**Faculty of Software Engineering and Computer Systems**

# Programming

## Lecture #1. Methods.
## Syntax constructions.

Instructor of faculty
Pismak Alexey Evgenievich
Kronverksky Pr. 49, 374 room

alexey.pismak@cs.ifmo.ru

Saint-Petersburg

# Command line arguments

```java
public class Main {

    public static void main(String[] args) {

        if(args.length > 0) {
            System.out.println ( args[0] );
        }

    }
}
```

```
1. javac Main.java  // run `javac` with arguments
2. java Main        // run `java` with arguments
3. java Main 3 2 4  // all arguments after class name will be
                    //           send to Java program
```

# Command line arguments

```
public class Main {

   public static void main(String[] args) {

      if(args.length > 0) {
          System.out.println ( args[0] );
      }

   }
}
```

```
java Main 3  2  4
```

# Classpath & Imports

```java
import static java.lang.Math.*;

/**
 *  Безысходники (game of words: sources + hopelessness)
 */
public class PracticMath {

  public static void main(String[] args) {

    double x = 5.1, y = 3.57;

    double res = sin(( x + 1) / 3*PI) * 8*cos(y);

  }

}
```

# Syntax constructions

# Conditional expressions

```
if ( condition ) expr
```

```
if ( condition ) expr else expr_2
```

```
if ( condition ) expr else if ( condition ) expr_x ...
```

```
final int LIMIT_TEMPERATURE = 25;

int t = 21;
boolean isSwitchedOff = false;

if(t > LIMIT) {
    …
}
```

# Dangerous!

```
boolean conditional = a * b > c;

if ( conditional ) { … }
```

```
if ( conditional == true ) { … }
if ( conditional != false ) { … }
```

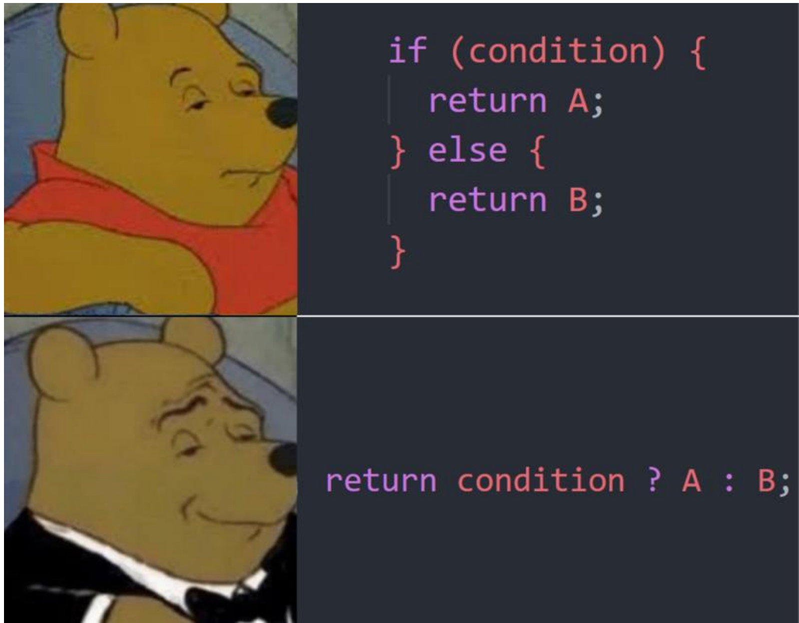
```
if ( String.valueOf(conditional ).equals(“true”) ) { … }
if ( conditional == true && conditional != false ) { … }
```

# Ternary operator

```
condition ? expression : expression;

// expression must return value
```

```
int delta = x > 0 ? x : Math.abs(x);
```

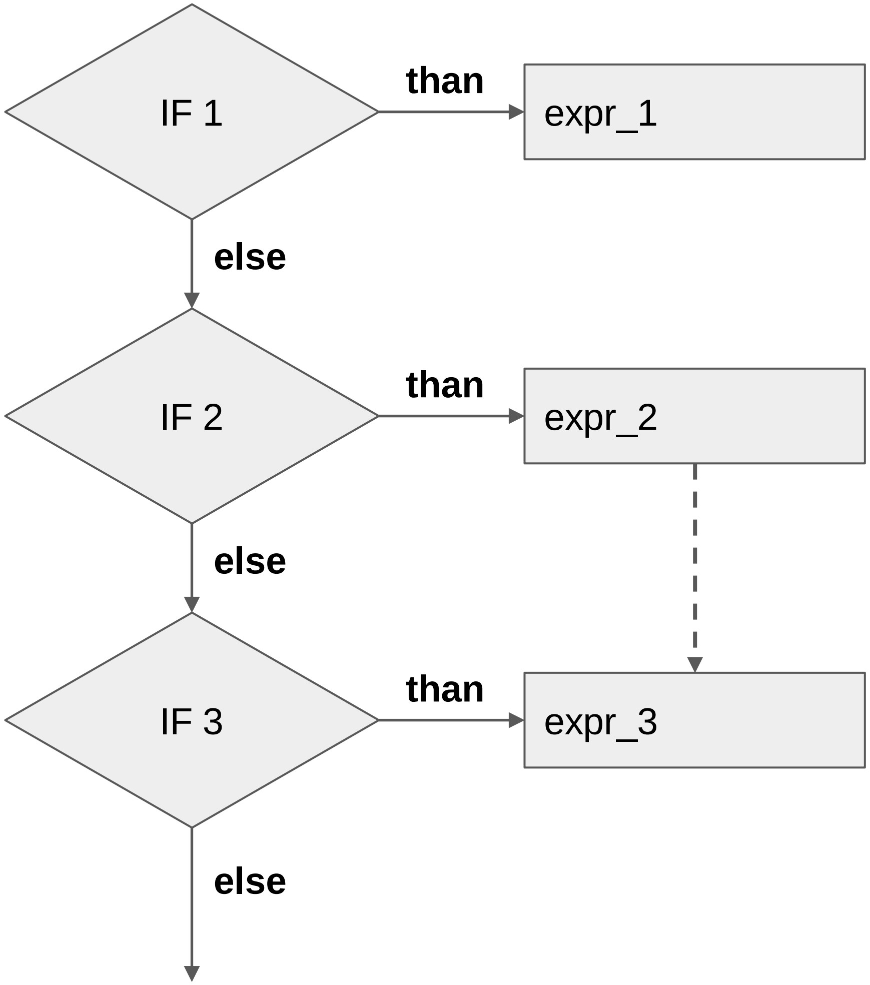
```
int delta;
if(x > 0) {
  delta = x;
} else {
  delta = Math.abs(x);
}
```



```
if (condition) {
  return A;
} else {
  return B;
}
```

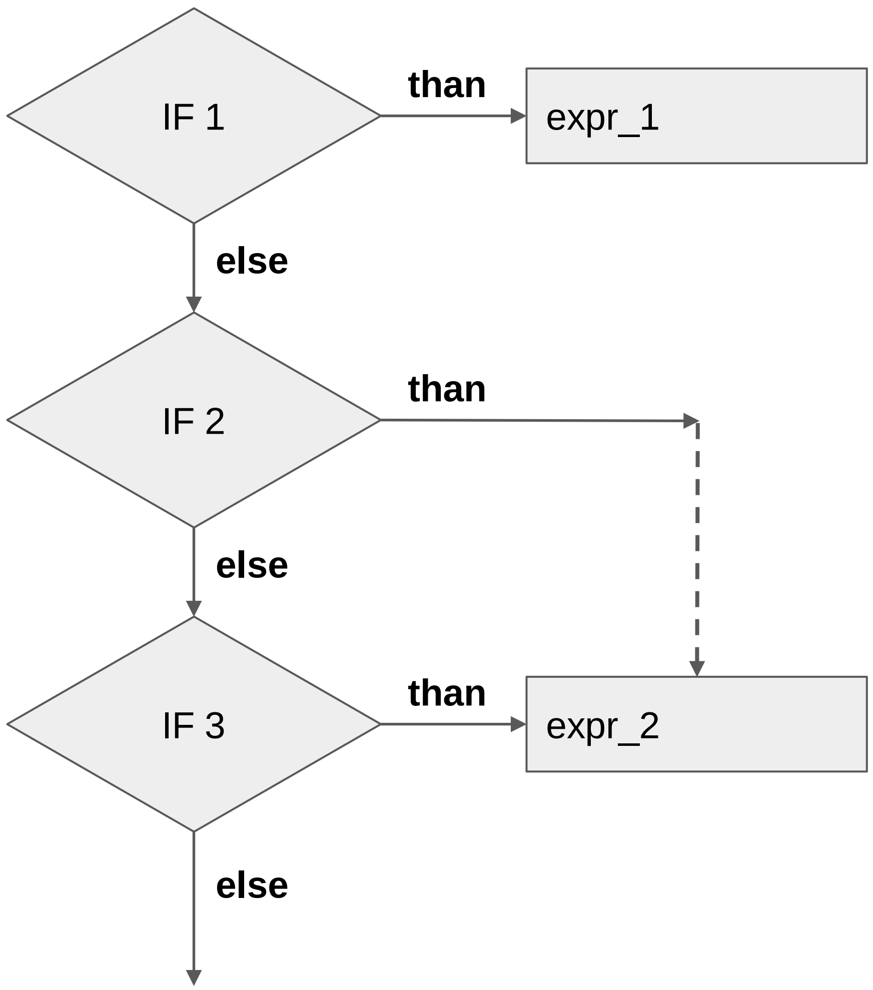
```
return condition ? A : B;
```

# Multivariate branching (classic)



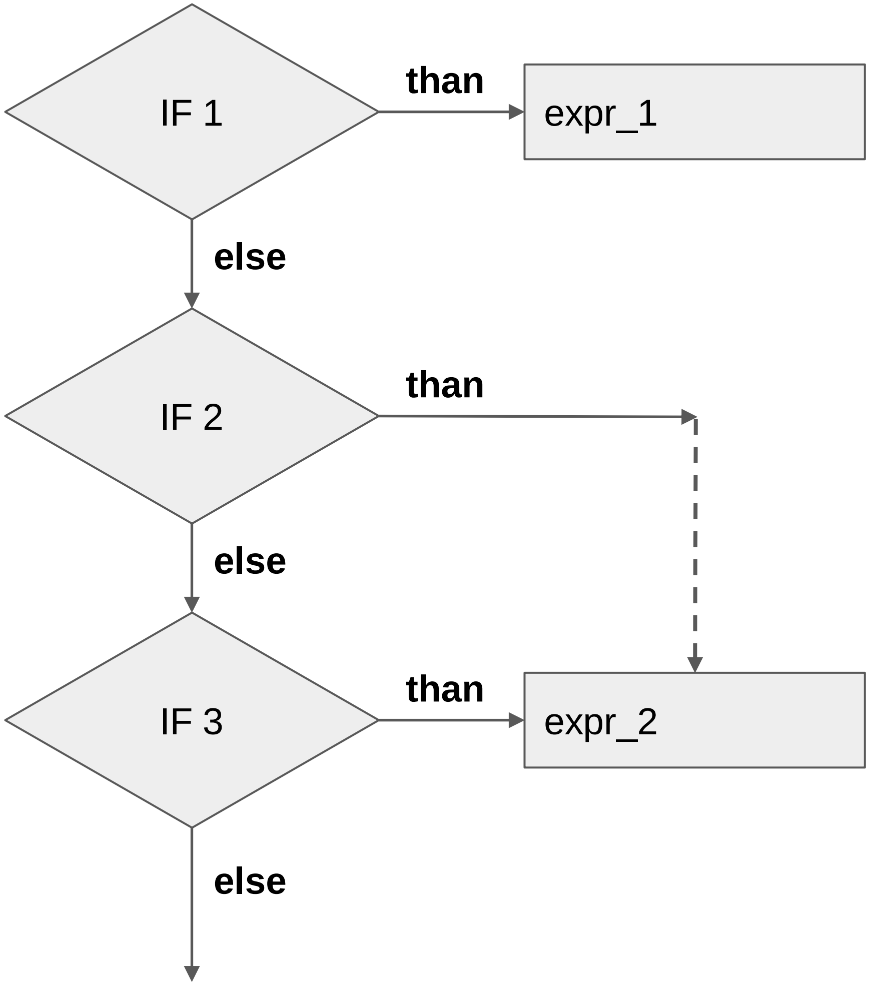
```
switch ( x ) {

case 1 : expr_1;
      break;

case 2 :
case 3 : expr_2;

default : expr_n;

}
```

# Multivariate branching (upgrade#1)



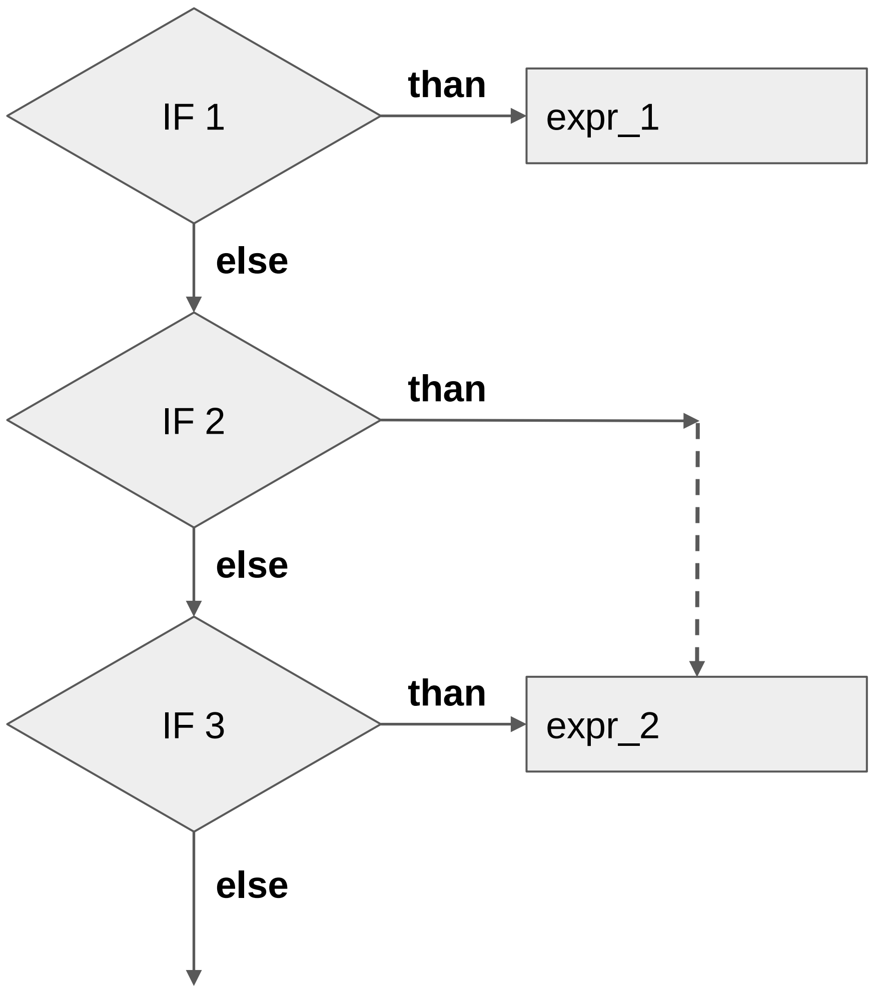
```
switch ( x ) {

case 1 : expr_1;
         break;

case 2,3 : expr_2;

default : expr_n;

}
```

# Multivariate branching (upgrade#2)



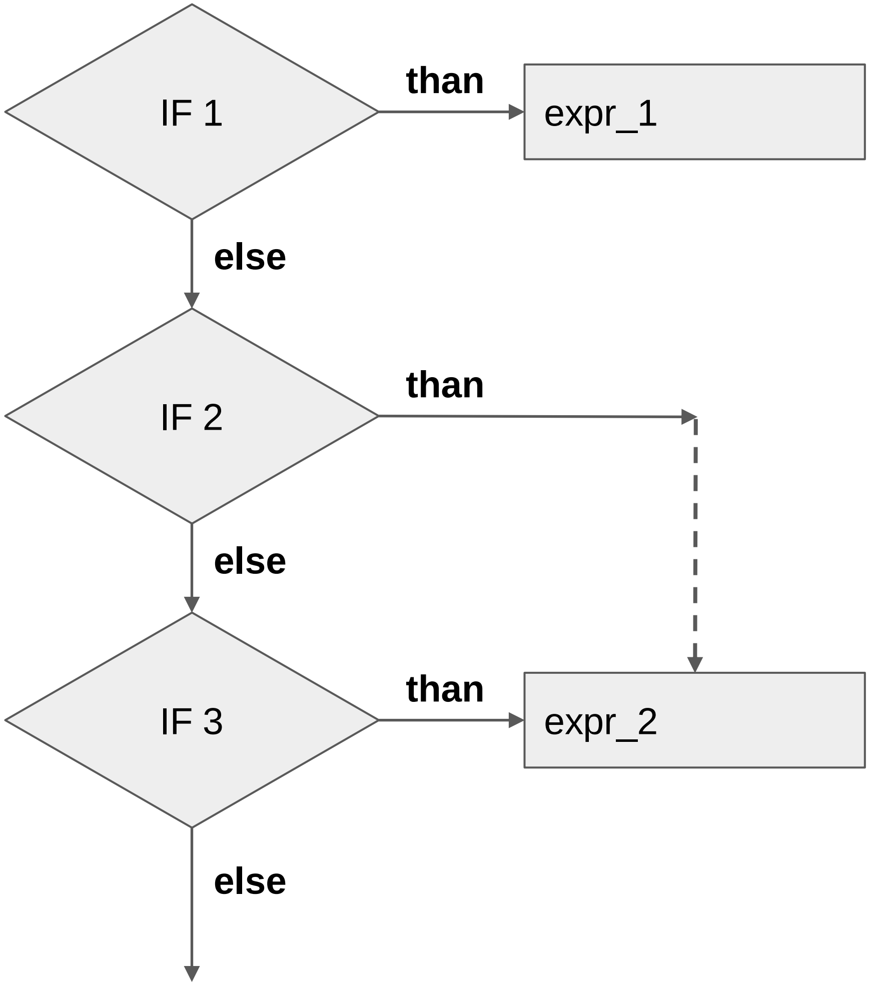
```
switch ( x ) {

case 1 -> expr_1;

case 2,3 -> expr_2;

default -> expr_n;

}
```

# Multivariate branching (upgrade#3)



```
var y = switch ( x ) {

case 1 -> 1;

case 2,3 -> 2;

default -> 10;

};
```

# Multivariate branching (upgrade#4)



```
var y = switch ( x ) {

    case 1 -> 1;

    case 2,3 -> {
        // calc result
        yield result;
    };

    default -> 10;

};
```

# Indefinite loops

```
while ( condition ) expression;


while ( true ) {
  // do something
}


while ( x > 0 ) {     // this may never executed
       // code here
}
```

# Indefinite loops

```
do expression while ( condition );
```

```
do {
  // do something
} while ( true );
```

```
do {     // executed at least once
       // code here
} while ( x > 0 );
```

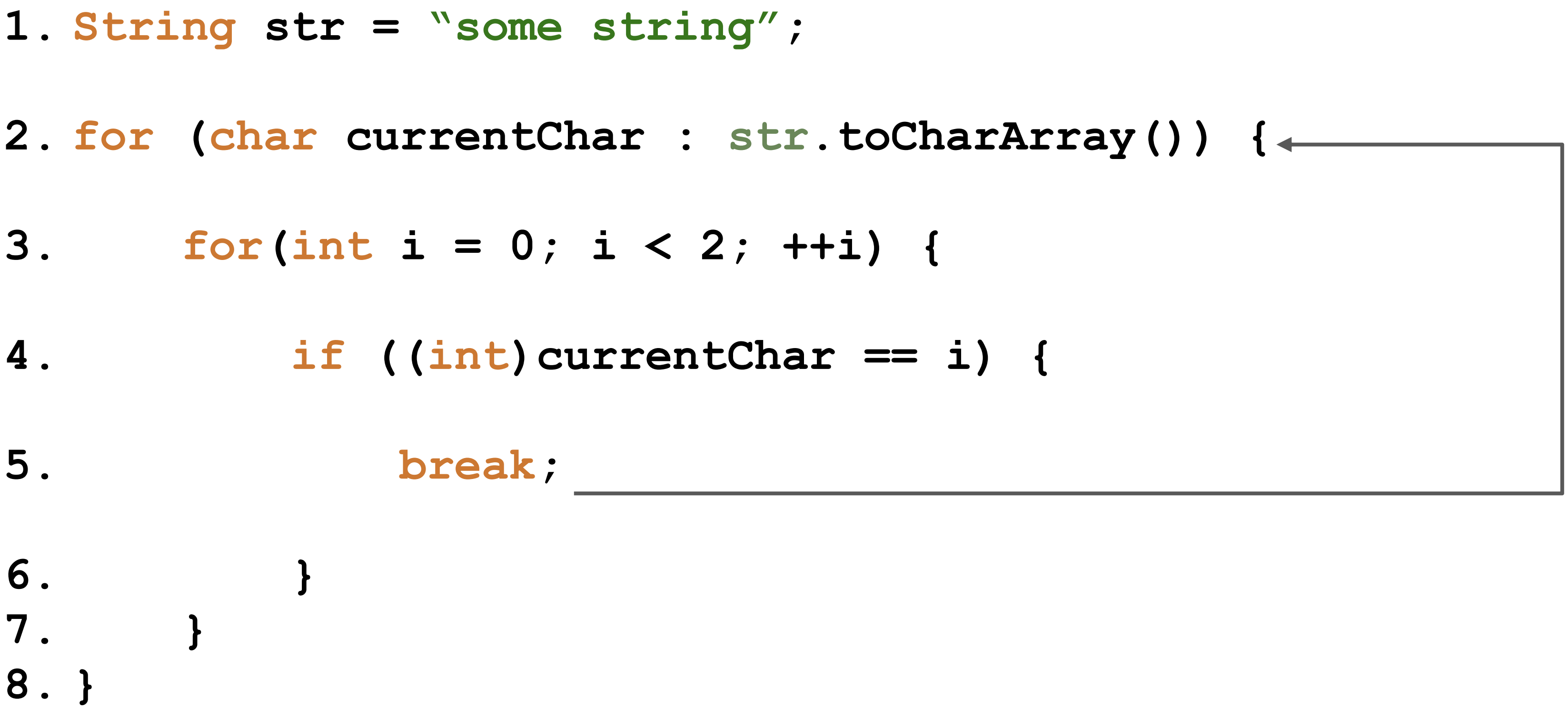
# Definite loop `for`

```
for ( init_block; condition; calc_block ) expression;


/*
 * Print numerals
 */
for (int i = 0; i < 10; ++i) {

    System.out.println( i );

}


// square table
for (int i = 0, j = 0; i < 10 && j < 10; ++i, ++j) {

    System.out.println(“%d * %d = %d”, i, j, i * j );

}
```

# Iterable loop `for`

```
for ( def_variable : set ) expression;


String str = “some string”;

for (char currentChar : str.toCharArray()) {
    if (currentChar != ‘s’) {
        System.out.print(currentChar);      //  ome tring
    }
}
```

# Interrupt execution



```java
String str = “some string”;
for (char currentChar : str.toCharArray()) {
    for(int i = 0; i < 2; ++i) {
        if ((int)currentChar == i) {
            break;
        }
    }
}
```

# Interrupt execution

```
String str = “some string”;
for (char currentChar : str.toCharArray()) {
    for(int i = 0; i < 2; ++i) {
        if ((int)currentChar == i) {
            continue;
        }
        // ...
    }
}
```

# Interrupt all loops

```
String str = “some string”;
full:
for (char c : str.toCharArray()) {
    for(int i = 0; i < 2; ++i) {
        if ((int)c == i) {
            break full;
        }
        // ...
    }
}
```

# Blocks and scope

```
{
    // expressions, operators etc.
}
```

```
                                       if ( condition ) {
if ( condition ) expression   →            // code
                                       }
```

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

# “Subprograms” (methods)

```
public static void printMessage (String msg) {
    System.out.println (msg);
}
public static void main(String[] args) {
    printMessage (“I am liquid”);
}
```

# Methods

```
public static int cube (int arg) {
    return arg * arg * arg;
}

public static void main(String[] args) {
    printMessage (“5^3 = ” + cube(5));
}

public static void printMessage (String msg) {
    System.out.println (msg);
}
```

# How to write a method with arguments like this System.out.printf ?

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

    System.out.printf(“5^3 = %d”, cube(5));

    System.out.printf(“%d = %d”, cube(5), 5*5*5);

}
```

String + one argument

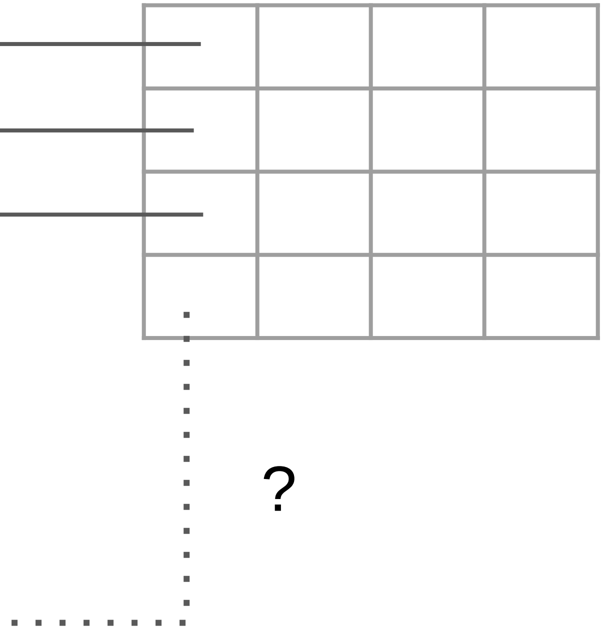
String + two arguments

# Variable arguments (VARARGS)

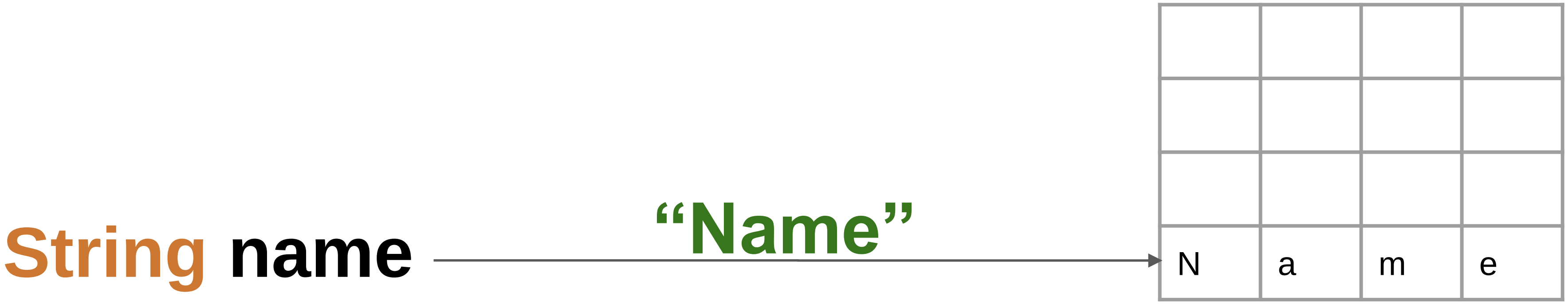
```
public static void main(String[] args) {

    printMessage (“This ”, “Bob”);
    printMessage (“Я”, “угадаю”, “как”, “тебя”, “зовут”);
}

public static void printMessage (String ... msg) {
    for(String s : msg) {
        System.out.println (s);
    }
}
```

# Ref- and valuable data types

```
int x = 5;

double arg = 1.544;

char c = ‘A’;

String name = “Name”;
```



# Ref- and valuable data types



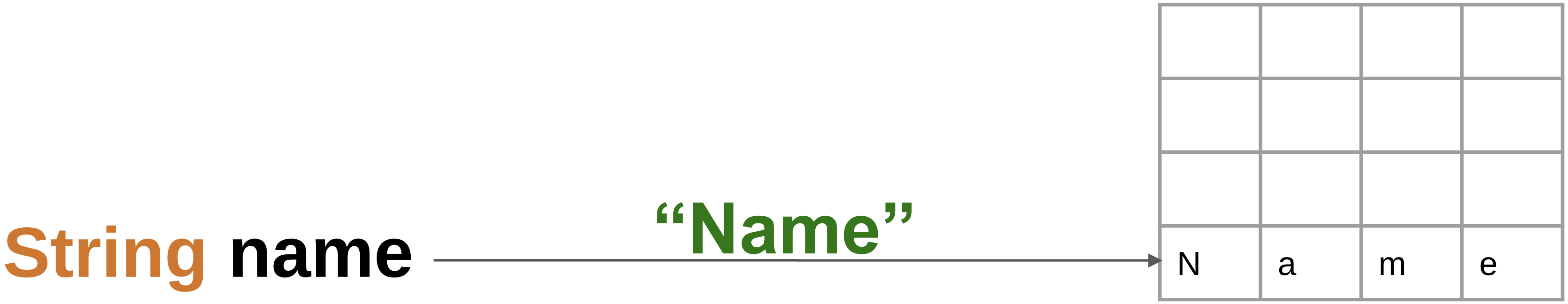
String name

“Name”

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| N | a | m | e |

```
String name1 = new String(“Name”);

String name2 = new String(“Name”);

System.out.println ( name1 == name2 );
```

# Ref- and valuable data types



String name

“Name”

N a m e

```
String name1 = new String(“Name”);
String name2 = new String(“Name”);

String name3 = “Name”;
String name4 = “Name”;
```

# Operator `new`

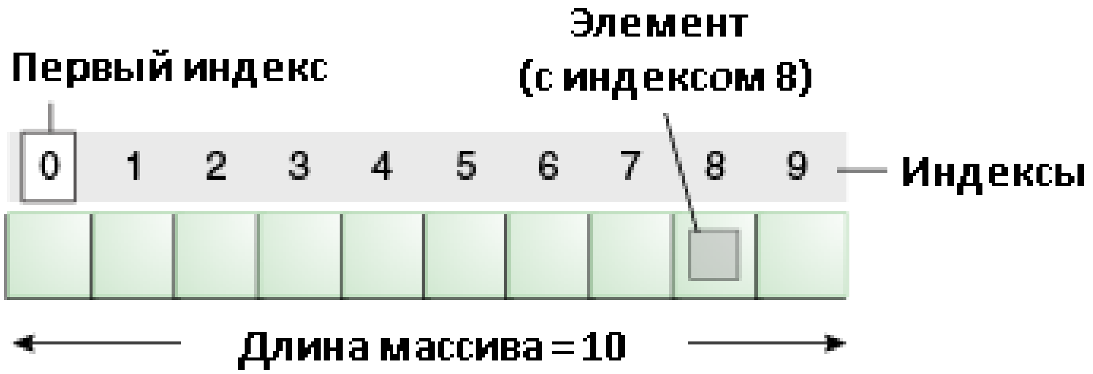
```
int[] y = new int[2];

String str = new String(“I am liquid”);
```

How reset reference variable to uninitialized value?

# **Value** ‘null’

```
int[] y = null;

String str = null;

y.length     // after this operations
str.trim()   // will errors
```

# Arrays



```
int[] a;

int b[];

int[] x = {5, 2};

int[] y = new int[2];
```

# Arrays

```
int[] x = {5, 2};
int count = x.length;  // property


java.util.Arrays         // work with array
        sort
        search
        copy
```

# Arrays

**Sort:**

**Arrays.sort ( … )**

**Arrays.parallelSort ( … )**

**Search:**

**Arrays.binarySearch ( … )**

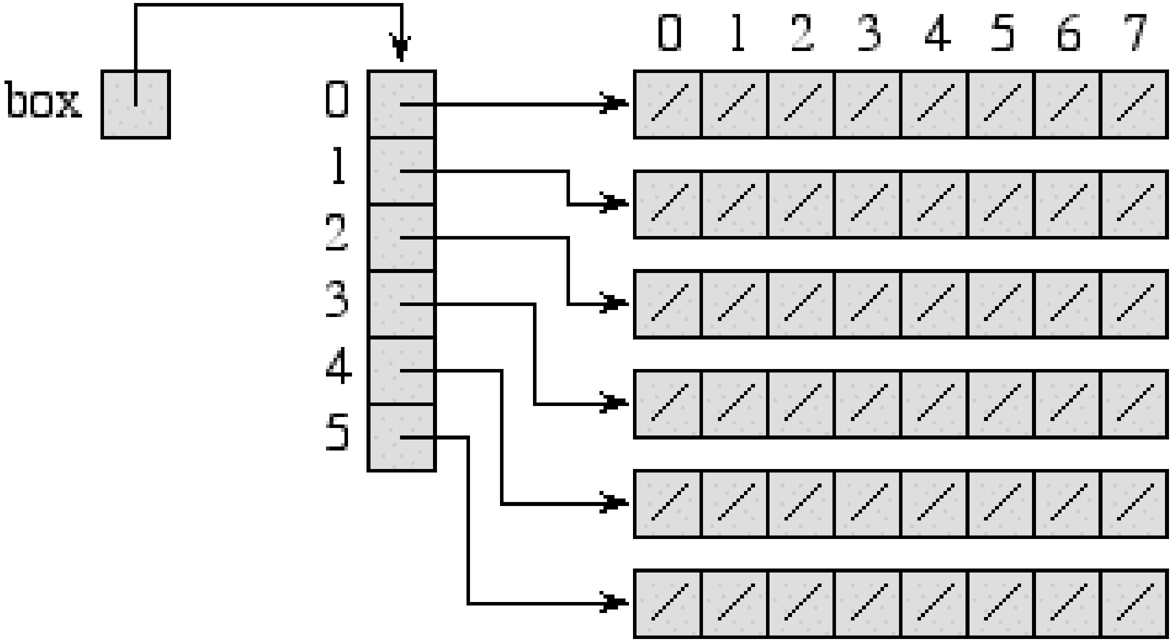
**Copy:**

**System.arrayCopy ( … )**

**Arrays.copyOfRange ( … )**

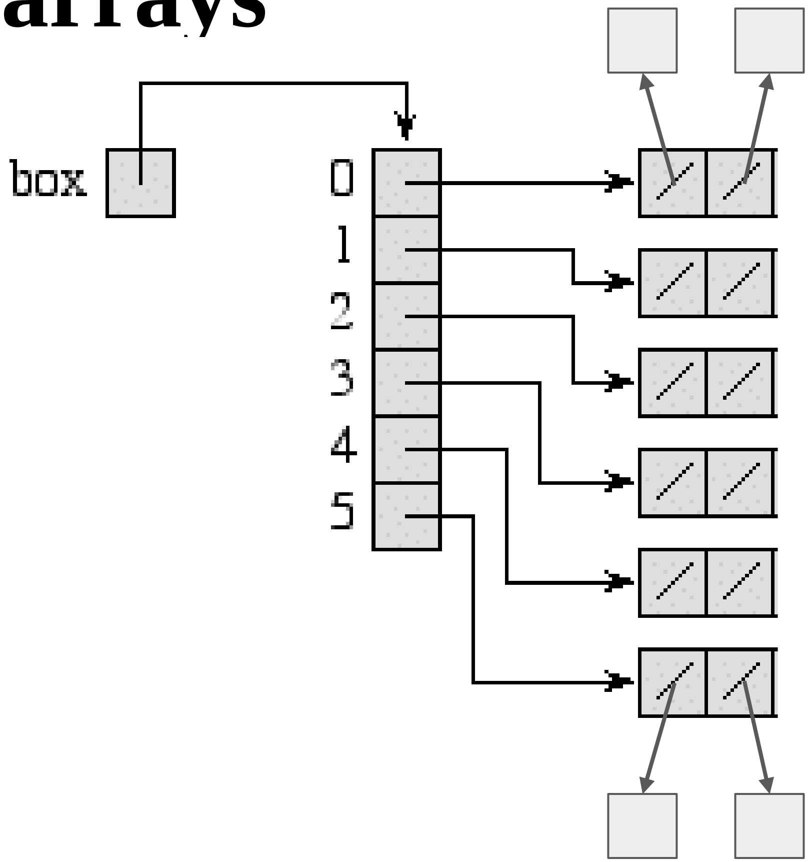
**filling, applying specific math expression, set default values etc.**

# Arrays



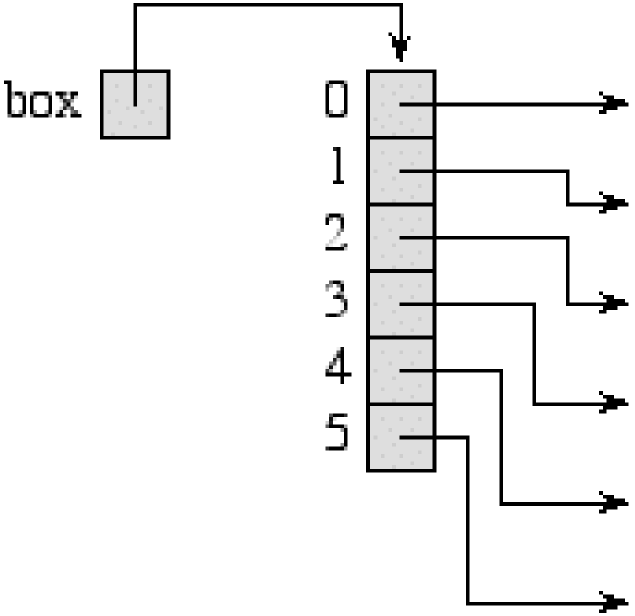
```
int[][] matrix;

int[][] box = new int[6][8];

int[][] box = { {1, 2, 3},  {4, 5, 6} };
```

# Crazy arrays



```
int[][][] box = new int[6][2][1];
```

# Arrays



```
int[][] box;

int[] box[] = new int[6][8];

int[][] box = new int[6][];

int box[][] = { {1, 2, 3},  {4, 5, 6} };
```