_1
    1:   putstatic     #2; //Field a:I
    4:   iconst_2
    5:   putstatic     #3; //Field b:I
    8:   return
```

static {}

```
1  public class Initializer {
2     static int a;
3     static int b;
4
5     static { a = 1; }
6     static { b = 2; }
7  }
```

<clinit>

```
static {};
  Code:
    0:   iconst_1
    1:   putstatic      #2; //Field a:I
    4:   iconst_2
    5:   putstatic      #3; //Field b:I
    8:   return
```

new

new

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

new

public Initializer();

Code:

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

new

```
public Initializer();
  Code:
    0:   aload_0
```

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

new

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

public Initializer();
Code:
0: aload_0
1: invokespecial #1; //Method java/lang/Object."<init>":()V

new

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

public Initializer();
Code:
0: aload_0
1: invokespecial #1; //Method java/lang/Object."<init>":()V
4: aload_0

new

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

```
public Initializer();
  Code:
    0:   aload_0
    1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
    4:   aload_0
    5:   new     #2; //class java/lang/Object
    8:   dup
```

new

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

```
public Initializer();
  Code:
    0:   aload_0
    1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
    4:   aload_0
    5:   new     #2; //class java/lang/Object
    8:   dup
    9:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   12:   putfield        #3; //Field o:Ljava/lang/Object;
```

new

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

```
public Initializer();
  Code:
    0:   aload_0
    1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
    4:   aload_0
    5:   new     #2; //class java/lang/Object
    8:   dup
    9:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   12:   putfield        #3; //Field o:Ljava/lang/Object;
   15:   return
```

new

```
1  public class Initializer{
2
3    Object o;
4
5    public Initializer(){
6      o = new Object();
7    }
8  }
```

```
public Initializer();
  Code:
    0:   aload_0
    1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
    4:   aload_0
    5:   new     #2; //class java/lang/Object
    8:   dup
    9:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   12:   putfield        #3; //Field o:Ljava/lang/Object;
   15:   return
```

{ }

```
1   public class Initializer {
2     int a;
3     int b;
4     int c;
5
6     { a = 1; }
7
8     public Initializer(int b) {
9       this.b = b;
10    }
11
12    { c = 2; }
13
14  }
15
```

{ }

```
1   public class Initializer {
2     int a;
3     int b;
4     int c;
5
6     { a = 1; }
7
8     public Initializer(int b) {
9       this.b = b;
10    }
11
12    { c = 2; }
13
14  }
15
```

{ }

```
1   public class Initializer {
2     int a;
3     int b;
4     int c;
5
6     { a = 1; }
7
8     public Initializer(int b) {
9       this.b = b;
10    }
11
12    { c = 2; }
13
14  }
15
```

```
public Initializer(int);
  Code:
    0: aload_0
    1: invokespecial  #1; // ..<init>
    4: aload_0
    5: iconst_1
    6: putfield       #2; //Field a:I
    9: aload_0
   10: iconst_2
   11: putfield       #3; //Field c:I
   14: aload_0
   15: iload_1
   16: putfield       #4; //Field b:I
   19: return
```

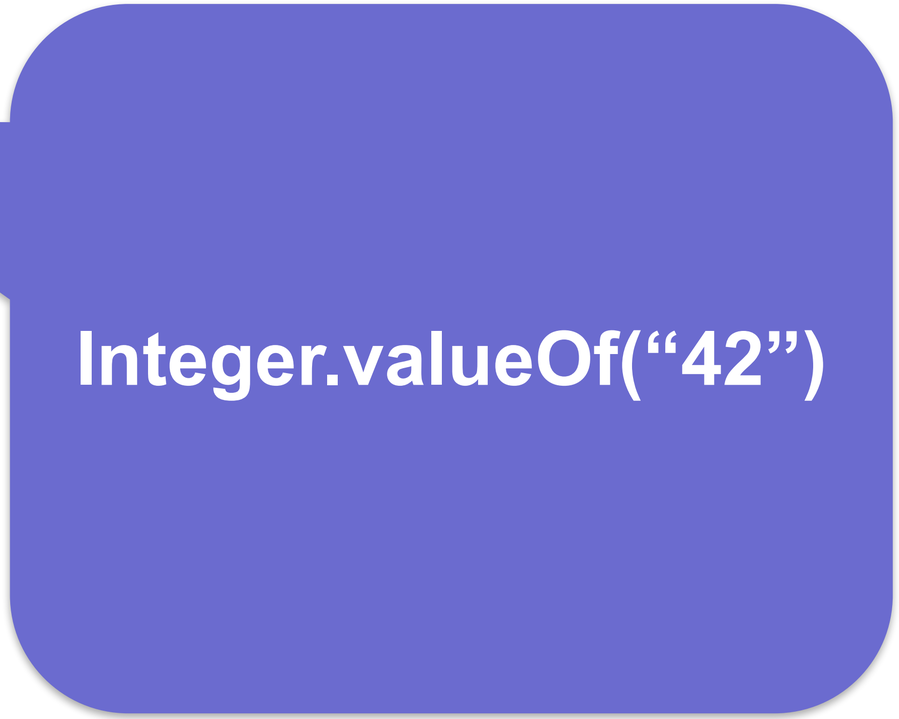
ВЫЗОВЫ МЕТОДОВ

invokeXXX

- invokestatic
- invokespecial
- invokevirtual
- invokeinterface
- invokedynamic

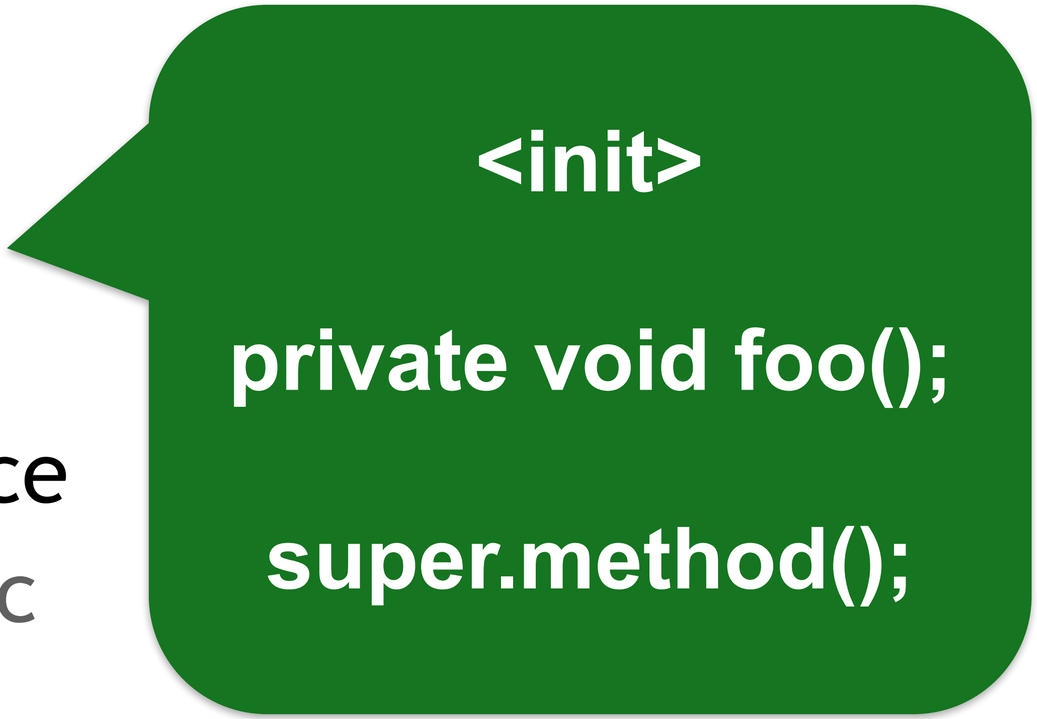
invokestatic

- invokestatic
- invokespecial
- invokevirtual
- invokeinterface
- invokedynamic



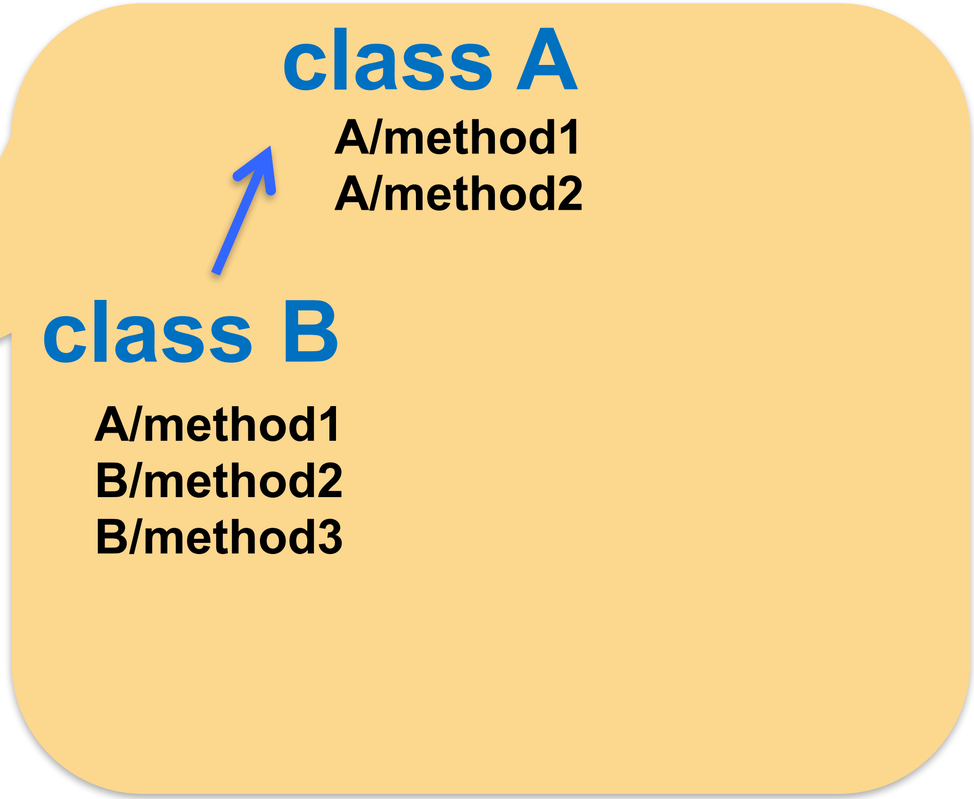
invokespecial

- invokestatic
- invokespecial
- invokevirtual
- invokeinterface
- invokedynamic



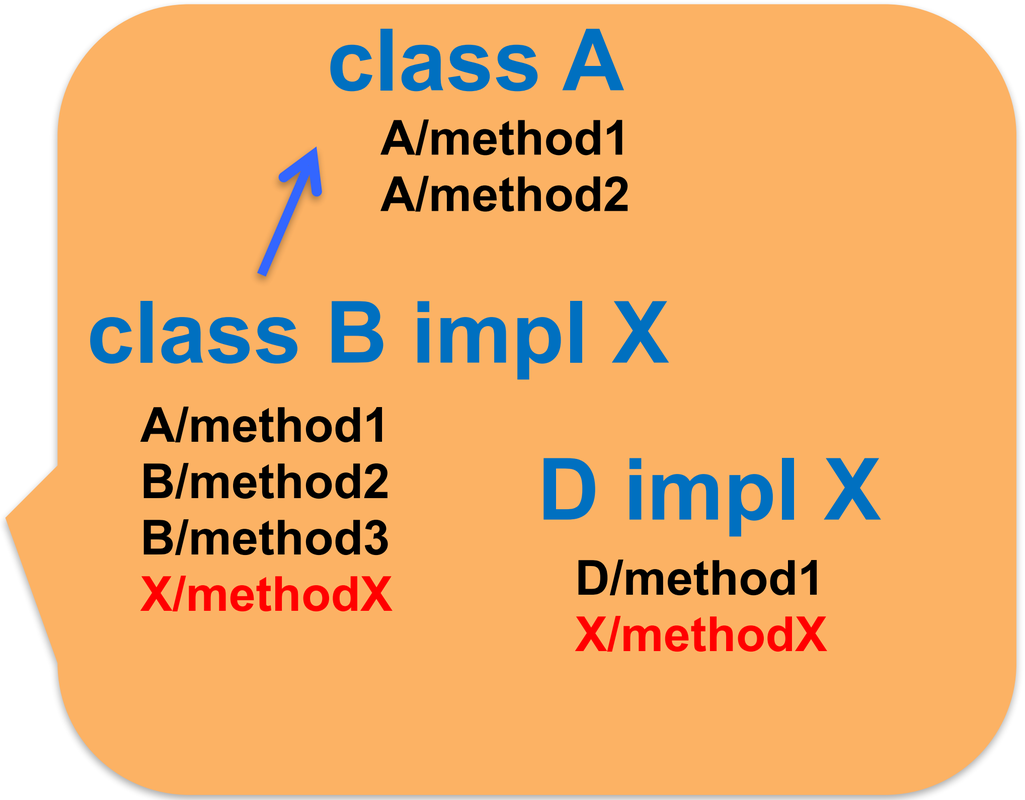
invokevirtual

- invokestatic
- invokespecial
- invokevirtual
- invokeinterface
- invokedynamic



invokeinterface

- invokestatic
- invokespecial
- invokevirtual
- invokeinterface
- invokedynamic



Efficient Implementation of Java Interfaces: *Invokeinterface* Considered Harmless, Bowen Alpern, Anthony Cocchi, Stephen Fink, David Grove, and Derek Lieber, OOPSLA'01

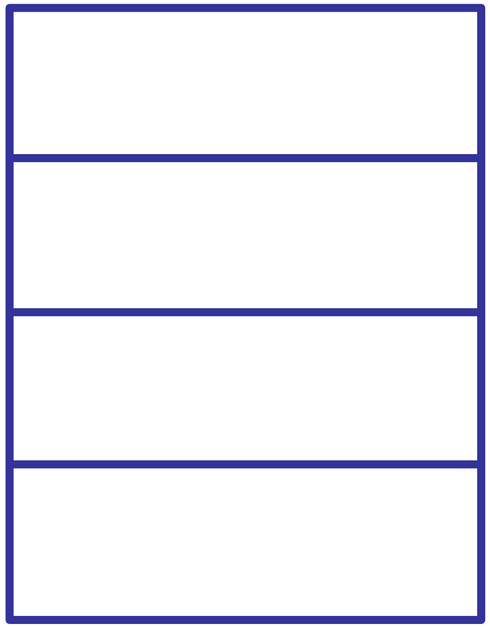
Вызов метода

Вызов метода

```
obj.method(param1, param2);
```

Вызов метода

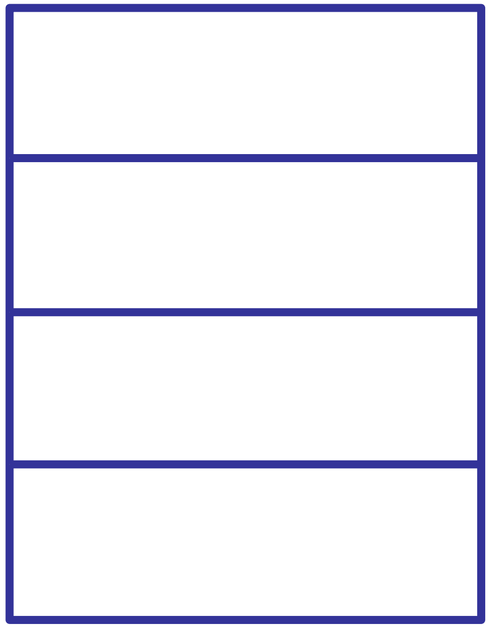
```
obj.method(param1, param2);
```



Вызов метода

```
obj.method(param1, param2);
```

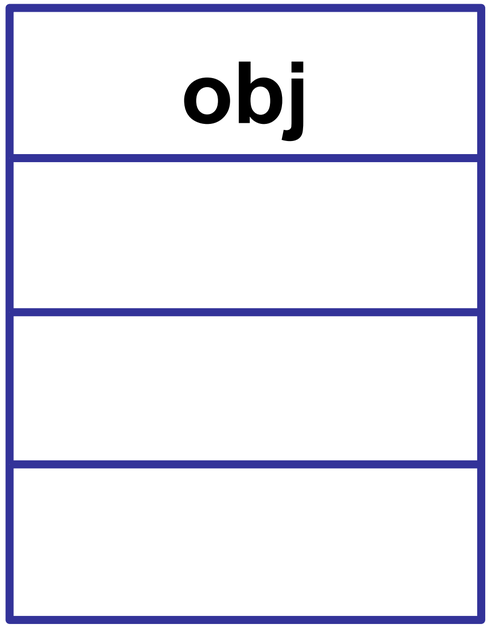
```
push obj
push param1
push param2
call method
```



Вызов метода

```
obj.method(param1, param2);
```

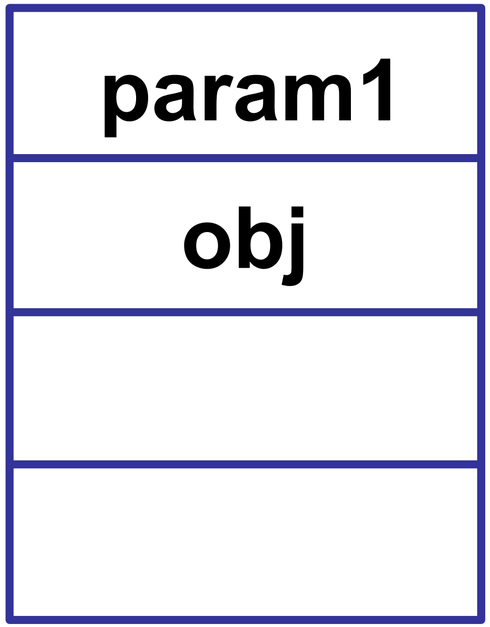
```
push obj
push param1
push param2
call method
```



Вызов метода

```
obj.method(param1, param2);
```

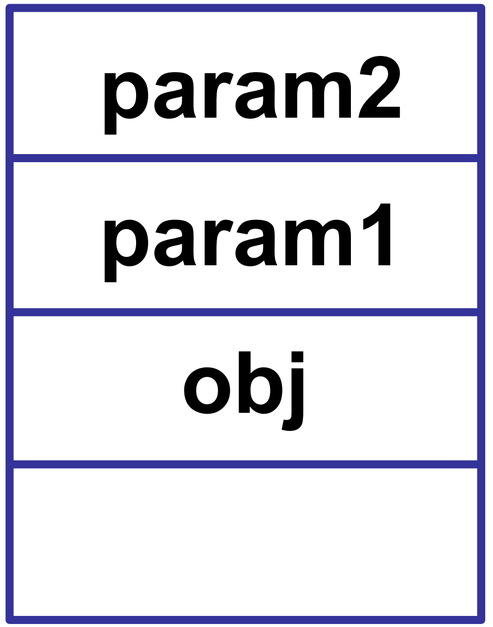
```
push obj
push param1
push param2
call method
```



Вызов метода

```
obj.method(param1, param2);
```

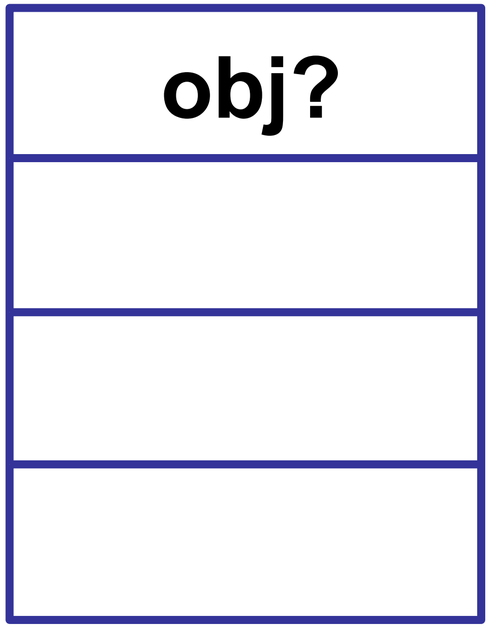
```
push obj
push param1
push param2
call method
```



Вызов метода

```
obj.method(param1, param2);
```

```
push obj
push param1
push param2
call method
```



Вызов метода

```
this.add(1, 2);
```

```
0:   aload_0
1:   iconst_1
2:   iconst_2
3:   invokevirtual   #2; //Method add:(II)I
```

INNER CLASSES

ВНУТЕРННИЕ КЛАССЫ

Внутренние классы

```
1   public class Car {
2
3       class Engine {
4           public void start() {
5               move();
6           }
7       }
8
9       private void move() {
10      }
11
12  }
```

Внутренние классы

```
class Car$Engine extends j.l.Object{
final Car this$0;

Car$Engine(Car);

public void start();
  Code:
    0:   aload_0
    1:   getfield        #1; //Field this$0:LCar;
    4:   invokestatic    #3; // Car.access$000:(LCar;)V
    7:   return

}
```

Внутренние классы

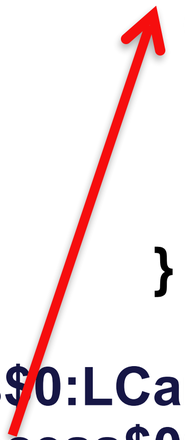
```
class Car$Engine extends j.l.Object{
final Car this$0;

Car$Engine(Car);

public void start();
  Code:
    0:   aload_0
    1:   getfield        #1; //Field this$0:LCar;
    4:   invokestatic    #3; // Car.access$000:(LCar;)V
    7:   return

}
```

```
public class Car extends j.l.Object{
public Car();
private void move();

static void access$000(Car);
  Code:
    0:   aload_0
    1:   invokespecial   #1; // move: ()V;
    4:   return
}
```

Внутренние классы

```
class Car$Engine extends j.l.Object{
final Car this$0;

Car$Engine(Car);

public void start();
  Code:
    0:   aload_0
    1:   getfield       #1; //Field this$0:LCar;
    4:   invokestatic   #3; // Car.access$000:(LCar;)V
    7:   return

}
```

```
public class Car extends j.l.Object{
public Car();
private void move();

static void access$000(Car);
  Code:
    0:   aload_0
    1:   invokespecial   #1; // move: ()V;
    4:   return
}
```



@antonarhipov

anton@zeroturnaround.com