Маленькие оптимизации

Java 9-14

Тагир Валеев

Joker<?> 2016

Sander Mak

Luminis Technologies

Java 9 Modularity in Action

https://youtu.be/6ewlhP-k1fA

Joker<?> 2017

Cay Horstmann

San Jose State University

Java 9: the good parts (not modules)

https://youtu.be/_cvgg9szPH8

The New HTTP Client API in Java 11

Sergey Kuksenko
Java Platform Group, Oracle
October, 2018

https://youtu.be/lAW_NhJ3kqs

Fun With Var

Nicolai Parlog
codefx.org / @nipafx

J-Fall
#jfall

Nicolai Parlog
Fun with `var` in Java 11

https://youtu.be/BdnaXuthR-U

The Lord of the Strings: Two Scours

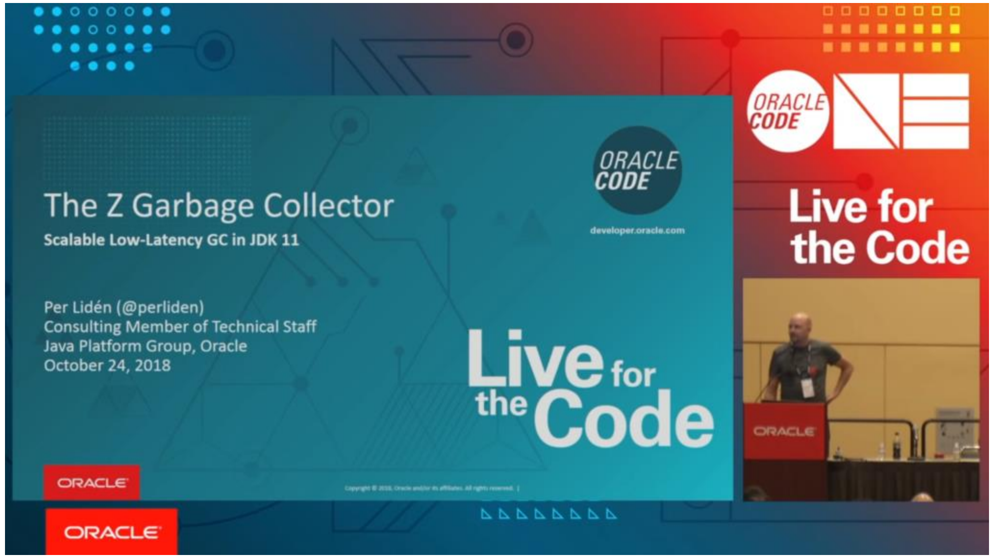
Алексей Шипилёв
Oracle

https://youtu.be/v2XLg4SvCF8

Chris Thalinger
Twitter

Graal: how to use the new JVM JIT compiler in real life

https://youtu.be/OPOHmQORG6M



The Z Garbage Collector
Scalable Low-Latency GC in JDK 11

Per Lidén (@perliden)
Consulting Member of Technical Staff
Java Platform Group, Oracle
October 24, 2018

https://youtu.be/kF_r3GE3zOo

Application Class Data Sharing
Archiving and Sharing Class Metadata and Java Objects in HotSpot VM to Improve Startup Performance and Reduce Footprint

Ioi Lam, Jiangli Zhou
Hotspot JVM developers
Oracle
October 23, 2018

https://youtu.be/nniYSR4GAH4

Маленькие оптимизации

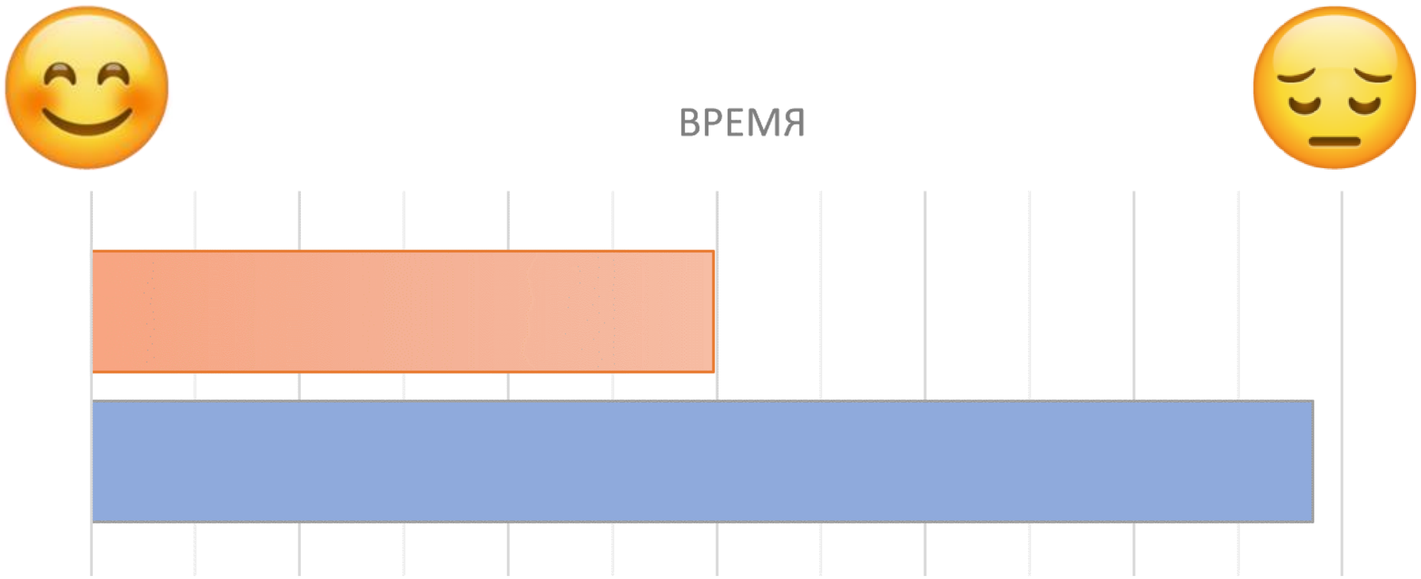
драма в трёх актах

Акт 1. Строки

Акт 2. Коллекции

Акт 3. Числа

- ✓ Intel Core i7-6820HQ CPU @ 2.70GHz, 4 Core(s), HT
- ✓ Windows 10
- ✓ -XX:+UseParallelGC



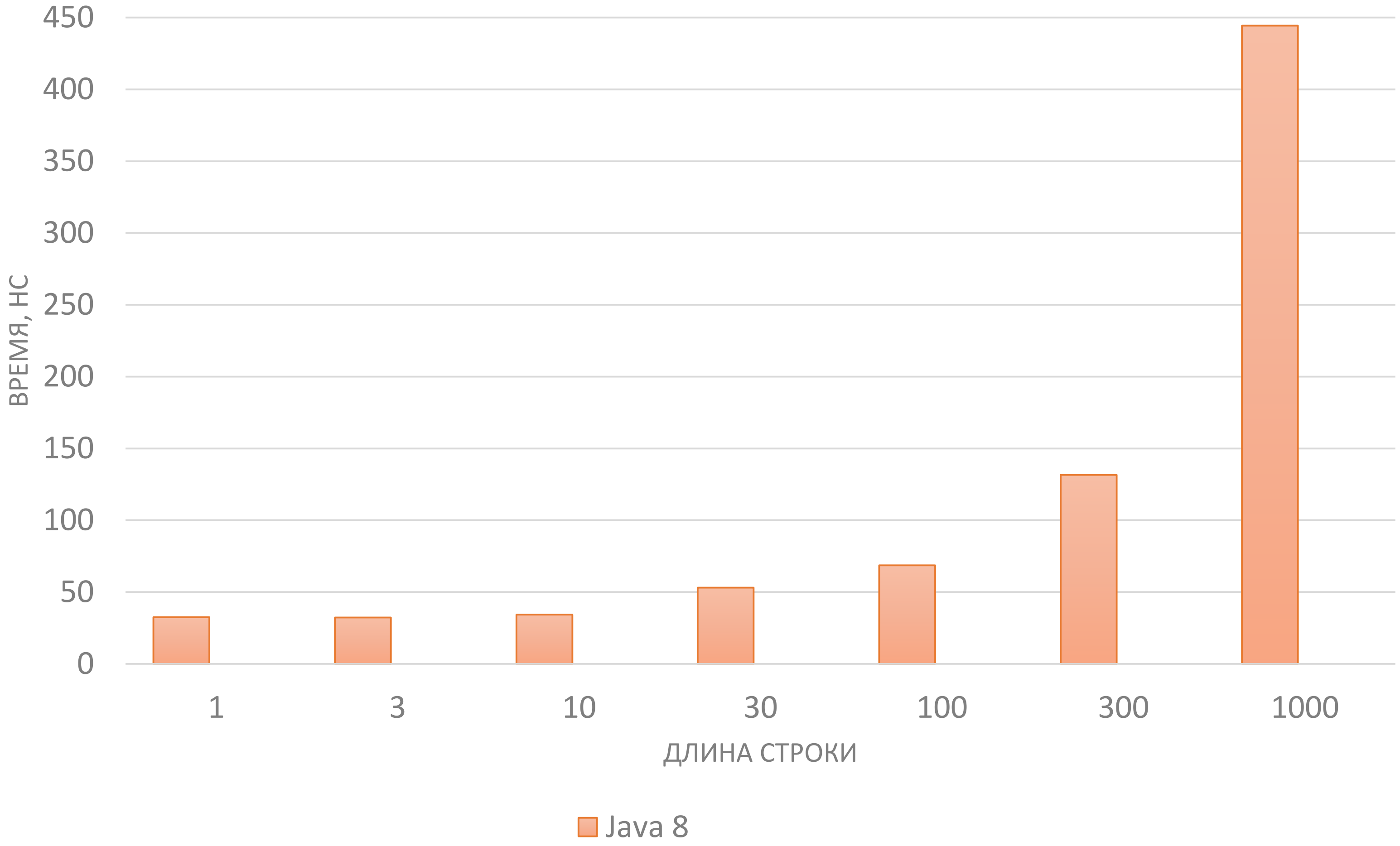
Developers, developers, developers, developers, deve

String.join

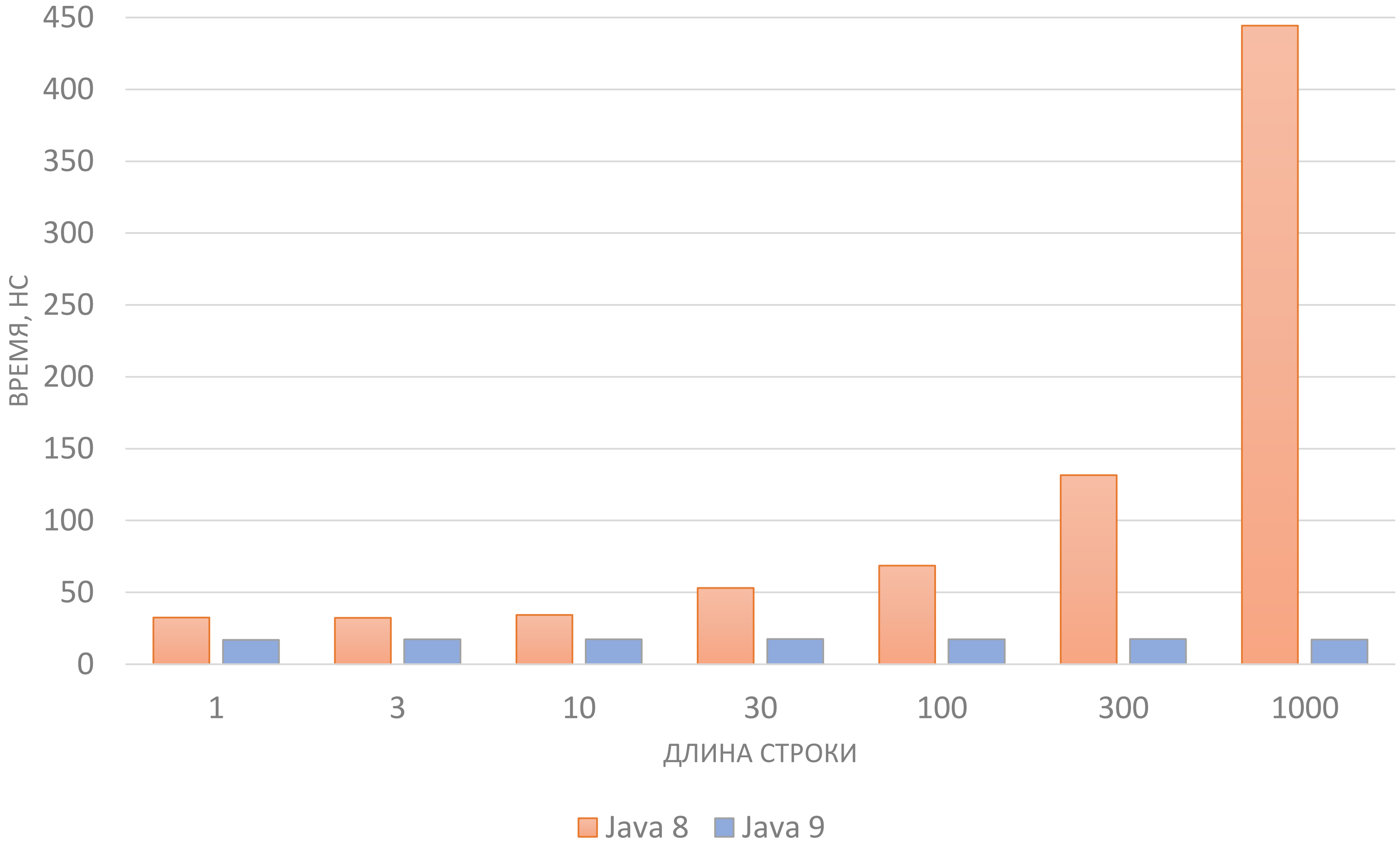
```java
public static String join(CharSequence delimiter, CharSequence... elements) {
    Objects.requireNonNull(delimiter);
    Objects.requireNonNull(elements);
    // Number of elements not likely worth Arrays.stream overhead.
    StringJoiner joiner = new StringJoiner(delimiter);
    for (CharSequence cs: elements) {
        joiner.add(cs);
    }
    return joiner.toString();
}
```

```java
@Param({"1", "3", "10", "30", "100", "300", "1000"})
int stringLen;

String[] data;

@Setup
public void setup() {
    char[] x = new char[stringLen];
    Arrays.fill(x, '*');
    data = new String[] {new String(x)};
}

@Benchmark
public String join() {
    return String.join(", ", data);
}
```

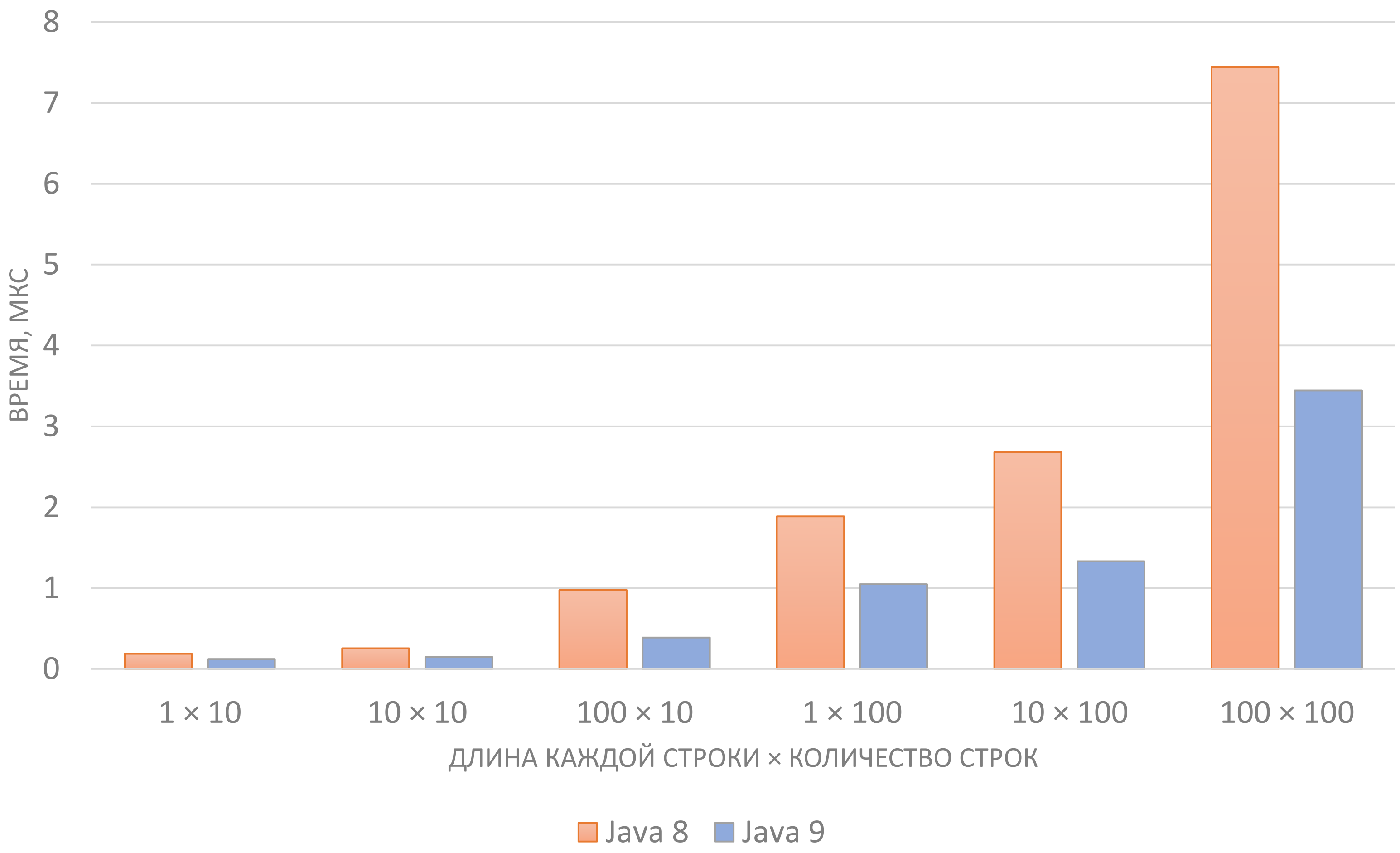
String.join; одна строка



String.join; одна строка



String.join; много строк



Java 8

```
public final class StringJoiner {

  private StringBuilder value;

  public StringJoiner add(CharSequence newElement) {
    prepareBuilder().append(newElement);
    return this;
  }

  private StringBuilder prepareBuilder() {
    if (value != null) {
      value.append(delimiter);
    } else {
      value = new StringBuilder().append(prefix);
    }
    return value;
  }
}
```

Java 9

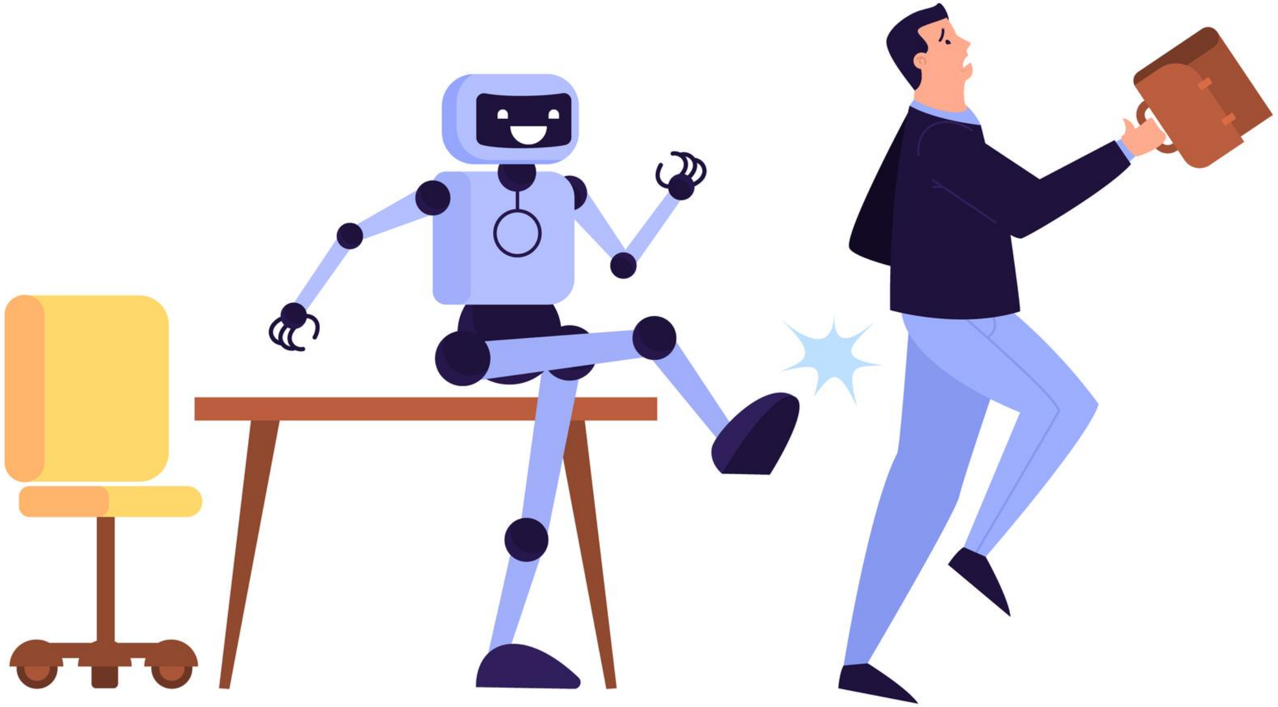
```
public final class StringJoiner {

  private String[] elts; // JDK-8054221
  private int len;

  public StringJoiner add(CharSequence newElement) {
    final String elt = String.valueOf(newElement);
    if (elts == null) {
      elts = new String[8];
    } else {
      if (size == elts.length)
        elts = Arrays.copyOf(elts, 2 * size);
      len += delimiter.length();
    }
    len += elt.length();
    elts[size++] = elt;
    return this;
  }
}
```

Java 9

```java
public final class StringJoiner {

  private void compactElts() {
    if (size > 1) {
      final char[] chars = new char[len];
      int i = 1, k = getChars(elts[0], chars, 0);
      do {
        k += getChars(delimiter, chars, k);
        k += getChars(elts[i], chars, k);
        elts[i] = null;
      } while (++i < size);
      size = 1;
      elts[0] = new String(chars);
    }
  }
}
```

String.replace

—

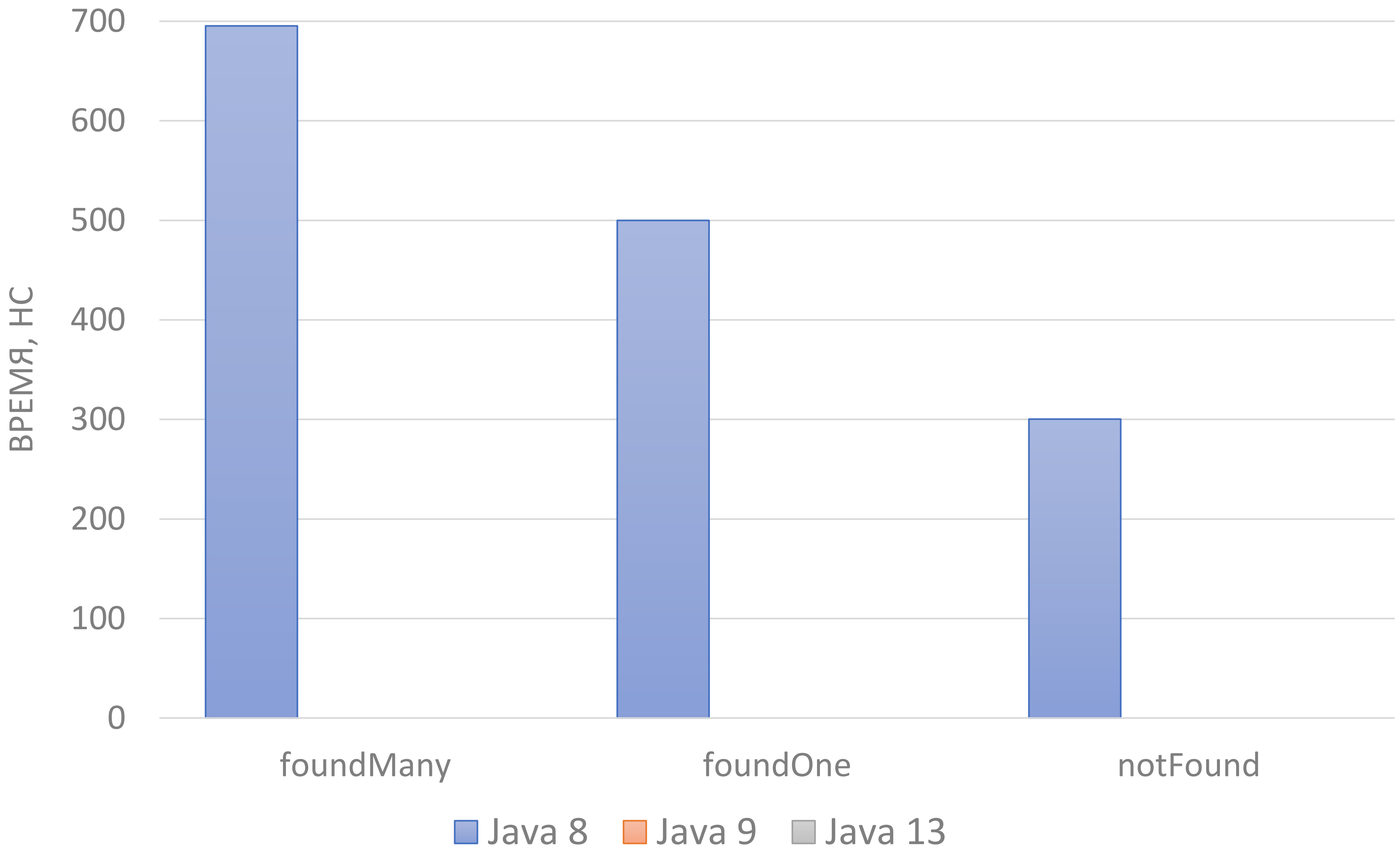


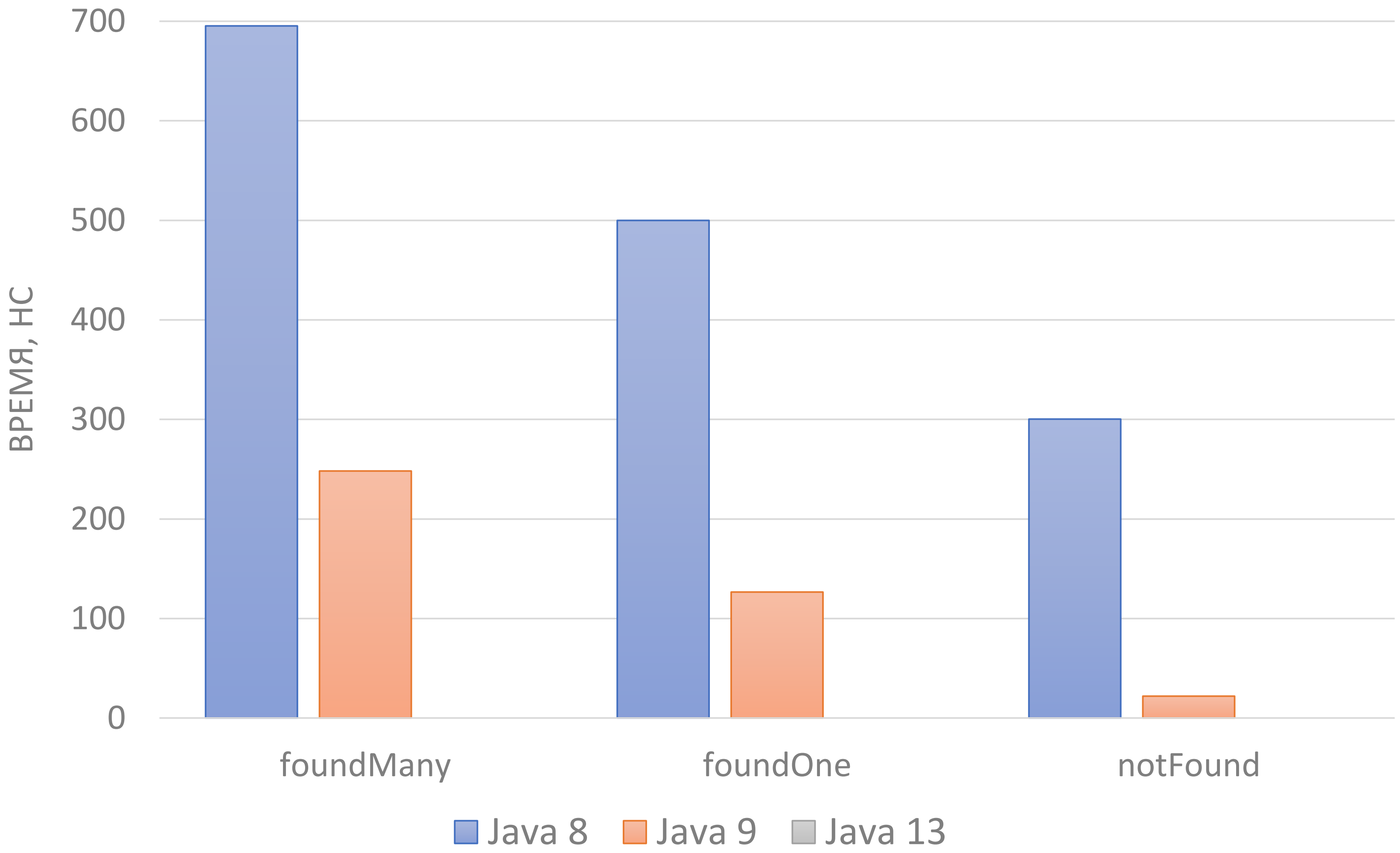
"краб крабу сделал грабли, подарил грабли крабу: «Грабь граблями гравий, краб»."

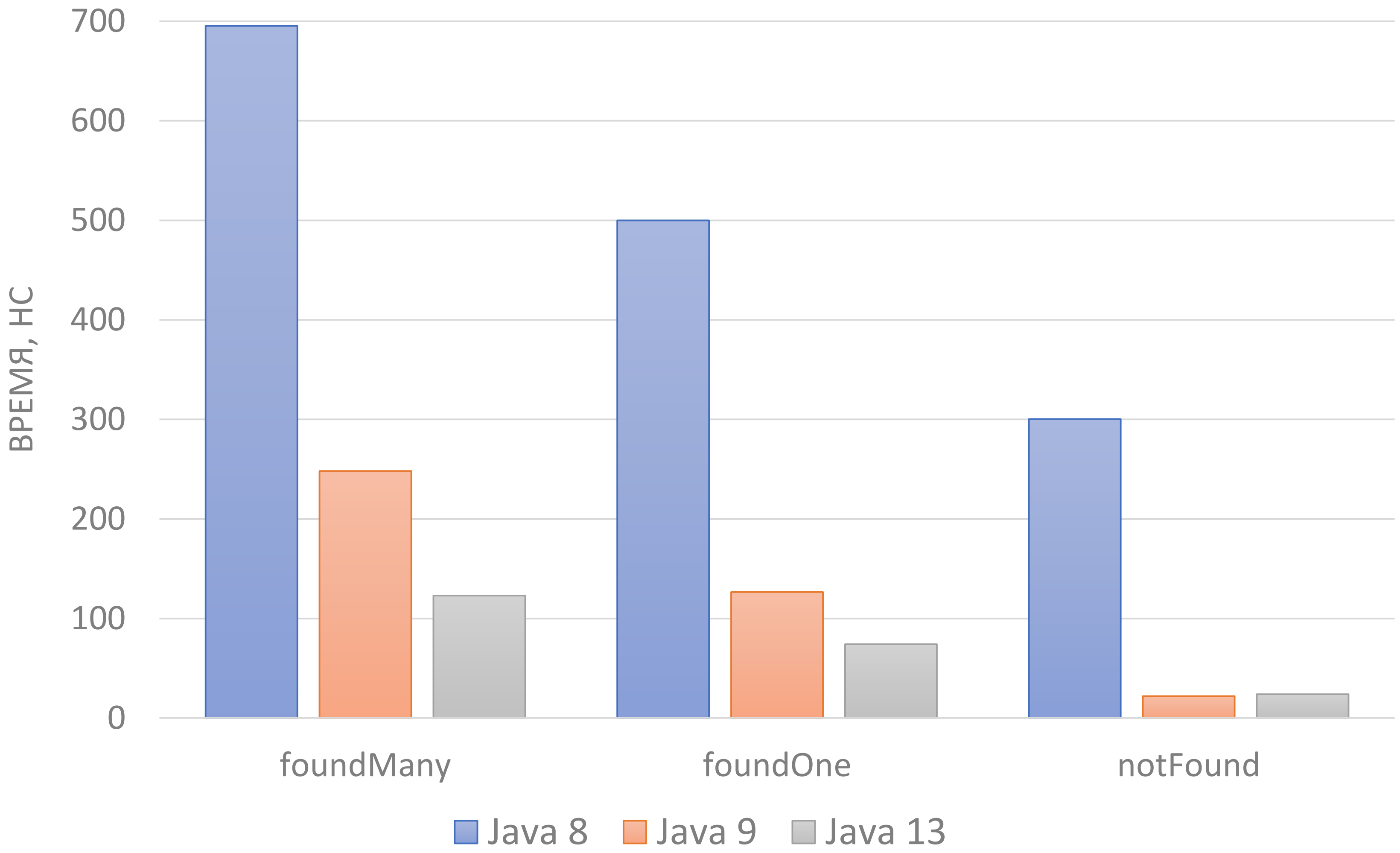
"завлаб завлабу сделал грабли, подарил грабли завлабу: «Грабь граблями гравий, завлаб»."

```
s = s.replace("краб", "завлаб");

s = s.replaceAll("краб", "завлаб");
```

```java
static final String КРАБ =
  "краб крабу сделал грабли, подарил грабли крабу: «Грабь граблями гравий, краб».";

@Benchmark
public String foundMany() {
    return КРАБ.replace("краб", "завлаб");
}

@Benchmark
public String foundOne() {
    return КРАБ.replace("гравий", "гнусавей");
}

@Benchmark
public String notFound() {
    return КРАБ.replace("завлаб", "краб");
}
```


700
600
500
400
300
200
100
0
ВРЕМЯ, НС
foundMany
foundOne
notFound
Java 8
Java 9
Java 13


700
600
500
400
300
200
100
0
ВРЕМЯ, НС
foundMany
foundOne
notFound
Java 8
Java 9
Java 13


700
600
500
400
300
200
100
0
ВРЕМЯ, НС
foundMany
foundOne
notFound
Java 8
Java 9
Java 13

Java 8

```
public String replace(CharSequence target, CharSequence replacement) {
    return Pattern.compile(target.toString(), Pattern.LITERAL).matcher(
            this).replaceAll(Matcher.quoteReplacement(replacement.toString()));
}
```

Java 9

```java
public String replace(CharSequence target, CharSequence replacement) {
    String tgtStr = target.toString();
    String replStr = replacement.toString();
    int j = indexOf(tgtStr);
    if (j < 0) return this;
    int tgtLen = tgtStr.length();
    int tgtLen1 = Math.max(tgtLen, 1);
    int thisLen = length();

    int newLenHint = thisLen - tgtLen + replStr.length();
    if (newLenHint < 0) {
        throw new OutOfMemoryError();
    }
    StringBuilder sb = new StringBuilder(newLenHint);
    int i = 0;
    do {
        sb.append(this, i, j).append(replStr);
        i = j + tgtLen;
    } while (j < thisLen && (j = indexOf(tgtStr, j + tgtLen1)) > 0);
    return sb.append(this, i, thisLen).toString();
}
```

Java 9

```java
public String replace(CharSequence target, CharSequence replacement) {
    String tgtStr = target.toString();
    String replStr = replacement.toString();
    int j = indexOf(tgtStr);
    if (j < 0) return this;
    int tgtLen = tgtStr.length();
    int tgtLen1 = Math.max(tgtLen, 1);
    int thisLen = length();

    int newLenHint = thisLen - tgtLen + replStr.length();
    if (newLenHint < 0) {
        throw new OutOfMemoryError();
    }
    StringBuilder sb = new StringBuilder(newLenHint);
    int i = 0;
    do {
        sb.append(this, i, j).append(replStr);
        i = j + tgtLen;
    } while (j < thisLen && (j = indexOf(tgtStr, j + tgtLen1)) > 0);
    return sb.append(this, i, thisLen).toString();
}
```

Java 9

```java
public String replace(CharSequence target, CharSequence replacement) {
    String tgtStr = target.toString();
    String replStr = replacement.toString();
    int j = indexOf(tgtStr);
    if (j < 0) return this;
    int tgtLen = tgtStr.length();
    int tgtLen1 = Math.max(tgtLen, 1);
    int thisLen = length();

    int newLenHint = thisLen - tgtLen + replStr.length();
    if (newLenHint < 0) {
        throw new OutOfMemoryError();
    }
    StringBuilder sb = new StringBuilder(newLenHint);
    int i = 0;
    do {
        sb.append(this, i, j).append(replStr);
        i = j + tgtLen;
    } while (j < thisLen && (j = indexOf(tgtStr, j + tgtLen1)) > 0);
    return sb.append(this, i, thisLen).toString();
}
```

Java 9

```java
public String replace(CharSequence target, CharSequence replacement) {
    String tgtStr = target.toString();
    String replStr = replacement.toString();
    int j = indexOf(tgtStr);
    if (j < 0) return this;
    int tgtLen = tgtStr.length();
    int tgtLen1 = Math.max(tgtLen, 1);
    int thisLen = length();

    int newLenHint = thisLen - tgtLen + replStr.length();
    if (newLenHint < 0) {
        throw new OutOfMemoryError();
    }
    StringBuilder sb = new StringBuilder(newLenHint);
    int i = 0;
    do {
        sb.append(this, i, j).append(replStr);
        i = j + tgtLen;
    } while (j < thisLen && (j = indexOf(tgtStr, j + tgtLen1)) > 0);
    return sb.append(this, i, thisLen).toString();
}
```

Java 13

```java
if (trgtLen > 0) {
    if (trgtLen == 1 && replLen == 1) {
        return replace(trgtStr.charAt(0), replStr.charAt(0));
    }

    boolean thisIsLatin1 = this.isLatin1();
    boolean trgtIsLatin1 = trgtStr.isLatin1();
    boolean replIsLatin1 = replStr.isLatin1();
    String ret = (thisIsLatin1 && trgtIsLatin1 && replIsLatin1)
            ? StringLatin1.replace(value, thisLen,
                                   trgtStr.value, trgtLen,
                                   replStr.value, replLen)
            : StringUTF16.replace(value, thisLen, thisIsLatin1,
                                  trgtStr.value, trgtLen, trgtIsLatin1,
                                  replStr.value, replLen, replIsLatin1);
    if (ret != null) {
        return ret;
    }
    return this;
} else { // trgtLen == 0
    …
}
```

Java 13

```java
if (trgtLen > 0) {
    if (trgtLen == 1 && replLen == 1) {
        return replace(trgtStr.charAt(0), replStr.charAt(0));
    }

    boolean thisIsLatin1 = this.isLatin1();
    boolean trgtIsLatin1 = trgtStr.isLatin1();
    boolean replIsLatin1 = replStr.isLatin1();
    String ret = (thisIsLatin1 && trgtIsLatin1 && replIsLatin1)
            ? StringLatin1.replace(value, thisLen,
                                   trgtStr.value, trgtLen,
                                   replStr.value, replLen)
            : StringUTF16.replace(value, thisLen, thisIsLatin1,
                                  trgtStr.value, trgtLen, trgtIsLatin1,
                                  replStr.value, replLen, replIsLatin1);
    if (ret != null) {
        return ret;
    }
    return this;
} else { // trgtLen == 0
    …
}
```

Java 13

```java
if (trgtLen > 0) {
    if (trgtLen == 1 && replLen == 1) {
        return replace(trgtStr.charAt(0), replStr.charAt(0));
    }

    boolean thisIsLatin1 = this.isLatin1();
    boolean trgtIsLatin1 = trgtStr.isLatin1();
    boolean replIsLatin1 = replStr.isLatin1();
    String ret = (thisIsLatin1 && trgtIsLatin1 && replIsLatin1)
            ? StringLatin1.replace(value, thisLen,
                                   trgtStr.value, trgtLen,
                                   replStr.value, replLen)
            : StringUTF16.replace(value, thisLen, thisIsLatin1,
                                  trgtStr.value, trgtLen, trgtIsLatin1,
                                  replStr.value, replLen, replIsLatin1);
    if (ret != null) {
        return ret;
    }
    return this;
} else { // trgtLen == 0
    …
}
```

Java 13

```java
if (trgtLen > 0) {
    if (trgtLen == 1 && replLen == 1) {
        return replace(trgtStr.charAt(0), replStr.charAt(0));
    }

    boolean thisIsLatin1 = this.isLatin1();
    boolean trgtIsLatin1 = trgtStr.isLatin1();
    boolean replIsLatin1 = replStr.isLatin1();
    String ret = (thisIsLatin1 && trgtIsLatin1 && replIsLatin1)
            ? StringLatin1.replace(value, thisLen,
                                   trgtStr.value, trgtLen,
                                   replStr.value, replLen)
            : StringUTF16.replace(value, thisLen, thisIsLatin1,
                                  trgtStr.value, trgtLen, trgtIsLatin1,
                                  replStr.value, replLen, replIsLatin1);
    if (ret != null) {
        return ret;
    }
    return this;
} else { // trgtLen == 0
    …
}
```

Java 13

```
if (trgtLen > 0) {
    if (trgtLen == 1 && replLen == 1) {
        return replace(trgtStr.charAt(0), replStr.charAt(0));
    }

    boolean thisIsLatin1 = this.isLatin1();
    boolean trgtIsLatin1 = trgtStr.isLatin1();
    boolean replIsLatin1 = replStr.isLatin1();
    String ret = (thisIsLatin1 && trgtIsLatin1 && replIsLatin1)
            ? StringLatin1.replace(value, thisLen,
                                   trgtStr.value, trgtLen,
                                   replStr.value, replLen)
            : StringUTF16.replace(value, thisLen, thisIsLatin1,
                                  trgtStr.value, trgtLen, trgtIsLatin1,
                                  replStr.value, replLen, replIsLatin1);
    if (ret != null) {
        return ret;
    }
    return this;
} else { // trgtLen == 0
    …
}
```

#	VALUE	TARGET	REPL	RESULT
1	Latin1	Latin1	UTF16	null or UTF16
2	Latin1	UTF16	Latin1	null
3	Latin1	UTF16	UTF16	null
4	UTF16	Latin1	Latin1	null or UTF16
5	UTF16	Latin1	UTF16	null or UTF16
6	UTF16	UTF16	Latin1	null, Latin1 or UTF16
7	UTF16	UTF16	UTF16	null or UTF16

#	VALUE	TARGET	REPL	RESULT
1	Latin1	Latin1	UTF16	null or UTF16
2	Latin1	UTF16	Latin1	null
3	Latin1	UTF16	UTF16	null
4	UTF16	Latin1	Latin1	null or UTF16
5	UTF16	Latin1	UTF16	null or UTF16
6	UTF16	UTF16	Latin1	null, Latin1 or UTF16
7	UTF16	UTF16	UTF16	null or UTF16

#	VALUE	TARGET	REPL	RESULT
1	Latin1	Latin1	UTF16	null or UTF16
2	Latin1	UTF16	Latin1	null
3	Latin1	UTF16	UTF16	null
4	UTF16	Latin1	Latin1	null or UTF16
5	UTF16	Latin1	UTF16	null or UTF16
6	UTF16	UTF16	Latin1	null, Latin1 or UTF16
7	UTF16	UTF16	UTF16	null or UTF16

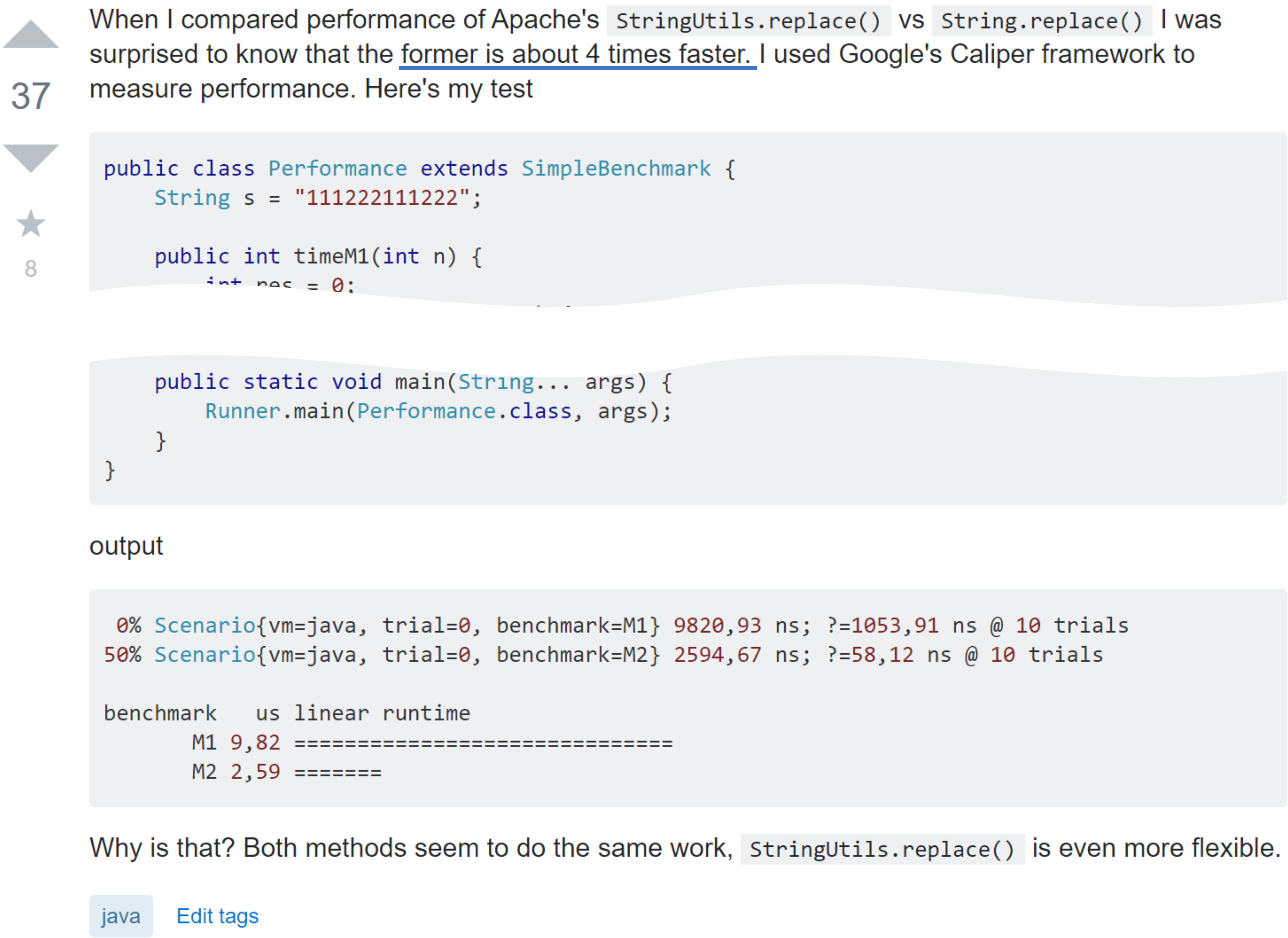
```
public static String replace(byte[] value, int valLen, byte[] targ,
                             int targLen, byte[] repl, int replLen)
{
    assert targLen > 0;
    int i, j, p = 0;
    if (valLen == 0 || (i = indexOf(value, valLen, targ, targLen, 0)) < 0) {
        return null; // for string to return this;
    }
```

```
public static String replace(byte[] value, int valLen, byte[] targ,
                             int targLen, byte[] repl, int replLen)
{
    …
    // find and store indices of substrings to replace
    int[] pos = new int[16];
    pos[0] = i;
    i += targLen;
    while ((j = indexOf(value, valLen, targ, targLen, i)) > 0) {
        if (++p == pos.length) {
            pos = Arrays.copyOf(pos, ArraysSupport.newLength(p, 1, p >> 1));
        }
        pos[p] = j;
        i = j + targLen;
    }
```

```java
public static String replace(byte[] value, int valLen, byte[] targ,
                             int targLen, byte[] repl, int replLen)
{
    …
    int resultLen;
    try {
        resultLen = Math.addExact(valLen,
                Math.multiplyExact(++p, replLen - targLen));
    } catch (ArithmeticException ignored) {
        throw new OutOfMemoryError();
    }
    if (resultLen == 0) {
        return "";
    }
    byte[] result = StringConcatHelper.newArray(resultLen);
```

```
public static String replace(byte[] value, int valLen, byte[] targ,
                             int targLen, byte[] repl, int replLen)
{
    …
    int resultLen;
    try {
        resultLen = Math.addExact(valLen,
                Math.multiplyExact(++p, replLen - targLen));
    } catch (ArithmeticException ignored) {
        throw new OutOfMemoryError();
    }
    if (resultLen == 0) {
        return "";
    }
    byte[] result = StringConcatHelper.newArray(resultLen);
```

```
public static String replace(byte[] value, int valLen, byte[] targ,
                             int targLen, byte[] repl, int replLen)
{
    …
    int posFrom = 0, posTo = 0;
    for (int q = 0; q < p; ++q) {
        int nextPos = pos[q];
        while (posFrom < nextPos) {
            result[posTo++] = value[posFrom++];
        }
        posFrom += targLen;
        for (int k = 0; k < replLen; ++k) {
            result[posTo++] = repl[k];
        }
    }
    while (posFrom < valLen) {
        result[posTo++] = value[posFrom++];
    }
    return new String(result, LATIN1);
}
```



When I compared performance of Apache's `StringUtils.replace()` vs `String.replace()` I was surprised to know that the former is about 4 times faster. I used Google's Caliper framework to measure performance. Here's my test

```
public class Performance extends SimpleBenchmark {
    String s = "111222111222";

    public int timeM1(int n) {
        int res = 0;
```

```
    public static void main(String... args) {
        Runner.main(Performance.class, args);
    }
}
```

output

```
 0% Scenario{vm=java, trial=0, benchmark=M1} 9820,93 ns; ?=1053,91 ns @ 10 trials
50% Scenario{vm=java, trial=0, benchmark=M2} 2594,67 ns; ?=58,12 ns @ 10 trials

benchmark   us linear runtime
       M1 9,82 ==============================
       M2 2,59 =======
```

Why is that? Both methods seem to do the same work, `StringUtils.replace()` is even more flexible.

Java 8

```
 0% Scenario{vm=java, trial=0, benchmark=M1} 291.42 ns; σ=6.56 ns @ 10 trials
50% Scenario{vm=java, trial=0, benchmark=M2} 70.34 ns; σ=0.15 ns @ 3 trials

benchmark    ns linear runtime
       M1 291.4 ==============================
       M2  70.3 =======
```

Java 9

```
 0% Scenario{vm=java, trial=0, benchmark=M2} 99,15 ns; σ=8,34 ns @ 10 trials
50% Scenario{vm=java, trial=0, benchmark=M1} 103,43 ns; σ=9,01 ns @ 10 trials

benchmark    ns linear runtime
       M2  99,1 ============================
       M1 103,4 ==============================
```

Java 13

```
 0% Scenario{vm=java, trial=0, benchmark=M2} 91,64 ns; σ=5,12 ns @ 10 trials
50% Scenario{vm=java, trial=0, benchmark=M1} 57,38 ns; σ=2,51 ns @ 10 trials

benchmark   ns linear runtime
       M2 91,6 ==============================
       M1 57,4 ==================
```

HashSet.toArray

```java
public class HashSet<E>
    extends AbstractSet<E>
    implements Set<E>, Cloneable, java.io.Serializable
{
  private transient HashMap<E,Object> map;
  // Dummy value to associate with an Object in the backing Map
  private static final Object PRESENT = new Object();

  public HashSet() { map = new HashMap<>(); }

  public Iterator<E> iterator() { return map.keySet().iterator(); }

  public int size() { return map.size(); }

  public boolean add(E e) { return map.put(e, PRESENT)==null; }

  public boolean remove(Object o) { return map.remove(o)==PRESENT; }
  …
}
```

```
public <T> T[] toArray(T[] a) {
    // Estimate size of array; be prepared to see more or fewer elements
    int size = size();
    T[] r = a.length >= size ? a :
              (T[])java.lang.reflect.Array
              .newInstance(a.getClass().getComponentType(), size);
    Iterator<E> it = iterator();
    for (int i = 0; i < r.length; i++) {
      if (! it.hasNext()) { // fewer elements than expected
        if (a == r) {
          r[i] = null; // null-terminate
        } else if (a.length < i) {
          return Arrays.copyOf(r, i);
        } else {
          System.arraycopy(r, 0, a, 0, i);
          if (a.length > i) a[i] = null;
        }
        return a;
      }
      r[i] = (T)it.next();
    }
    // more elements than expected
    return it.hasNext() ? finishToArray(r, it) : r;
}
```

```java
public <T> T[] toArray(T[] a) {
    // Estimate size of array; be prepared to see more or fewer elements
    int size = size();
    T[] r = a.length >= size ? a :
              (T[])java.lang.reflect.Array
              .newInstance(a.getClass().getComponentType(), size);
    Iterator<E> it = iterator();
    for (int i = 0; i < r.length; i++) {
      if (! it.hasNext()) { // fewer elements than expected
        if (a == r) {
          r[i] = null; // null-terminate
        } else if (a.length < i) {
          return Arrays.copyOf(r, i);
        } else {
          System.arraycopy(r, 0, a, 0, i);
          if (a.length > i) a[i] = null;
        }
        return a;
      }
      r[i] = (T)it.next();
    }
    // more elements than expected
    return it.hasNext() ? finishToArray(r, it) : r;
}
```

```
public <T> T[] toArray(T[] a) {
    // Estimate size of array; be prepared to see more or fewer elements
    int size = size();
    T[] r = a.length >= size ? a :
              (T[])java.lang.reflect.Array
              .newInstance(a.getClass().getComponentType(), size);
    Iterator<E> it = iterator();
    for (int i = 0; i < r.length; i++) {
      if (! it.hasNext()) { // fewer elements than expected
        if (a == r) {
          r[i] = null; // null-terminate
        } else if (a.length < i) {
          return Arrays.copyOf(r, i);
        } else {
          System.arraycopy(r, 0, a, 0, i);
          if (a.length > i) a[i] = null;
        }
        return a;
      }
      r[i] = (T)it.next();
    }
    // more elements than expected
    return it.hasNext() ? finishToArray(r, it) : r;
}
```

```java
public <T> T[] toArray(T[] a) {
  // Estimate size of array; be prepared to see more or fewer elements
  int size = size();
  T[] r = a.length >= size ? a :
           (T[])java.lang.reflect.Array
           .newInstance(a.getClass().getComponentType(), size);
  Iterator<E> it = iterator();
  for (int i = 0; i < r.length; i++) {
    if (! it.hasNext()) { // fewer elements than expected
      if (a == r) {
        r[i] = null; // null-terminate
      } else if (a.length < i) {
        return Arrays.copyOf(r, i);
      } else {
        System.arraycopy(r, 0, a, 0, i);
        if (a.length > i) a[i] = null;
      }
      return a;
    }
    r[i] = (T)it.next();
  }
  // more elements than expected
  return it.hasNext() ? finishToArray(r, it) : r;
}
```

Java 14

```java
public class HashMap<K,V> extends AbstractMap<K,V>
    implements Map<K,V>, Cloneable, Serializable {


  final <T> T[] prepareArray(T[] a) { // JDK-8225339
    int size = this.size;
    if (a.length < size) {
      return (T[]) java.lang.reflect.Array
            .newInstance(a.getClass().getComponentType(), size);
    }
    if (a.length > size) {
      a[size] = null;
    }
    return a;
  }
}
```

Java 14

```
public class HashMap<K,V> extends AbstractMap<K,V>
    implements Map<K,V>, Cloneable, Serializable {


  <T> T[] keysToArray(T[] a) { // JDK-8225339
    Object[] r = a;
    Node<K,V>[] tab;
    int idx = 0;
    if (size > 0 && (tab = table) != null) {
      for (Node<K,V> e : tab) {
        for (; e != null; e = e.next) {
          r[idx++] = e.key;
        }
      }
    }
    return a;
  }
}
```

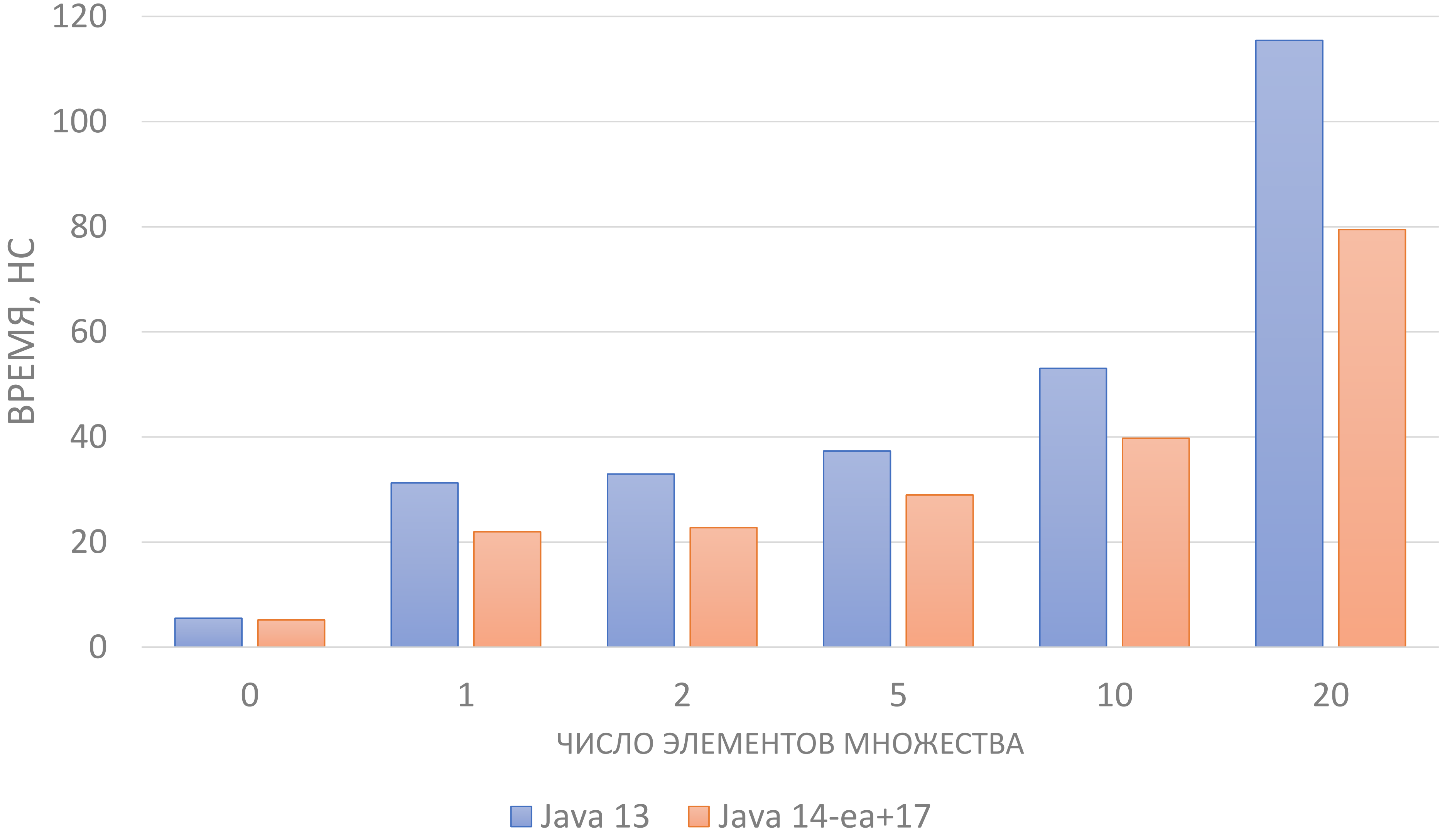
Java 14

```java
public class HashSet<E>
    extends AbstractSet<E>
    implements Set<E>, Cloneable, java.io.Serializable
{
  …
  @Override
  public Object[] toArray() {
    return map.keysToArray(new Object[map.size()]);
  }

  @Override
  public <T> T[] toArray(T[] a) {
    return map.keysToArray(map.prepareArray(a));
  }
}
```

Java 14

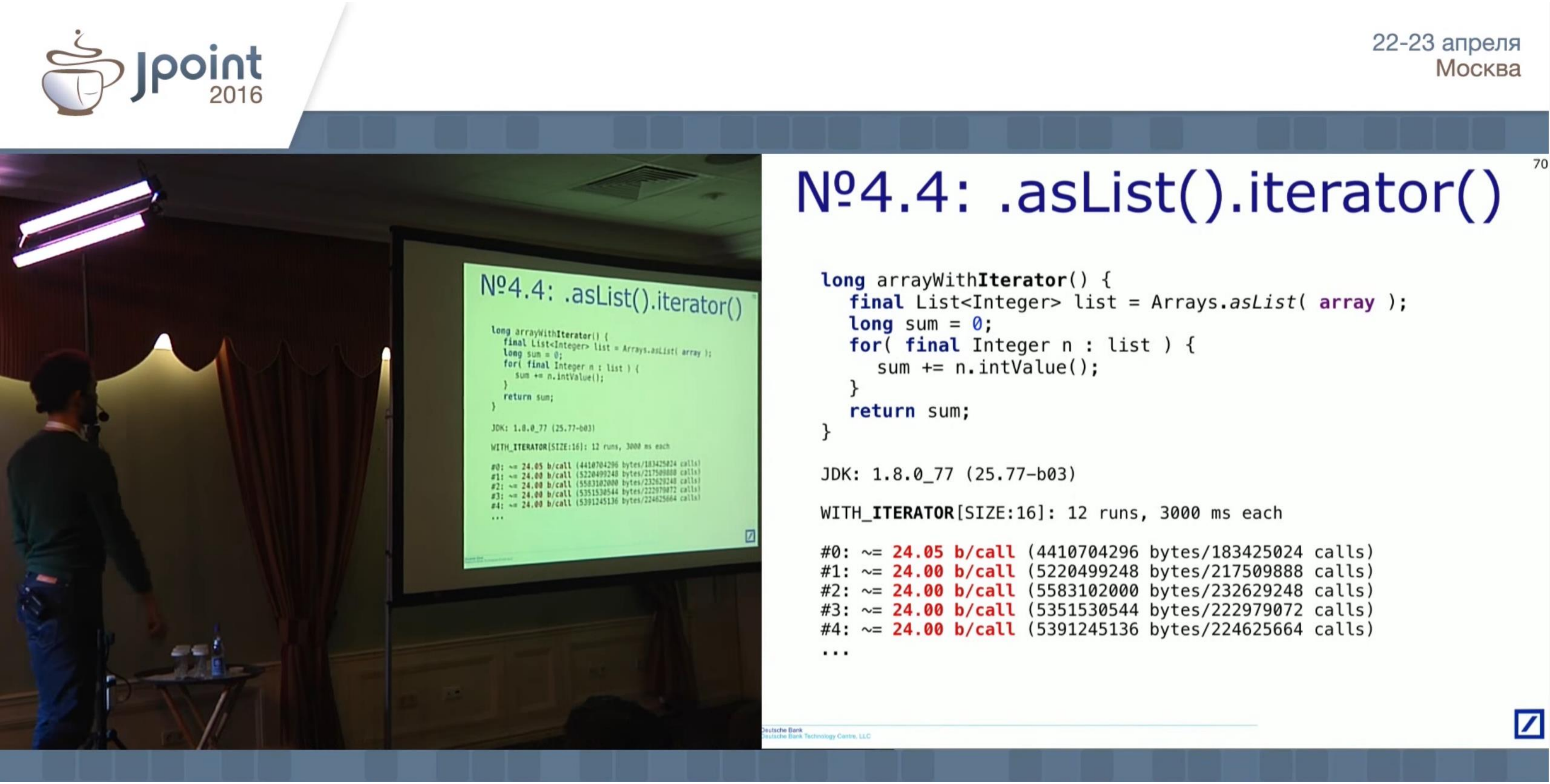
```java
public class LinkedHashMap<K,V> extends HashMap<K,V>
    implements Map<K,V> {

  @Override
  final <T> T[] keysToArray(T[] a) {
    Object[] r = a;
    int idx = 0;
    for (LinkedHashMap.Entry<K,V> e = head; e != null; e = e.after) {
      r[idx++] = e.key;
    }
    return a;
  }
}
```

hashSet.toArray(new String[0])



```
public ArrayList(Collection<? extends E> c) {
    elementData = c.toArray();
    if ((size = elementData.length) != 0) {
        // c.toArray might (incorrectly) not return Object[] (see 6260652)
        if (elementData.getClass() != Object[].class)
            elementData = Arrays.copyOf(elementData, size, Object[].class);
    } else {
        // replace with empty array.
        this.elementData = EMPTY_ELEMENTDATA;
    }
}
```

Arrays.asList().iterator()



№4.4: .asList().iterator()

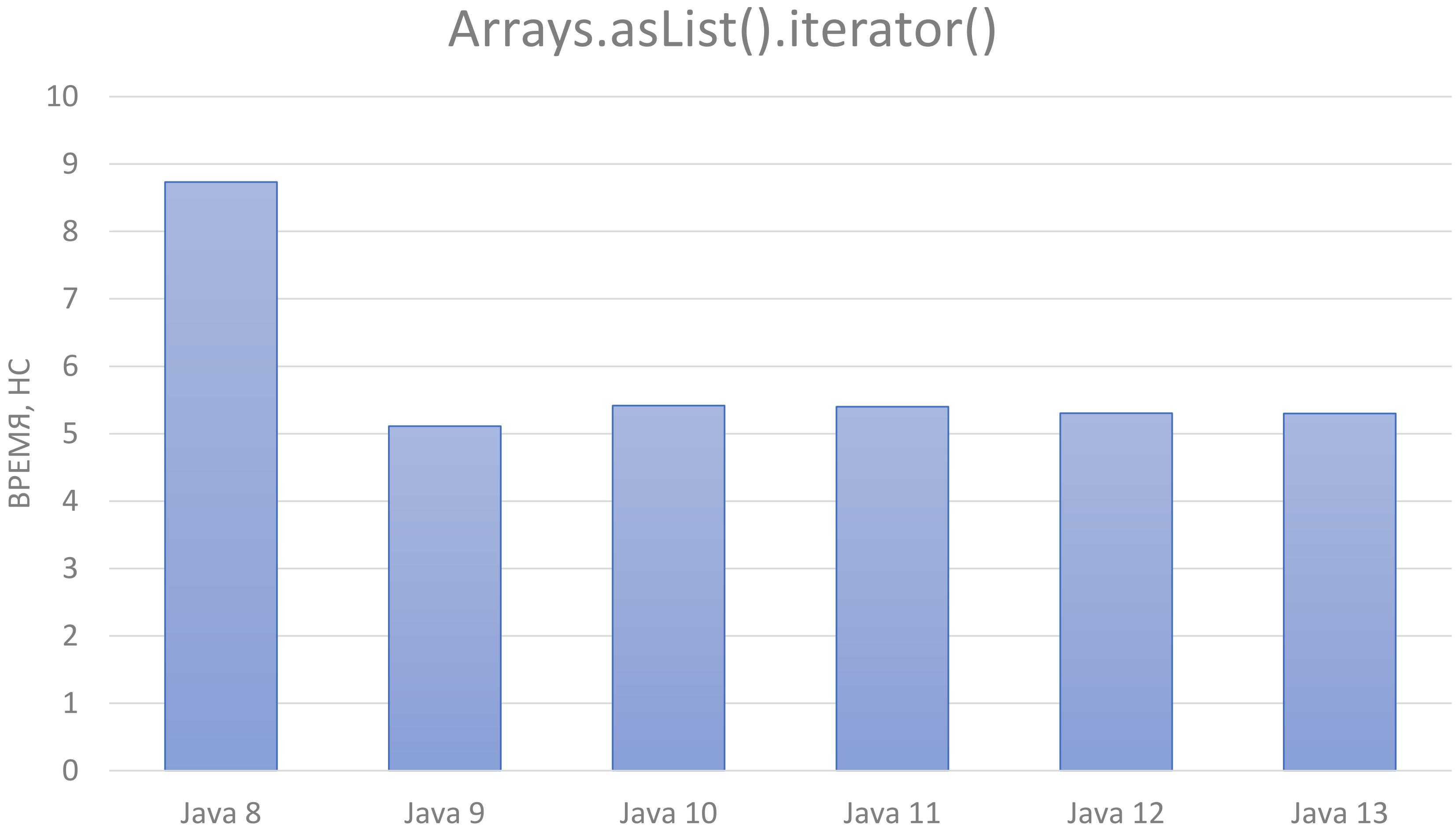
```
long arrayWithIterator() {
    final List<Integer> list = Arrays.asList( array );
    long sum = 0;
    for( final Integer n : list ) {
        sum += n.intValue();
    }
    return sum;
}

JDK: 1.8.0_77 (25.77-b03)

WITH_ITERATOR[SIZE:16]: 12 runs, 3000 ms each

#0: ~= 24.05 b/call (4410704296 bytes/183425024 calls)
#1: ~= 24.00 b/call (5220499248 bytes/217509888 calls)
#2: ~= 24.00 b/call (5583102000 bytes/232629248 calls)
#3: ~= 24.00 b/call (5351530544 bytes/222979072 calls)
#4: ~= 24.00 b/call (5391245136 bytes/224625664 calls)
...
```

https://youtu.be/K6c3W6vhQOA Руслан Черёмин — Escape Analysis и скаляризация

```
 @SafeVarargs
 @SuppressWarnings("varargs")
 public static <T> List<T> asList(T... a) {
     return new ArrayList<>(a);
 }

 /**
  * @serial include
  */
 private static class ArrayList<E> extends AbstractList<E>
     implements RandomAccess, java.io.Serializable
 {
     …

+    @Override // JDK-8155600
+    public Iterator<E> iterator() {
+        return new ArrayItr<>(a);
+    }
 }
```

```java
private static class ArrayItr<E> implements Iterator<E> {
    private int cursor;
    private final E[] a;

    ArrayItr(E[] a) {
        this.a = a;
    }

    @Override
    public boolean hasNext() {
        return cursor < a.length;
    }

    @Override
    public E next() {
        int i = cursor;
        if (i >= a.length) throw new NoSuchElementException();
        cursor = i + 1;
        return a[i];
    }
}
```

```java
private final String[] data = {"foo", "bar", "baz"};

@Benchmark
public int sum() {
 return sumList(Arrays.asList(data));
}

private int sumList(List<String> list) {
 int sum = 0;
 for (String s : list) {
   sum += s.length();
 }
 return sum;
}
```


Arrays.asList().iterator()
10
9
8
7
6
5
4
3
2
1
0
ВРЕМЯ, НС
Java 8
Java 9
Java 10
Java 11
Java 12
Java 13

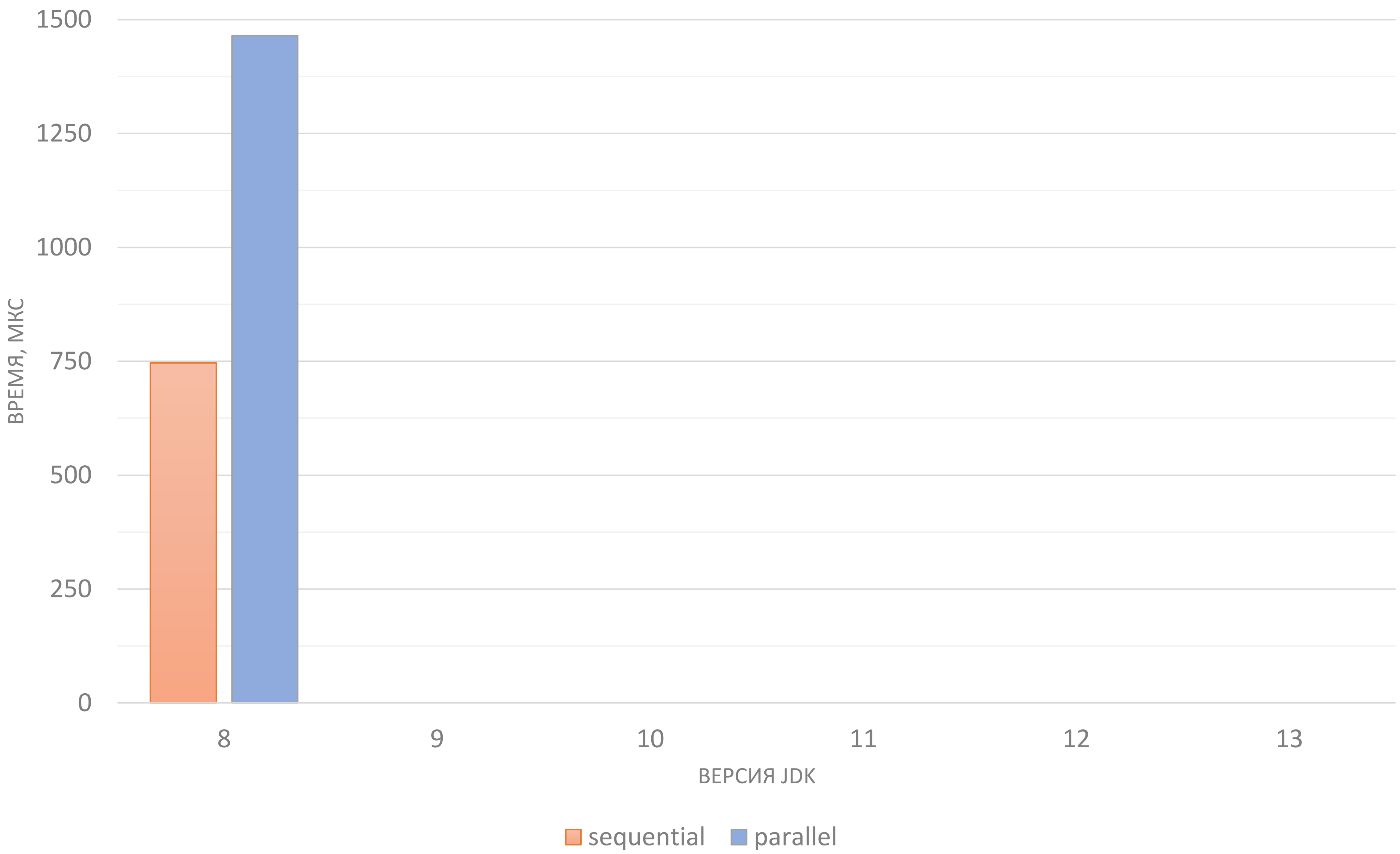
Properties.getProperty

Java 8

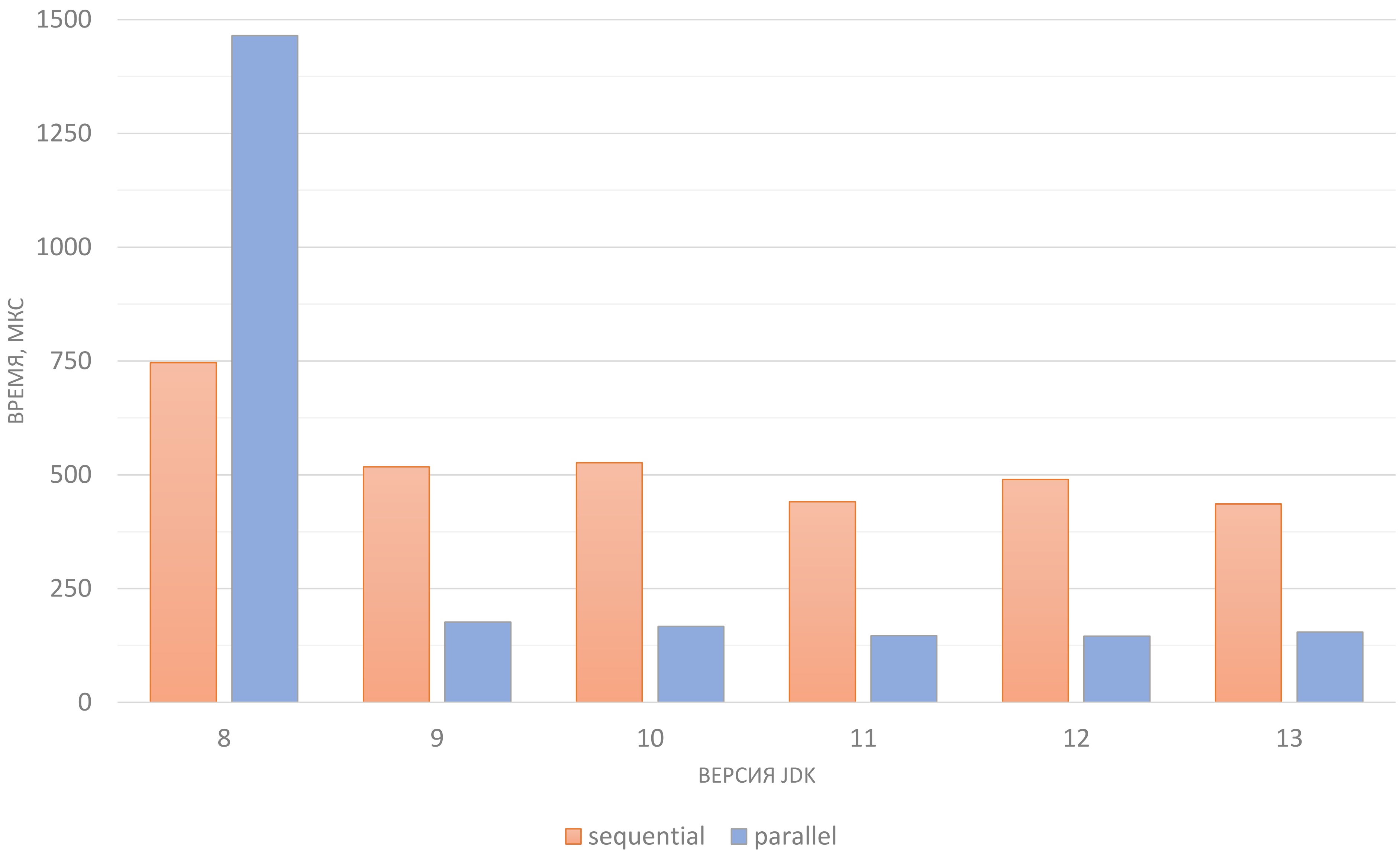
```
public
class Properties extends Hashtable<Object,Object> {
  …

  public String getProperty(String key) {
    Object oval = super.get(key);
    String sval = (oval instanceof String) ? (String)oval : null;
    return ((sval == null) && (defaults != null)) ?
           defaults.getProperty(key) : sval;
  }

  public synchronized Object setProperty(String key, String value) {
    return put(key, value);
  }
  …
}
```

```
public class Hashtable<K,V>
    extends Dictionary<K,V>
    implements Map<K,V>, Cloneable, java.io.Serializable {
  …

  public synchronized V get(Object key) {
    …
  }

  public synchronized V put(K key, V value) {
    …
  }
  …
}
```

```java
public class PropertiesGet {
    private Properties properties = IntStream.range(0, 10000).boxed()
            .collect(toMap(String::valueOf, i -> String.valueOf(i * i),
                    (a, b) -> a, Properties::new));

    @Benchmark
    public int sumOfSquares() {
        return IntStream.range(0, 10000)
                .map(x -> Integer.parseInt(properties.getProperty(String.valueOf(x))))
                .sum();
    }

    @Benchmark
    public int sumOfSquaresParallel() {
        return IntStream.range(0, 10000).parallel()
                .map(x -> Integer.parseInt(properties.getProperty(String.valueOf(x))))
                .sum();
    }
}
```



1500
1250
1000
750
500
250
0
ВРЕМЯ, МКС
8
9
10
11
12
13
ВЕРСИЯ JDK
sequential
parallel



1500
1250
1000
750
500
250
0
ВРЕМЯ, МКС
8
9
10
11
12
13
ВЕРСИЯ JDK
sequential
parallel

Java 9

```
public
class Properties extends Hashtable<Object,Object> {
  …

  /** JDK-8029891
   * Properties does not store values in its inherited Hashtable, but instead
   * in an internal ConcurrentHashMap.  Synchronization is omitted from
   * simple read operations.  Writes and bulk operations remain synchronized,
   * as in Hashtable.
   */
  private transient volatile ConcurrentHashMap<Object, Object> map;
  …
}
```

Java 9

```java
public
class Properties extends Hashtable<Object,Object> {
  …

  @Override
  public boolean containsKey(Object key) {
    return map.containsKey(key);
  }

  @Override
  public Object get(Object key) {
    return map.get(key);
  }

  @Override
  public synchronized Object put(Object key, Object value) {
    return map.put(key, value);
  }
  …
}
```

Java 9

```
public
class Properties extends Hashtable<Object,Object> {
  …
  private Properties(Properties defaults, int initialCapacity) {
    // use package-private constructor to
    // initialize unused fields with dummy values
    super((Void) null);
    map = new ConcurrentHashMap<>(initialCapacity);
    this.defaults = defaults;

    // Ensure writes can't be reordered
    UNSAFE.storeFence();
  }
  …
}
```

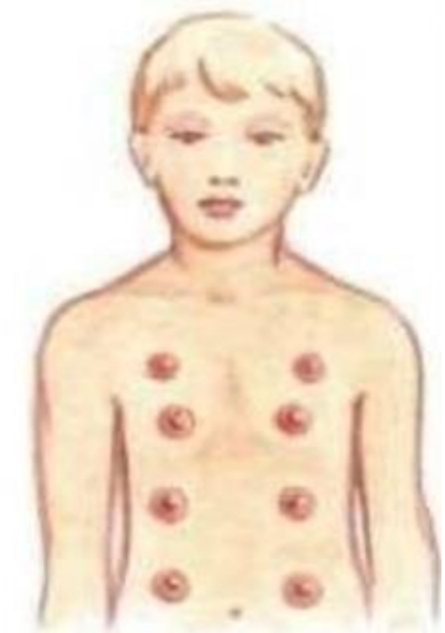
Java 9

```
public class Hashtable<K,V>
    extends Dictionary<K,V>
    implements Map<K,V>, Cloneable, java.io.Serializable {
  …
  /**
   * A constructor chained from {@link Properties} keeps Hashtable fields
   * uninitialized since they are not used.
   *
   * @param dummy a dummy parameter
   */
  Hashtable(Void dummy) {}
  …
}
```

Примеры атавизмов

Хвостатый мальчик

Волосатый человек



Многососковость

```
public
class Properties extends
Hashtable<Object,Object>
{
…
}
```

Родительский класс Properties

Collections hashCodes

java.util.Set<E> int **hashCode()**

Returns the hash code value for this set. The hash code of a set is defined to be the sum of the hash codes of the elements in the set, where the hash code of a null element is defined to be zero. This ensures that `s1.equals(s2)` implies that `s1.hashCode()==s2.hashCode()` for any two sets `s1` and `s2`, as required by the general contract of `Object.hashCode`.

java.util.List<E> int **hashCode()**

Returns the hash code value for this list. The hash code of a list is defined to be the result of the following calculation:

```
int hashCode = 1;
for (E e : list)
    hashCode = 31*hashCode + (e==null ? 0 : e.hashCode());
```

This ensures that `list1.equals(list2)` implies that `list1.hashCode()==list2.hashCode()` for any two lists, `list1` and `list2`, as required by the general contract of `Object.hashCode`.

```java
/**
 * The empty set (immutable).  This set is serializable.
 *
 * @see #emptySet()
 */
@SuppressWarnings("rawtypes")
public static final Set EMPTY_SET = new EmptySet<>();


private static class EmptySet<E>
    extends AbstractSet<E>
    implements Serializable
{
    …

}
```

Java 9

```
 /**
  * The empty set (immutable).  This set is serializable.
  *
  * @see #emptySet()
  */
 @SuppressWarnings("rawtypes")
 public static final Set EMPTY_SET = new EmptySet<>();


 private static class EmptySet<E>
     extends AbstractSet<E>
     implements Serializable
 {
     …

+    @Override
+    public int hashCode() { // JDK-8166365
+        return 0;
+    }
 }
```

```java
/**
 * Returns an immutable set containing only the specified object.
 * The returned set is serializable.
 *
 * @param  <T> the class of the objects in the set
 * @param o the sole object to be stored in the returned set.
 * @return an immutable set containing only the specified object.
 */
public static <T> Set<T> singleton(T o) {
    return new SingletonSet<>(o);
}

private static class SingletonSet<E>
    extends AbstractSet<E>
    implements Serializable
{
    …
}
```

Java 9

```
 /**
  * Returns an immutable set containing only the specified object.
  * The returned set is serializable.
  *
  * @param  <T> the class of the objects in the set
  * @param o the sole object to be stored in the returned set.
  * @return an immutable set containing only the specified object.
  */
 public static <T> Set<T> singleton(T o) {
     return new SingletonSet<>(o);
 }

 private static class SingletonSet<E>
     extends AbstractSet<E>
     implements Serializable
 {
     …
+    @Override
+    public int hashCode() { // JDK-8166365
+        return Objects.hashCode(element);
+    }
 }
```

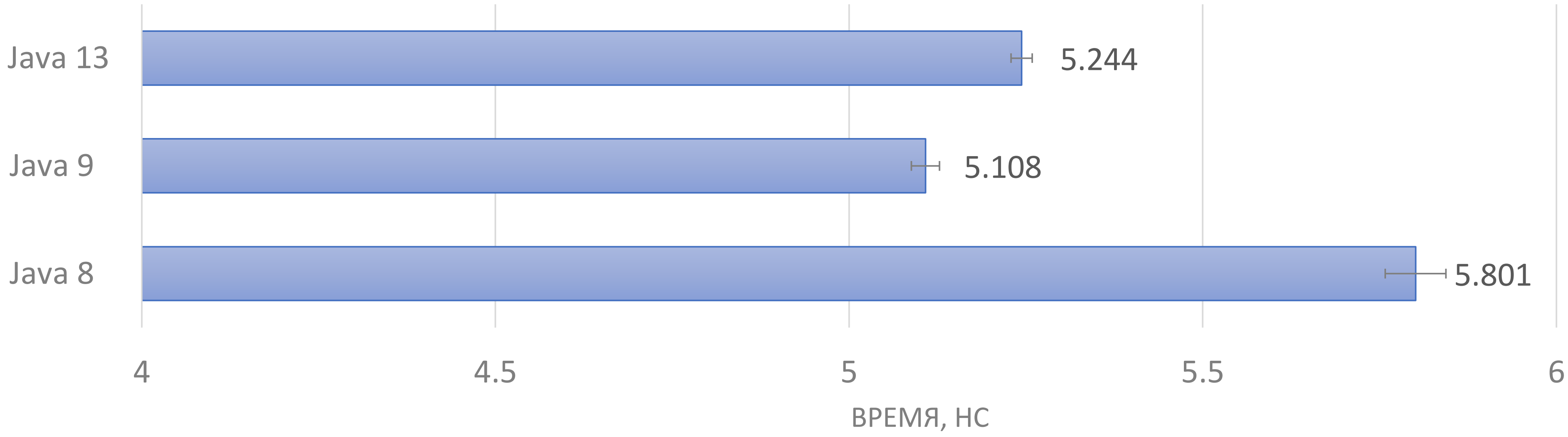
```java
/**
 * Returns an immutable list containing only the specified object.
 * The returned list is serializable.
 *
 * @param  <T> the class of the objects in the list
 * @param o the sole object to be stored in the returned list.
 * @return an immutable list containing only the specified object.
 * @since 1.3
 */
public static <T> List<T> singletonList(T o) {
    return new SingletonList<>(o);
}

private static class SingletonList<E>
    extends AbstractList<E>
    implements RandomAccess, Serializable {
    …
}
```

Java 9

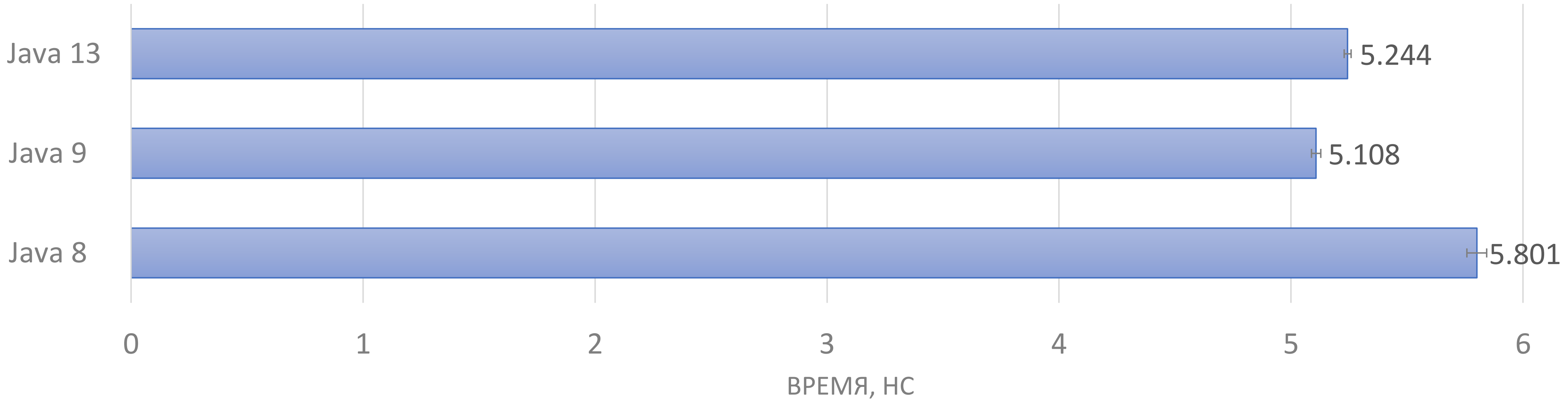
```
 /**
  * Returns an immutable list containing only the specified object.
  * The returned list is serializable.
  *
  * @param  <T> the class of the objects in the list
  * @param o the sole object to be stored in the returned list.
  * @return an immutable list containing only the specified object.
  * @since 1.3
  */
 public static <T> List<T> singletonList(T o) {
     return new SingletonList<>(o);
 }

 private static class SingletonList<E>
     extends AbstractList<E>
     implements RandomAccess, Serializable {
     …
+    @Override
+    public int hashCode() { // JDK-8166365
+        return 31 + Objects.hashCode(element);
+    }
 }
```

```
private Collection<?>[] collections = {
        Collections.emptySet(), Collections.singleton("foo"),
        Collections.singletonList("foo")
};

@Benchmark
public int hashCodes() {
    return collections[0].hashCode()+collections[1].hashCode()+
           collections[2].hashCode();
}
```

```
private Collection<?>[] collections = {
        Collections.emptySet(), Collections.singleton("foo"),
        Collections.singletonList("foo")
};

@Benchmark
public int hashCodes() {
    return collections[0].hashCode()+collections[1].hashCode()+
            collections[2].hashCode();
}
```



```
private Collection<?>[] collections = {
        Collections.emptySet(), Collections.singleton("foo"),
        Collections.singletonList("foo")
};

@Benchmark
public int hashCodes() {
    return collections[0].hashCode()+collections[1].hashCode()+
           collections[2].hashCode();
}
```



```java
public static <T> List<T> nCopies(int n, T o) {
    if (n < 0)
        throw new IllegalArgumentException("List length = " + n);
    return new CopiesList<>(n, o);
}
```

```java
public static <T> List<T> nCopies(int n, T o) {
    if (n < 0)
        throw new IllegalArgumentException("List length = " + n);
    return new CopiesList<>(n, o);
}

private static class CopiesList<E>
    extends AbstractList<E>
    implements RandomAccess, Serializable
{
    final int n;
    final E element;

    CopiesList(int n, E e) {
        assert n >= 0;
        this.n = n;
        element = e;
    }
    …
}
```

```
// AbstractList::hashCode
public int hashCode() {
    int hashCode = 1;
    for (E e : this)
        hashCode = 31*hashCode + (e==null ? 0 : e.hashCode());
    return hashCode;
}
```

```
// AbstractList::hashCode
public int hashCode() {
    int hashCode = 1;
    for (E e : this)
        hashCode = 31*hashCode + (e==null ? 0 : e.hashCode());
    return hashCode;
}
```

```
public int hashCode() {
    int hashCode = 1;
    final int elementHashCode = (element == null) ? 0 : element.hashCode();
    for (int i = 0; i < n; i++) {
        hashCode = 31*hashCode + elementHashCode;
    }
    return hashCode;
}
```

Zheka Kozlov

@ZhekaKozlov

Straight from the top of my DOM

http://mail.openjdk.java.net/pipermail/core-libs-dev/2018-November/056843.html

H – хэш-код коллекции
h – хэш-код элемента
n – число элементов

$$H\ = h + 31\big(h + 31 \ldots \big(h + 31(h + 31)\big) \ldots \big)$$
$$(n\ \text{раз})$$

H – хэш-код коллекции
h – хэш-код элемента
n – число элементов

$$H = h + 31\big(h + 31 \ldots \big(h + 31(h + 31)\big) \ldots \big)$$

$$(n \text{ раз})$$

$$H = 31^n + h \sum_{i=0}^{n-1} 31^i$$

H – хэш-код коллекции

h – хэш-код элемента

n – число элементов

$$H = h + 31\big(h + 31 \ldots \big(h + 31(h + 31)\big) \ldots \big)$$

$$(n \text{ раз})$$

$$H = 31^n + h \sum_{i=0}^{n-1} 31^i = 31^n + h\frac{31^n - 1}{31 - 1}$$

H – хэш-код коллекции
h – хэш-код элемента
n – число элементов

$$H = h + 31\big(h + 31 \ldots \big(h + 31(h + 31)\big) \ldots\big)$$

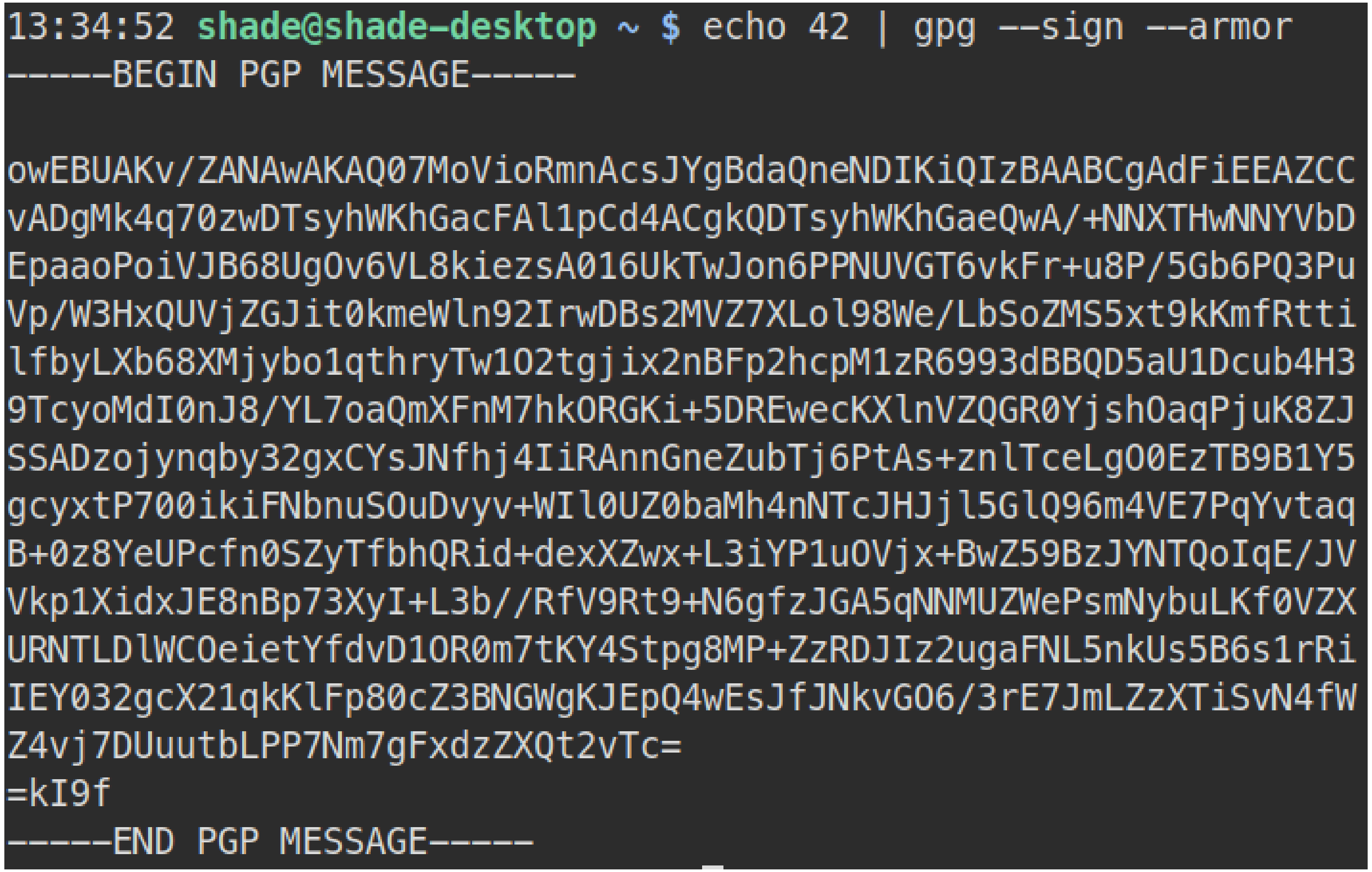
$$(n \text{ раз})$$

… но с переполнением

$$2147483647 + 1 = -2147483648$$

$\mathbb{Z}_{2^{32}} = \{0..2^{32} - 1;\ +; \times\}$ – кольцо вычетов по модулю 2^{32}

Signed integer



```
13:34:52 shade@shade-desktop ~ $ echo 42 | gpg --sign --armor
-----BEGIN PGP MESSAGE-----

owEBUAKv/ZANAwAKAQ07MoVioRmnAcsJYgBdaQneNDIKiQIzBAABCgAdFiEEAZCC
vADgMk4q70zwDTsyhWKhGacFAl1pCd4ACgkQDTsyhWKhGaeQwA/+NNXTHwNNYVbD
EpaaoPoiVJB68UgOv6VL8kiezsA016UkTwJon6PPNUVGT6vkFr+u8P/5Gb6PQ3Pu
Vp/W3HxQUVjZGJit0kmeWln92IrwDBs2MVZ7XLol98We/LbSoZMS5xt9kKmfRtti
lfbyLXb68XMjybo1qthryTw1O2tgjix2nBFp2hcpM1zR6993dBBQD5aU1Dcub4H3
9TcyoMdI0nJ8/YL7oaQmXFnM7hkORGKi+5DREwecKXlnVZQGR0YjshOaqPjuK8ZJ
SSADzojynqby32gxCYsJNfhj4IiRAnnGneZubTj6PtAs+znlTceLgO0EzTB9B1Y5
gcyxtP700ikiFNbnuSOuDvyv+WIl0UZ0baMh4nNTcJHJjl5GlQ96m4VE7PqYvtaq
B+0z8YeUPcfn0SZyTfbhQRid+dexXZwx+L3iYP1uOVjx+BwZ59BzJYNTQoIqE/JV
Vkp1XidxJE8nBp73XyI+L3b//RfV9Rt9+N6gfzJGA5qNNMUZWePsmNybuLKf0VZX
URNTLDlWCOeietYfdvD1OR0m7tKY4Stpg8MP+ZzRDJIz2ugaFNL5nkUs5B6s1rRi
IEY032gcX21qkKlFp80cZ3BNGWgKJEpQ4wEsJfJNkvGO6/3rE7JmLZzXTiSvN4fW
Z4vj7DUuutbLPP7Nm7gFxdzZXQt2vTc=
=kI9f
-----END PGP MESSAGE-----
```

Image: https://twitter.com/shipilev/status/1167400545952423936

$\mathbb{Z}_{2^{32}} = \{0..2^{32} - 1;\ +; \times\}$ – кольцо вычетов по модулю 2^{32}; беззнаковые числа

$\mathbb{S}_{2^{32}} = \{-2^{31} \dots 2^{31} - 1;\ +; \times\}$ – знаковые числа

$f\colon \mathbb{S}_{2^{32}} \to \mathbb{Z}_{2^{32}}$

$$f(x) = \begin{cases} x, x > 0 \\ x + 2^{32}, x \leq 0 \end{cases}$$

$$f^{-1}(x) = \begin{cases} x, x < 2^{31} \\ x - 2^{32}, x \geq 2^{31} \end{cases}$$

$\mathbb{Z}_{2^{32}} = \{0..2^{32} - 1;\ +; \times\}$ – кольцо вычетов по модулю 2^{32}; беззнаковые числа

$\mathbb{S}_{2^{32}} = \{-2^{31} \dots 2^{31} - 1;\ +; \times\}$ – знаковые числа

$f\colon \mathbb{S}_{2^{32}} \to \mathbb{Z}_{2^{32}}$ $\qquad f(x) = \begin{cases} x, x > 0 \\ x + 2^{32}, x \le 0 \end{cases}$ $\qquad f^{-1}(x) = \begin{cases} x, x < 2^{31} \\ x - 2^{32}, x \ge 2^{31} \end{cases}$

Теорема:

$\forall s, t \in \mathbb{S}_{2^{32}}$:

$$f(s +_{\mathbb{S}} t) = f(s) +_{\mathbb{Z}} f(t)$$

$$f(s \times_{\mathbb{S}} t) = f(s) \times_{\mathbb{Z}} f(t)$$

H – хэш-код коллекции

h – хэш-код элемента

n – число элементов

$$H = h + 31\big(h + 31 \ldots \big(h + 31(h + 31)\big) \ldots\big) \qquad (n \text{ раз})$$

$$H = 31^n + h \sum_{i=0}^{n-1} 31^i$$

H – хэш-код коллекции

h – хэш-код элемента

n – число элементов

$$H = h + 31\big(h + 31 \ldots \big(h + 31(h + 31)\big) \ldots\big) \qquad (n \text{ раз})$$

$$H = 31^n + h \sum_{i=0}^{n-1} 31^i$$

$$31^{2n} = (31^n)^2$$

$$31^{n+1} = 31 \cdot 31^n$$

H – хэш-код коллекции

h – хэш-код элемента

n – число элементов

$$H = h + 31\big(h + 31 \ldots \big(h + 31(h + 31)\big) \ldots \big) \qquad (n \text{ раз})$$

$$H = 31^n + h \sum_{i=0}^{n-1} 31^i$$

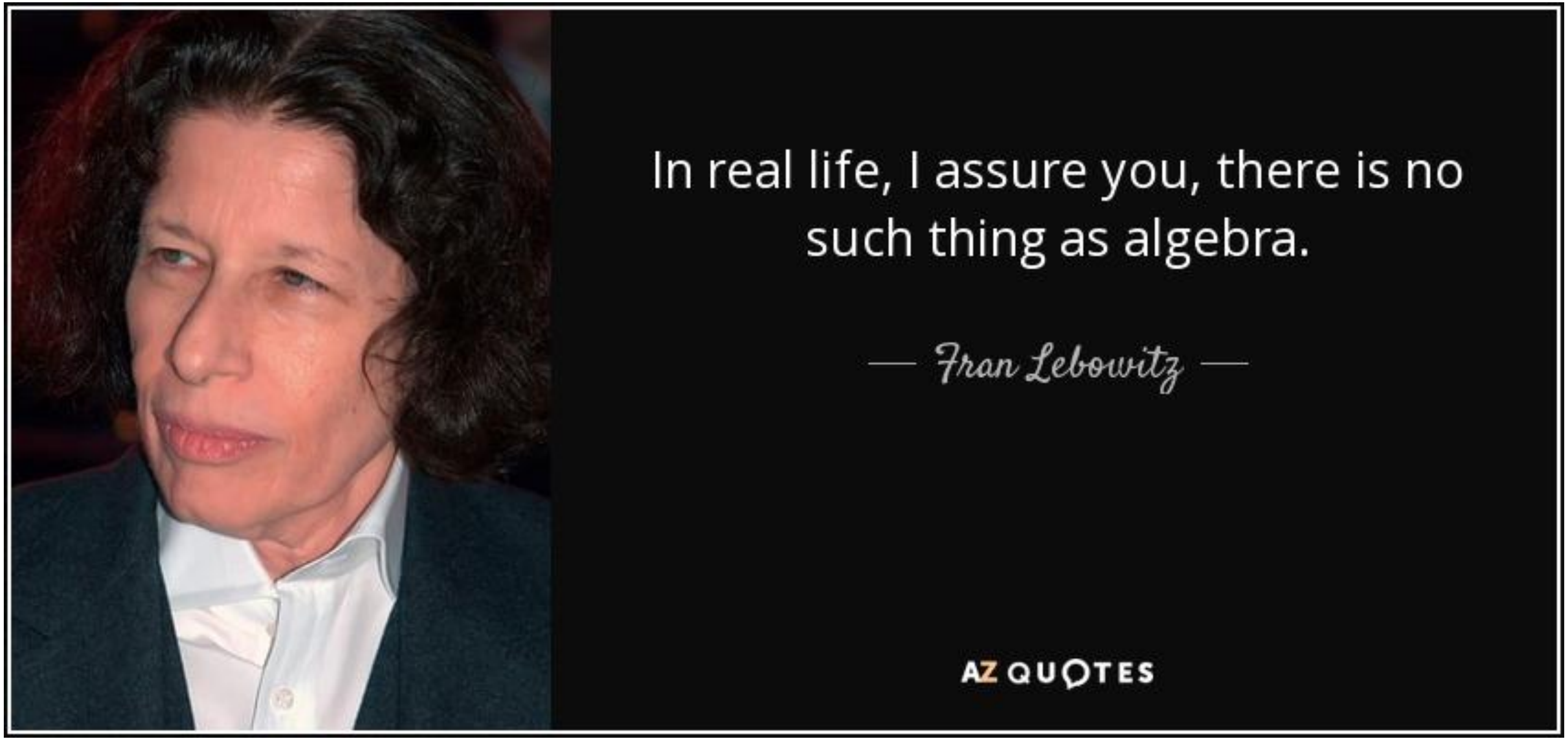
$$31^{2n} = (31^n)^2$$
$$31^{n+1} = 31 \cdot 31^n$$

$$\sum_{i=0}^{2n-1} 31^i = (31^n + 1) \sum_{i=0}^{n-1} 31^i$$

Java 13

```java
@Override
public int hashCode() {  // JDK-8214687
    if (n == 0) return 1;
    int pow = 31;
    int sum = 1;
    for (int i = Integer.numberOfLeadingZeros(n) + 1; i < Integer.SIZE; i++) {
        sum *= pow + 1;
        pow *= pow;
        if ((n << i) < 0) {
            pow *= 31;
            sum = sum * 31 + 1;
        }
    }
    return pow + sum * (element == null ? 0 : element.hashCode());
}
```

n	Zheka’s, ns	Optimized, ns
0	2,6	2,3
1	2,9	2,7
2	3,7	4,1
3	4,3	4,3
5	5,3	5,5
10	8,3	6,9
30	24,8	9,1
100	89,0	10,1
1000	923,8	14,1
10000	9157,4	17,0
100000	91705,6	20,8
1000000	919723,5	23,6

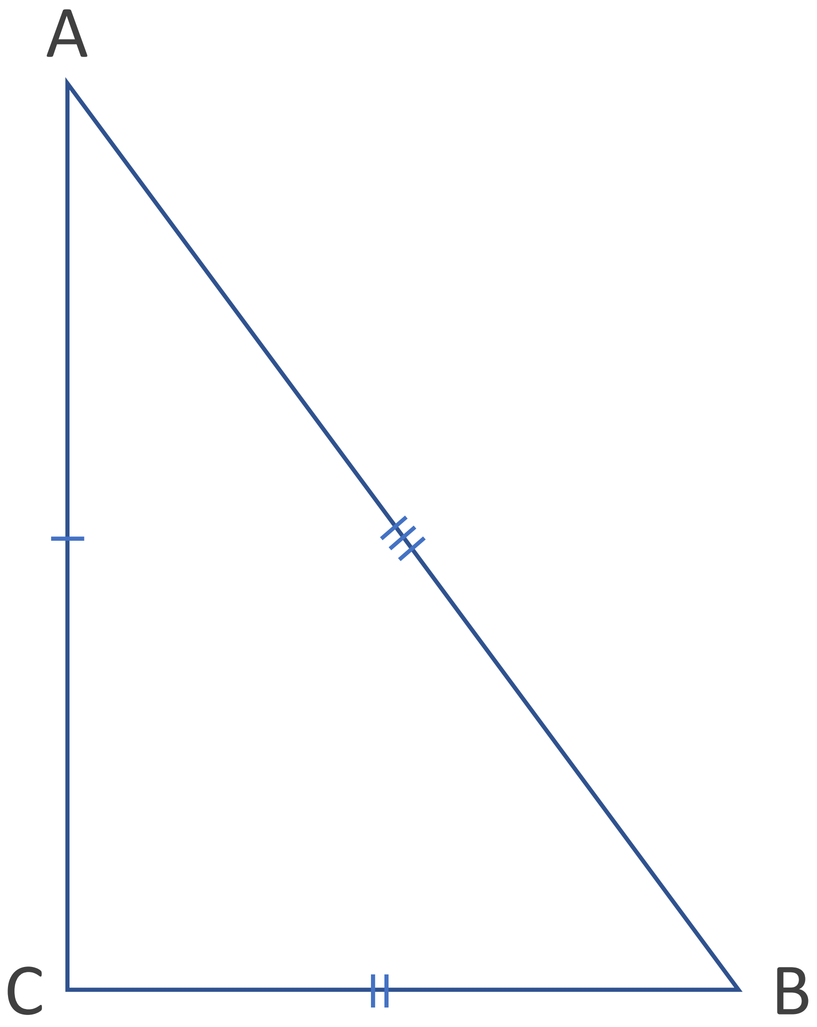


In real life, I assure you, there is no such thing as algebra.

— Fran Lebowitz —

AZ QUOTES

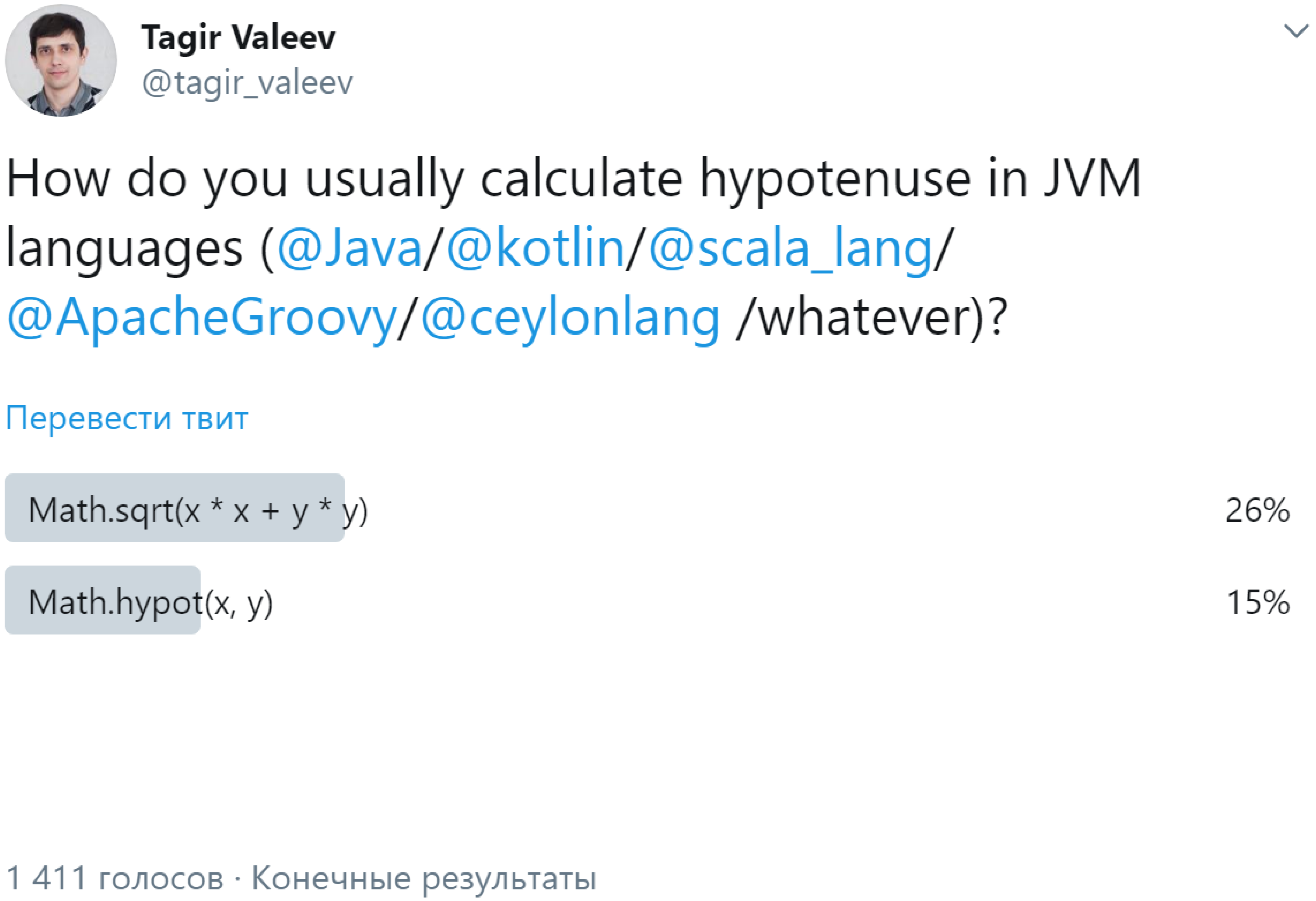
Math.hypot



```
|  Welcome to JShell -- Version 11
|  For an introduction type: /help intro

jshell> Math.hypot(3, 4)
$1 ==> 5.0

jshell> Math.sqrt(3 * 3 + 4 * 4)
$2 ==> 5.0
```

```
|  Welcome to JShell -- Version 11
|  For an introduction type: /help intro

jshell> Math.hypot(3, 4)
$1 ==> 5.0

jshell> Math.sqrt(3 * 3 + 4 * 4)
$2 ==> 5.0

jshell> Math.hypot(0.1, 0.2)
$3 ==> 0.22360679774997896

jshell> Math.sqrt(0.1 * 0.1 + 0.2 * 0.2)
$4 ==> 0.223606797749979
```

```
|  Welcome to JShell -- Version 11
|  For an introduction type: /help intro

jshell> Math.hypot(3, 4)
$1 ==> 5.0

jshell> Math.sqrt(3 * 3 + 4 * 4)
$2 ==> 5.0

jshell> Math.hypot(0.1, 0.2)
$3 ==> 0.22360679774997896

jshell> Math.sqrt(0.1 * 0.1 + 0.2 * 0.2)
$4 ==> 0.223606797749979

jshell> Math.hypot(1e200, 1e200)
$5 ==> 1.414213562373095E200

jshell> Math.sqrt(1e200 * 1e200 + 1e200 * 1e200)
$6 ==> Infinity
```



Tagir Valeev
@tagir_valeev

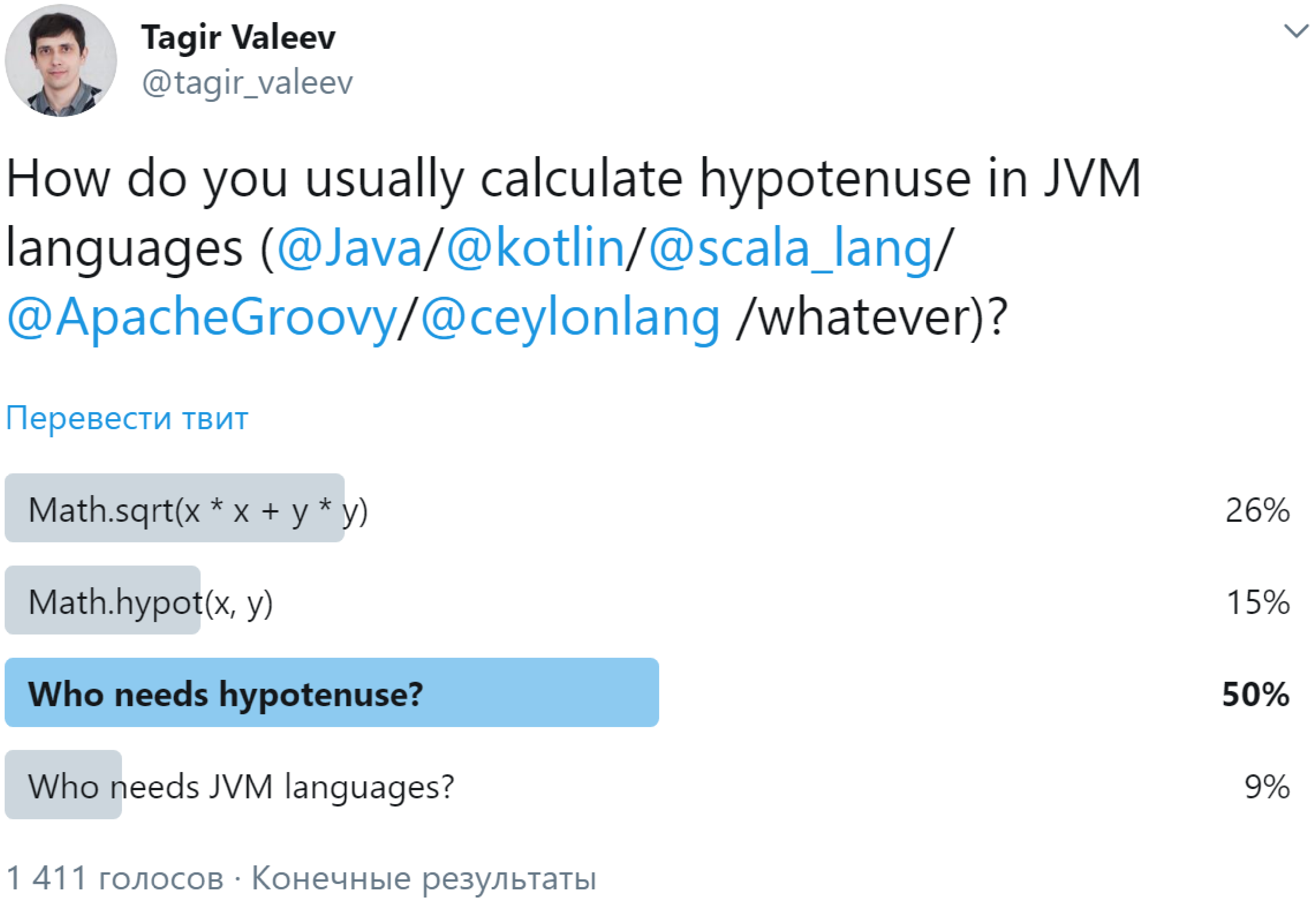
How do you usually calculate hypotenuse in JVM languages (@Java/@kotlin/@scala_lang/ @ApacheGroovy/@ceylonlang /whatever)?

Перевести твит

Math.sqrt(x * x + y * y) 26%

Math.hypot(x, y) 15%

1 411 голосов · Конечные результаты



Tagir Valeev
@tagir_valeev

How do you usually calculate hypotenuse in JVM languages (@Java/@kotlin/@scala_lang/ @ApacheGroovy/@ceylonlang /whatever)?

Перевести твит

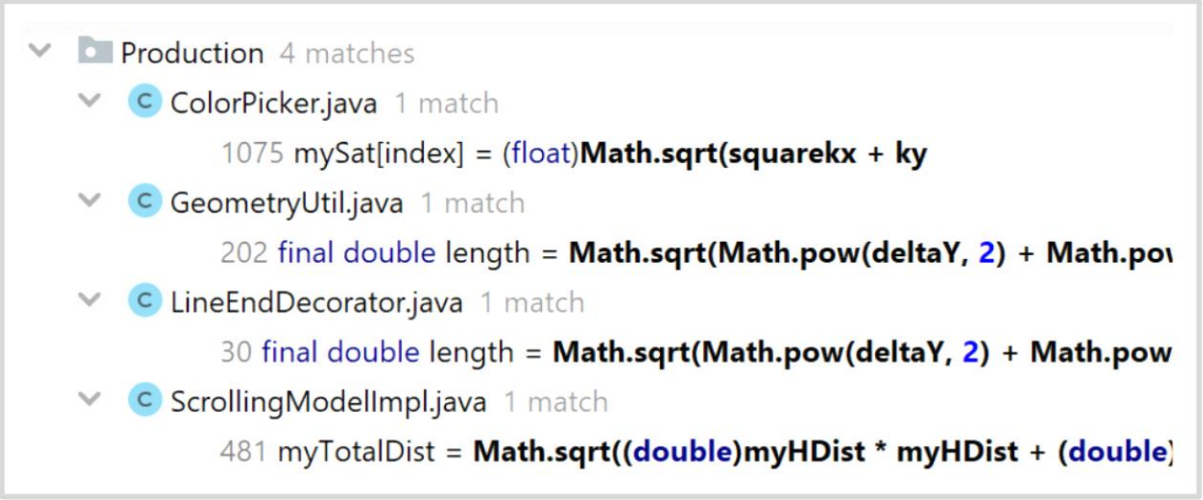
Math.sqrt(x * x + y * y)	26%
Math.hypot(x, y)	15%
Who needs hypotenuse?	**50%**
Who needs JVM languages?	9%

1 411 голосов · Конечные результаты

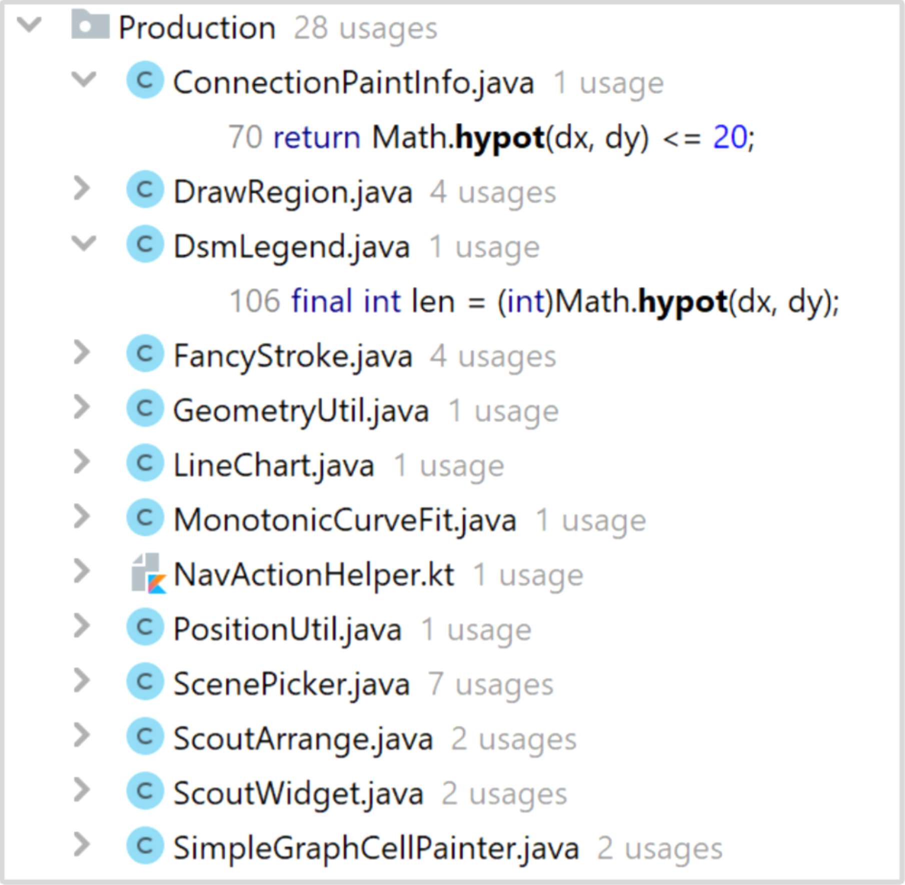
IntelliJ IDEA sources

```
Math.sqrt(x * x + y * y);
```

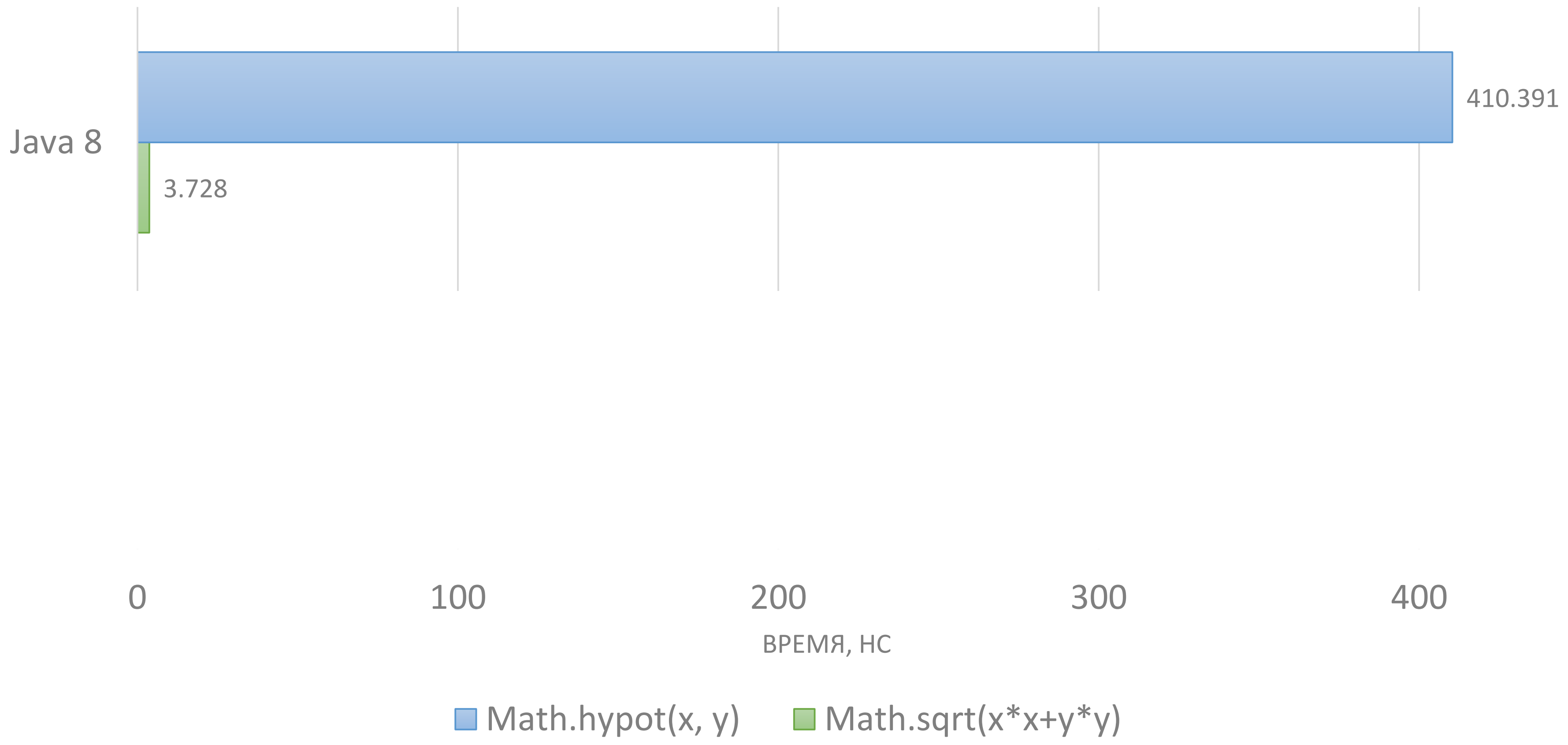
```
Math.hypot(x, y);
```

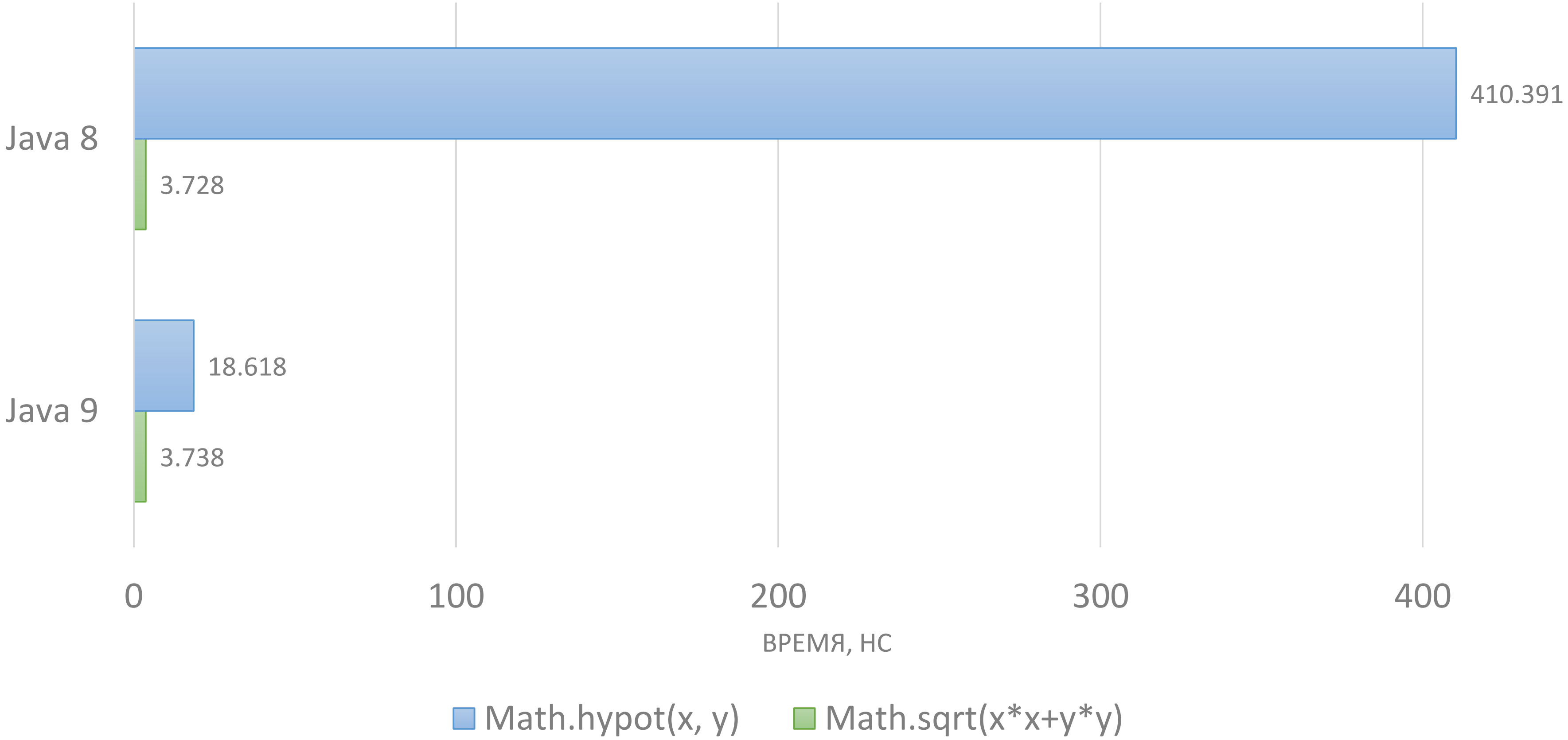


```
Production 4 matches
    ColorPicker.java 1 match
        1075 mySat[index] = (float)Math.sqrt(squarekx + ky
    GeometryUtil.java 1 match
        202 final double length = Math.sqrt(Math.pow(deltaY, 2) + Math.po
    LineEndDecorator.java 1 match
        30 final double length = Math.sqrt(Math.pow(deltaY, 2) + Math.pow
    ScrollingModelImpl.java 1 match
        481 myTotalDist = Math.sqrt((double)myHDist * myHDist + (double
```



```
Production 28 usages
    ConnectionPaintInfo.java 1 usage
        70 return Math.hypot(dx, dy) <= 20;
    DrawRegion.java 4 usages
    DsmLegend.java 1 usage
        106 final int len = (int)Math.hypot(dx, dy);
    FancyStroke.java 4 usages
    GeometryUtil.java 1 usage
    LineChart.java 1 usage
    MonotonicCurveFit.java 1 usage
    NavActionHelper.kt 1 usage
    PositionUtil.java 1 usage
    ScenePicker.java 7 usages
    ScoutArrange.java 2 usages
    ScoutWidget.java 2 usages
    SimpleGraphCellPainter.java 2 usages
```


Java 8
410.391
3.728
0
100
200
300
400
ВРЕМЯ, НС
Math.hypot(x, y)
Math.sqrt(x*x+y*y)

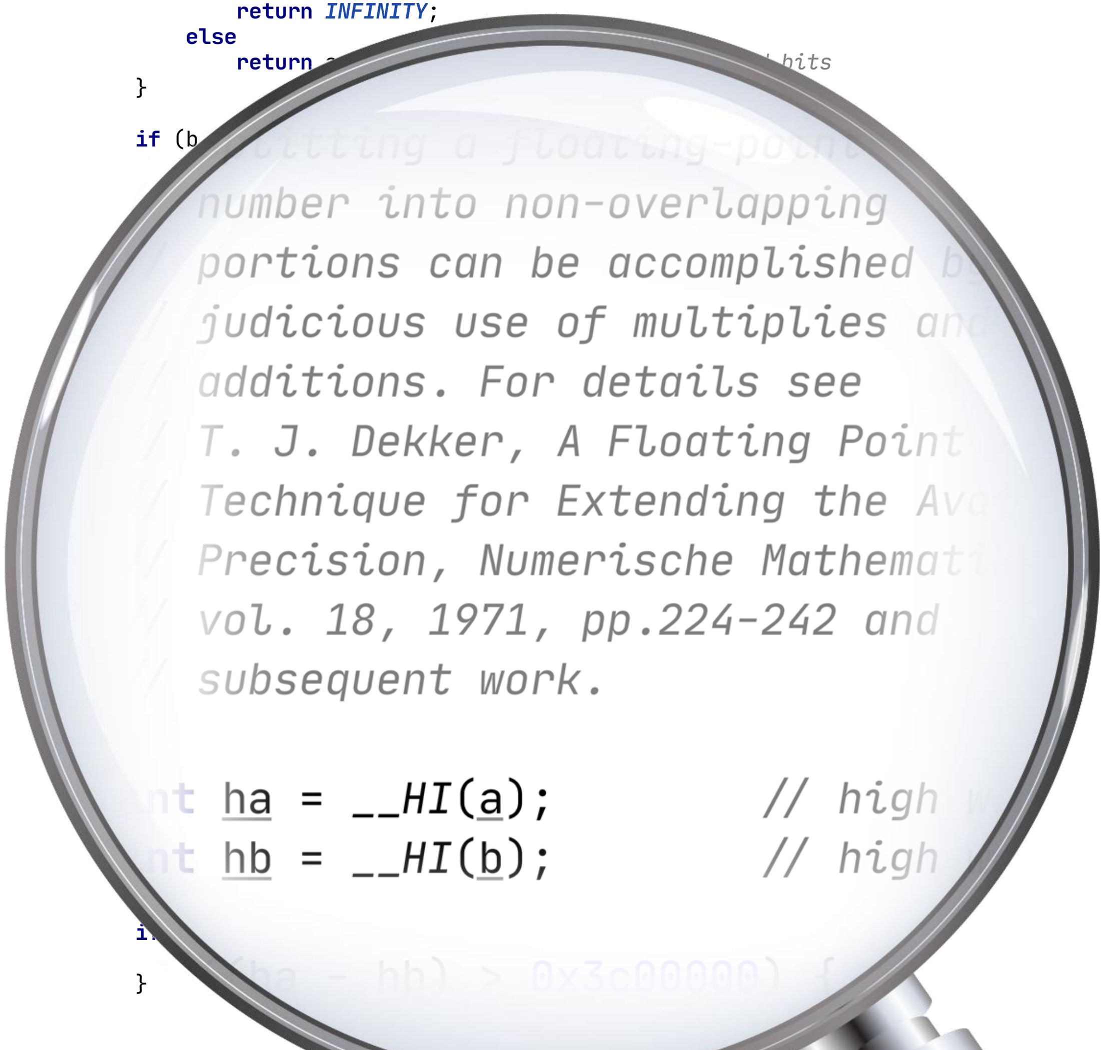

Java 8
410.391
3.728
Java 9
18.618
3.738
0
100
200
300
400
ВРЕМЯ, НС
Math.hypot(x, y)
Math.sqrt(x*x+y*y)

Java 9

```java
public static class Hypot {
    public static final double TWO_MINUS_600 = 0x1.0p-600;
    public static final double TWO_PLUS_600  = 0x1.0p+600;

    private Hypot() {
        throw new UnsupportedOperationException();
    }

    public static strictfp double compute(double x, double y) {
        double a = Math.abs(x);
        double b = Math.abs(y);

        if (!Double.isFinite(a) || !Double.isFinite(b)) {
            if (a == INFINITY || b == INFINITY)
                return INFINITY;
            else
                return a + b; // Propagate NaN significand bits
        }

        if (b > a) {
            double tmp = a;
            a = b;
            b = tmp;
        }
        assert a >= b;

        // Doing bitwise conversion after screening for NaN allows
        // the code to not worry about the possibility of
        // "negative" NaN values.

        // Note: the ha and hb variables are the high-order
        // 32-bits of a and b stored as integer values. The ha and
        // hb values are used first for a rough magnitude
        // comparison of a and b and second for simulating higher
        // precision by allowing a and b, respectively, to be
        // decomposed into non-overlapping portions. Both of these
        // uses could be eliminated. The magnitude comparison
        // could be eliminated by extracting and comparing the
        // exponents of a and b or just be performing a
        // floating-point divide.  Splitting a floating-point
        // number into non-overlapping portions can be
        // accomplished by judicious use of multiplies and
        // additions. For details see T. J. Dekker, A Floating
        // Point Technique for Extending the Available Precision ,
        // Numerische Mathematik, vol. 18, 1971, pp.224-242 and
        // subsequent work.

        int ha = __HI(a);        // high word of a
        int hb = __HI(b);        // high word of b

        if ((ha - hb) > 0x3c00000) {
            return a + b;  // x / y > 2**60
        }
        int k = 0;
        if (a > 0x1.00000_ffff_ffffp500) {   // a > ~2**500
            // scale a and b by 2**-600
            ha -= 0x25800000;
            hb -= 0x25800000;
            a = a * TWO_MINUS_600;
            b = b * TWO_MINUS_600;
            k += 600;
        }
        double t1, t2;
        if (b < 0x1.0p-500) {   // b < 2**-500
            if (b < Double.MIN_NORMAL) {      // subnormal b or 0 */
                if (b == 0.0)
                    return a;
                t1 = 0x1.0p1022;   // t1 = 2^1022
                b *= t1;
                a *= t1;
                k -= 1022;
            } else {            // scale a and b by 2^600
                ha += 0x25800000;       // a *= 2^600
                hb += 0x25800000;       // b *= 2^600
                a = a * TWO_PLUS_600;
                b = b * TWO_PLUS_600;
                k -= 600;
            }
        }
        // medium size a and b
        double w = a - b;
        if (w > b) {
            t1 = 0;
            t1 = __HI(t1, ha);
            t2 = a - t1;
            w  = Math.sqrt(t1*t1 - (b*(-b) - t2 * (a + t1)));
        } else {
            double y1, y2;
            a  = a + a;
            y1 = 0;
            y1 = __HI(y1, hb);
            y2 = b - y1;
            t1 = 0;
            t1 = __HI(t1, ha + 0x00100000);
            t2 = a - t1;
            w  = Math.sqrt(t1*y1 - (w*(-w) - (t1*y2 + t2*b)));
        }
        if (k != 0) {
            return Math.powerOfTwoD(k) * w;
        } else
            return w;
    }
}
```

Java 9

```
public static class Hypot {
    public static final double TWO_MINUS_600 = 0x1.0p-600;
    public static final double TWO_PLUS_600  = 0x1.0p+600;

    private Hypot() {
        throw new UnsupportedOperationException();
    }

    public static strictfp double compute(double x, double y) {
        double a = Math.abs(x);
        double b = Math.abs(y);

        if (!Double.isFinite(a) || !Double.isFinite(b)) {
            if (a == INFINITY || b == INFINITY)
                return INFINITY;
            else
                return a                                  bits
        }

        if (b

        i

        }
```



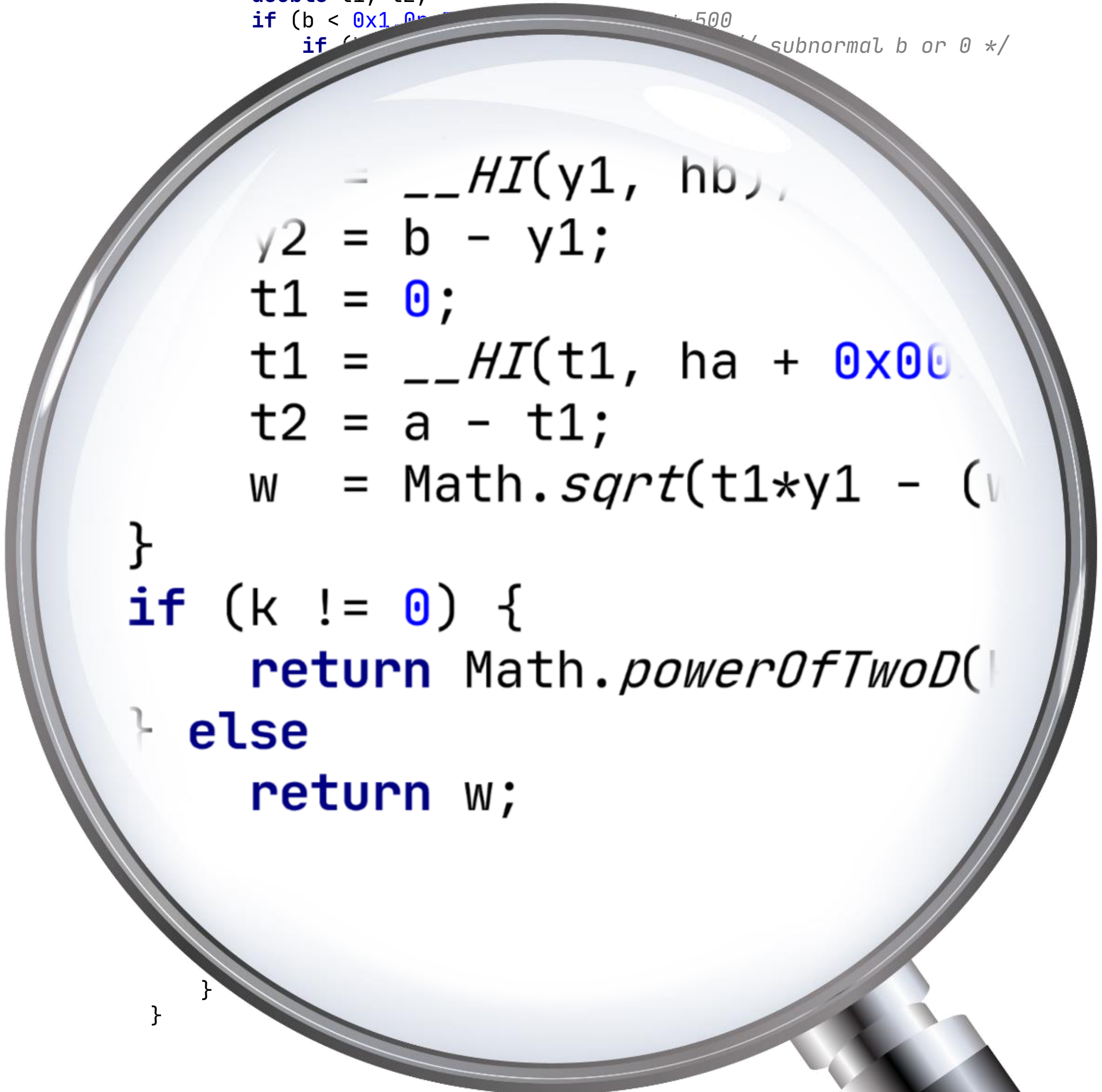
```
   itting a floating-point
// number into non-overlapping
// portions can be accomplished b
// judicious use of multiplies an
// additions. For details see
// T. J. Dekker, A Floating Point
// Technique for Extending the Av
// Precision, Numerische Mathemat
// vol. 18, 1971, pp.224-242 and
// subsequent work.

nt ha = __HI(a);        // high
nt hb = __HI(b);        // high

 (ha - hb) > 0x3c00000) {
```

```
        int k = 0;
        if (a > 0x1.00000_ffff_ffffp500) {   // a > ~2**500
            // scale a and b by 2**-600
            ha -= 0x25800000;
            hb -= 0x25800000;
            a = a * TWO_MINUS_600;
            b = b * TWO_MINUS_600;
            k += 600;
        }
        double t1, t2;
        if (b < 0x1.0p-500) {   // b < 2**-500
            if (b < Double.MIN_NORMAL) {      // subnormal b or 0 */
                if (b == 0.0)
                    return a;
                t1 = 0x1.0p1022;   // t1 = 2^1022
                b *= t1;
                a *= t1;
                k -= 1022;
            } else {                // scale a and b by 2^600
                ha += 0x25800000;       // a *= 2^600
                hb += 0x25800000;       // b *= 2^600
                a = a * TWO_PLUS_600;
                b = b * TWO_PLUS_600;
                k -= 600;
            }
        }
        // medium size a and b
        double w = a - b;
        if (w > b) {
            t1 = 0;
            t1 = __HI(t1, ha);
            t2 = a - t1;
            w  = Math.sqrt(t1*t1 - (b*(-b) - t2 * (a + t1)));
        } else {
            double y1, y2;
            a  = a + a;
            y1 = 0;
            y1 = __HI(y1, hb);
            y2 = b - y1;
            t1 = 0;
            t1 = __HI(t1, ha + 0x00100000);
            t2 = a - t1;
            w  = Math.sqrt(t1*y1 - (w*(-w) - (t1*y2 + t2*b)));
        }
        if (k != 0) {
            return Math.powerOfTwoD(k) * w;
        } else
            return w;
    }
}
```

Java 9

```java
public static class Hypot {
    public static final double TWO_MINUS_600 = 0x1.0p-600;
    public static final double TWO_PLUS_600  = 0x1.0p+600;

    private Hypot() {
        throw new UnsupportedOperationException();
    }

    public static strictfp double compute(double x, double y) {
        double a = Math.abs(x);
        double b = Math.abs(y);

        if (!Double.isFinite(a) || !Double.isFinite(b)) {
            if (a == INFINITY || b == INFINITY)
                return INFINITY;
            else
                return a + b; // Propagate NaN significand bits
        }

        if (b > a) {
            double tmp = a;
            a = b;
            b = tmp;
        }
        assert a >= b;

        // Doing bitwise conversion after screening for NaN allows
        // the code to not worry about the possibility of
        // "negative" NaN values.

        // Note: the ha and hb variables are the high-order
        // 32-bits of a and b stored as integer values. The ha and
        // hb values are used first for a rough magnitude
        // comparison of a and b and second for simulating higher
        // precision by allowing a and b, respectively, to be
        // decomposed into non-overlapping portions. Both of these
        // uses could be eliminated. The magnitude comparison
        // could be eliminated by extracting and comparing the
        // exponents of a and b or just be performing a
        // floating-point divide.  Splitting a floating-point
        // number into non-overlapping portions can be
        // accomplished by judicious use of multiplies and
        // additions. For details see T. J. Dekker, A Floating
        // Point Technique for Extending the Available Precision ,
        // Numerische Mathematik, vol. 18, 1971, pp.224-242 and
        // subsequent work.

        int ha = __HI(a);        // high word of a
        int hb = __HI(b);        // high word of b

        if ((ha - hb) > 0x3c00000) {
            return a + b;  // x / y > 2**60
        }
```

```java
        int k = 0;
        if (a > 0x1.00000_ffff_ffffp500) {   // a > ~2**500
            // scale a and b by 2**-600
            ha -= 0x25800000;
            hb -= 0x25800000;
            a = a * TWO_MINUS_600;
            b = b * TWO_MINUS_600;
            k += 600;
        }
        double t1, t2;
        if (b < 0x1.0p                    500
            if (                               // subnormal b or 0 */
```



```java
        = __HI(y1, hb);
    y2 = b - y1;
    t1 = 0;
    t1 = __HI(t1, ha + 0x00
    t2 = a - t1;
    w  = Math.sqrt(t1*y1 - (
}
if (k != 0) {
    return Math.powerOfTwoD(
} else
    return w;
```

```java
    }
}
```

Java 8

```
public static native double hypot(double x, double y);
```

Math.abs

—

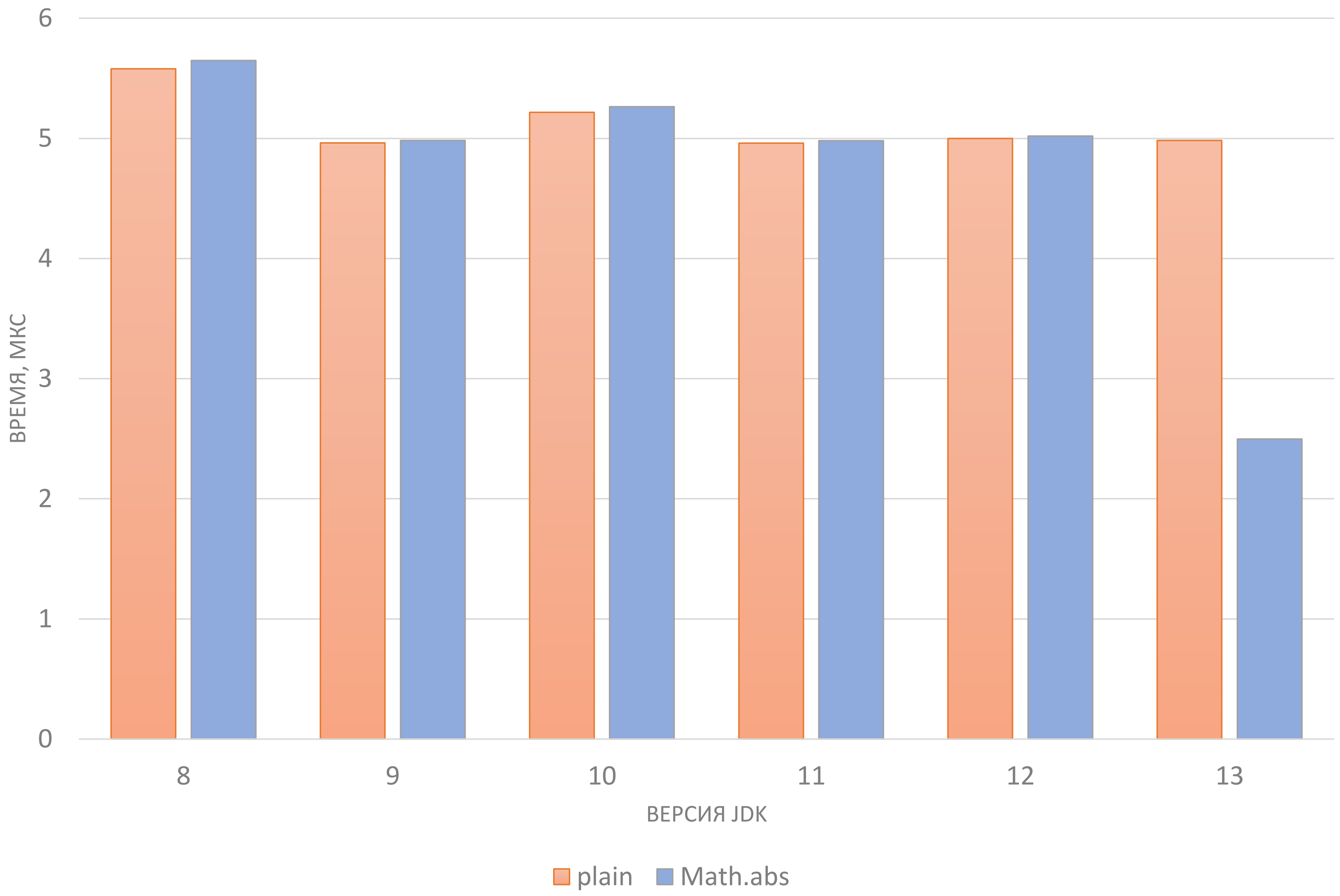
Image: http://rl337.org/2012/07/31/in-java-when-is-math-abs-negative/

```java
/**
 * Returns the absolute value of an {@code int} value.
 * If the argument is not negative, the argument is returned.
 * If the argument is negative, the negation of the argument is returned.
 *
 * <p>Note that if the argument is equal to the value of
 * {@link Integer#MIN_VALUE}, the most negative representable
 * {@code int} value, the result is that same value, which is
 * negative.
 *
 * @param   a   the argument whose absolute value is to be determined
 * @return  the absolute value of the argument.
 */
public static int abs(int a) {
    return (a < 0) ? -a : a;
}
```

```java
static int sumViaAbs(int[] data) {
    int result = 0;
    for (int x : data) {
        result += Math.abs(x);
    }
    return result;
}
```

```java
static int sumViaAbs(int[] data) {
    int result = 0;
    for (int x : data) {
        result += Math.abs(x);
    }
    return result;
}

static int sumPlain(int[] data) {
    int result = 0;
    for (int x : data) {
        result += x < 0 ? -x : x;
    }
    return result;
}
```

```java
static int sumViaAbs(int[] data) {
    int result = 0;
    for (int x : data) {
        result += Math.abs(x);
    }
    return result;
}

static int sumPlain(int[] data) {
    int result = 0;
    for (int x : data) {
        result += x < 0 ? -x : x;
    }
    return result;
}

int[] data = new SplittableRandom(0).ints(10_000).toArray();
```


6
5
4
3
2
1
0
ВРЕМЯ, МКС
8
9
10
11
12
13
ВЕРСИЯ JDK
plain
Math.abs

```
:loop
vmovq   rdx,xmm0
mov     r10d,dword ptr [rdx+rbp*4+2ch]
mov     r11d,dword ptr [rdx+rbp*4+28h]
mov     r9d,dword ptr [rdx+rbp*4+24h]
mov     r8d,dword ptr [rdx+rbp*4+20h]
mov     ebx,dword ptr [rdx+rbp*4+1ch]
mov     edi,dword ptr [rdx+rbp*4+18h]
mov     esi,dword ptr [rdx+rbp*4+14h]
mov     r14d,dword ptr [rdx+rbp*4+10h]
vmovq   xmm0,rdx
mov     edx,r10d
neg     edx
test    r10d,r10d
cmovl   r10d,edx
mov     edx,r14d
neg     edx
test    r14d,r14d
cmovl   r14d,edx
add     eax,r14d
mov     r14d,r11d
neg     r14d
test    r11d,r11d
cmovl   r11d,r14d
mov     edx,r9d
neg     edx
test    r9d,r9d
```

```
cmovl   r9d,edx
mov     r14d,r8d
neg     r14d
test    r8d,r8d
cmovl   r8d,r14d
mov     edx,esi
neg     edx
test    esi,esi
cmovl   esi,edx
add     eax,esi
mov     edx,ebx
neg     edx
test    ebx,ebx
cmovl   ebx,edx
mov     esi,edi
neg     esi
test    edi,edi
cmovl   edi,esi
add     eax,edi
add     eax,ebx
add     eax,r8d
add     eax,r9d
add     eax,r11d
add     eax,r10d
add     ebp,8h
cmp     ebp,ecx
jl      loop
```

Начало цикла

```
:loop
vmovq   rdx,xmm0
mov     r10d,dword ptr [rdx+rbp*4+2ch]
mov     r11d,dword ptr [rdx+rbp*4+28h]
mov     r9d,dword ptr [rdx+rbp*4+24h]
mov     r8d,dword ptr [rdx+rbp*4+20h]
mov     ebx,dword ptr [rdx+rbp*4+1ch]
mov     edi,dword ptr [rdx+rbp*4+18h]
mov     esi,dword ptr [rdx+rbp*4+14h]
mov     r14d,dword ptr [rdx+rbp*4+10h]
vmovq   xmm0,rdx
```

data[rbp+7] → r10d
data[rbp+6] → r11d
data[rbp+5] → r9d
data[rbp+4] → r8d
data[rbp+3] → ebx
data[rbp+2] → edi
data[rbp+1] → esi
data[rbp+0] → r14d

Восемь раз для каждого значения

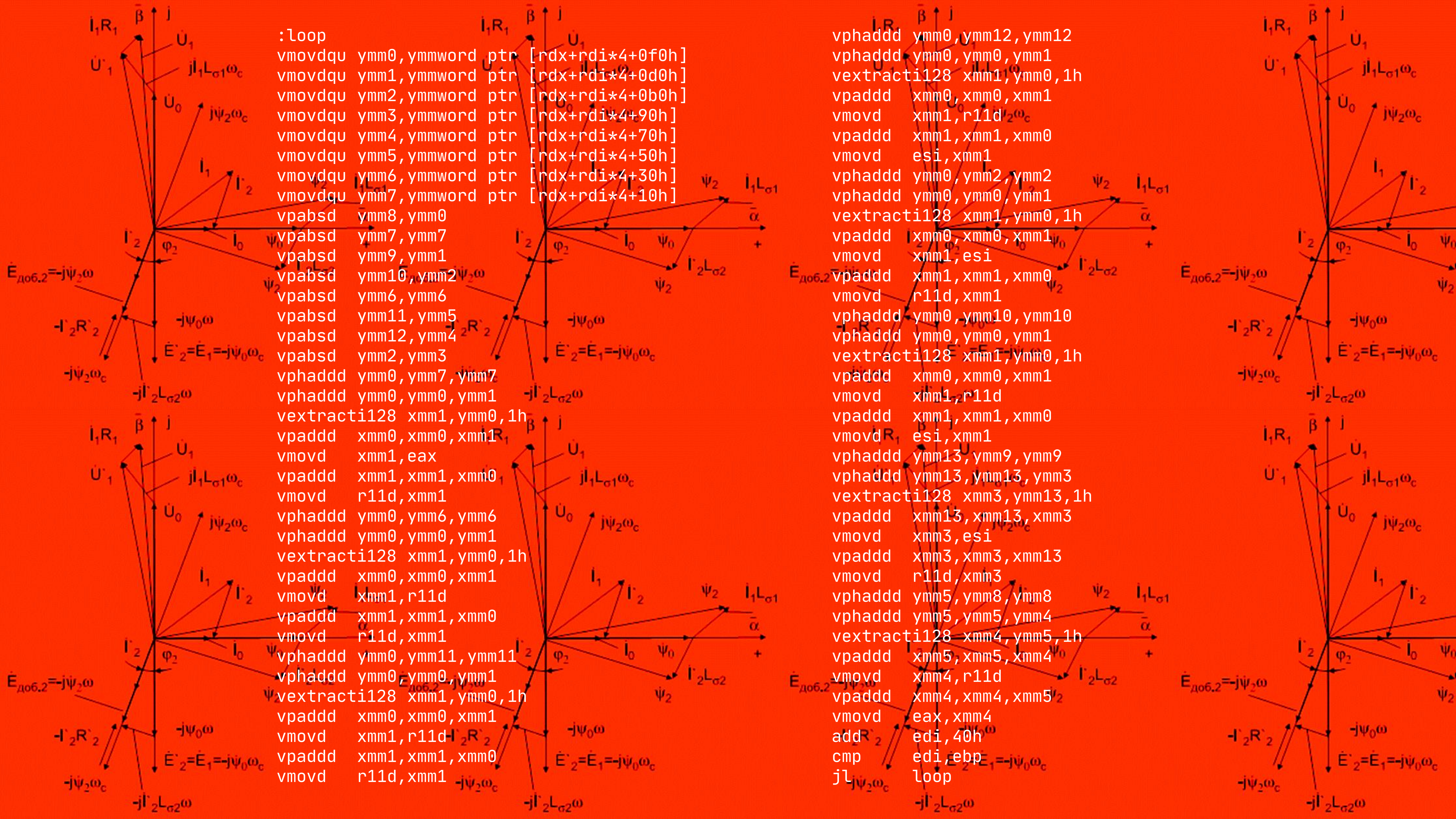
```
mov     r14d,r11d
neg     r14d
test    r11d,r11d
cmovl   r11d,r14d
add     eax,r11d
```

Копируем r11d → r14d
Инвертируем r14d
Проверяем r11d
Если меньше 0, то r14d → r11d
Добавляем результат в eax

```
add      ebp,8h
cmp      ebp,ecx
jl       loop
```

ebp+=8

Если меньше ecx, то идём назад



```
:loop
vmovdqu ymm0,ymmword ptr [rdx+rdi*4+0f0h]
vmovdqu ymm1,ymmword ptr [rdx+rdi*4+0d0h]
vmovdqu ymm2,ymmword ptr [rdx+rdi*4+0b0h]
vmovdqu ymm3,ymmword ptr [rdx+rdi*4+90h]
vmovdqu ymm4,ymmword ptr [rdx+rdi*4+70h]
vmovdqu ymm5,ymmword ptr [rdx+rdi*4+50h]
vmovdqu ymm6,ymmword ptr [rdx+rdi*4+30h]
vmovdqu ymm7,ymmword ptr [rdx+rdi*4+10h]
vpabsd  ymm8,ymm0
vpabsd  ymm7,ymm7
vpabsd  ymm9,ymm1
vpabsd  ymm10,ymm2
vpabsd  ymm6,ymm6
vpabsd  ymm11,ymm5
vpabsd  ymm12,ymm4
vpabsd  ymm2,ymm3
vphaddd ymm0,ymm7,ymm7
vphaddd ymm0,ymm0,ymm1
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,eax
vpaddd  xmm1,xmm1,xmm0
vmovd   r11d,xmm1
vphaddd ymm0,ymm6,ymm6
vphaddd ymm0,ymm0,ymm1
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,r11d
vpaddd  xmm1,xmm1,xmm0
vmovd   r11d,xmm1
vphaddd ymm0,ymm11,ymm11
vphaddd ymm0,ymm0,ymm1
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,r11d
vpaddd  xmm1,xmm1,xmm0
vmovd   r11d,xmm1
```

```
vphaddd ymm0,ymm12,ymm12
vphaddd ymm0,ymm0,ymm1
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,r11d
vpaddd  xmm1,xmm1,xmm0
vmovd   esi,xmm1
vphaddd ymm0,ymm2,ymm2
vphaddd ymm0,ymm0,ymm1
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,esi
vpaddd  xmm1,xmm1,xmm0
vmovd   r11d,xmm1
vphaddd ymm0,ymm10,ymm10
vphaddd ymm0,ymm0,ymm1
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,r11d
vpaddd  xmm1,xmm1,xmm0
vmovd   esi,xmm1
vphaddd ymm13,ymm9,ymm9
vphaddd ymm13,ymm13,ymm3
vextracti128 xmm3,ymm13,1h
vpaddd  xmm13,xmm13,xmm3
vmovd   xmm3,esi
vpaddd  xmm3,xmm3,xmm13
vmovd   r11d,xmm3
vphaddd ymm5,ymm8,ymm8
vphaddd ymm5,ymm5,ymm4
vextracti128 xmm4,ymm5,1h
vpaddd  xmm5,xmm5,xmm4
vmovd   xmm4,r11d
vpaddd  xmm4,xmm4,xmm5
vmovd   eax,xmm4
add     edi,40h
cmp     edi,ebp
jl      loop
```

Начало цикла

```
:loop
vmovdqu ymm0,ymmword ptr [rdx+rdi*4+0f0h]
vmovdqu ymm1,ymmword ptr [rdx+rdi*4+0d0h]
vmovdqu ymm2,ymmword ptr [rdx+rdi*4+0b0h]
vmovdqu ymm3,ymmword ptr [rdx+rdi*4+90h]
vmovdqu ymm4,ymmword ptr [rdx+rdi*4+70h]
vmovdqu ymm5,ymmword ptr [rdx+rdi*4+50h]
vmovdqu ymm6,ymmword ptr [rdx+rdi*4+30h]
vmovdqu ymm7,ymmword ptr [rdx+rdi*4+10h]
```

data[rbp+56..63] → ymm0
data[rbp+48..55] → ymm1
data[rbp+40..47] → ymm2
data[rbp+32..39] → ymm3
data[rbp+24..31] → ymm4
data[rbp+16..23] → ymm5
data[rbp+8..15] → ymm6
data[rbp+0..7] → ymm7

```
vpabsd  ymm8,ymm0
vpabsd  ymm7,ymm7
vpabsd  ymm9,ymm1
vpabsd  ymm10,ymm2
vpabsd  ymm6,ymm6
vpabsd  ymm11,ymm5
vpabsd  ymm12,ymm4
vpabsd  ymm2,ymm3
```

Abs(ymm0) → ymm8

Abs(ymm7) → ymm7

Abs(ymm1) → ymm9

Abs(ymm2) → ymm10

Abs(ymm6) → ymm6

Abs(ymm5) → ymm11

Abs(ymm4) → ymm12

Abs(ymm3) → ymm2

Восемь раз для каждого значения

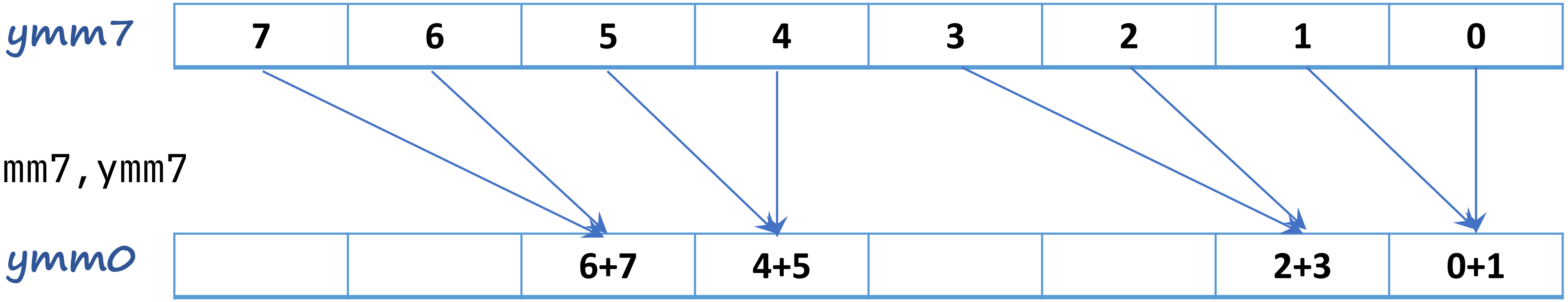
```
vphaddd ymm0,ymm7,ymm7
vphaddd ymm0,ymm0,ymm1
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,eax
vpaddd  xmm1,xmm1,xmm0
vmovd   r11d,xmm1
```

ymm7	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

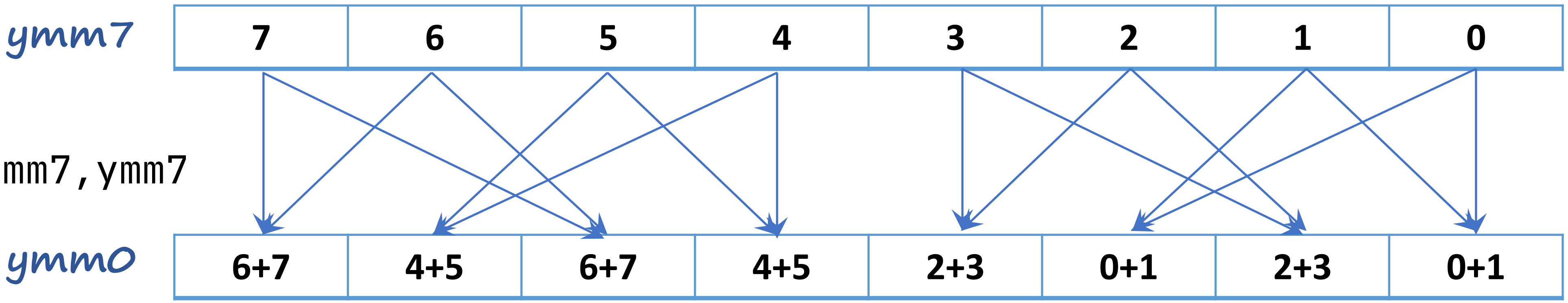
```
vphaddd
  ymm0,ymm7,ymm7
```

ymm0								

```
vphaddd
  ymm0,ymm7,ymm7
```



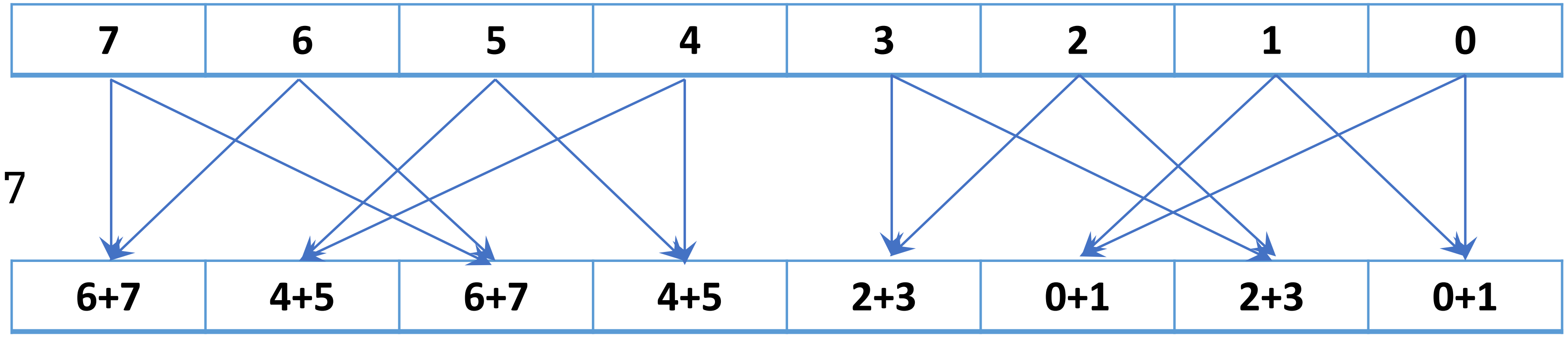
ymm7	7	6	5	4	3	2	1	0
ymm0			6+7	4+5			2+3	0+1



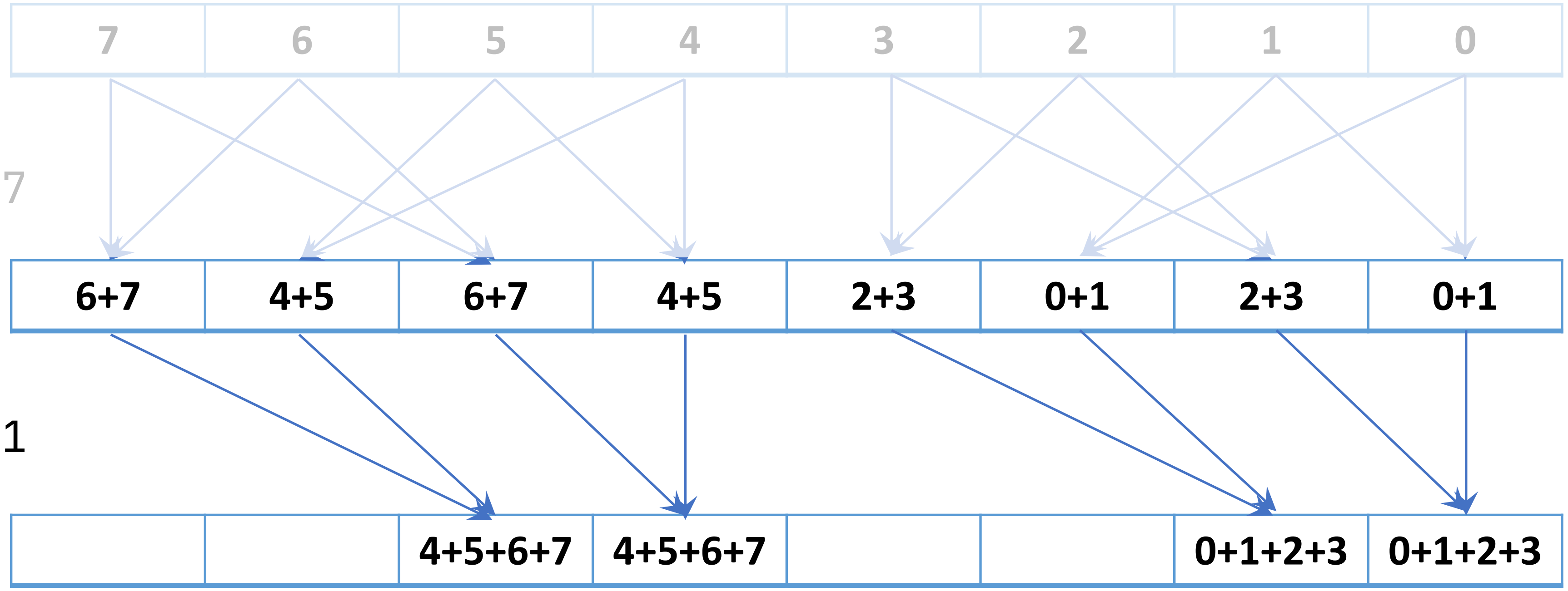
ymm7	7	6	5	4	3	2	1	0
ymm0	6+7	4+5	6+7	4+5	2+3	0+1	2+3	0+1

```
vphaddd
  ymm0,ymm7,ymm7
```

```
vphaddd
   ymm0,ymm7,ymm7
```



ymm7	7	6	5	4	3	2	1	0
ymm0	6+7	4+5	6+7	4+5	2+3	0+1	2+3	0+1



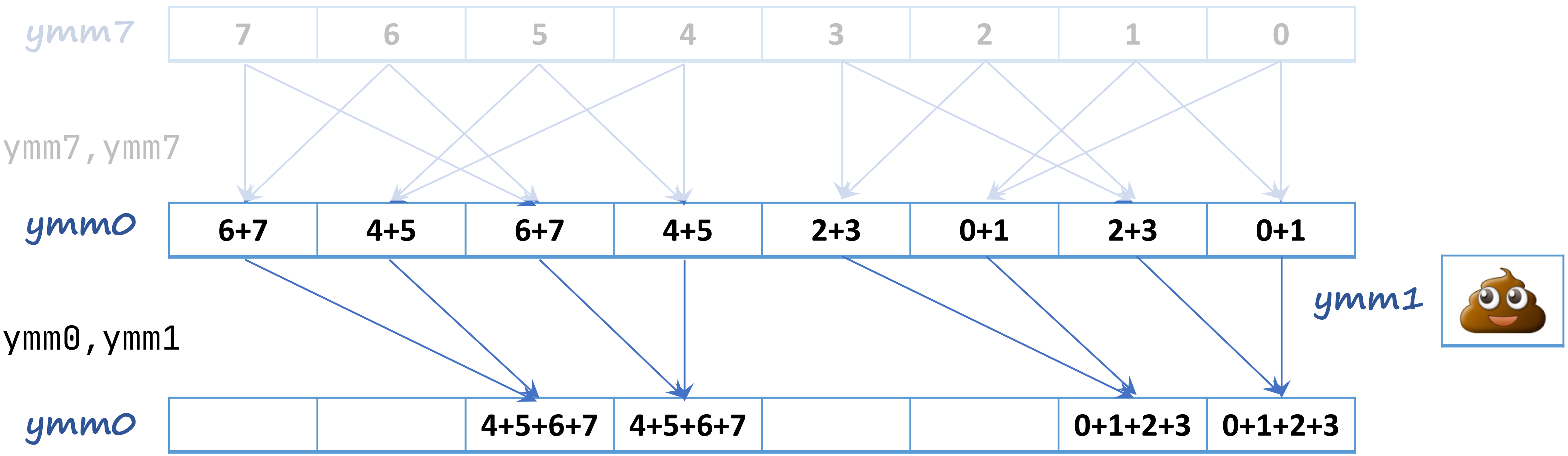
ymm7	7	6	5	4	3	2	1	0

```
vphaddd
  ymm0,ymm7,ymm7
```

ymm0	6+7	4+5	6+7	4+5	2+3	0+1	2+3	0+1

```
vphaddd
  ymm0,ymm0,ymm1
```

ymm0			4+5+6+7	4+5+6+7			0+1+2+3	0+1+2+3

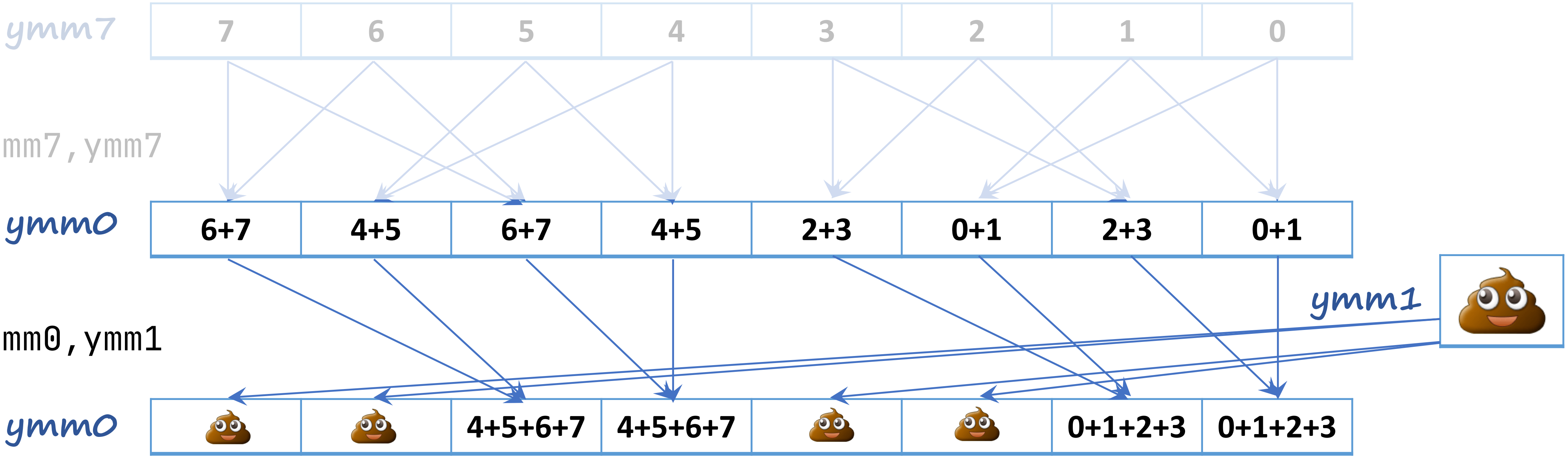


```
vphaddd
  ymm0,ymm7,ymm7
```

ymm7	7	6	5	4	3	2	1	0
ymm0	6+7	4+5	6+7	4+5	2+3	0+1	2+3	0+1

```
vphaddd
  ymm0,ymm0,ymm1
```

ymm0			4+5+6+7	4+5+6+7			0+1+2+3	0+1+2+3



ymm7	7	6	5	4	3	2	1	0

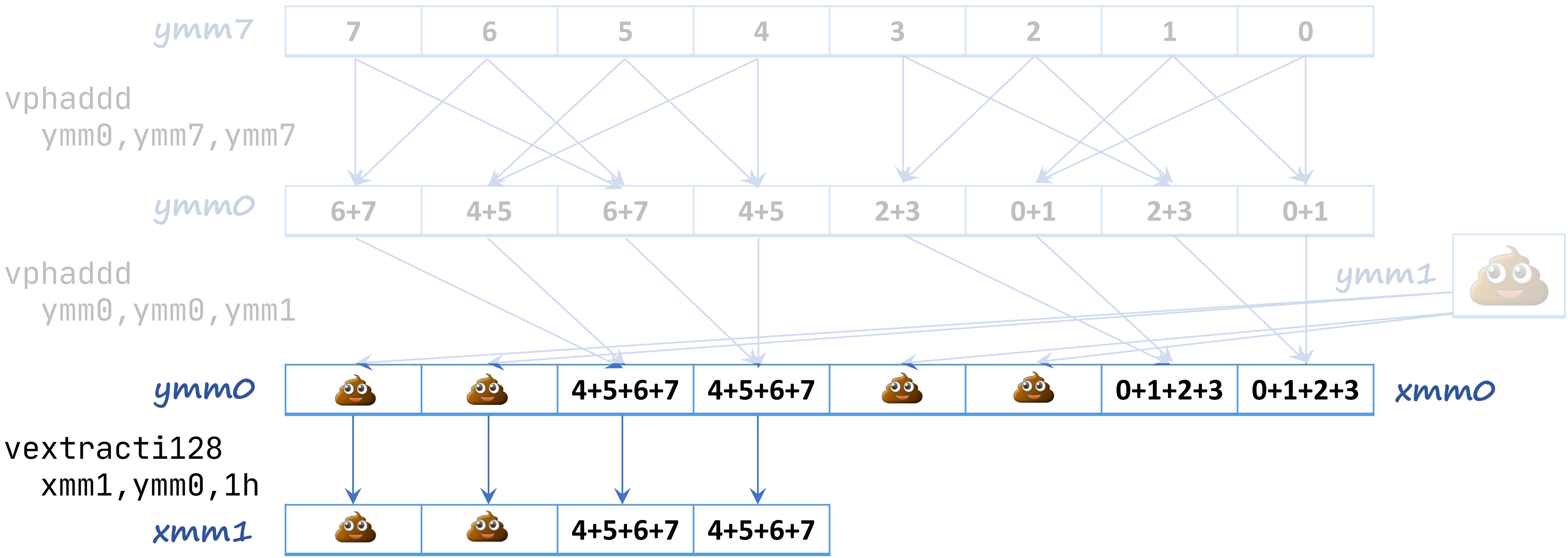
```
vphaddd
  ymm0,ymm7,ymm7
```

ymm0	6+7	4+5	6+7	4+5	2+3	0+1	2+3	0+1

```
vphaddd
  ymm0,ymm0,ymm1
```

ymm1	💩

ymm0	💩	💩	4+5+6+7	4+5+6+7	💩	💩	0+1+2+3	0+1+2+3



ymm7

7	6	5	4	3	2	1	0

```
vphaddd
  ymm0,ymm7,ymm7
```

ymm0

6+7	4+5	6+7	4+5	2+3	0+1	2+3	0+1

```
vphaddd
  ymm0,ymm0,ymm1
```

ymm1 💩

ymm0

💩	💩	4+5+6+7	4+5+6+7	💩	💩	0+1+2+3	0+1+2+3

xmm0

```
vextracti128
  xmm1,ymm0,1h
```

xmm1

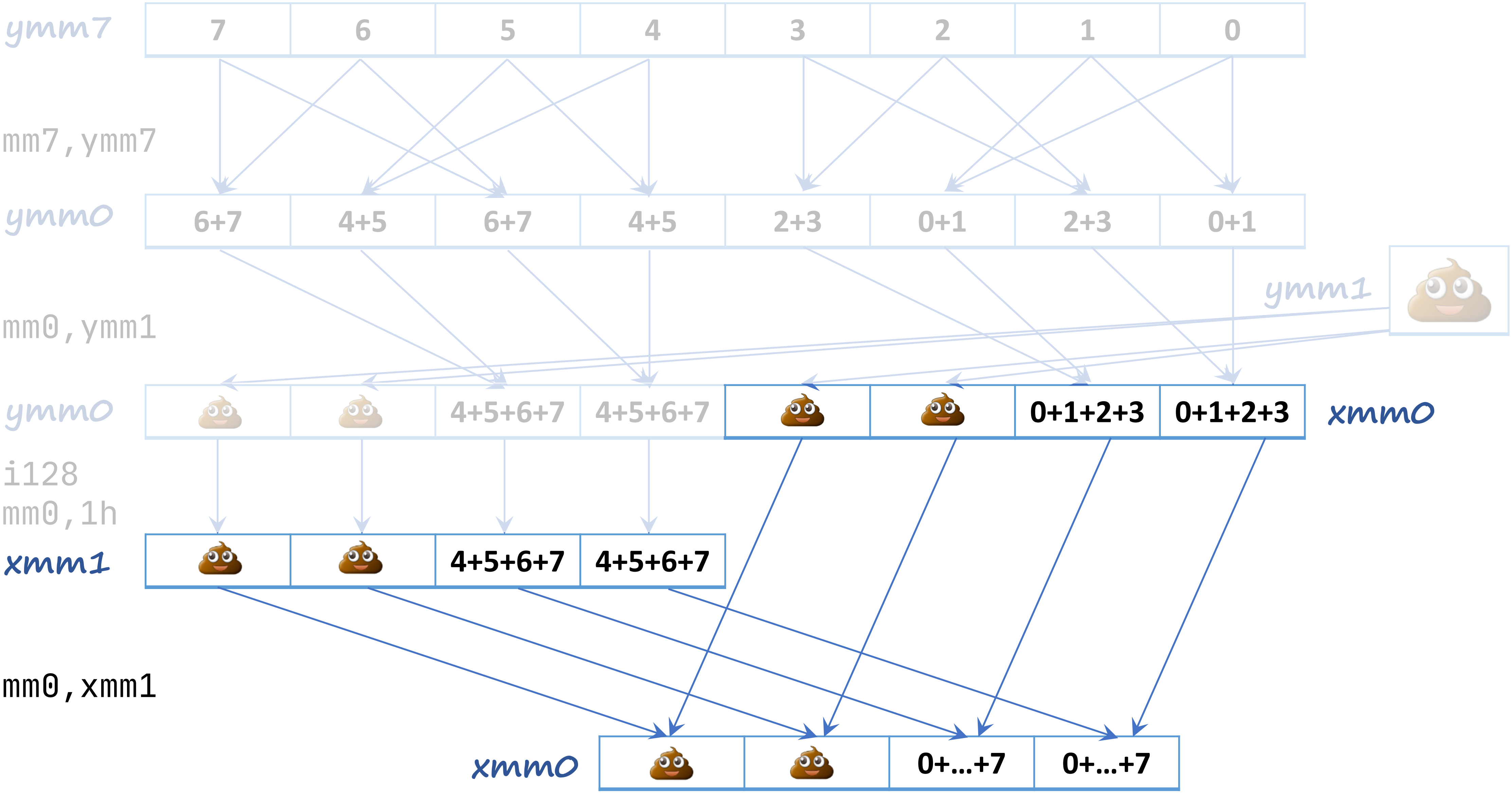
💩	💩	4+5+6+7	4+5+6+7

```
vphaddd
  ymm0,ymm7,ymm7
```

```
vphaddd
  ymm0,ymm0,ymm1
```

```
vextracti128
  xmm1,ymm0,1h
```

```
vpaddd
  xmm0,xmm0,xmm1
```



Восемь раз для каждого значения

```
vphaddd ymm0,ymm7,ymm7
vphaddd ymm0,ymm0,ymm1
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,eax
vpaddd  xmm1,xmm1,xmm0
vmovd   r11d,xmm1
```

Суммируем компоненты ymm7

eax + xmm0 → r11d

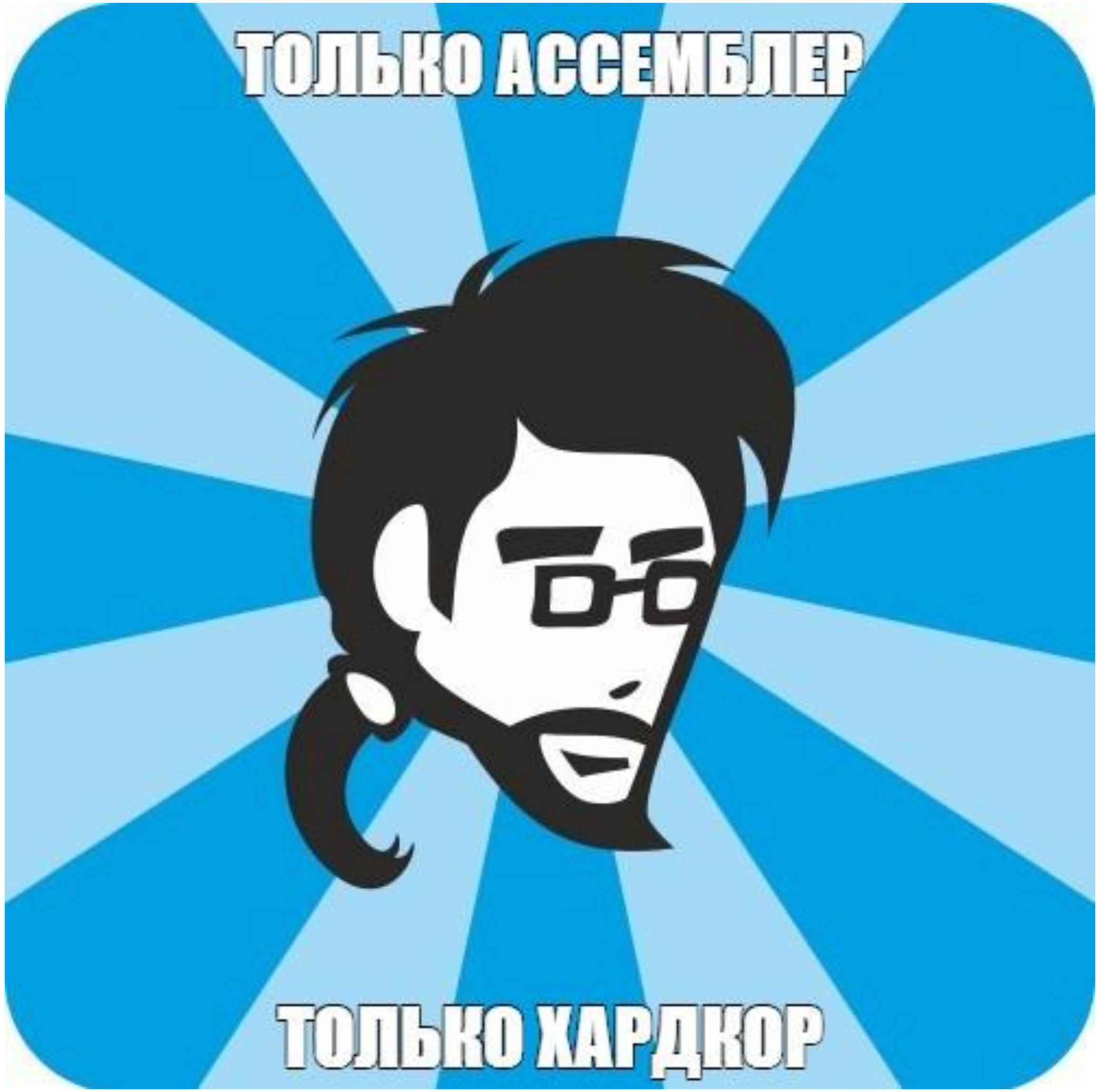
```
add     edi,40h
cmp     edi,ebp
jl      loop
```

edi+=64

Если меньше ebp, то идём назад

```
…
vpaddd  ymm0,ymm0,ymm1
vpaddd  ymm0,ymm0,ymm2
vpaddd  ymm0,ymm0,ymm3
vpaddd  ymm0,ymm0,ymm4
vpaddd  ymm0,ymm0,ymm5
vpaddd  ymm0,ymm0,ymm6
vpaddd  ymm0,ymm0,ymm7
vphaddd ymm0,ymm0,ymm0
vphaddd ymm0,ymm0,ymm0
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,eax
vpaddd  xmm1,xmm1,xmm0
vmovd   eax,xmm1
…
```

```
…
vpaddd  ymm0,ymm0,ymm1
vpaddd  ymm0,ymm0,ymm2
vpaddd  ymm0,ymm0,ymm3
vpaddd  ymm0,ymm0,ymm4
vpaddd  ymm0,ymm0,ymm5
vpaddd  ymm0,ymm0,ymm6
vpaddd  ymm0,ymm0,ymm7
vphaddd ymm0,ymm0,ymm0
vphaddd ymm0,ymm0,ymm0
vextracti128 xmm1,ymm0,1h
vpaddd  xmm0,xmm0,xmm1
vmovd   xmm1,eax
vpaddd  xmm1,xmm1,xmm0
vmovd   eax,xmm1
…
```



Итого

Issue ID	Method/Class	Fix version
JDK-8054221	StringJoiner	9
JDK-8058779	String.replace	9
JDK-8222955	String.replace	13
JDK-8225339	HashSet.toArray()	14
JDK-8029891	Properties.getProperty()	9
JDK-8155600	Arrays.asList().iterator()	9
JDK-8166365	emptyList().hashCode()	9
JDK-8214687	nCopies().hashCode()	13
JDK-7130085	Math.hypot()	9
JDK-8222074	Math.abs()	13

Итого

Issue ID	Method/Class	Fix version
JDK-8054221	StringJoiner	9
JDK-8058779	String.replace	9
JDK-8222955	String.replace	13
JDK-8225339	HashSet.toArray()	14
JDK-8029891	Properties.getProperty()	9
JDK-8155600	Arrays.asList().iterator()	9
JDK-8166365	emptyList().hashCode()	9
JDK-8214687	nCopies().hashCode()	13, 8u231
JDK-7130085	Math.hypot()	9
JDK-8222074	Math.abs()	13

A DAY MAY COME WHEN I UPDATE JAVA
BUT IT IS NOT THIS DAY

Спасибо
за внимание

https://twitter.com/tagir_valeev
https://habrahabr.ru/users/lany
https://github.com/amaembo
tagir.valeev@jetbrains.com