
JAVA PUZZLERS NG SO3
ОТКУДА ВЫ ВСЕ ЛЕЗЕТЕ-ТО?!


CLICK AND HACK
THE TYPE-IT BROTHERS

55,746 REPUTATION

9 107 193

Tagir Valeev

top 0.27% this year

Author of StreamEx library.

IntelliJ IDEA Developer.

FindBugs project contributor.

Tech blog in Russian on Habrahabr.

1,264 answers **12** questions **~4.1m** people reached

Novosibirsk, Russia

amaembo

Member for 2 years, 5 months

4,658 profile views

Last seen yesterday

Communities (3)

Stack Overflow	55.7k
Stack Overflow на р…	4.8k
Code Review	111

Top Meta Posts

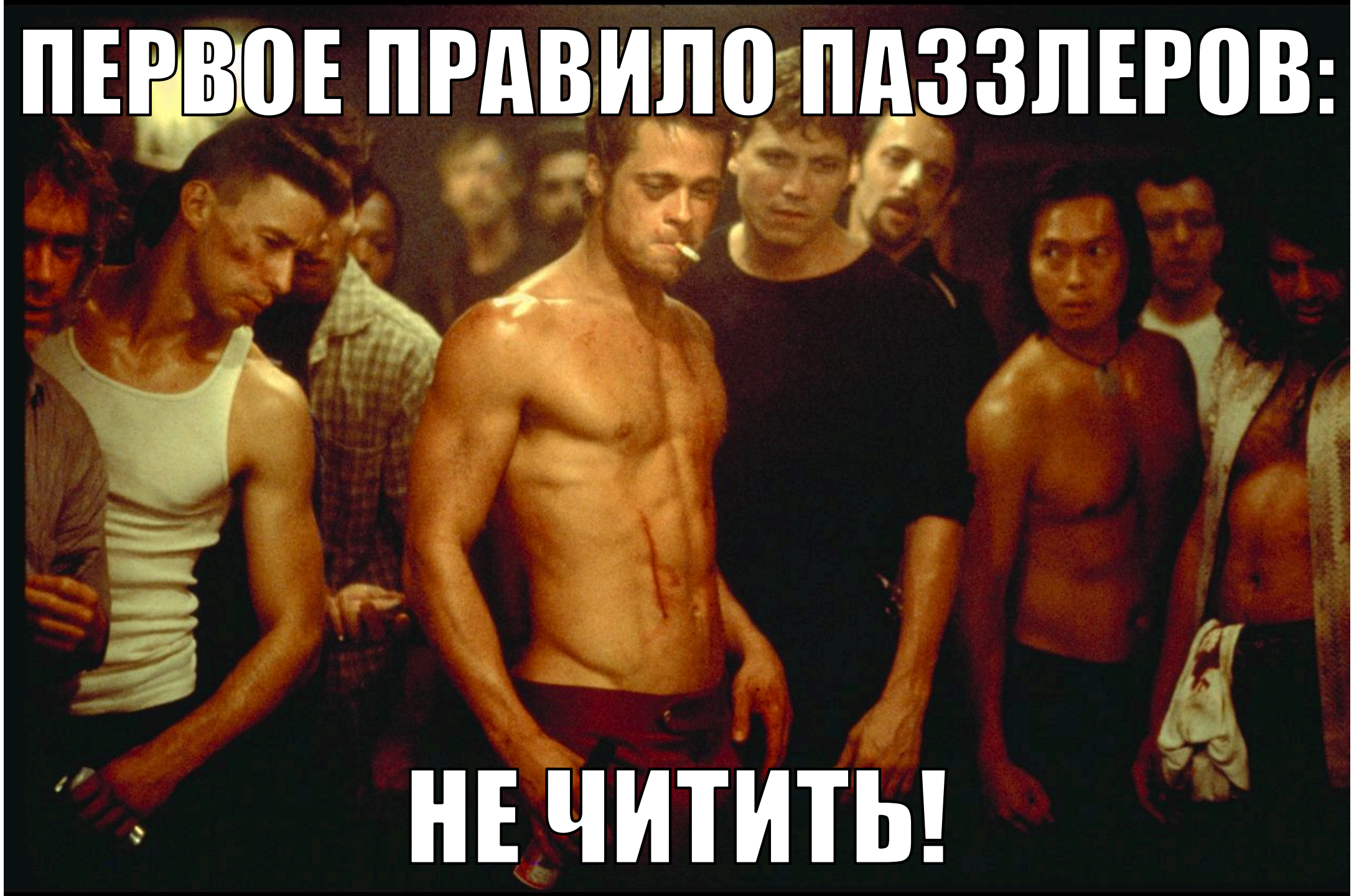
3 1

Top Tags (582)

java	SCORE 4,883	POSTS 1,166	POSTS % 91
java-8	SCORE 3,154	POSTS 560	
java-stream	SCORE 2,122	POSTS 426	
lambda	SCORE 658	POSTS 142	
generics	SCORE 246	POSTS 69	
collections	SCORE 235	POSTS 52	

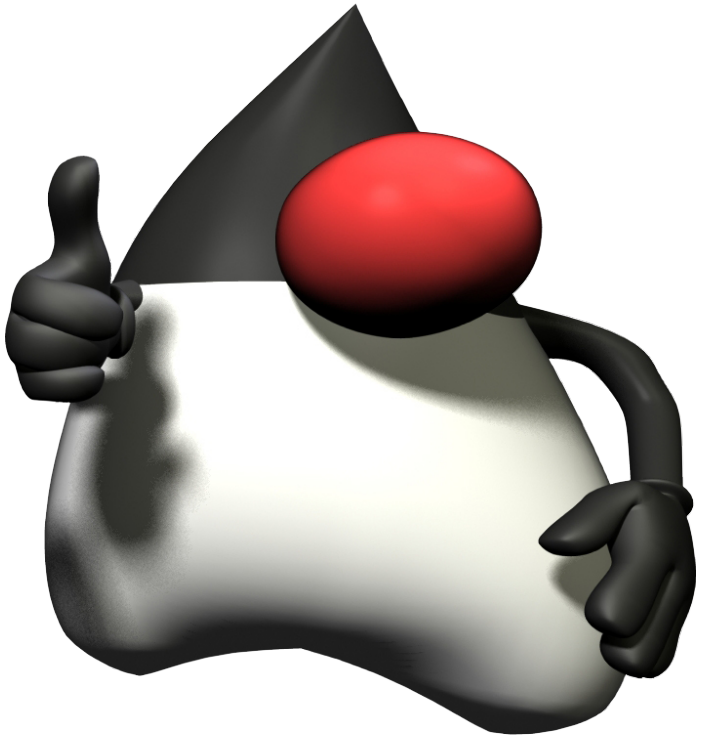
Baruch Sadogursky, JFrog Developer Advocate, @jbaruch

1. ДВА КЛЕВЫХ ПАРНЯ НА СЦЕНЕ
2. СМЕШНЫЕ ГОЛОВОЛОМКИ
3. ВЫ ГОЛОСУЕТЕ ЗА ОТВЕТЫ!
4. МЫ КИДАЕМСЯ ФУТБОЛКАМИ ОТ JFROG (СПАСИБО ИЛЛ!)
5. ОФИЦИАЛЬНЫЕ ХЭШТЭГИ: #JAVAPUZZLERSNG #JOKERCONF


ПЕРВОЕ ПРАВИЛО ПАЗЗЛЕРОВ:
НЕ ЧИТИТЬ!

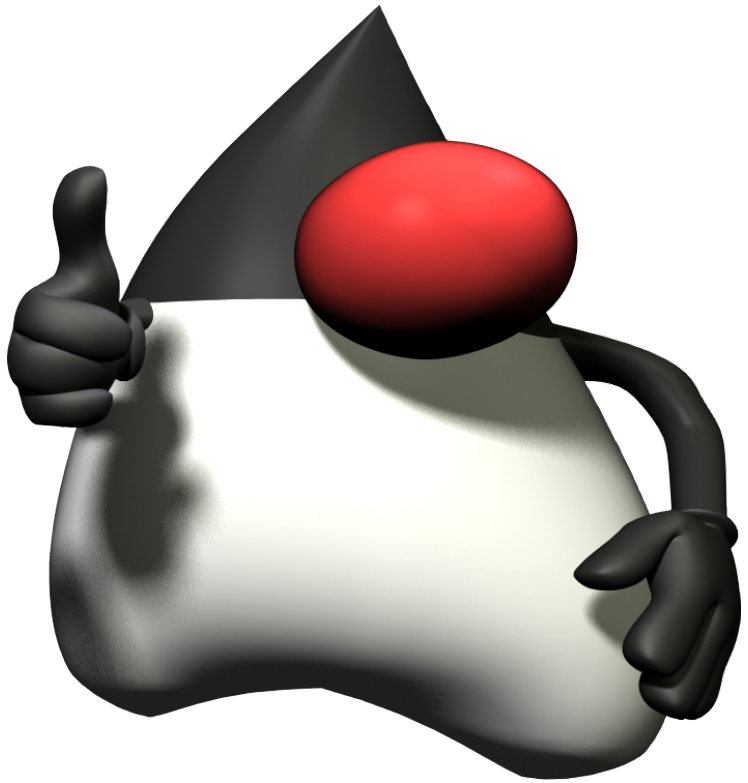
Проверка системы голосования! Какая у вас Java?

A. Java 7
B. Java 8
C. Java 9
D. Java 6
E. Java 5
F. Java 2



Готовьтесь, сейчас будет вот так:

Всё работает (ну, или не работает) на последней версии Java 8 и 9 соответственно.




ТРЕТЬИМ БУДЕШЬ?

"мать...мать..." привычно ответило эхо

```
// module-info.java

module module {
    requires requires;
    exports exports;
    opens opens;
    uses uses;
}
```

1. Ошибка в каждой строчке
2. Нельзя экспортировать пакет с именем exports
3. Нельзя объявлять в uses имя uses
4. Компилируется без ошибок

```
// module-info.java

module module {
    requires requires;
    exports exports;
    opens opens;
    uses uses;
}
```

1. Ошибка в каждой строчке
2. Нельзя экспортировать пакет с именем exports
3. Нельзя объявлять в uses имя uses
4. Компилируется без ошибок

```
// module-info.java

module module {
    requires requires;
    exports exports;
    opens opens;
    uses uses;
}
```

1. Ошибка в каждой строчке
2. Нельзя экспортировать пакет с именем exports
3. Нельзя объявля uses имя uses
4. Компилируется без ошибок

Как мы это чиним?

```
// module-info.java

import opens.uses;
module module {
    requires requires;
    exports exports;
    opens opens;
    uses uses;
}
```


42

```
System.out.println(isUltimateQuestion ? 42 : null);
```

7 МИЛЛИОНОВ ЛЕТ СПУСТЯ...

```
System.out.println(isUltimateQuestion ? 42 :
                   isUltimateQuestion ? 42 : null);
```

```
System.out.println(isUltimateQuestion ? 42 :
    isUltimateQuestion ? 42 : null);
```

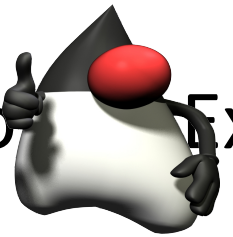
```
isUltimateQuestion = false
```

1. null
2. Не скомпилируется
3. 42
4. NullPointerException

```
System.out.println(isUltimateQuestion ? 42 :
    isUltimateQuestion ? 42 : null);
```

```
isUltimateQuestion = false
```

1. null
2. Не скомпилируется
3. 42
4. NullPo[illegible]Exception



Какой тип у `(b ? 42 : null)`?

Смотрим jls-15.25:

The following tables summarize the rules above by giving the type of a conditional expression for all possible types of its second and third operands. `bnp(..)` means to apply binary numeric promotion. The form "T | bnp(..)" is used where one operand is a constant expression of type `int` and may be representable in type T, where binary numeric promotion is used if the operand is not representable in type T. The operand type `Object` means any reference type other than the `null` type and the eight wrapper classes `Boolean`, `Byte`, `Short`, `Character`, `Integer`, `Long`, `Float`, `Double`.

Table 15.25-A. Conditional expression type (Primitive 3rd operand, Part I)

3rd →	byte	short	char	int
2nd ↓				
byte	byte	short	bnp(byte,char)	byte | bnp(byte,int)
Byte	byte	short	bnp(Byte,char)	byte | bnp(Byte,int)
short	short	short	bnp(short,char)	short | bnp(short,int)
Short	short	short	bnp(Short,char)	short | bnp(Short,int)
char	bnp(char,byte)	bnp(char,short)	char	char | bnp(char,int)
Character	bnp(Character,byte)	bnp(Character,short)	char	char | bnp(Character,int)
int	byte | bnp(int,byte)	short | bnp(int,short)	char | bnp(int,char)	int
Integer	bnp(Integer,byte)	bnp(Integer,short)	bnp(Integer,char)	int
long	bnp(long,byte)	bnp(long,short)	bnp(long,char)	bnp(long,int)
Long	bnp(Long,byte)	bnp(Long,short)	bnp(Long,char)	bnp(Long,int)
float	bnp(float,byte)	bnp(float,short)	bnp(float,char)	bnp(float,int)
Float	bnp(Float,byte)	bnp(Float,short)	bnp(Float,char)	bnp(Float,int)
double	bnp(double,byte)	bnp(double,short)	bnp(double,char)	bnp(double,int)
Double	bnp(Double,byte)	bnp(Double,short)	bnp(Double,char)	bnp(Double,int)
boolean	lub(Boolean,Byte)	lub(Boolean,Short)	lub(Boolean,Character)	lub(Boolean,Integer)
Boolean	lub(Boolean,Byte)	lub(Boolean,Short)	lub(Boolean,Character)	lub(Boolean,Integer)
null	lub(null,Byte)	lub(null,Short)	lub(null,Character)	lub(null,Integer)
Object	lub(Object,Byte)	lub(Object,Short)	lub(Object,Character)	lub(Object,Integer)

Table 15.25-B. Conditional expression type (Primitive 3rd operand, Part II)

3rd →	long	float	double	boolean
2nd ↓				
byte	bnp(byte,long)	bnp(byte,float)	bnp(byte,double)	lub(byte,Boolean)
Byte	bnp(byte,long)	bnp(byte,float)	bnp(byte,double)	lub(byte,Boolean)
short	bnp(short,long)	bnp(short,float)	bnp(short,double)	lub(Short,Boolean)
Short	bnp(Short,long)	bnp(Short,float)	bnp(Short,double)	lub(Short,Boolean)
char	bnp(char,long)	bnp(char,float)	bnp(char,double)	lub(Character,Boolean)
Character	bnp(Character,long)	bnp(Character,float)	bnp(Character,double)	lub(Character,Boolean)
int	bnp(int,long)	bnp(int,float)	bnp(int,double)	lub(Integer,Boolean)
Integer	bnp(Integer,long)	bnp(Integer,float)	bnp(Integer,double)	lub(Integer,Boolean)
long	long	bnp(long,float)	bnp(long,double)	lub(Long,Boolean)
Long	long	bnp(Long,float)	bnp(Long,double)	lub(Long,Boolean)
float	bnp(float,long)	float	bnp(float,double)	lub(Float,Boolean)
Float	bnp(Float,long)	float	bnp(Float,double)	lub(Float,Boolean)
double	bnp(double,long)	bnp(double,float)	double	lub(Double,Boolean)
Double	bnp(Double,long)	bnp(Double,float)	double	lub(Double,Boolean)
boolean	lub(Boolean,Long)	lub(Boolean,Float)	lub(Boolean,Double)	boolean
Boolean	lub(Boolean,Long)	lub(Boolean,Float)	lub(Boolean,Double)	boolean
null	lub(null,Long)	lub(null,Float)	lub(null,Double)	lub(null,Boolean)
Object	lub(Object,Long)	lub(Object,Float)	lub(Object,Double)	lub(Object,Boolean)

Table 15.25-C. Conditional expression type (Reference 3rd operand, Part I)

3rd →	Byte	Short	Character	Integer
2nd ↓				
byte	byte	short	bnp(byte,Character)	bnp(byte,Integer)
Byte	Byte	short	bnp(Byte,Character)	bnp(Byte,Integer)
short	short	short	bnp(short,Character)	bnp(short,Integer)
Short	short	short	bnp(Short,Character)	bnp(Short,Integer)
char	bnp(char,Byte)	bnp(char,Short)	char	bnp(char,Integer)
Character	bnp(Character,Byte)	bnp(Character,Short)	Character	bnp(Character,Integer)
int	byte | bnp(int,Byte)	short | bnp(int,Short)	char | bnp(int,Character)	int
Integer	bnp(Integer,Byte)	bnp(Integer,Short)	bnp(Integer,Character)	Integer
long	bnp(long,Byte)	bnp(long,Short)	bnp(long,Character)	bnp(long,Integer)
Long	bnp(Long,Byte)	bnp(Long,Short)	bnp(Long,Character)	bnp(Long,Integer)
float	bnp(float,Byte)	bnp(float,Short)	bnp(float,Character)	bnp(float,Integer)
Float	bnp(Float,Byte)	bnp(Float,Short)	bnp(Float,Character)	bnp(Float,Integer)
double	bnp(double,Byte)	bnp(double,Short)	bnp(double,Character)	bnp(double,Integer)
Double	bnp(Double,Byte)	bnp(Double,Short)	bnp(Double,Character)	bnp(Double,Integer)
boolean	lub(Boolean,Byte)	lub(Boolean,Short)	lub(Boolean,Character)	lub(Boolean,Integer)
Boolean	lub(Boolean,Byte)	lub(Boolean,Short)	lub(Boolean,Character)	lub(Boolean,Integer)
null	Byte	Short	Character	Integer
Object	lub(Object,Byte)	lub(Object,Short)	lub(Object,Character)	lub(Object,Integer)

Table 15.25-D. Conditional expression type (Reference 3rd operand, Part II)

3rd →	Long	Float	Double	Boolean
2nd ↓				
byte	bnp(byte,Long)	bnp(byte,Float)	bnp(byte,Double)	lub(byte,Boolean)
Byte	bnp(Byte,Long)	bnp(Byte,Float)	bnp(Byte,Double)	lub(byte,Boolean)
short	bnp(short,Long)	bnp(short,Float)	bnp(short,Double)	lub(Short,Boolean)
Short	bnp(Short,Long)	bnp(Short,Float)	bnp(Short,Double)	lub(Short,Boolean)
char	bnp(char,Long)	bnp(char,Float)	bnp(char,Double)	lub(Character,Boolean)
Character	bnp(Character,Long)	bnp(Character,Float)	bnp(Character,Double)	lub(Character,Boolean)
int	bnp(int,Long)	bnp(int,Float)	bnp(int,Double)	lub(Integer,Boolean)
Integer	bnp(Integer,Long)	bnp(Integer,Float)	bnp(Integer,Double)	lub(Integer,Boolean)
long	Long	bnp(Long,Float)	bnp(Long,Double)	lub(Long,Boolean)
Long	Long	bnp(Long,Float)	bnp(Long,Double)	lub(Long,Boolean)
float	bnp(float,Long)	float	bnp(float,Double)	lub(Float,Boolean)
Float	bnp(Float,Long)	Float	bnp(Float,Double)	lub(Float,Boolean)
double	bnp(double,Long)	bnp(double,Float)	double	lub(Double,Boolean)
Double	bnp(Double,Long)	bnp(Double,Float)	Double	lub(Double,Boolean)
boolean	lub(Boolean,Long)	lub(Boolean,Float)	lub(Boolean,Double)	boolean
Boolean	lub(Boolean,Long)	lub(Boolean,Float)	lub(Boolean,Double)	Boolean
null	Long	Float	Double	Boolean
Object	lub(Object,Long)	lub(Object,Float)	lub(Object,Double)	lub(Object,Boolean)

Table 15.25-E. Conditional expression type (Reference 3rd operand, Part III)

3rd →	null	Object
2nd ↓		
byte	lub(Byte,null)	lub(Byte,Object)
Byte	Byte	lub(Byte,Object)
short	lub(Short,null)	lub(Short,Object)
Short	Short	lub(Short,Object)
char	lub(Character,null)	lub(Character,Object)
Character	Character	lub(Character,Object)
int	lub(Integer,null)	lub(Integer,Object)
Integer	Integer	lub(Integer,Object)
long	lub(Long,null)	lub(Long,Object)
Long	Long	lub(Long,Object)
float	lub(Float,null)	lub(Float,Object)
Float	Float	lub(Float,Object)
double	lub(Double,null)	lub(Double,Object)
Double	Double	lub(Double,Object)
boolean	lub(Boolean,null)	lub(Boolean,Object)
Boolean	Boolean	lub(Boolean,Object)
null	null	lub(null,Object)
Object	Object	Object

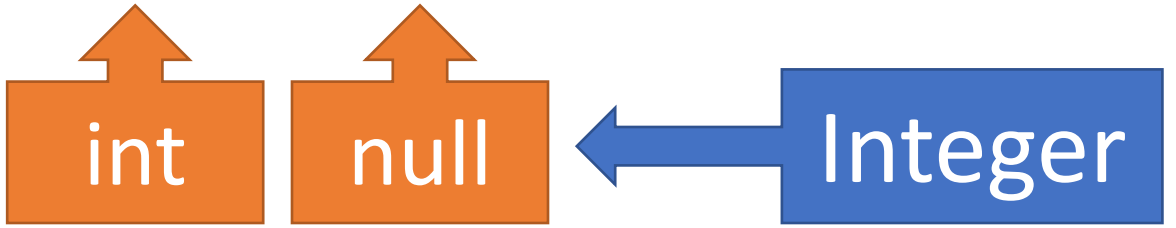
At run time, the first operand expression of the conditional expression is evaluated first. If necessary, unboxing conversion is performed on the result.

The resulting `boolean` value is then used to choose either the second or the third operand expression:

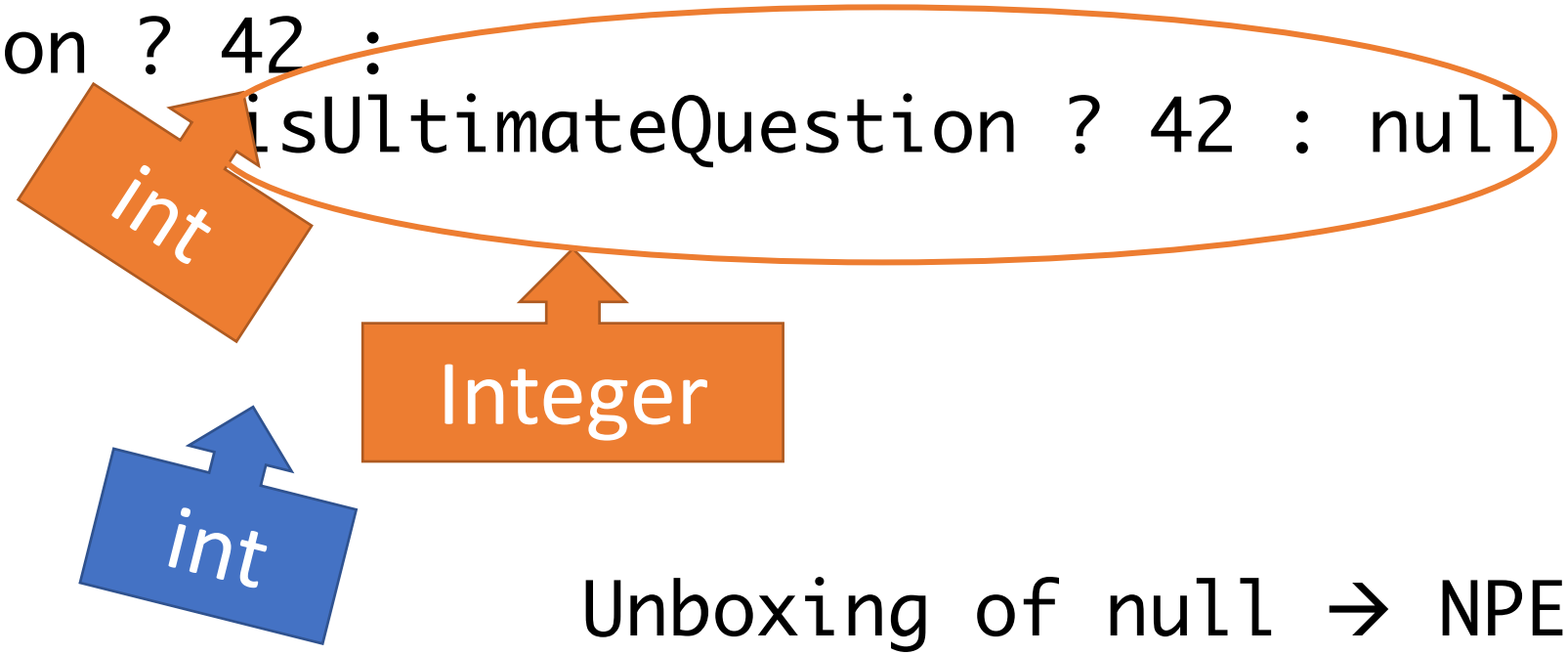
- If the value of the first operand is `true`, then the second operand expression is chosen.

Какой тип у `(b ? 42 : null)`?

```
isUltimateQuestion ? 42 : null
```




```
isUltimateQuestion ? 42 :
                     isUltimateQuestion ? 42 : null
```



Unboxing of null → NPE


DRAKULA
SPORT
PLAYOFF
Goliat
POWER

```
static short bitCount(short s) {
    short bits = 0;
    while (s != 0) {
        bits += s & 1;
        s >>>= 1;
    }
    return bits;
}
```

bitCount((short)-1)

1. 16
2. 32
3. 1
4. Ничего

```
static short bitCount(short s) {
    short bits = 0;
    while (s != 0) {
        bits += s & 1;
        s >>>= 1;
    }
    return bits;
}
```

```
bitCount((short)-1)
```

1. Ошибка компиляции
2. Зацикливается
3. ArithmeticException
4. StackOverflowError


gifs.com

```
static short bitCount(short s) {
    short bits = 0;
    while (s != 0) {
        bits += s & 1;
        s >>>= 1;
    }
    return bits;
}
```

```
bitCount((short)-1)
```

1. Ошибка компиляции
2. Зацикливается
3. ArithmeticException
4. StackOverflowError

Давайте разбираться!

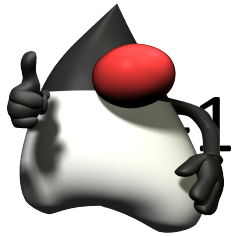
```
-1 short в двоичной: 0b                 11111111_11111111

-1 int в двоичной:   0b11111111_11111111_11111111_11111111

>>> 1:               0b01111111_11111111_11111111_11111111

Обратно в short:     0b                 11111111_11111111
```

```
System.out.println((short)Double.NEGATIVE_INFINITY);
System.out.println((short)Double.POSITIVE_INFINITY);
```

1. 0 и 0
2. 0 и -1
3. -32768 и 32767
4. Упадёт с ArithmeticException


AP
Begeruch

```
System.out.println((short)Double.NEGATIVE_INFINITY);
System.out.println((short)Double.POSITIVE_INFINITY);
```

1. 0 и 0
2.  1
3. -32768 и 32767
4. Упадёт с ArithmeticException

Двухшаговое приведение

double → int → short

Двухшаговое приведение

```
Double.NEGATIVE_INFINITY → Integer.MIN_VALUE
Double.POSITIVE_INFINITY → Integer.MAX_VALUE
```

Двухшаговое приведение

```
Integer.MAX_VALUE:  0b01111111_11111111_11111111_11111111
B short:                    0b                  11111111_11111111

Integer.MIN_VALUE:  0b10000000_00000000_00000000_00000000
B short :                   0b                  00000000_00000000
```

```
public class Superhero {

    private Long strength;

    public Superhero(Long strength) {
        this.strength = strength;
    }

    public Long getStrength() {
        return strength;
    }
}
```

Какой вариант правильный?

```
class Superhero implements Comparable<Superhero> {

  …
  @Override
  public int compareTo(Superhero this,
                       Superhero that) {
    return this.strength.compareTo(that.strength);
  }
}
```

1

```
class Superhero implements Comparable<Superhero> {

  …
  public int compareTo(Superhero me, Superhero you)
{
    return me.strength.compareTo(you.strength);
  }
}
```

2

```
class Superhero implements Comparator<Superhero> {

  …
  @Override
  public int compare(Superhero batman, Superhero superman)
    return batman.strength.compareTo(superman.strength);
  }
}
```

3

4.Никакой!


CN

Какой вариант правильный?

```
class Superhero implements Comparable<Superhero> {

  …
  @Override
  public int compareTo(Superhero this, Superhero
that) {
    return this.strengt    eTo(that.strength);
  }
}
```

1

```
class Superhero implements Comparable<Superhero> {

  …
  public int compareTo(Superhero me, Superhero you)
{
    return me.strength.compareTo(you.strength);
  }
}
```

2

```
class Superhero implements Comparator<Superhero> {

  …
  @Override
  public int compare(Superhero batman, Superhero superman)
    return batman.strength.compareTo(superman.strength);
  }
}
```

3

4.Никакой!

8.4.1. Formal Parameters

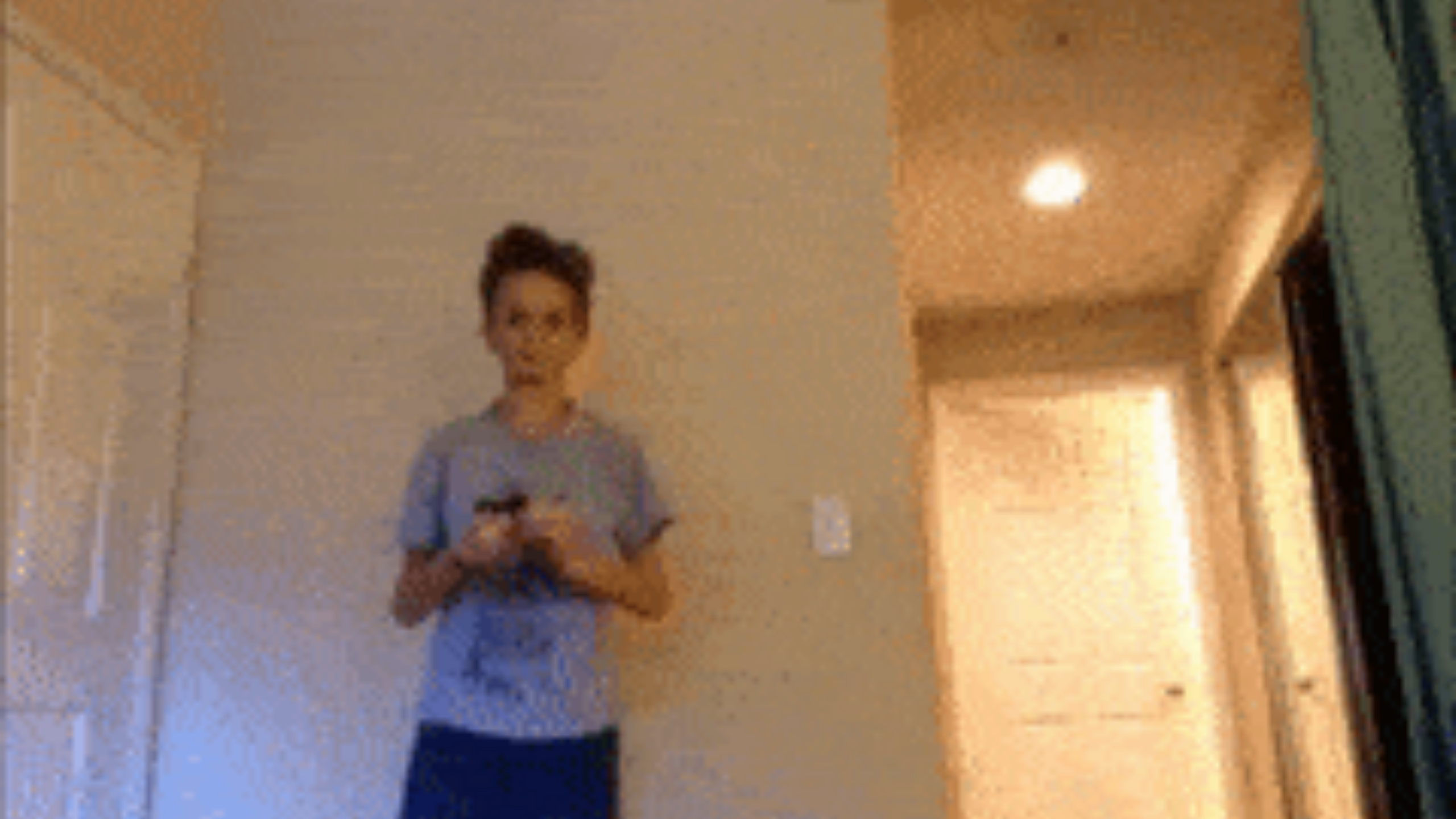
The *formal parameters* of a method or constructor, if any, are specified by a list of comma-separated parameter specifiers. Each parameter specifier consists of a type (optionally preceded by the `final` modifier and/or one or more annotations) and an identifier (optionally followed by brackets) that specifies the name of the parameter.

If a method or constructor has no formal parameters, only an empty pair of parentheses appears in the declaration of the method or constructor.


```
FormalParameterList:
  ReceiverParameter
  FormalParameters , LastFormalParameter
  LastFormalParameter

FormalParameters:
  FormalParameter {, FormalParameter}
  ReceiverParameter {, FormalParameter}

FormalParameter:
  {VariableModifier} UnannType VariableDeclaratorId

VariableModifier:
  (one of)
  Annotation final

ReceiverParameter:
  {Annotation} UnannType [Identifier .] this

LastFormalParameter:
  {VariableModifier} UnannType {Annotation} ... VariableDeclaratorId
  FormalParameter
```

```
System.out.println("Красивый спиннер\n" +
    "с логотипом Joker Conf\n"
      + "Цена: всего " +
+ '2' + " EUR");
```

1. Берём 100 штук, продадим участникам конфы по пять евро, выручку пропьём
2. Возьмём два - тебе и мне, 140 рублей с носа не такие большие деньги
3. Возьмём один на двоих, скинемся по 70, будем крутить по очереди
4. Вы что там с ума посходили с такими ценами???



```
System.out.println("Красивый спиннер\n" +
    "с логотипом Joker Conf\n"
      + "Цена: всего " +
+ '2' + " EUR");
```

1. Берём 100 штук, продадим участникам конфы по пять евро, выручку пропьём
2. Возьмём две - тебе и мне, 140 рублей с носа не такие большие деньги
3. Возьмём один на двоих, скинемся по 70, будем крутить по очереди
4. Вы что там с ума пос[illegible] с такими ценами???

```
System.out.println("Красивый спиннер\n" +
    "с логотипом Joker Conf\n"
      + "Цена: всего " +
+ '2' + " EUR");
```

1. Берём 100 штук, продадим участникам конфы по пять евро, выручку пропьём
2. Возьмём две - тебе и мне, 140 рублей с носа не такие большие деньги
3. Возьмём один на двоих, скинемся по 70, будем крутить по очереди
4. Вы что там с ума пос[illegible] с такими ценами???

```
import static java.util.stream.IntStream.range;
range(0, 10).forEach(System.out::println);
```

1. 0..10
2. IllegalArgumentException
3. Ничего
4. 10..0


MAKE GIFS AT GIFSOUP.COM

```
import static java.util.stream.IntStream.range;
range(10, 0).forEach(System.out::println);
```

1. 0..10
2. IllegalArgumentException
3. H 
4. 10..0

Как же всё таки сделать?!

```
IntStream.rangeClosed(0, 10)
         .map(x -> 10 - x)
         .forEach(System.out::println)
```

vs.

```
(10..0).each{println it}
```


Groovy

СКОТНЫЙ ДВОР

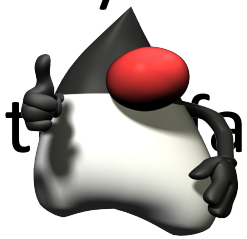
ДЖОРДЖ ОРУЭЛЛ

```
static void compare(double x, double y) {
    System.out.println(x == y);
    System.out.println(Objects.equals(x, y));
}
```

1. Может true/false, а вот false/true - никак
2. Может false/true, а вот true/false - никак
3. Может и true/false, и false/true
4. Всегда консистентны.

```
static void compare(double x, double y) {
    System.out.println(x == y);
    System.out.println(Objects.equals(x, y));
}
```

1. Может true/false, а вот false/true - никак
2. Может false/true, а вот true/false - никак
3. Может и t[illegible]alse, и false/true
4. Всегда консистентны.



```
compare(Double.NaN, Double.NaN) - false/true

compare(0, -0)                  - true/false
```


РОССИЯ 24
ПРЯМОЙ ЭФИР
ПО ДАННЫМ ЦИК
Свердловская область
1 Единая Россия 39,61%
2 Справедливая Россия 30,59%
3 КПРФ 18,64%
4 ЛДПР 17,67%
5 Яблоко 3,82%
6 Правое дело 2,75%
7 Патриоты России 2,27%
Партии, набравшие от 5% до 6% голосов, получат 1 место в ГД
00:50 USD/CHF 0,9206
TV screen shot

```
import static java.util.stream.LongStream.of;
long[] votes = new Random(42).longs(1000).toArray();

1. of(votes).average().getAsDouble();
2. of(votes).asDoubleStream().average().getAsDouble();
3. of(votes).asDoubleStream().parallel().average().getAsDouble();

4. double sum = 0.0;
for(long val : votes)
      sum += val;
double average = sum / votes.length;
```

1. Все результаты одинаковые
2. Три одинаковых
3. Два одинаковых
4. Все разные

```
import static java.util.stream.LongStream.of;
long[] votes = new Random(42).longs(1000).toArray();

1. of(votes).average().getAsDouble();
2. of(votes).asDoubleStream().average().getAsDouble();
3. of(votes).asDoubleStream().parallel().average().getAsDouble();

4. double sum = 0.0;
for(long val : votes)
      sum += val;
double average = sum / votes.length;
```

1. Все результаты одинаковые
2. Три одинаковых
3. Два одинаковых
4. В[illegible]зные

146%, конечно

```
1. of(votes).average().getAsDouble();
2. of(votes).asDoubleStream().average().getAsDouble();
3. of(votes).asDoubleStream().parallel().average().getAsDouble();

4. double sum = 0.0;
for(long val : votes)
      sum += val;
double average = sum / votes.length;
```

1. 6.895315673268 91E15
2. -3.066051176857 3549E17
3. -3.066051176857 3555E17
4. -3.066051176857 3491E17

Выводы


WITH GREAT
POWER
COMES GREAT
RESPONSIBILITY
PETER PARKER is SPIDER MAN

- ПИШИТЕ ЧИТАЕМЫЙ КОД!
- КОММЕНТИРУЙТЕ ВСЕ ТРЮКИ
- ИНОГДА ДАЖЕ В ДЖА
  БАГИ (В ЭТОМ СЕЗОН
- СТАТИЧЕСКИЕ АНАЛ
  КОДА РУЛЯТ! INTELL
- RTFM!
- НЕ БОЛЕЙТЕ СТРИМ

-САЛЛИ ПОНИЛЛАЕТЕ, JAVA 8 И 9 - КЛАДЕЗЬ ПАЗЗЛЕРОВ

-ЕСЛИ ВЫ НАТКНУЛИСЬ НА ПАЗЗЛЕР, ДАВАЙТЕ ЕГО СЮДА!

-PUZZLERS+JAVA@JFROG.COЛЛ

-ПОНРАВИЛОСЬ?
-ХВАЛИТЕ НАС СКОРЕЕ В ТВИТТЕРЕ!

-#JAVAPUZZLERSNG #JOKERCONF
-@TAGIR_VALEEV
-@JBARUCH

-НЕ ПОНРАВИЛОСЬ?
-/DEV/NULL