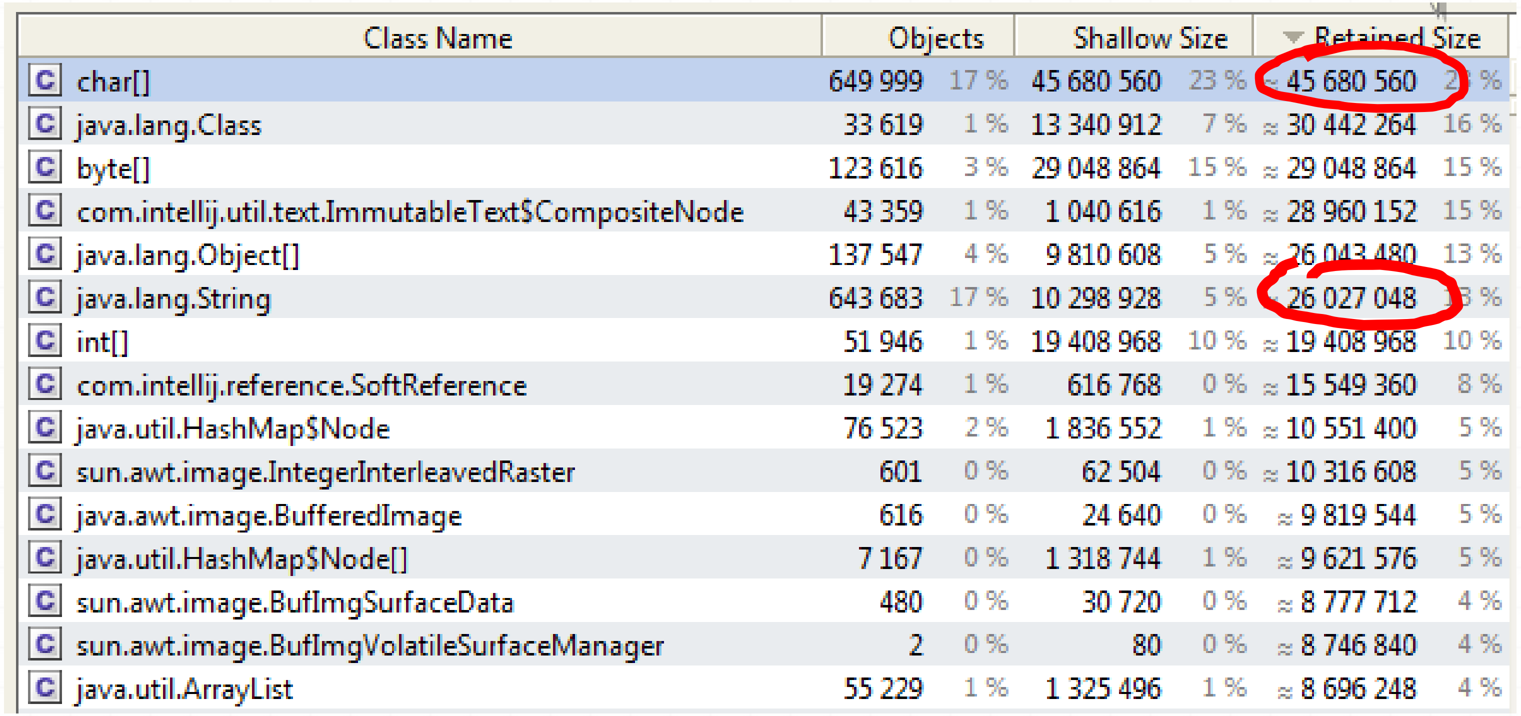
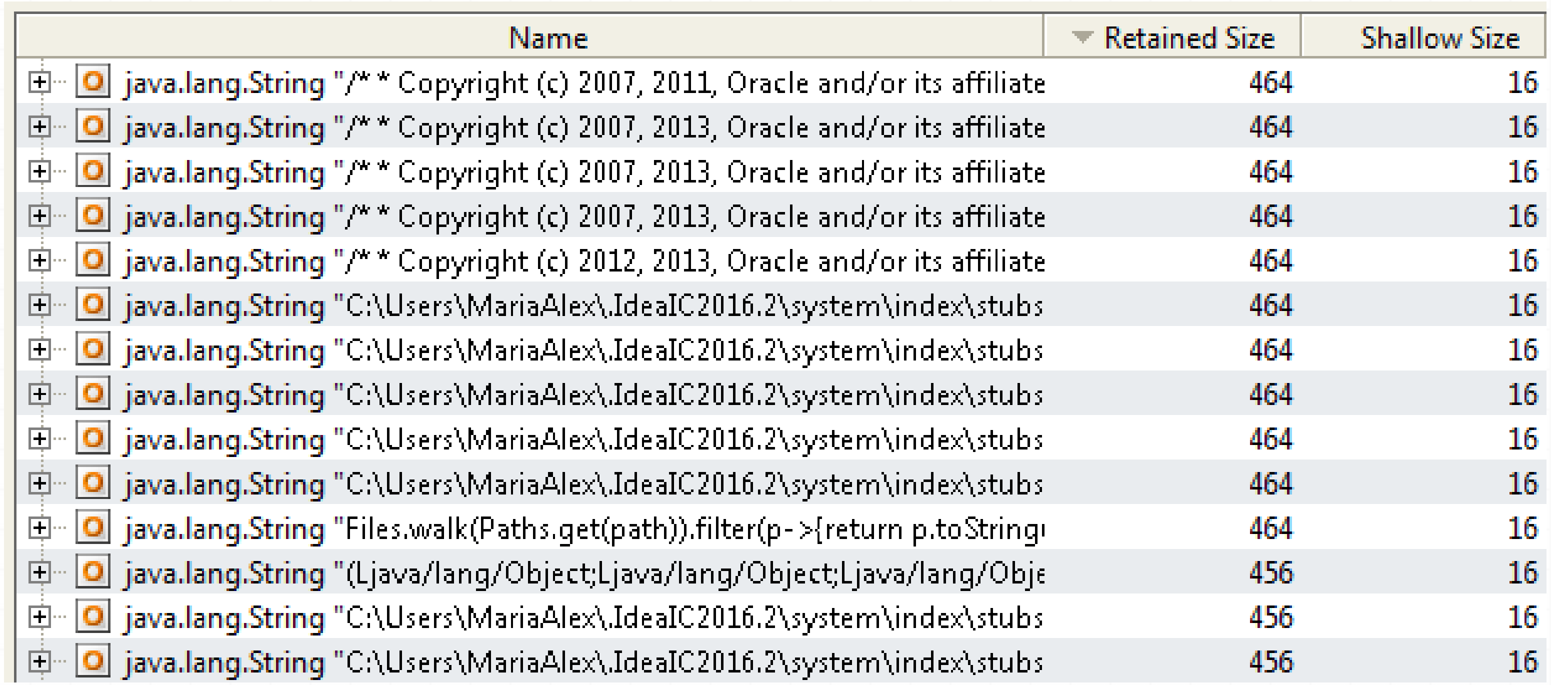
Таинственный дневник

Мы программируем строки



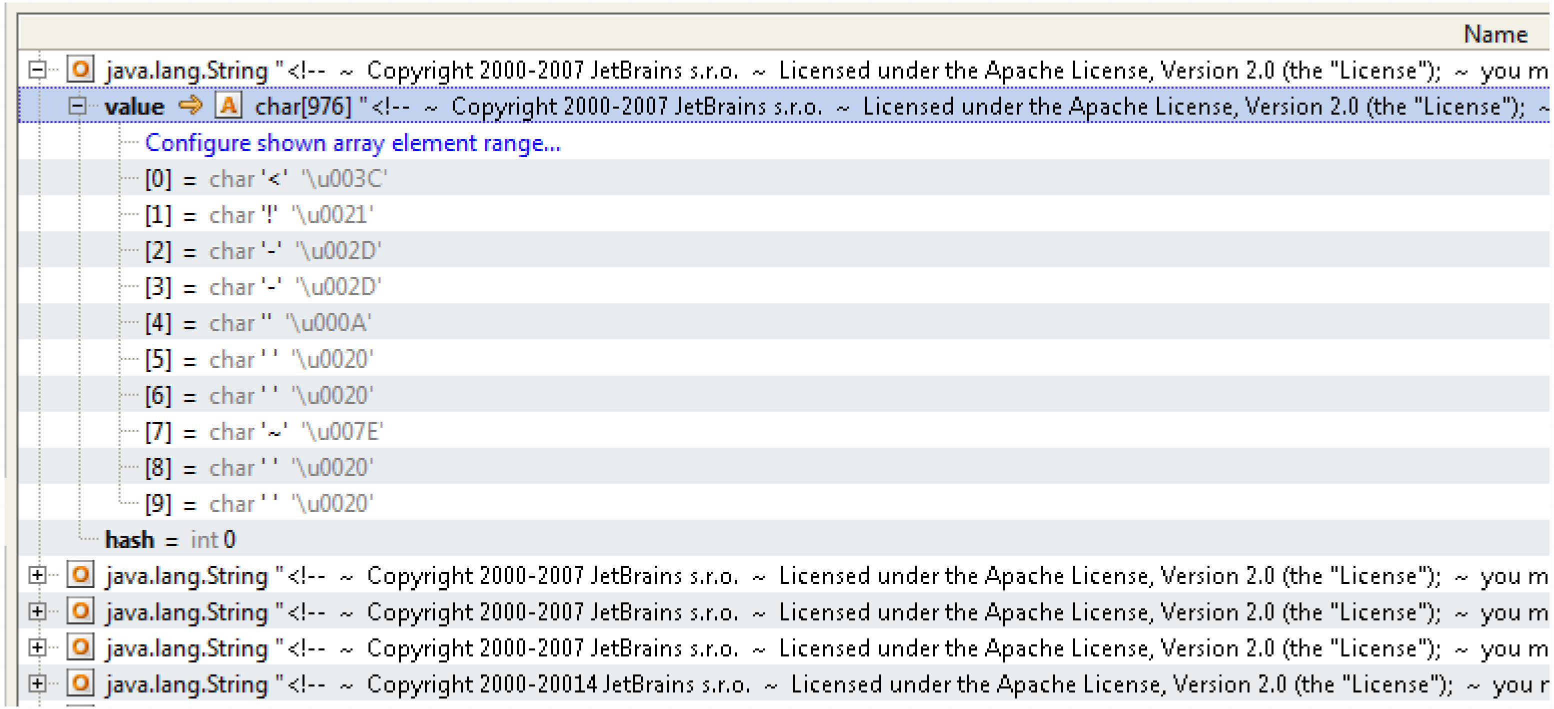
Class Name	Objects		Shallow Size		Retained Size	
char[]	649 999	17 %	45 680 560	23 %	≈ 45 680 560	23 %
java.lang.Class	33 619	1 %	13 340 912	7 %	≈ 30 442 264	16 %
byte[]	123 616	3 %	29 048 864	15 %	≈ 29 048 864	15 %
com.intellij.util.text.ImmutableText$CompositeNode	43 359	1 %	1 040 616	1 %	≈ 28 960 152	15 %
java.lang.Object[]	137 547	4 %	9 810 608	5 %	≈ 26 043 480	13 %
java.lang.String	643 683	17 %	10 298 928	5 %	≈ 26 027 048	13 %
int[]	51 946	1 %	19 408 968	10 %	≈ 19 408 968	10 %
com.intellij.reference.SoftReference	19 274	1 %	616 768	0 %	≈ 15 549 360	8 %
java.util.HashMap$Node	76 523	2 %	1 836 552	1 %	≈ 10 551 400	5 %
sun.awt.image.IntegerInterleavedRaster	601	0 %	62 504	0 %	≈ 10 316 608	5 %
java.awt.image.BufferedImage	616	0 %	24 640	0 %	≈ 9 819 544	5 %
java.util.HashMap$Node[]	7 167	0 %	1 318 744	1 %	≈ 9 621 576	5 %
sun.awt.image.BufImgSurfaceData	480	0 %	30 720	0 %	≈ 8 777 712	4 %
sun.awt.image.BufImgVolatileSurfaceManager	2	0 %	80	0 %	≈ 8 746 840	4 %
java.util.ArrayList	55 229	1 %	1 325 496	1 %	≈ 8 696 248	4 %

… с английским текстом



Name	Retained Size	Shallow Size
java.lang.String "/* * Copyright (c) 2007, 2011, Oracle and/or its affiliate	464	16
java.lang.String "/* * Copyright (c) 2007, 2013, Oracle and/or its affiliate	464	16
java.lang.String "/* * Copyright (c) 2007, 2013, Oracle and/or its affiliate	464	16
java.lang.String "/* * Copyright (c) 2007, 2013, Oracle and/or its affiliate	464	16
java.lang.String "/* * Copyright (c) 2012, 2013, Oracle and/or its affiliate	464	16
java.lang.String "C:\Users\MariaAlex\.IdeaIC2016.2\system\index\stubs	464	16
java.lang.String "C:\Users\MariaAlex\.IdeaIC2016.2\system\index\stubs	464	16
java.lang.String "C:\Users\MariaAlex\.IdeaIC2016.2\system\index\stubs	464	16
java.lang.String "C:\Users\MariaAlex\.IdeaIC2016.2\system\index\stubs	464	16
java.lang.String "C:\Users\MariaAlex\.IdeaIC2016.2\system\index\stubs	464	16
java.lang.String "Files.walk(Paths.get(path)).filter(p->{return p.toString	464	16
java.lang.String "(Ljava/lang/Object;Ljava/lang/Object;Ljava/lang/Obje	456	16
java.lang.String "C:\Users\MariaAlex\.IdeaIC2016.2\system\index\stubs	456	16
java.lang.String "C:\Users\MariaAlex\.IdeaIC2016.2\system\index\stubs	456	16

… из ASCII символов



```
                                                                                                  Name
⊟ java.lang.String "<!--  ~  Copyright 2000-2007 JetBrains s.r.o.  ~  Licensed under the Apache License, Version 2.0 (the "License");  ~  you m
   ⊟ value ⇒ char[976] "<!--  ~  Copyright 2000-2007 JetBrains s.r.o.  ~  Licensed under the Apache License, Version 2.0 (the "License");  ~
        Configure shown array element range...
        [0] = char '<'  '\u003C'
        [1] = char '!'  '\u0021'
        [2] = char '-'  '\u002D'
        [3] = char '-'  '\u002D'
        [4] = char ''  '\u000A'
        [5] = char ' '  '\u0020'
        [6] = char ' '  '\u0020'
        [7] = char '~'  '\u007E'
        [8] = char ' '  '\u0020'
        [9] = char ' '  '\u0020'
     hash = int 0
⊞ java.lang.String "<!--  ~  Copyright 2000-2007 JetBrains s.r.o.  ~  Licensed under the Apache License, Version 2.0 (the "License");  ~  you m
⊞ java.lang.String "<!--  ~  Copyright 2000-2007 JetBrains s.r.o.  ~  Licensed under the Apache License, Version 2.0 (the "License");  ~  you m
⊞ java.lang.String "<!--  ~  Copyright 2000-2007 JetBrains s.r.o.  ~  Licensed under the Apache License, Version 2.0 (the "License");  ~  you m
⊞ java.lang.String "<!--  ~  Copyright 2000-20014 JetBrains s.r.o.  ~  Licensed under the Apache License, Version 2.0 (the "License");  ~  you r
```

Вжух-метод

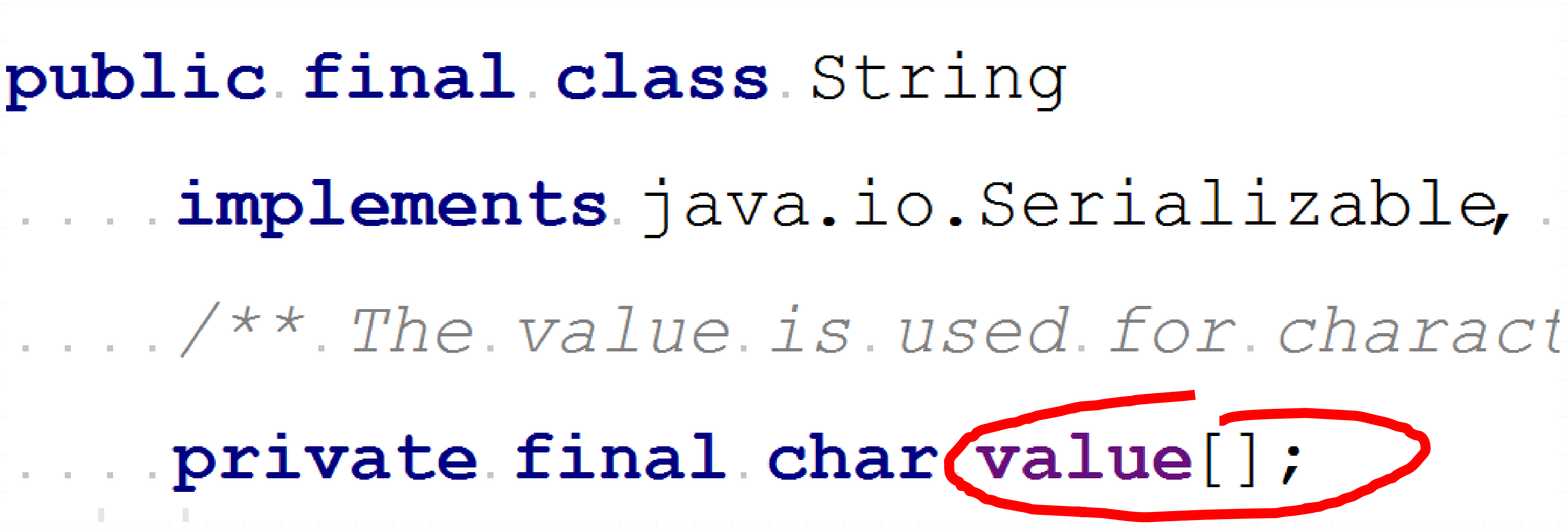
X2

JEP 254: Compact Strings in JDK9

JDK9 Issues: Оставь надежду

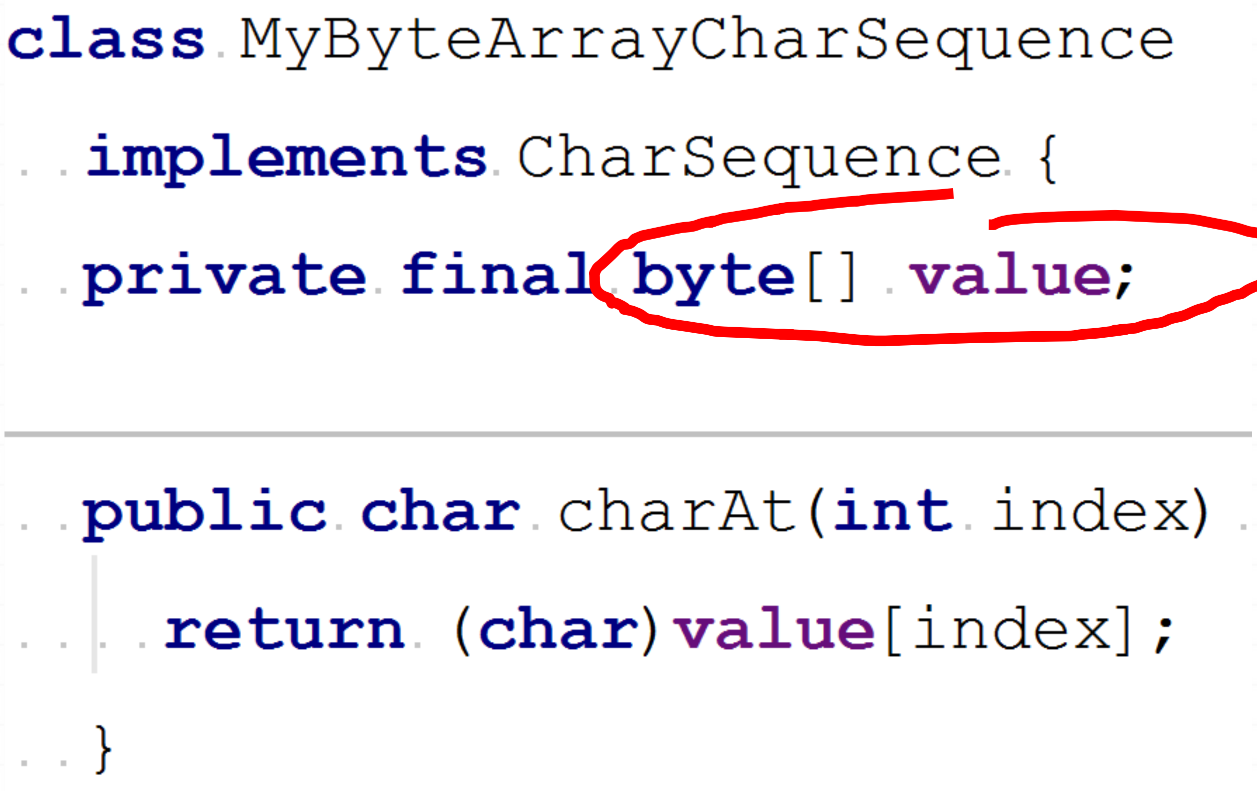
Status	Description
Unresolved	Copy-paste garbles line-endings: IDEA-129142, JDK-8058780
Unresolved	It's required to use "--add-exports" and "--add-opens" flags in JVM command line to access non-public API
Unresolved	"-Xbootclasspath/p" option is no longer supported
Unresolved	Forms compilation doesn’t work (no ASM for JDK 9)
Unresolved	Can’t inject *rt module classes to user's project classpath when it’s started as modular java application.
Unresolved	Everything is broken.

JDK8: Trump Strings great again



```
public final class String
    implements java.io.Serializable,
    /** The value is used for charact
    private final char value[];
```

JDK8: Trump Strings great again



```
class MyByteArrayCharSequence
  implements CharSequence {
  private final byte[] value;

  public char charAt(int index)
    return (char)value[index];
  }
```

JDK loves Strings

- java.lang.String: 64557 usages.

JDK loves Strings

- java.lang.String: 64557 usages.
- java.lang.CharSequence: 369 usages.

JDK loves Strings

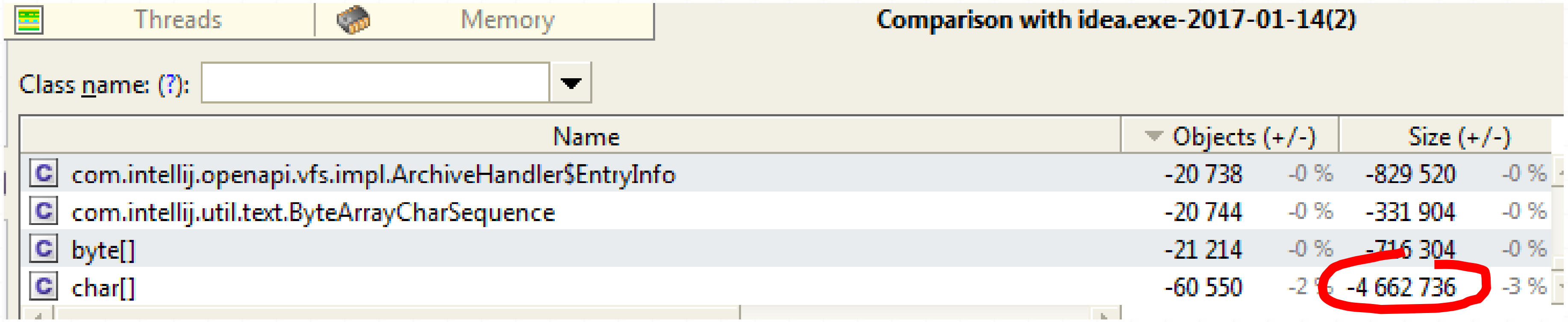
369

- java.lang.String: 64557 usages.
- java.lang.CharSequence: 369 usages.

JDK8 String de-duplication: new hope

```
-XX:+UseG1GC -XX:+UseStringDeduplication
```

String de-duplication: победа?



Comparison with idea.exe-2017-01-14(2)

Class name: (?):

Name	Objects (+/-)		Size (+/-)	
com.intellij.openapi.vfs.impl.ArchiveHandler$EntryInfo	-20 738	-0 %	-829 520	-0 %
com.intellij.util.text.ByteArrayCharSequence	-20 744	-0 %	-331 904	-0 %
byte[]	-21 214	-0 %	-716 304	-0 %
char[]	-60 550	-2 %	-4 662 736	-3 %

String de-duplication:Ужас

Candidate Selection

Candidate selection is done during young/mixed and full collections. This is a performance sensitive operation since it is applied to all visited objects. An object is considered a deduplication candidate if all of the following statements are true:

•The object is an instance of String,

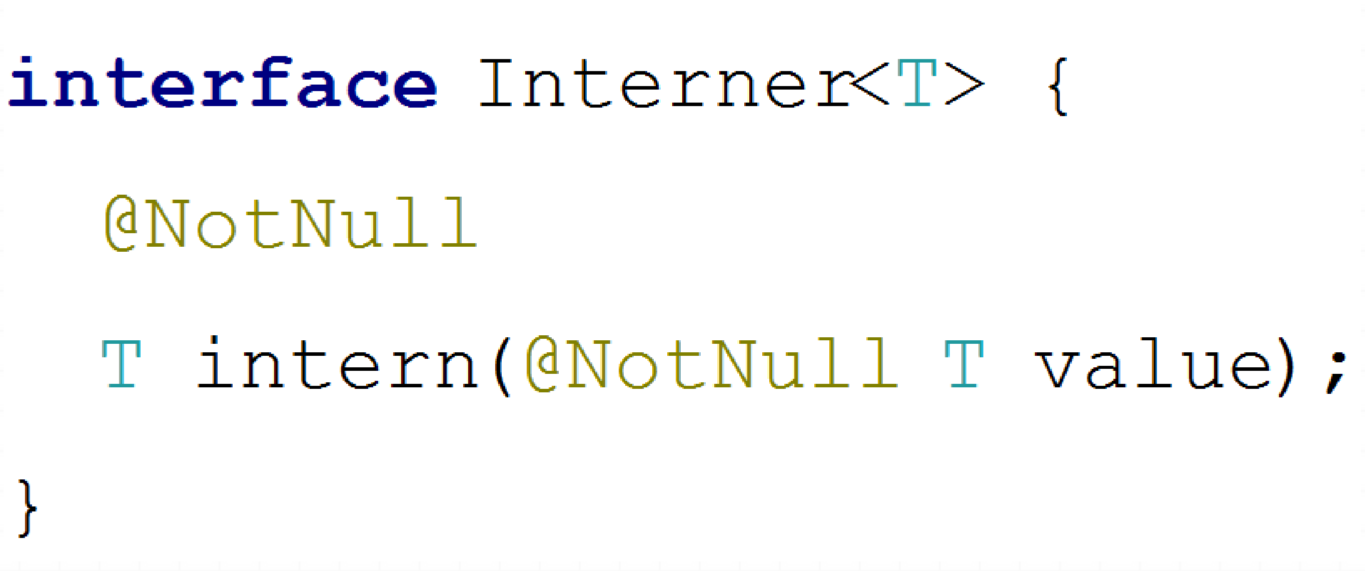
•The object is being evacuated *from* a young heap region, and

•The object is being evacuated *to* a young/survivor heap region and the object's age is *equal to* the deduplication age threshold, **or** the object is being evacuated *to* an old heap region and the object's age is *less than* the deduplication age threshold.

String de-duplication: Да ну на

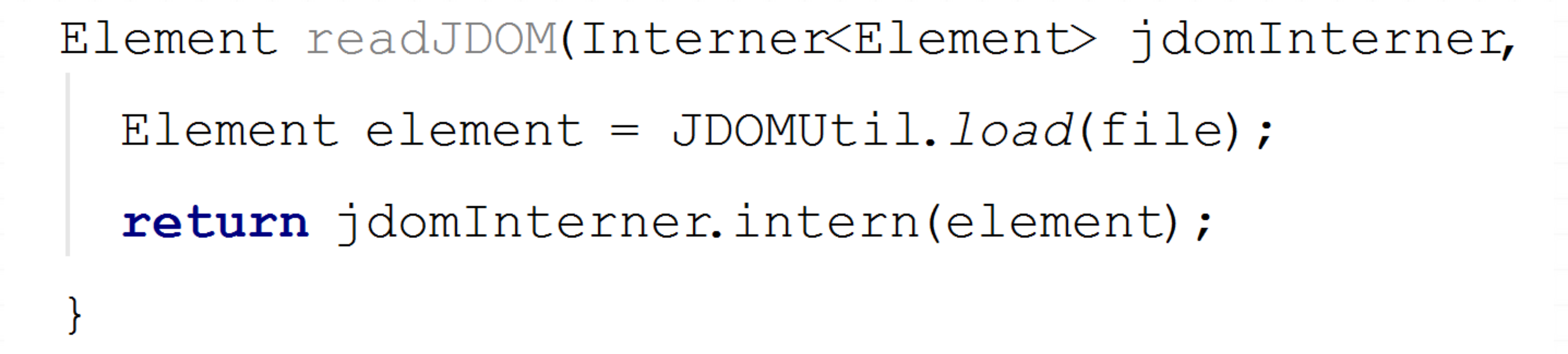
- No **String** objects count reduction
- Not all strings are de-duplication candidates
- Overhead (Deduplication hashtable, deduplication thread)

Хочешь хорошей дедупликации - сдедуплицируй дупликаты сам



```
interface Interner<T> {
  @NotNull
  T intern(@NotNull T value);
}
```

De-duplicate everything



```
Element readJDOM(Interner<Element> jdomInterner,
    Element element = JDOMUtil.load(file);
    return jdomInterner.intern(element);
}
```

Indirection kills

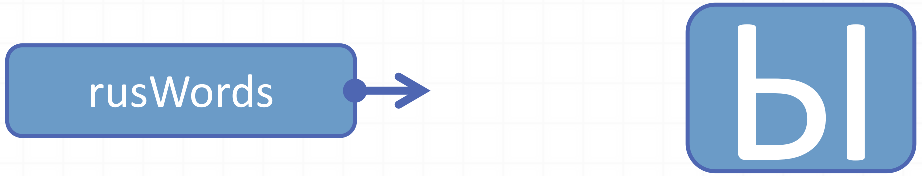
```
List<String> rusWords = new ArrayList<>();
rusWords.add("Ы");
```

Indirection kills

```
List<String> rusWords = new ArrayList<>();
rusWords.add("Ы");
```

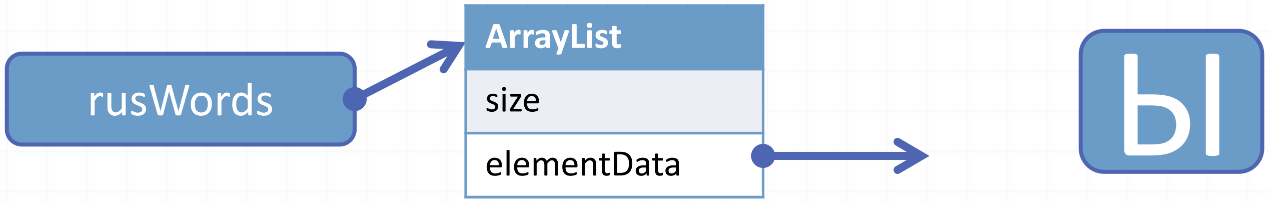
Indirection kills

```
List<String> rusWords = new ArrayList<>();
rusWords.add("Ы");
```



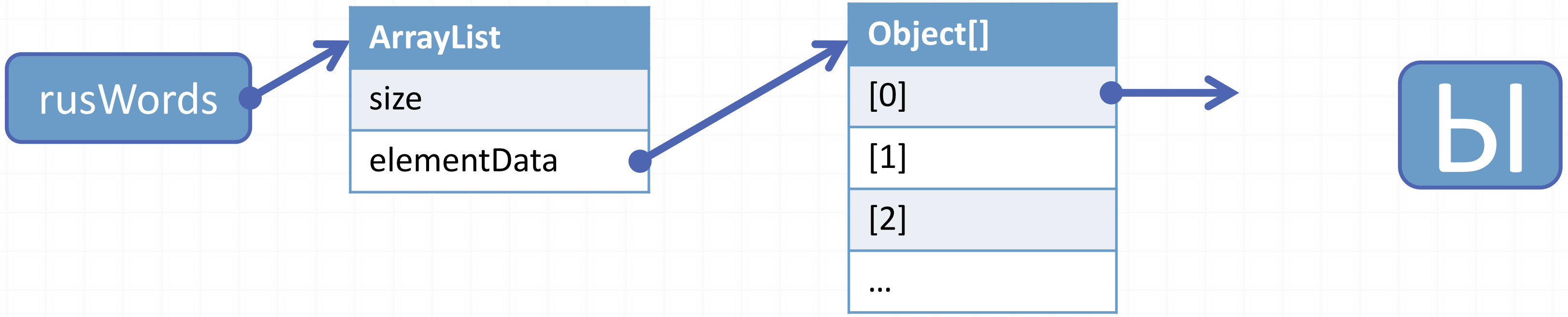
Indirection kills

```
List<String> rusWords = new ArrayList<>();
rusWords.add("Ы");
```



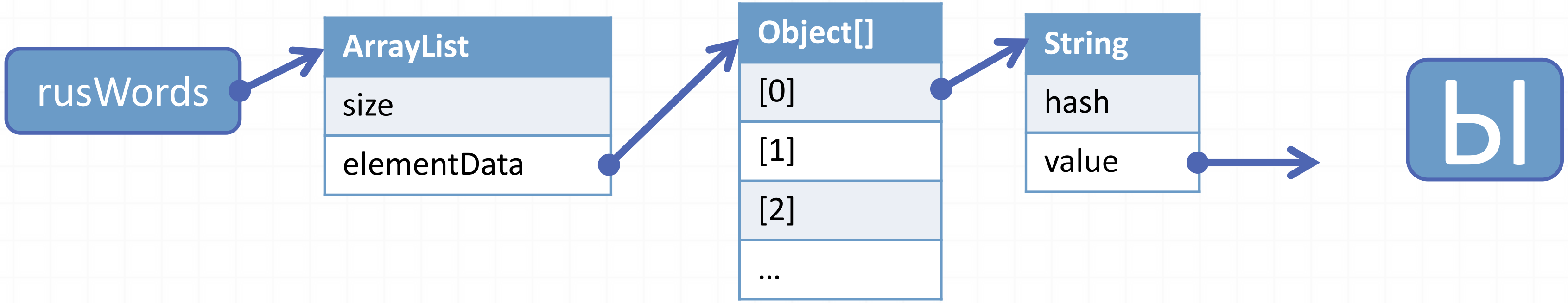
Indirection kills

```
List<String> rusWords = new ArrayList<>();
rusWords.add("Ы");
```



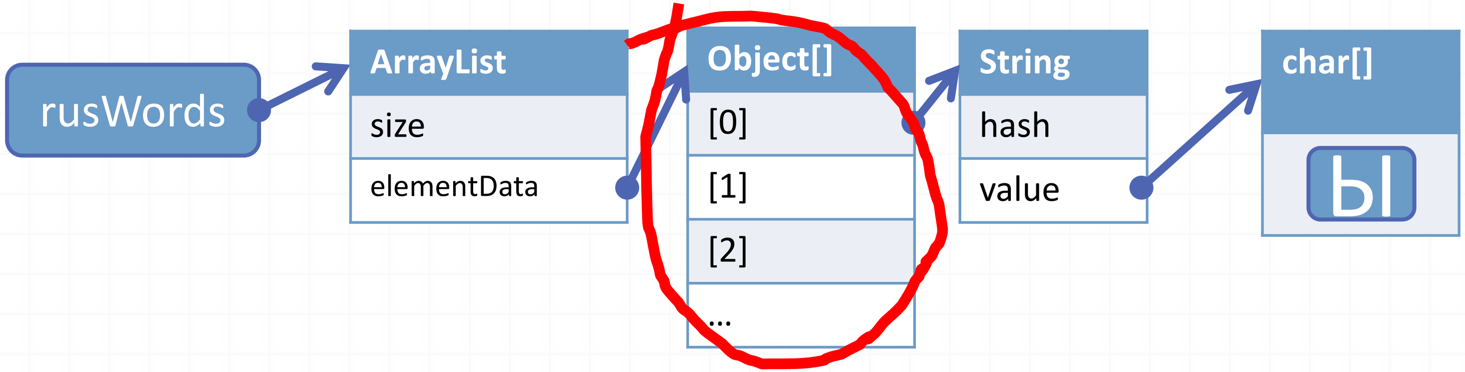
Indirection kills

```
List<String> rusWords = new ArrayList<>();
rusWords.add("Ы");
```

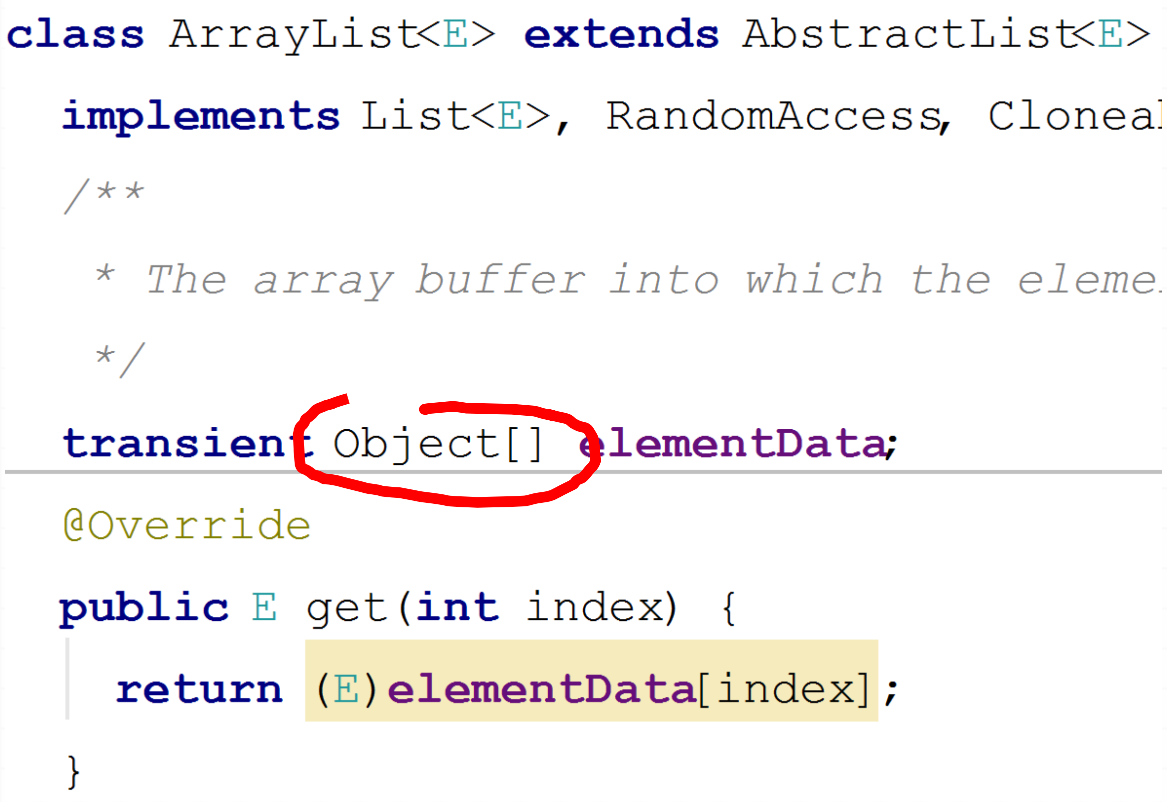


Indirection kills

```
List<String> rusWords = new ArrayList<>();
rusWords.add("Ы");
```

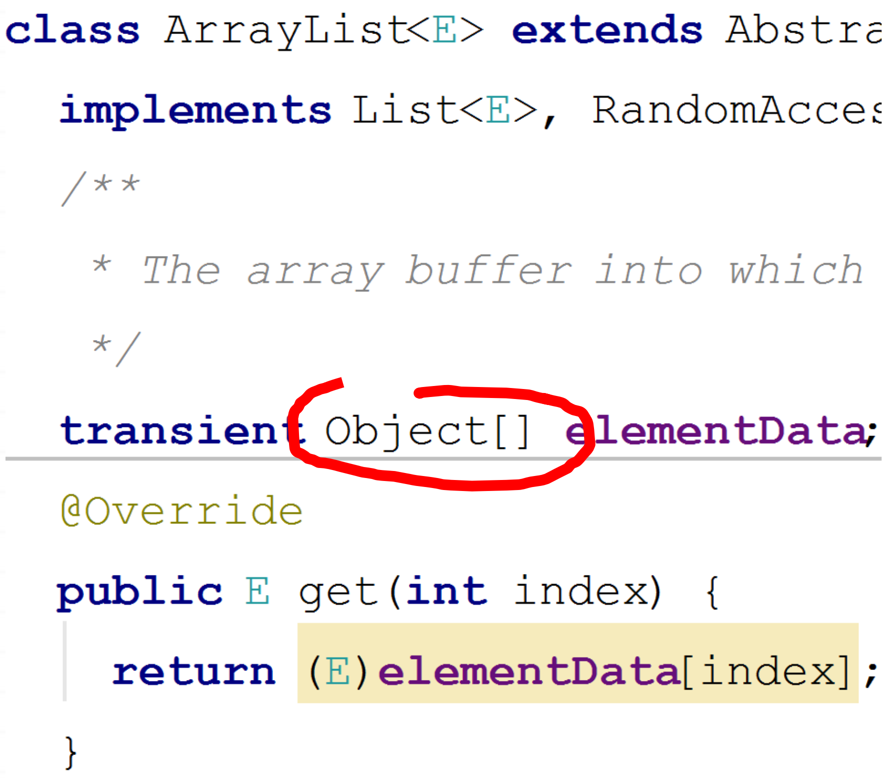


1-element lists FTW

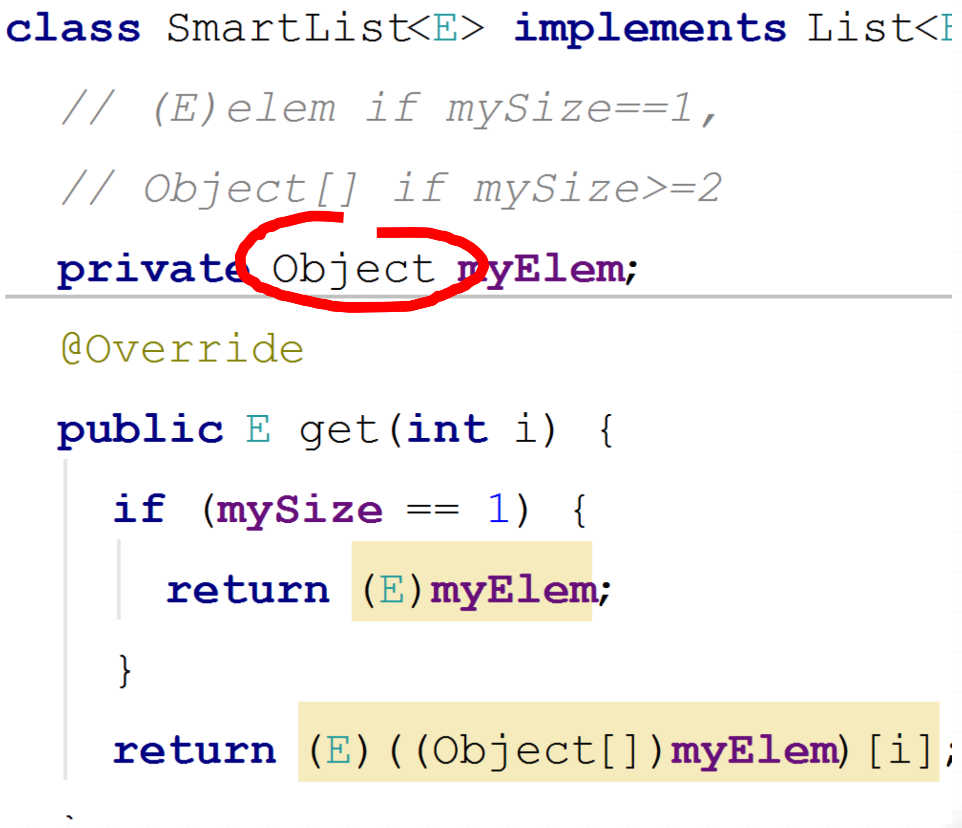


```
class ArrayList<E> extends AbstractList<E>
  implements List<E>, RandomAccess, Cloneal
  /**
   * The array buffer into which the eleme
   */
  transient Object[] elementData;
  @Override
  public E get(int index) {
    return (E)elementData[index];
  }
```

1-element lists FTW

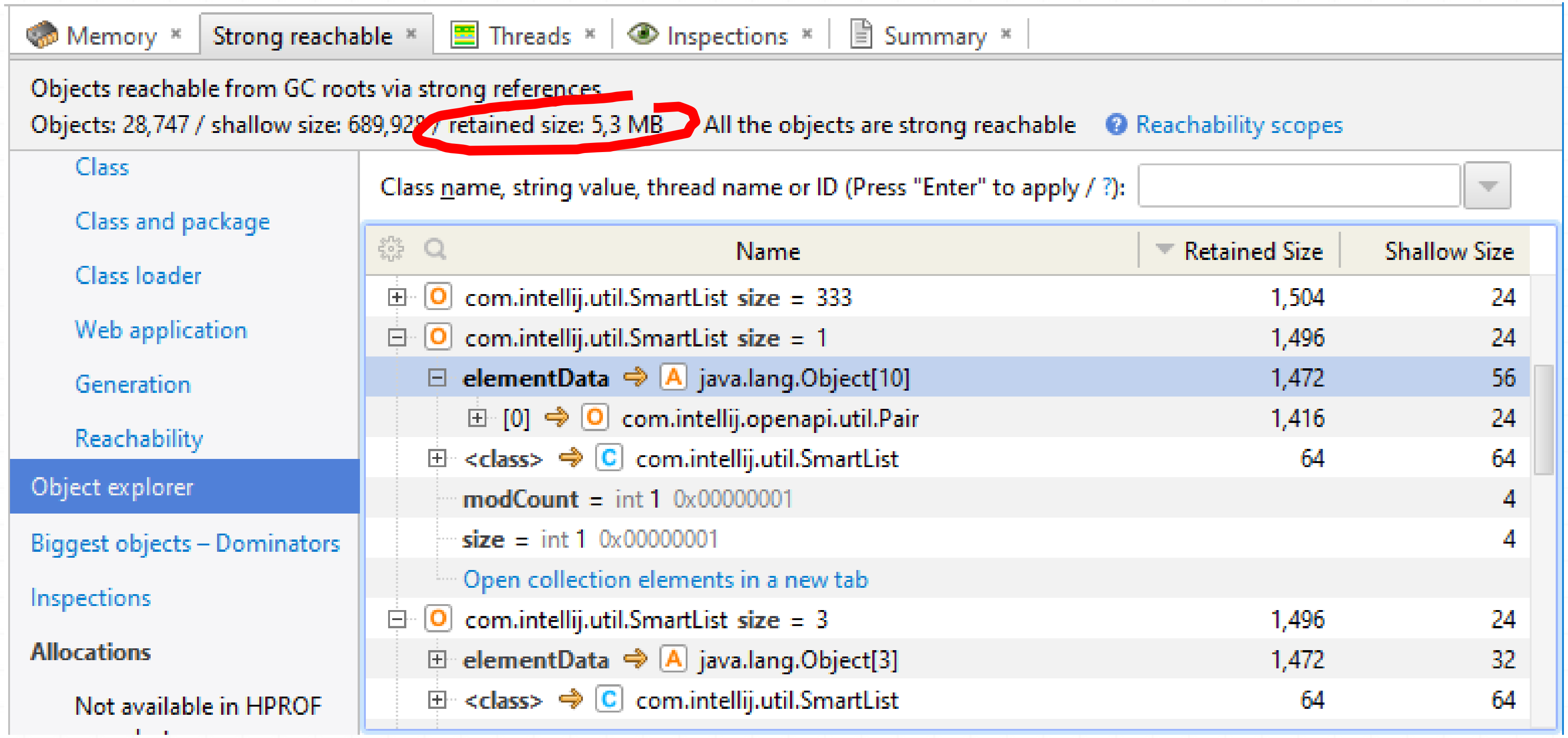


```
class ArrayList<E> extends Abstra
  implements List<E>, RandomAcce
  /**
   * The array buffer into which
   */
  transient Object[] elementData;
  @Override
  public E get(int index) {
    return (E)elementData[index];
  }
```



```
class SmartList<E> implements List<
  // (E)elem if mySize==1,
  // Object[] if mySize>=2
  private Object myElem;
  @Override
  public E get(int i) {
    if (mySize == 1) {
      return (E)myElem;
    }
    return (E)((Object[])myElem)[i];
```

ArrayList - before



Memory | Strong reachable | Threads | Inspections | Summary

Objects reachable from GC roots via strong references

Objects: 28,747 / shallow size: 689,928 / retained size: 5,3 MB All the objects are strong reachable Reachability scopes

Class

Class and package

Class loader

Web application

Generation

Reachability

Object explorer

Biggest objects – Dominators

Inspections

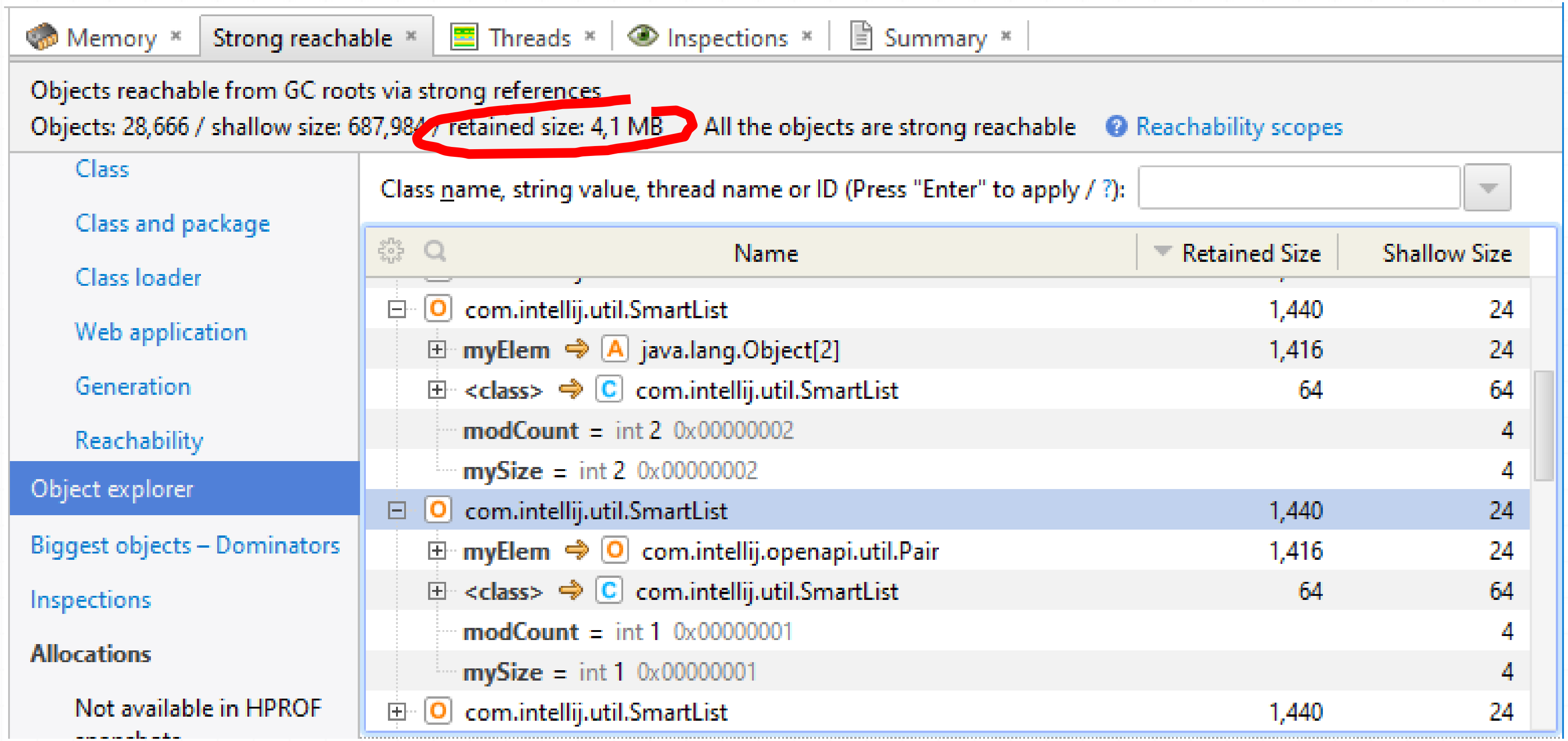
Allocations

Not available in HPROF

Class name, string value, thread name or ID (Press "Enter" to apply / ?):

Name	Retained Size	Shallow Size
com.intellij.util.SmartList size = 333	1,504	24
com.intellij.util.SmartList size = 1	1,496	24
elementData ⇒ java.lang.Object[10]	1,472	56
[0] ⇒ com.intellij.openapi.util.Pair	1,416	24
<class> ⇒ com.intellij.util.SmartList	64	64
modCount = int 1 0x00000001		4
size = int 1 0x00000001		4
Open collection elements in a new tab		
com.intellij.util.SmartList size = 3	1,496	24
elementData ⇒ java.lang.Object[3]	1,472	32
<class> ⇒ com.intellij.util.SmartList	64	64

SmartList - after



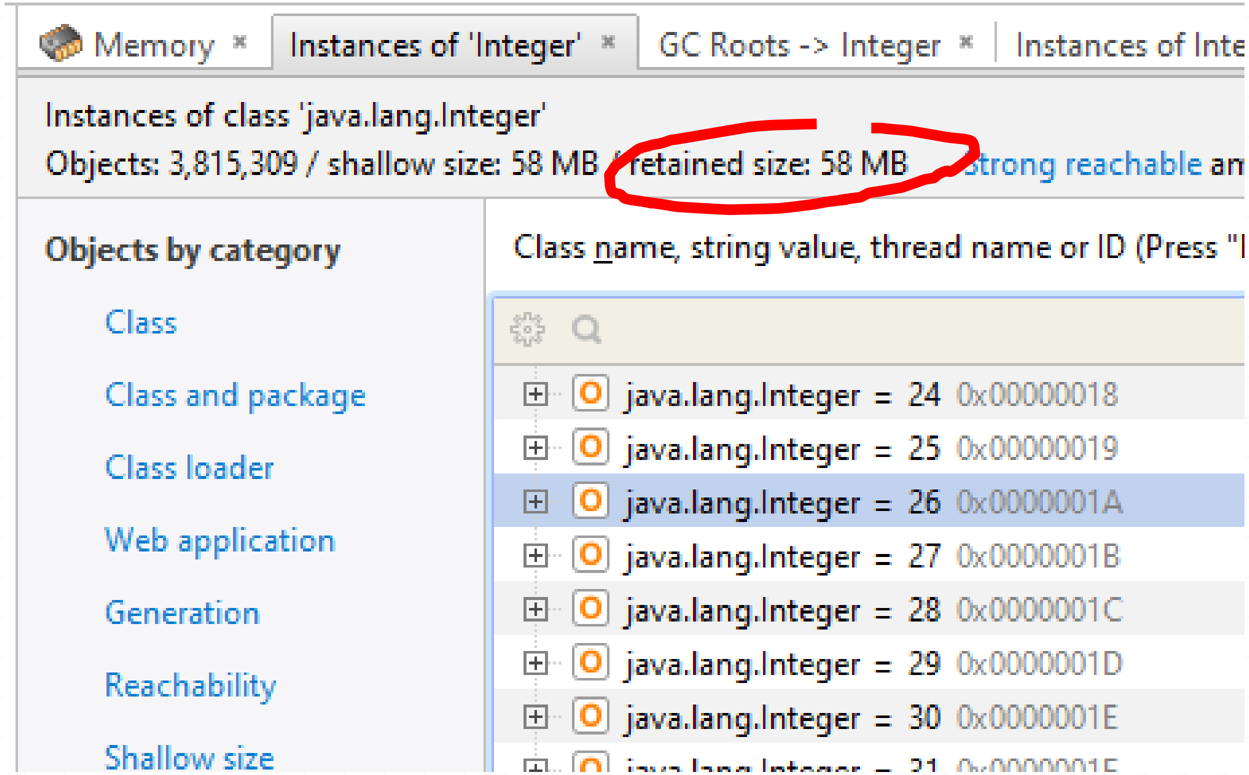
Memory | Strong reachable | Threads | Inspections | Summary

Objects reachable from GC roots via strong references
Objects: 28,666 / shallow size: 687,984 / retained size: 4,1 MB All the objects are strong reachable Reachability scopes

Class name, string value, thread name or ID (Press "Enter" to apply / ?):

Name	Retained Size	Shallow Size
com.intellij.util.SmartList	1,440	24
myElem ⇨ java.lang.Object[2]	1,416	24
<class> ⇨ com.intellij.util.SmartList	64	64
modCount = int 2 0x00000002		4
mySize = int 2 0x00000002		4
com.intellij.util.SmartList	1,440	24
myElem ⇨ com.intellij.openapi.util.Pair	1,416	24
<class> ⇨ com.intellij.util.SmartList	64	64
modCount = int 1 0x00000001		4
mySize = int 1 0x00000001		4
com.intellij.util.SmartList	1,440	24

Wrappers: чистое зло



Instances of class 'java.lang.Integer'

Objects: 3,815,309 / shallow size: 58 MB / retained size: 58 MB Strong reachable an

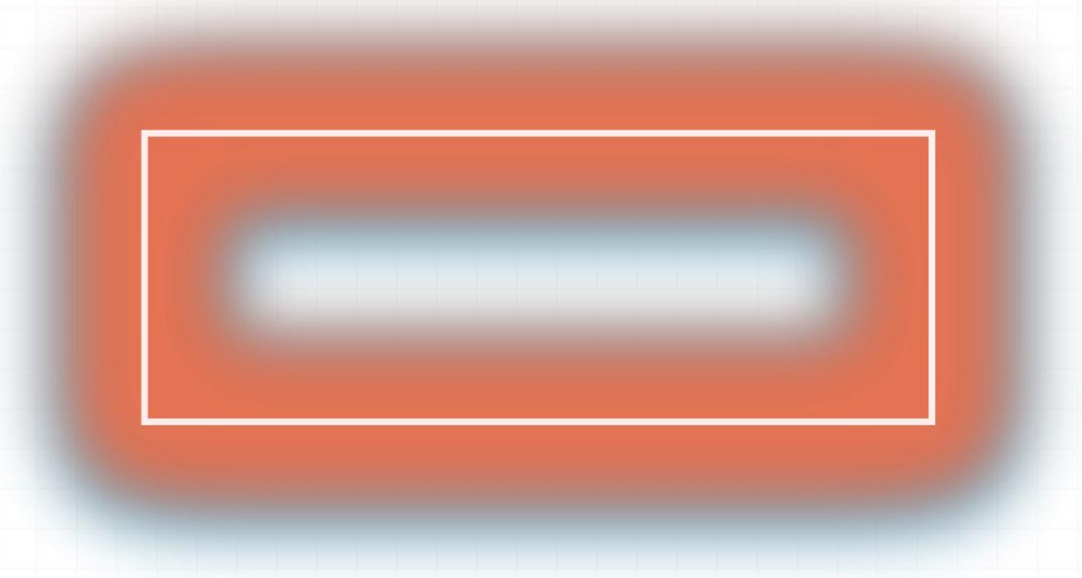
Objects by category

- Class
- Class and package
- Class loader
- Web application
- Generation
- Reachability
- Shallow size

Class name, string value, thread name or ID (Press "I

```
java.lang.Integer = 24 0x00000018
java.lang.Integer = 25 0x00000019
java.lang.Integer = 26 0x0000001A
java.lang.Integer = 27 0x0000001B
java.lang.Integer = 28 0x0000001C
java.lang.Integer = 29 0x0000001D
java.lang.Integer = 30 0x0000001E
java.lang.Integer = 31 0x0000001F
```

Спаситель: приди



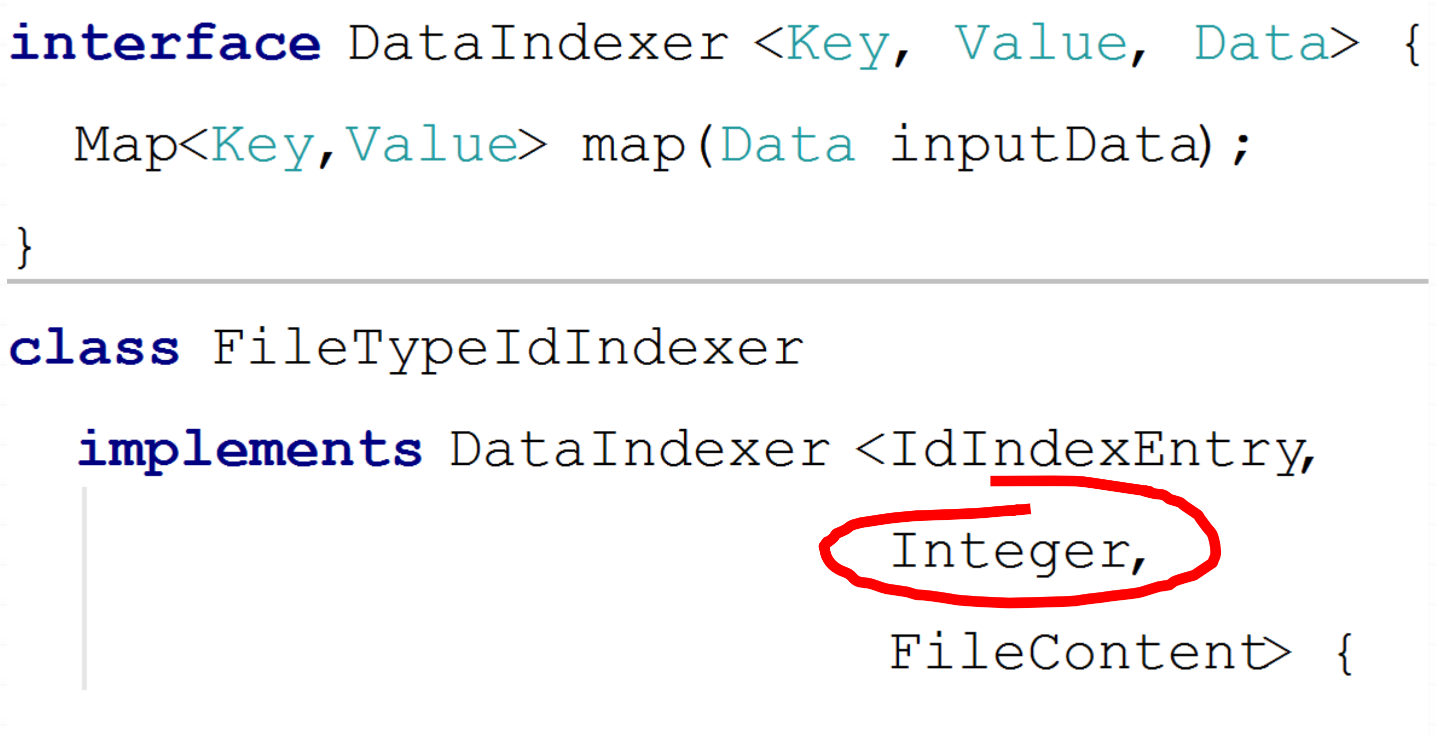
Спаситель: приди

Trove

Мы спасены

- HashSet<Long> → TLongHashSet
- HashMap<String, Double> → TObjectDoubleHashMap<String>
- ArrayList<Byte> → TByteArrayList

Мы обречены



```
interface DataIndexer <Key, Value, Data> {
  Map<Key,Value> map(Data inputData);
}
```

```
class FileTypeIdIndexer
  implements DataIndexer <IdIndexEntry,
                          Integer,
                          FileContent> {
```

Мы в ужасе

```
package java.util.stream;

import ...

interface IntStream extends BaseStream<Integer, IntStream> {

   IntStream filter(IntPredicate predicate);

   IntStream map(IntUnaryOperator mapper);

   <U> Stream<U> mapToObj(IntFunction<? extends U> mapper);

   LongStream mapToLong(IntToLongFunction mapper);
```

Когда можно использовать java.lang.Integer:

Когда можно использовать java.lang.Integer:

- **20%**: List<Integer> - бенчмарки с лекций Алексея Шипилева.

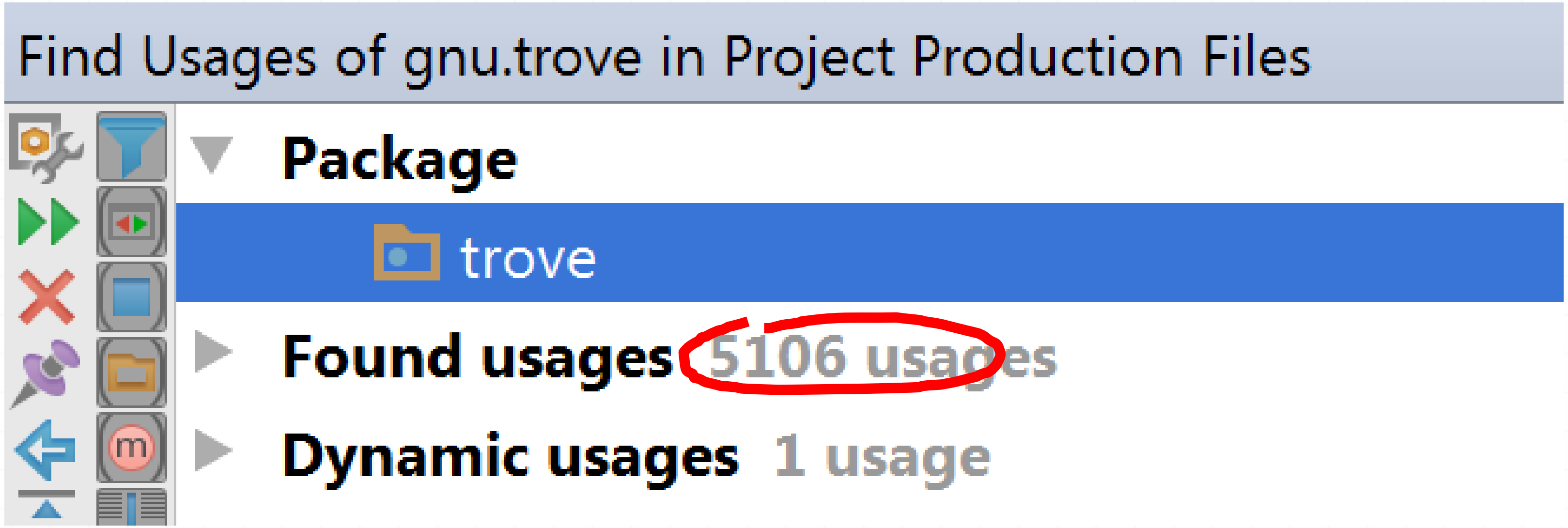
Когда можно использовать java.lang.Integer:

- **20%**: List<Integer> - бенчмарки с лекций Алексея Шипилева.
- **20%**: Stream<Integer> - бенчмарки с лекций Тагира Валеева.

Когда можно использовать java.lang.Integer:

- **20%**: List<Integer> - бенчмарки с лекций Алексея Шипилева.
- **20%**: Stream<Integer> - бенчмарки с лекций Тагира Валеева.
- Остальные **95%**: Ваша программа – отстой.

Trove: хорошие новости



Find Usages of gnu.trove in Project Production Files

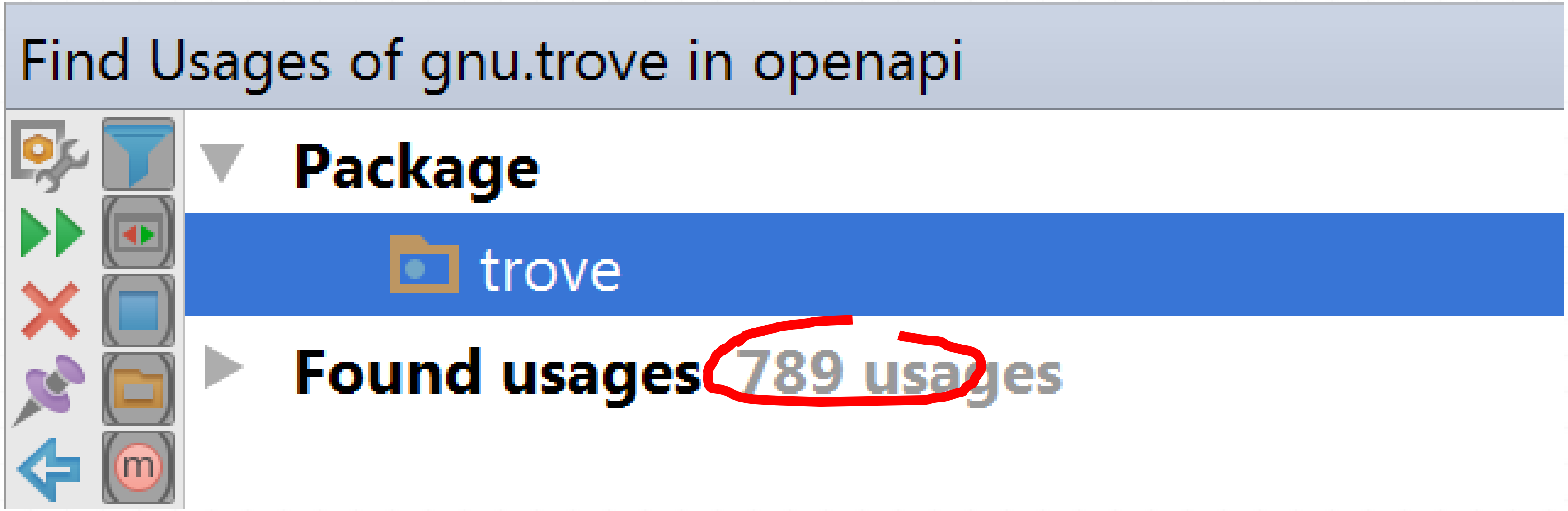
Package

trove

Found usages 5106 usages

Dynamic usages 1 usage

Trove: ужасные новости



Find Usages of gnu.trove in openapi

Package

trove

Found usages 789 usages

Bit hacks: когда boolean - толстый

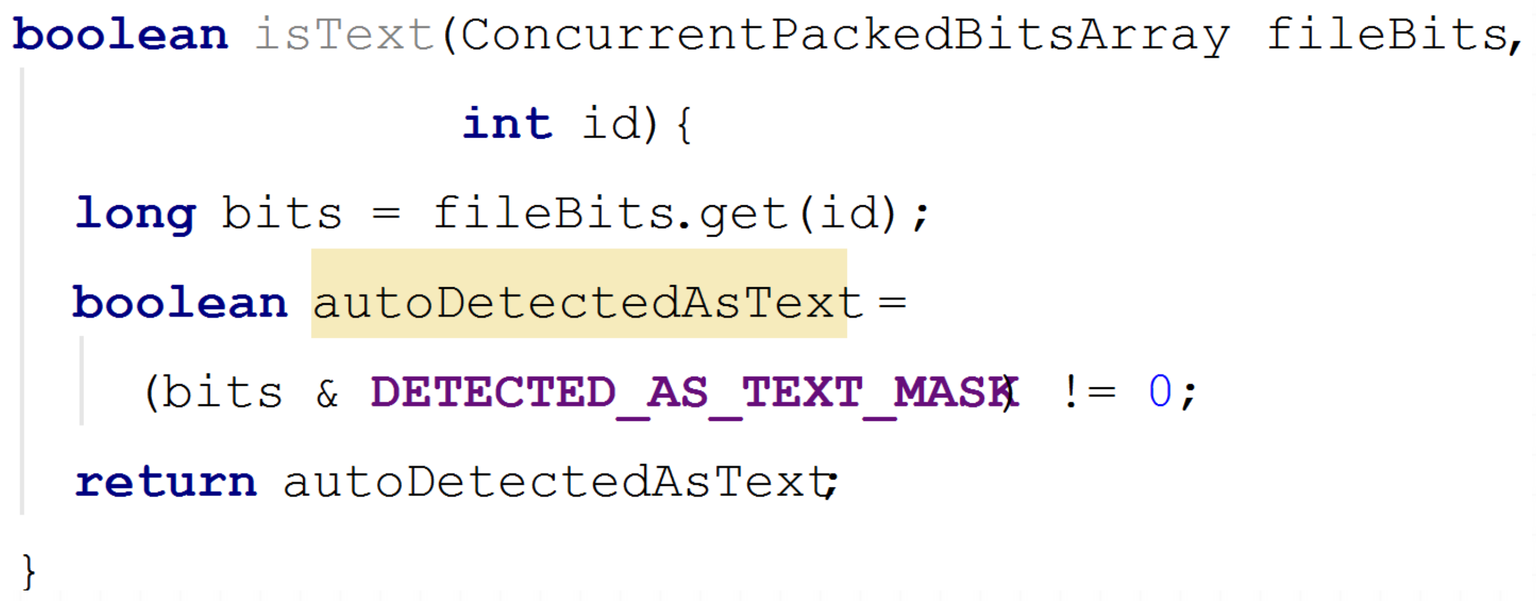
```
class FileTypeInfo {
  boolean autoDetectedAsText;
  boolean autoDetectedAsBinary;
  boolean autoDetectionWasRun;

  FileTypeInfo getFileInfo(int id) {
  void setFileInfo(int id, FileTypeInfo info)
```

Bit hacks: работаем руками

Txt0	**Bin0**	**Det0**	**Txt1**	**Bin1**	**Det1**	**Txt2**	**Bin2**
0	0	1	0	1	1	0	0
Flags0			Flags1			Flags2…	

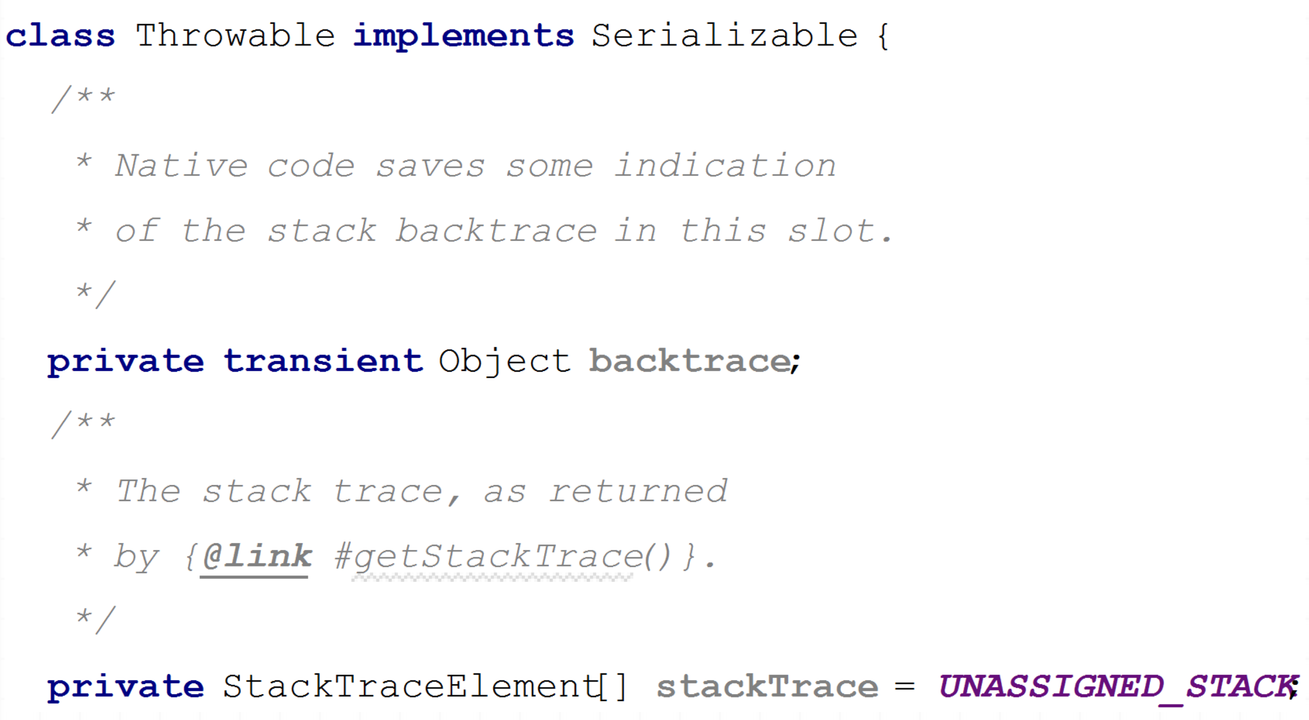
Bit hacks: работаем руками

Txt0	Bin0	Det0	Txt1	Bin1	Det1	Txt2	Bin2
0	0	1	0	1	1	0	0
Flags0			Flags1			Flags2…	



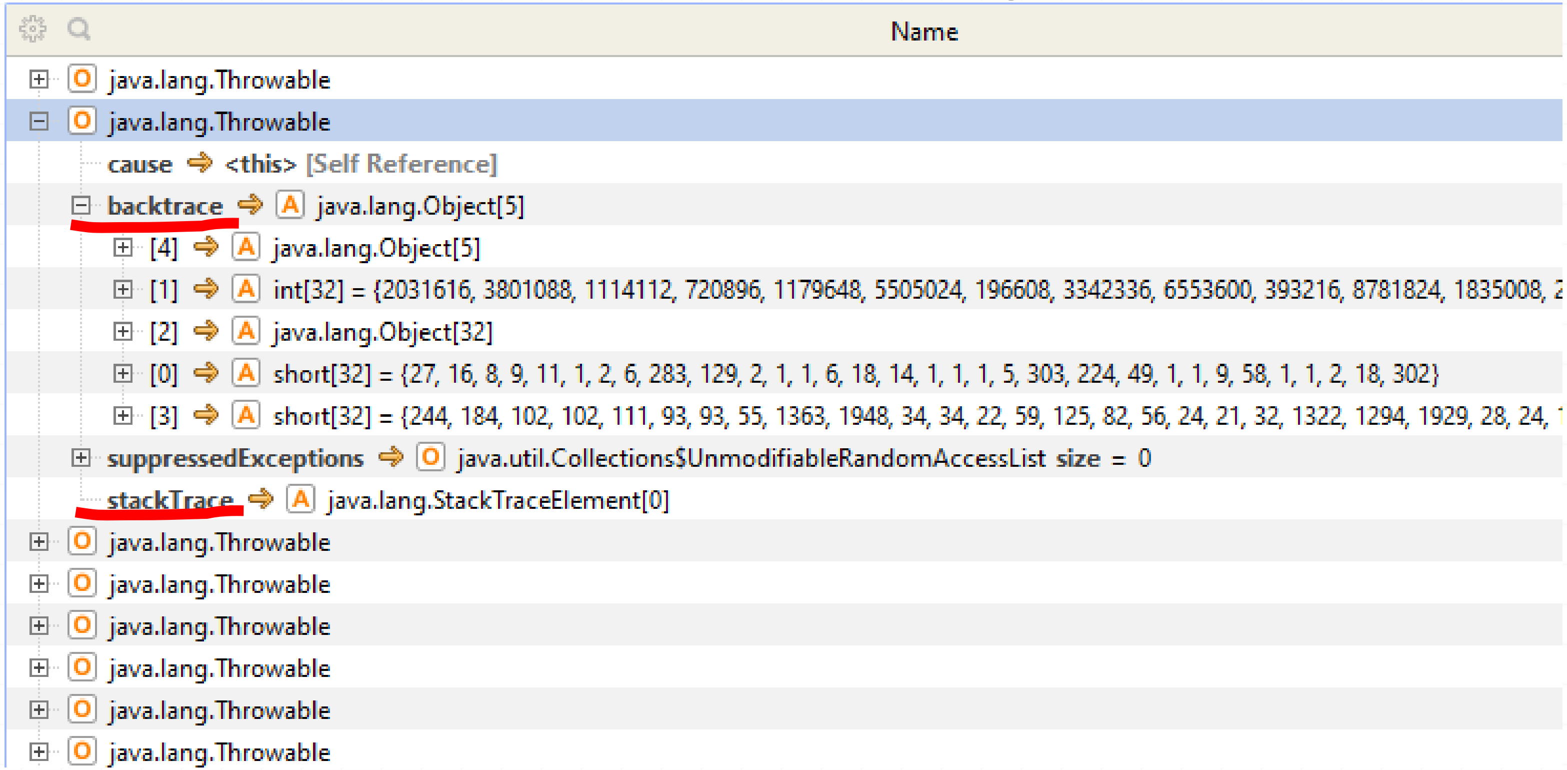
```
boolean isText(ConcurrentPackedBitsArray fileBits,
                int id) {
  long bits = fileBits.get(id);
  boolean autoDetectedAsText =
     (bits & DETECTED_AS_TEXT_MASK) != 0;
  return autoDetectedAsText;
}
```

Throwable – это жирный класс



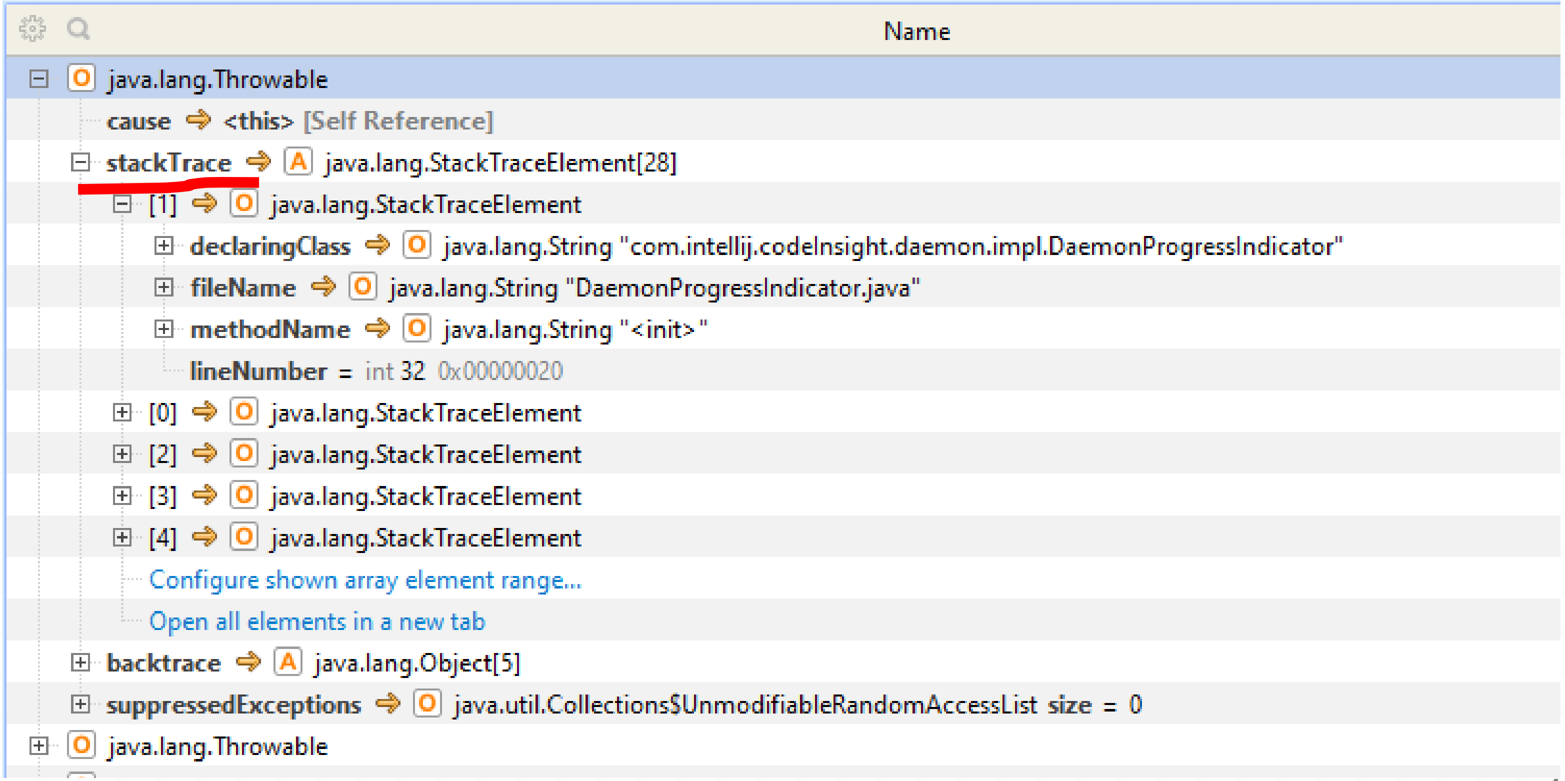
```
class Throwable implements Serializable {
  /**
   * Native code saves some indication
   * of the stack backtrace in this slot.
   */
  private transient Object backtrace;
  /**
   * The stack trace, as returned
   * by {@link #getStackTrace()}.
   */
  private StackTraceElement[] stackTrace = UNASSIGNED_STACK;
```

Throwable – это хитрый класс



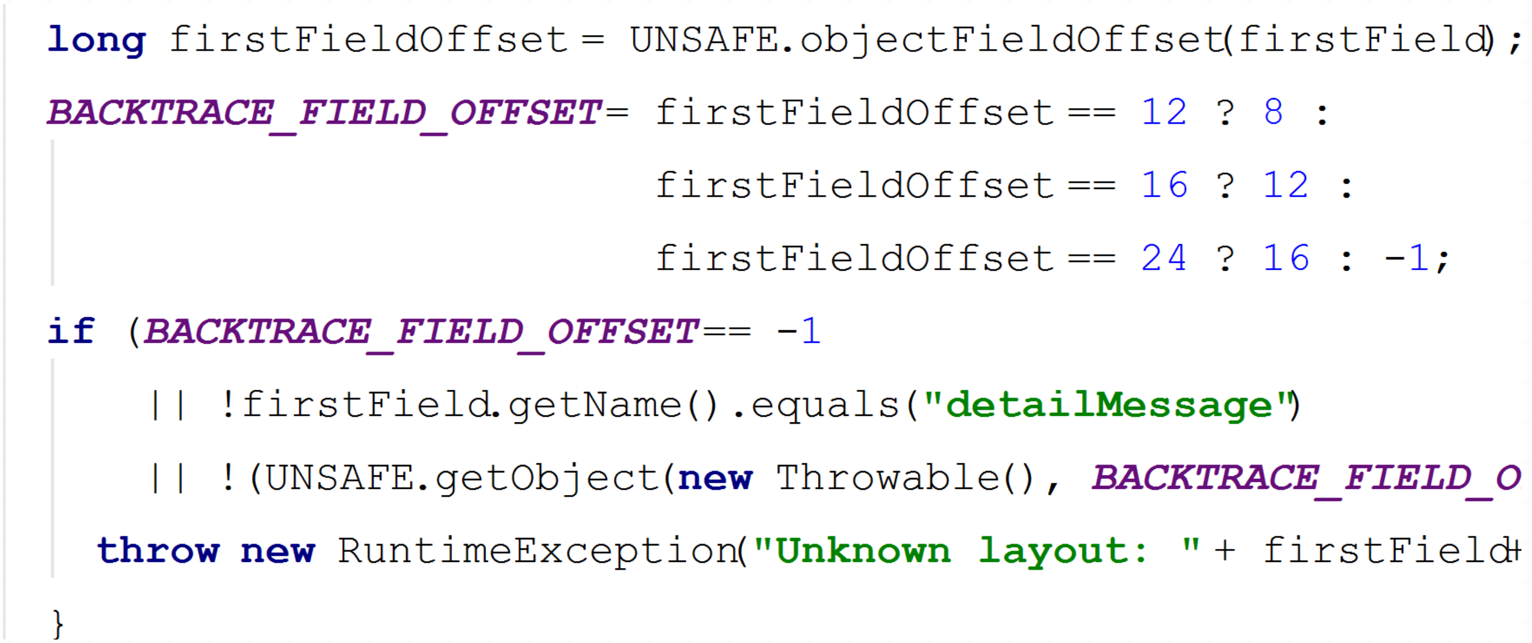
```
Name
⊞ O java.lang.Throwable
⊟ O java.lang.Throwable
    cause ⇨ <this> [Self Reference]
  ⊟ backtrace ⇨ A java.lang.Object[5]
      ⊞ [4] ⇨ A java.lang.Object[5]
      ⊞ [1] ⇨ A int[32] = {2031616, 3801088, 1114112, 720896, 1179648, 5505024, 196608, 3342336, 6553600, 393216, 8781824, 1835008, 2
      ⊞ [2] ⇨ A java.lang.Object[32]
      ⊞ [0] ⇨ A short[32] = {27, 16, 8, 9, 11, 1, 2, 6, 283, 129, 2, 1, 1, 6, 18, 14, 1, 1, 1, 5, 303, 224, 49, 1, 1, 9, 58, 1, 1, 2, 18, 302}
      ⊞ [3] ⇨ A short[32] = {244, 184, 102, 102, 111, 93, 93, 55, 1363, 1948, 34, 34, 22, 59, 125, 82, 56, 24, 21, 32, 1322, 1294, 1929, 28, 24, 1
  ⊞ suppressedExceptions ⇨ O java.util.Collections$UnmodifiableRandomAccessList size = 0
    stackTrace ⇨ A java.lang.StackTraceElement[0]
⊞ O java.lang.Throwable
⊞ O java.lang.Throwable
⊞ O java.lang.Throwable
⊞ O java.lang.Throwable
⊞ O java.lang.Throwable
⊞ O java.lang.Throwable
```

Throwable – это коварный класс



```
Name
⊟ java.lang.Throwable
    cause ⇨ <this> [Self Reference]
    ⊟ stackTrace ⇨ java.lang.StackTraceElement[28]
        ⊟ [1] ⇨ java.lang.StackTraceElement
            ⊞ declaringClass ⇨ java.lang.String "com.intellij.codeInsight.daemon.impl.DaemonProgressIndicator"
            ⊞ fileName ⇨ java.lang.String "DaemonProgressIndicator.java"
            ⊞ methodName ⇨ java.lang.String "<init>"
              lineNumber = int 32  0x00000020
        ⊞ [0] ⇨ java.lang.StackTraceElement
        ⊞ [2] ⇨ java.lang.StackTraceElement
        ⊞ [3] ⇨ java.lang.StackTraceElement
        ⊞ [4] ⇨ java.lang.StackTraceElement
          Configure shown array element range...
          Open all elements in a new tab
    ⊞ backtrace ⇨ java.lang.Object[5]
    ⊞ suppressedExceptions ⇨ java.util.Collections$UnmodifiableRandomAccessList size = 0
⊞ java.lang.Throwable
```

Throwable – это отчаяние

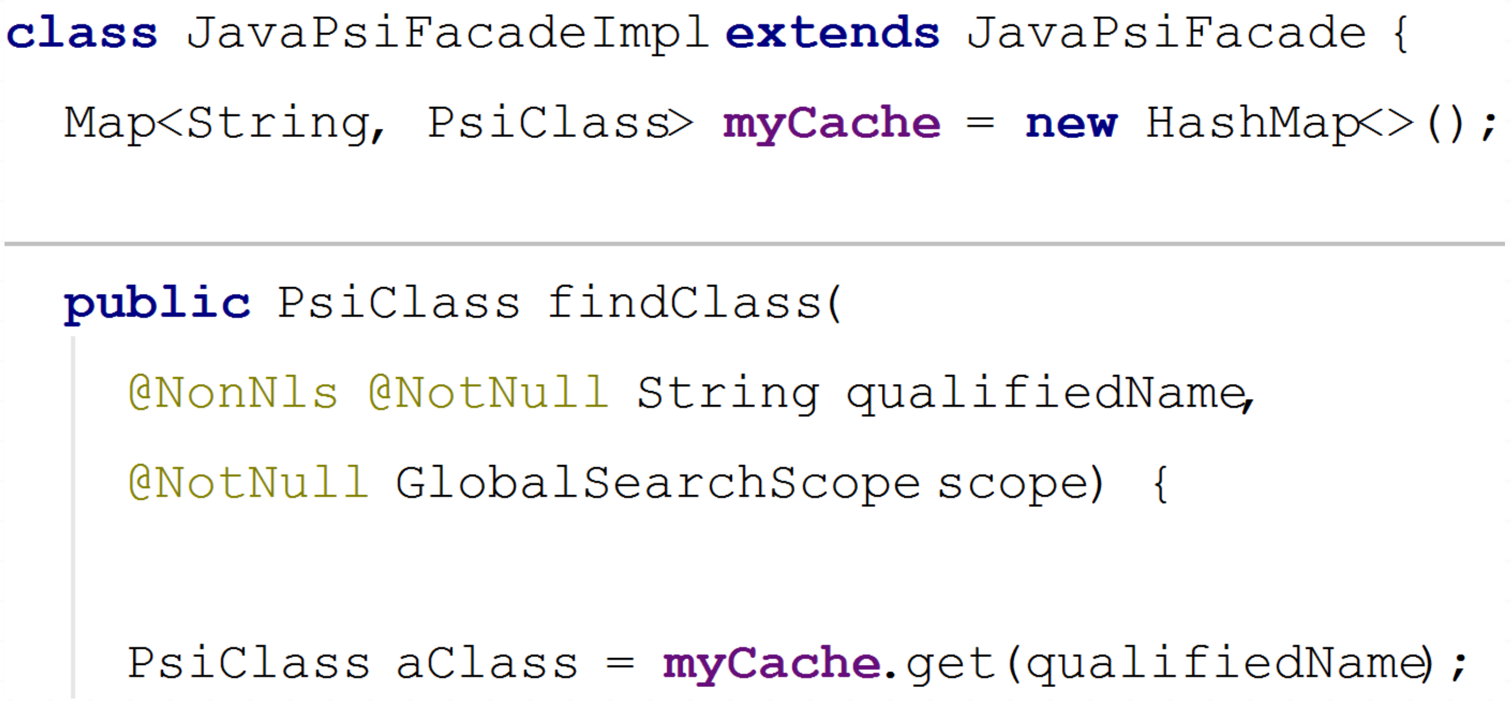


```
long firstFieldOffset = UNSAFE.objectFieldOffset(firstField);
BACKTRACE_FIELD_OFFSET = firstFieldOffset == 12 ? 8 :
                         firstFieldOffset == 16 ? 12 :
                         firstFieldOffset == 24 ? 16 : -1;
if (BACKTRACE_FIELD_OFFSET == -1
    || !firstField.getName().equals("detailMessage")
    || !(UNSAFE.getObject(new Throwable(), BACKTRACE_FIELD_O
  throw new RuntimeException("Unknown layout: " + firstField+
}
```

Cachification: прививка от тормозов

```java
class JavaPsiFacadeImpl extends JavaPsiFacade {
  /**
   * Searches the specified scope within the project
   * for a class with the specified full-qualified
   * name and returns one if it is found.
   */
  public PsiClass findClass(
    @NonNls @NotNull String qualifiedName,
    @NotNull GlobalSearchScope scope) {

    for (PsiElementFinder finder : finders()) {
```

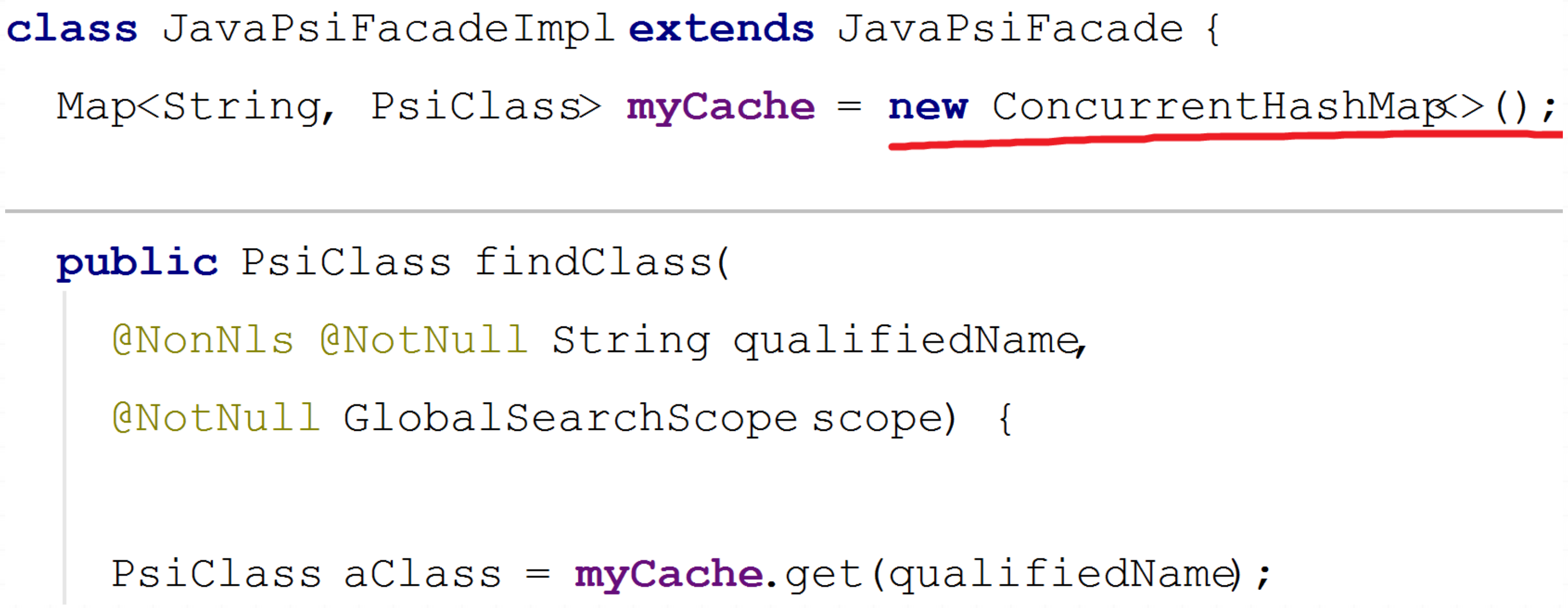
Cachification: да не вопрос



```java
class JavaPsiFacadeImpl extends JavaPsiFacade {
  Map<String, PsiClass> myCache = new HashMap<>();
```

```java
  public PsiClass findClass(
    @NonNls @NotNull String qualifiedName,
    @NotNull GlobalSearchScope scope) {

    PsiClass aClass = myCache.get(qualifiedName);
```

Cachification: Concurrency?



```
class JavaPsiFacadeImpl extends JavaPsiFacade {
  Map<String, PsiClass> myCache = new ConcurrentHashMap<>();

  public PsiClass findClass(
    @NonNls @NotNull String qualifiedName,
    @NotNull GlobalSearchScope scope) {

    PsiClass aClass = myCache.get(qualifiedName);
```

Cachification: Memory management?

```
class JavaPsiFacadeImpl extends JavaPsiFacade {
  Map<String, PsiClass> myCache =
    ContainerUtil.createConcurrentWeakKeySoftValueMap();
```

```
  public PsiClass findClass(
    @NonNls @NotNull String qualifiedName,
    @NotNull GlobalSearchScope scope) {

    PsiClass aClass = myCache.get(qualifiedName);
```

Cachification: Invalidation?

```
class JavaPsiFacadeImpl extends JavaPsiFacade {
  Map<String, PsiClass> myCache = new ConcurrentHashMap<>();

  public JavaPsiFacadeImpl(MessageBus bus, Project project) {
    bus.connect().subscribe(PsiModificationTracker.TOPIC,
                           () -> myCache.clear());
  }

  public PsiClass findClass(@NotNull String qualifiedName ,
                            @NotNull GlobalSearchScope scope) {
    PsiClass aClass = myCache.get(qualifiedName);
```

Cachification: More memory management?

```
class JavaPsiFacadeImpl extends JavaPsiFacade {
  Map<String, PsiClass> myCache = new ConcurrentHashMap<>();
```

```
  public JavaPsiFacadeImpl(MessageBus bus, Project project) {
    bus.connect().subscribe(PsiModificationTracker.TOPIC,
                                () -> myCache.clear());
    LowMemoryWatcher.register(() -> myCache.clear(), project);
  }
```

```
  public PsiClass findClass(@NotNull String qualifiedName ,
                            @NotNull GlobalSearchScope scope) {
    PsiClass aClass = myCache.get(qualifiedName);
```

Cachification: славные итоги

Repeat

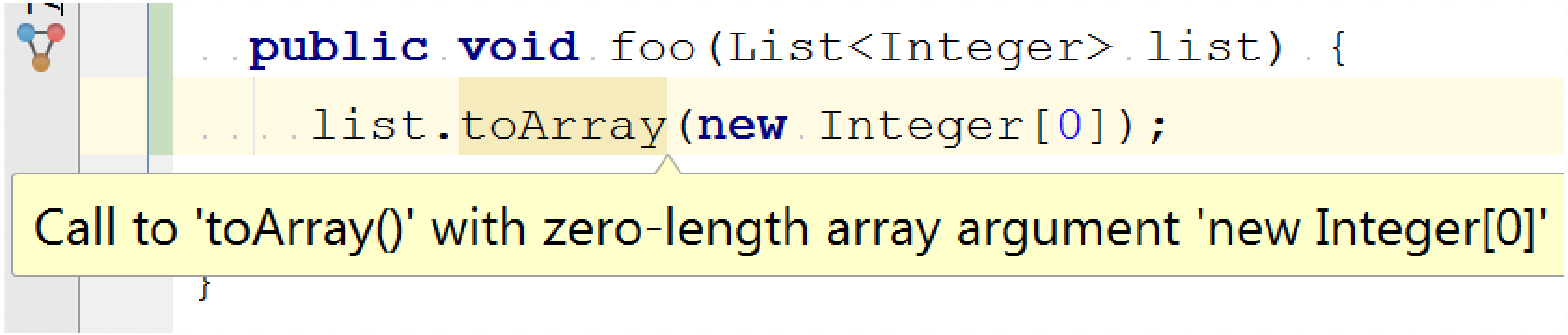
- 118 cache classes
- 19252 usages

Методология

shipilev.net/blog/2016/arrays-wisdom-ancients

```
List<Integer> list;
list.toArray(new Integer[list.size()]);
                    Vs.
list.toArray(new Integer[0]);
```

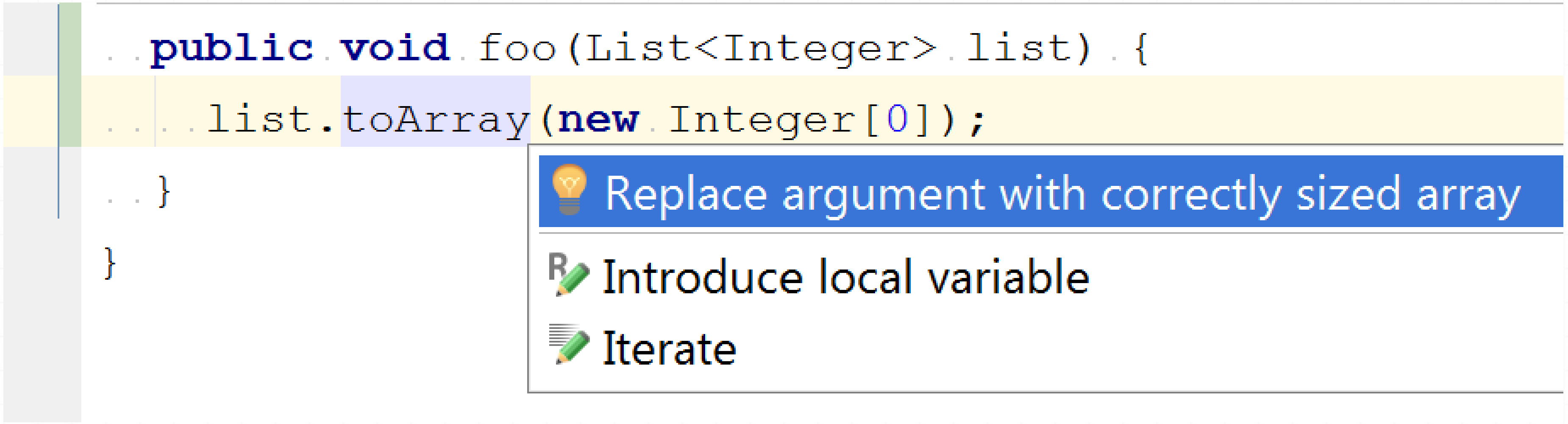
Методология



```
public void foo(List<Integer> list) {
    list.toArray(new Integer[0]);
}
```

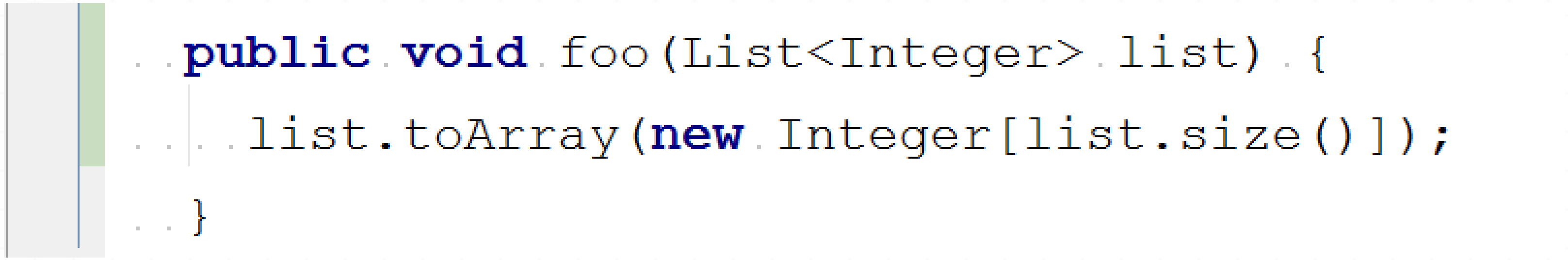
Call to 'toArray()' with zero-length array argument 'new Integer[0]'

Методология



```
  public void foo(List<Integer> list) {
      list.toArray(new Integer[0]);
  }
}
```

Методология



```
  public void foo(List<Integer> list) {
      list.toArray(new Integer[list.size()]);
  }
```

Методология

Мы - люди простые. Среднеквадратичные отклонения не высчитываем.

Методология

Перформанс - это
шоб не тормозило; ну и ваще.

Дезинфекция, Дератизация и Деквадратизация

AccidentallyQuadratic.tumblr.com

$$O(n^2)$$

Дезинфекция, Дератизация и Деквадратизация

AccidentallyQuadratic.tumblr.com

Dequadratisation: откуда берутся

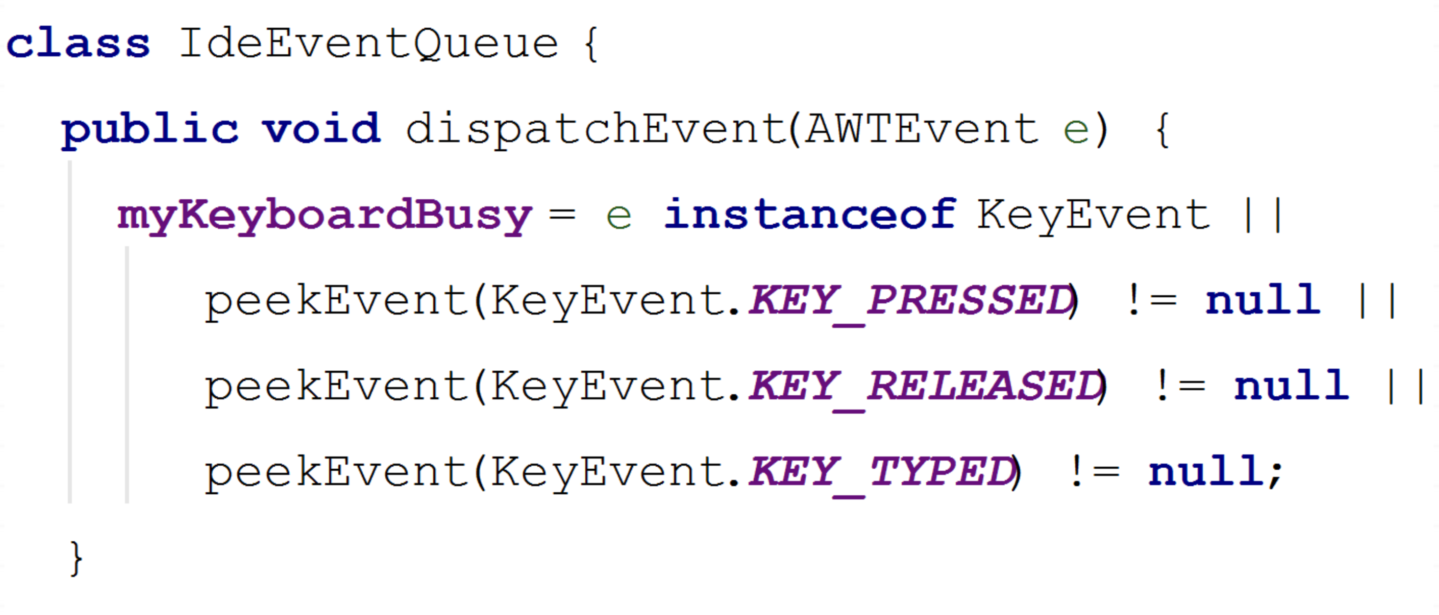
```
class MethodNamesDifferOnlyByCaseVisitor
  extends JavaElementVisitor {
  public void visitMethod (PsiMethod method)  {
    PsiClass aClass = method.getContainingClass();
    PsiMethod[] methods = aClass.getMethods();
    for (PsiMethod testMethod : methods)  {
      String name = testMethod.getName();
```

Dequadratisation: откуда берутся

```
class MethodNamesDifferOnlyByCaseVisitor
  extends JavaElementVisitor {
  public void visitMethod (PsiMethod method)  {
    PsiClass aClass = method.getContainingClass();
    PsiMethod[] methods = aClass.getMethods();
    for (PsiMethod testMethod : methods)  {
      String name = testMethod.getName();
```

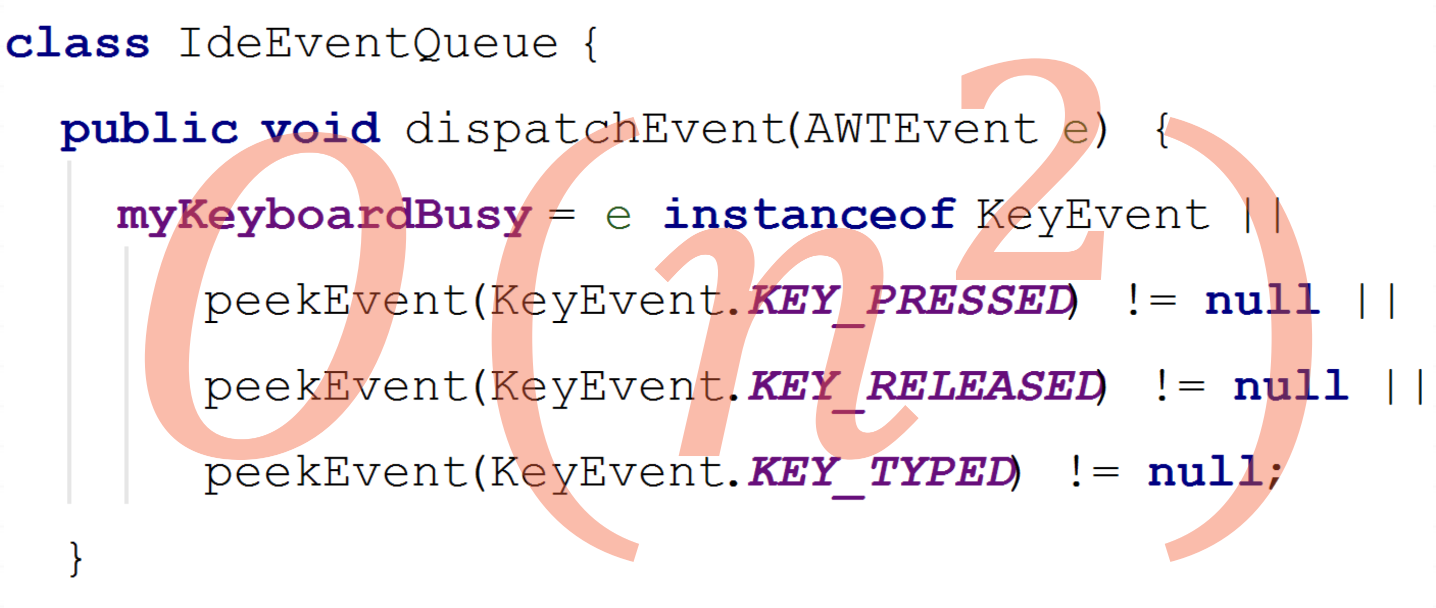
$$O(n^2)$$

Dequadratisation: да уйди



```
class IdeEventQueue {
  public void dispatchEvent(AWTEvent e) {
    myKeyboardBusy = e instanceof KeyEvent ||
        peekEvent(KeyEvent.KEY_PRESSED) != null ||
        peekEvent(KeyEvent.KEY_RELEASED) != null ||
        peekEvent(KeyEvent.KEY_TYPED) != null;
  }
```

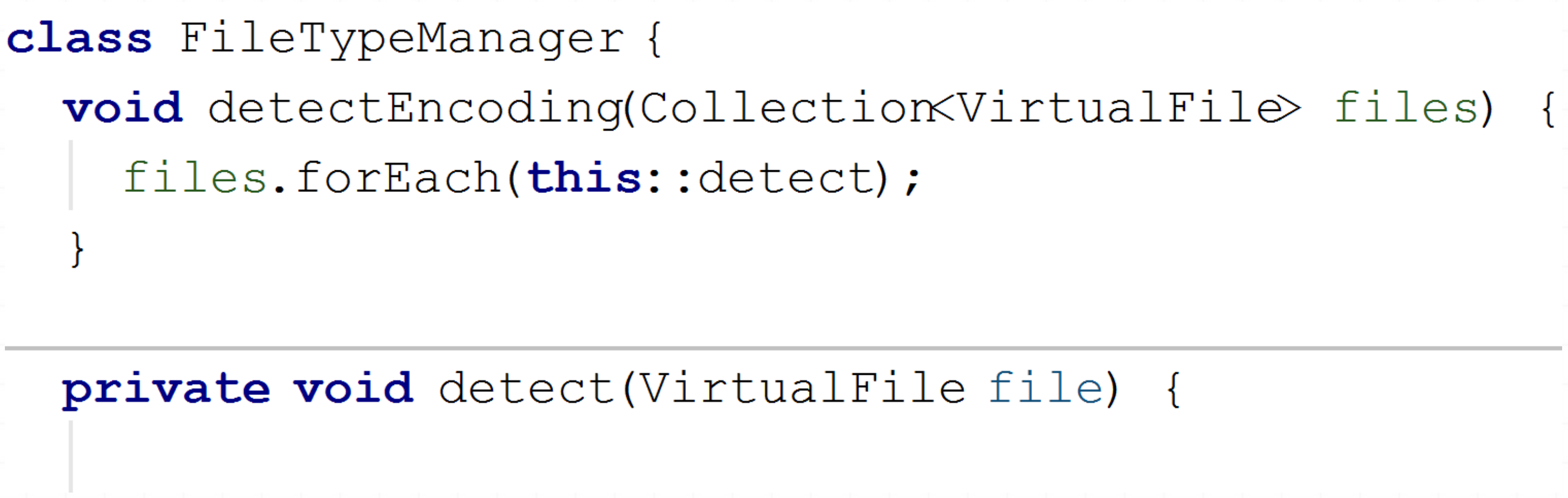
Dequadratisation: да уйди



```
class IdeEventQueue {
  public void dispatchEvent(AWTEvent e) {
    myKeyboardBusy = e instanceof KeyEvent ||
      peekEvent(KeyEvent.KEY_PRESSED) != null ||
      peekEvent(KeyEvent.KEY_RELEASED) != null ||
      peekEvent(KeyEvent.KEY_TYPED) != null;
  }
```

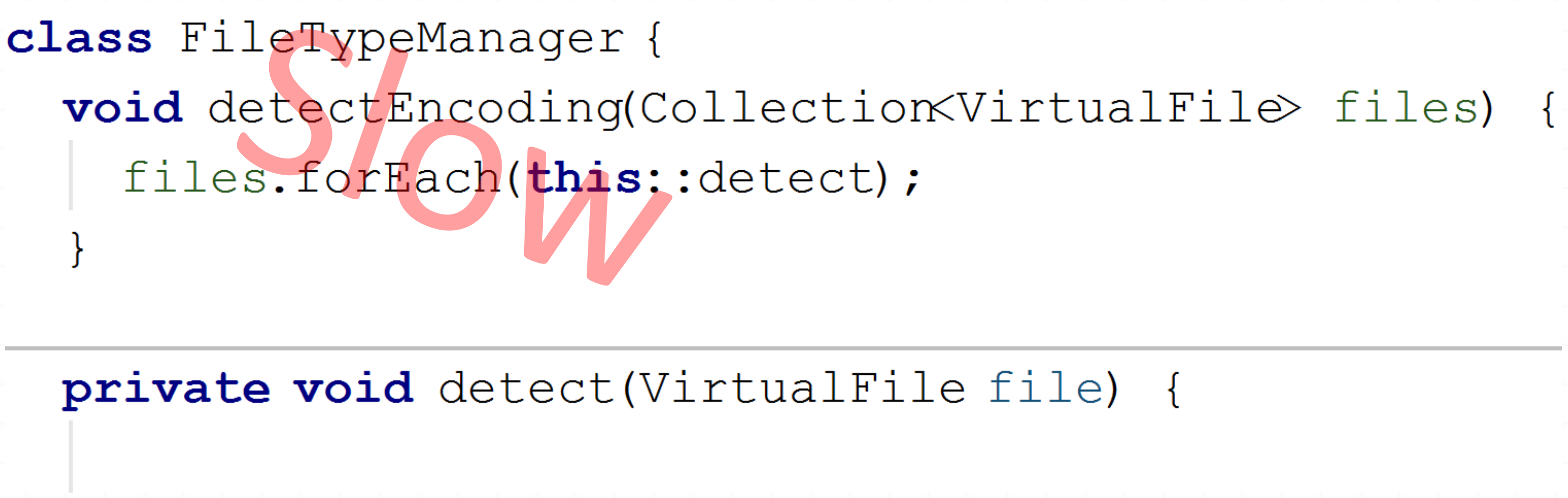
$O(n^2)$

Parallelization: наивняк



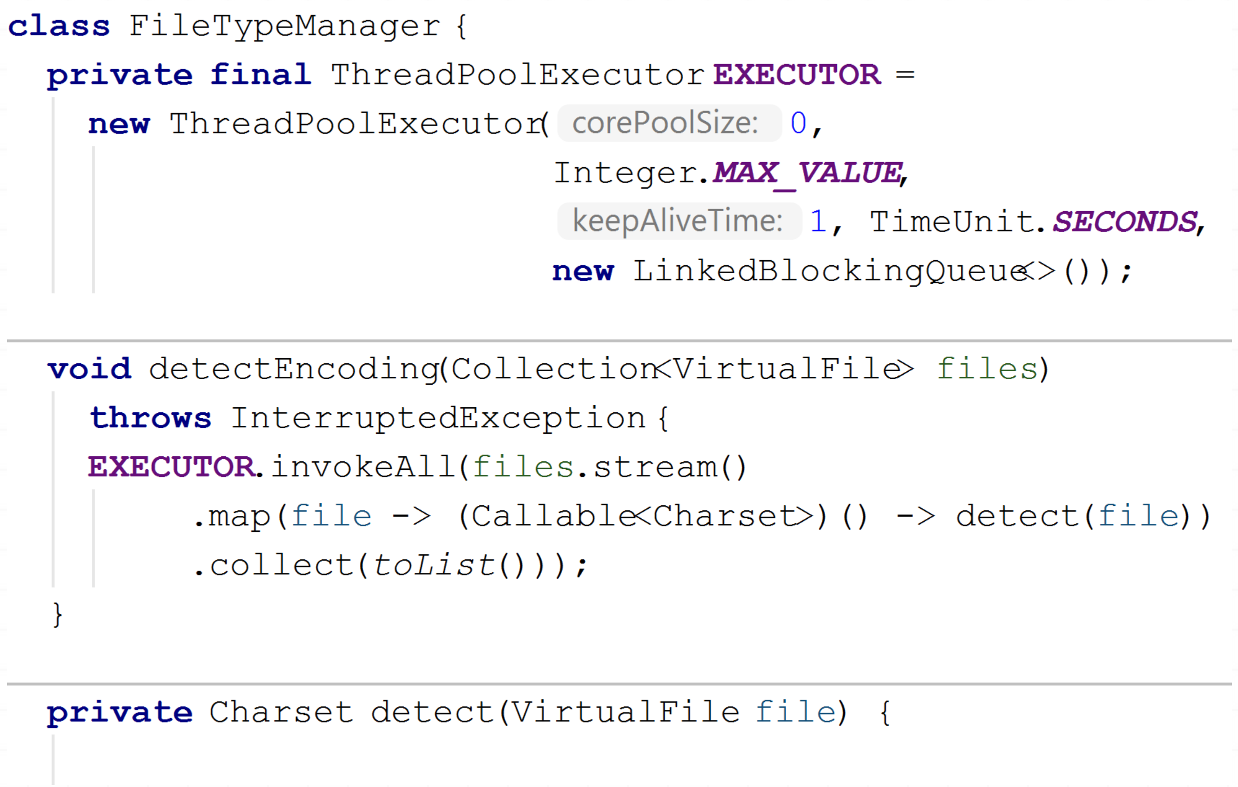
```
class FileTypeManager {
  void detectEncoding(Collection<VirtualFile> files) {
    files.forEach(this::detect);
  }

  private void detect(VirtualFile file) {

```

Parallelization: наивняк



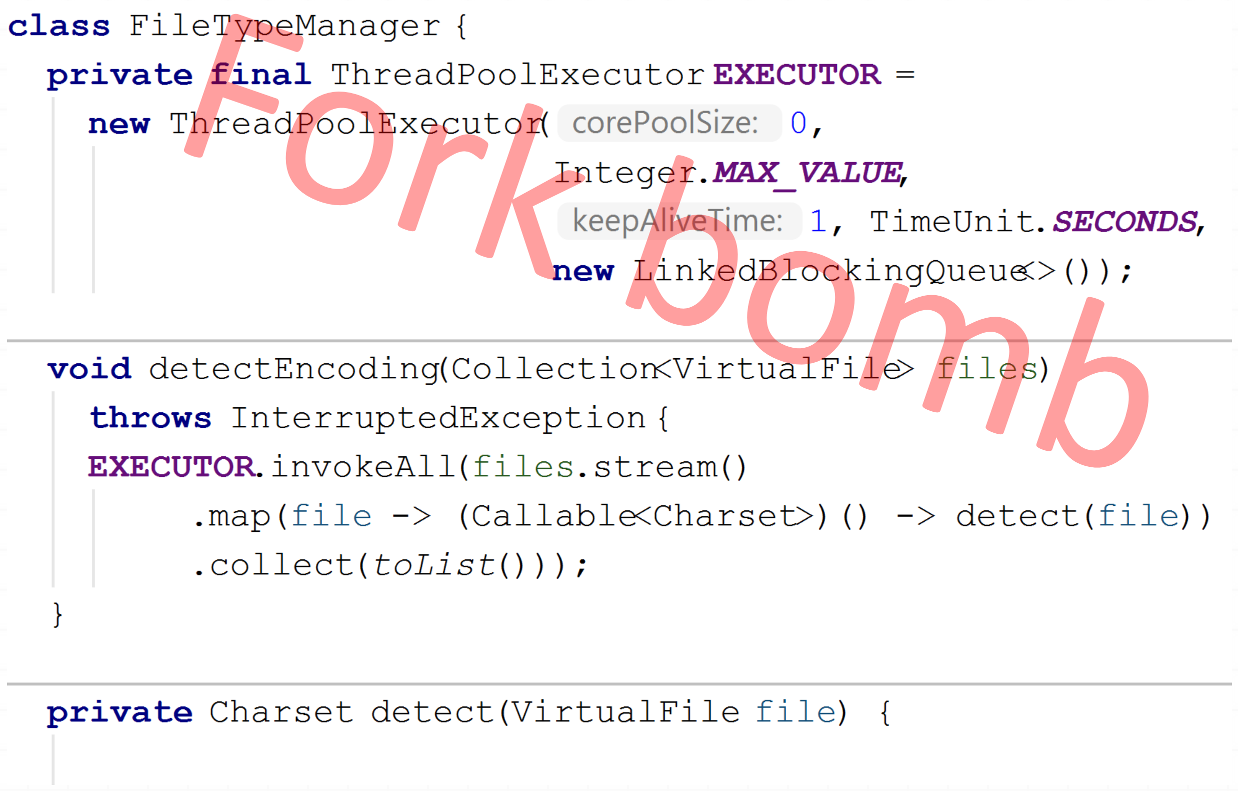
```
class FileTypeManager {
  void detectEncoding(Collection<VirtualFile> files) {
    files.forEach(this::detect);
  }

  private void detect(VirtualFile file) {
```

Parallelization: из пушки по мухам



```
class FileTypeManager {
  private final ThreadPoolExecutor EXECUTOR =
    new ThreadPoolExecutor( corePoolSize: 0,
                           Integer.MAX_VALUE,
                           keepAliveTime: 1, TimeUnit.SECONDS,
                           new LinkedBlockingQueue<>());

  void detectEncoding(Collection<VirtualFile> files)
    throws InterruptedException {
    EXECUTOR.invokeAll(files.stream()
          .map(file -> (Callable<Charset>) () -> detect(file))
          .collect(toList()));
  }

  private Charset detect(VirtualFile file)  {
```

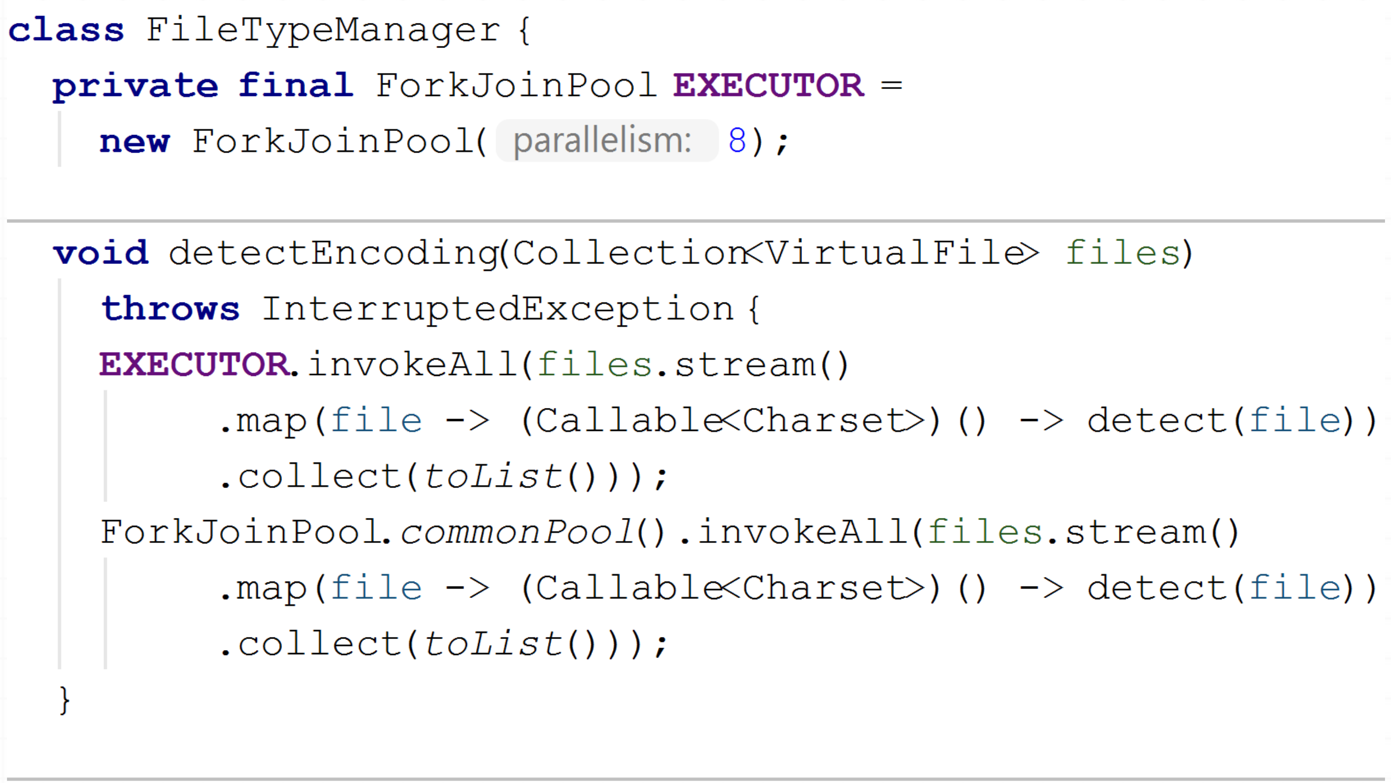
Parallelization: из пушки по мухам



```
class FileTypeManager {
  private final ThreadPoolExecutor EXECUTOR =
    new ThreadPoolExecutor( corePoolSize: 0,
                           Integer.MAX_VALUE,
                           keepAliveTime: 1, TimeUnit.SECONDS,
                           new LinkedBlockingQueue<>());

  void detectEncoding(Collection<VirtualFile> files)
    throws InterruptedException {
    EXECUTOR.invokeAll(files.stream()
         .map(file -> (Callable<Charset>) () -> detect(file))
         .collect(toList()));
  }

  private Charset detect(VirtualFile file)  {
```

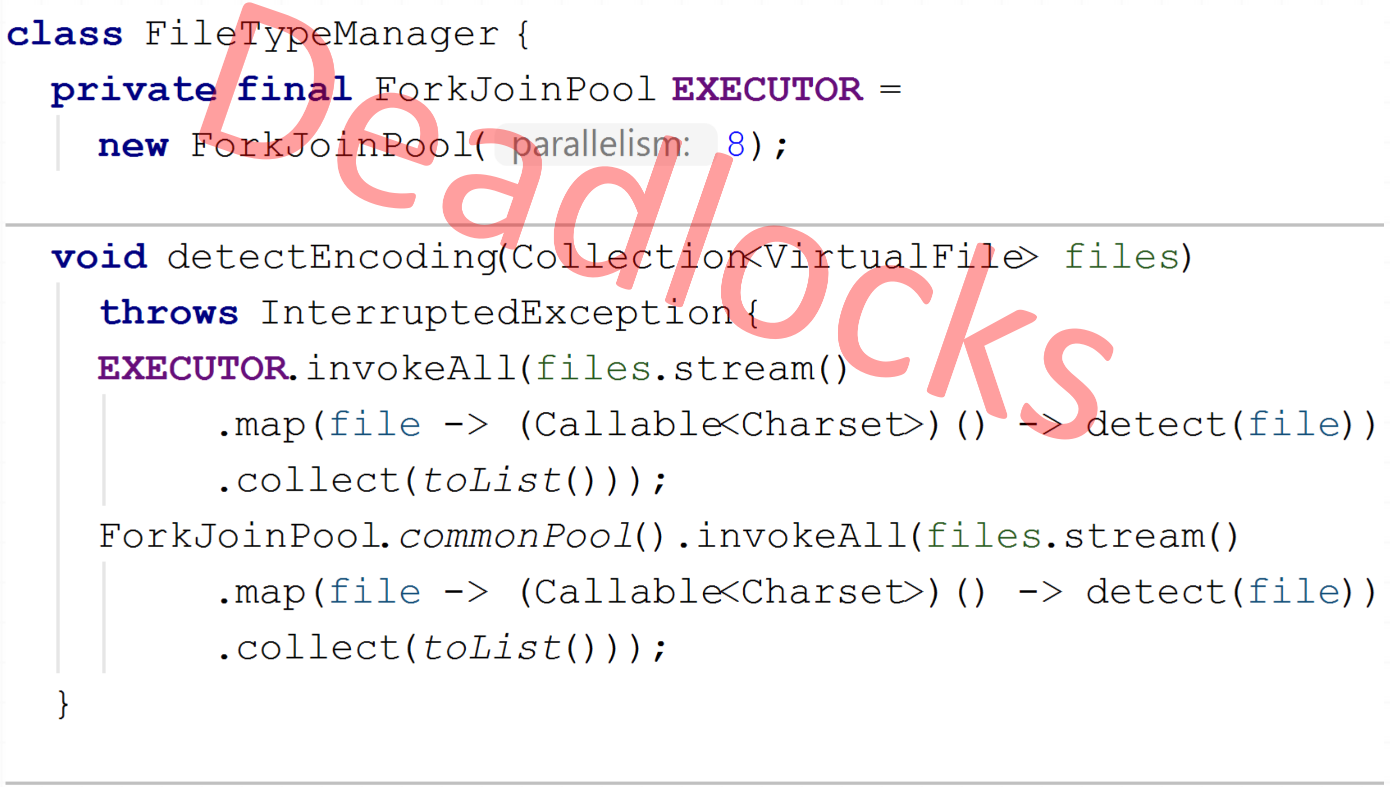
Fork bomb

Parallelization: из стримов по мухам



```
class FileTypeManager {
  private final ForkJoinPool EXECUTOR =
    new ForkJoinPool( parallelism: 8);

  void detectEncoding(Collection<VirtualFile> files)
    throws InterruptedException {
    EXECUTOR.invokeAll(files.stream()
          .map(file -> (Callable<Charset>) () -> detect(file))
          .collect(toList()));
    ForkJoinPool.commonPool().invokeAll(files.stream()
          .map(file -> (Callable<Charset>) () -> detect(file))
          .collect(toList()));
  }
```

Parallelization: из стримов по мухам



```
class FileTypeManager {
  private final ForkJoinPool EXECUTOR =
    new ForkJoinPool( parallelism: 8);

  void detectEncoding(Collection<VirtualFile> files)
    throws InterruptedException {
    EXECUTOR.invokeAll(files.stream()
          .map(file -> (Callable<Charset>) () -> detect(file))
          .collect(toList()));
    ForkJoinPool.commonPool().invokeAll(files.stream()
          .map(file -> (Callable<Charset>) () -> detect(file))
          .collect(toList()));
  }
```

Deadlocks

Parallelization: из пистолета по мухам

```
class FileTypeManager {
  private final ThreadPoolExecutor EXECUTOR =
    new ThreadPoolExecutor( corePoolSize: 0,
                            maximumPoolSize: 8,
                            keepAliveTime: 1, TimeUnit.SECONDS,
                           new LinkedBlockingQueue<>());

  void detectEncoding(Collection<VirtualFile> files)
    throws InterruptedException {
    EXECUTOR.invokeAll(files.stream()
          .map(file -> (Callable<Charset>) () -> detect(file))
          .collect(toList()));
  }
```

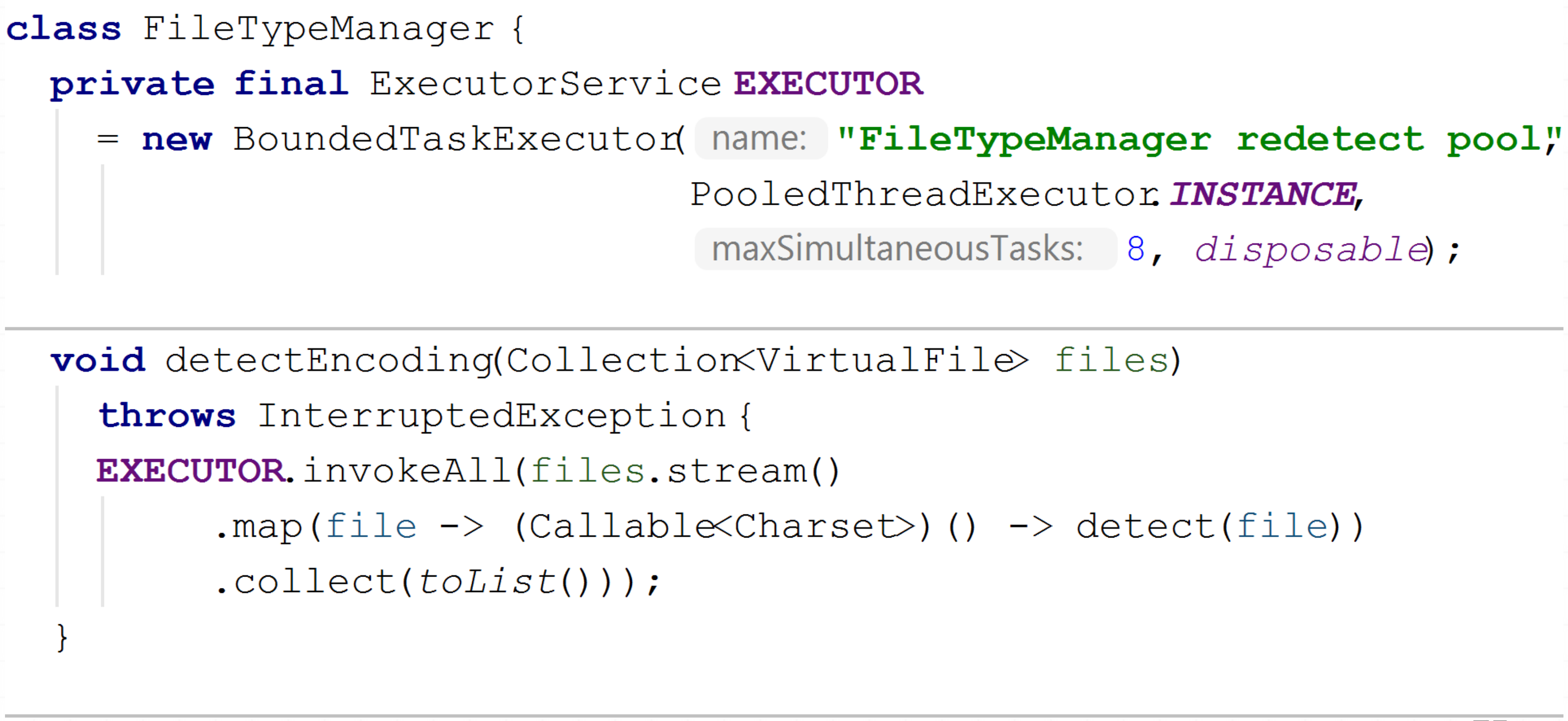
Parallelization: из пистолета по мухам

```
class FileTypeManager {
  private final ThreadPoolExecutor EXECUTOR =
    new ThreadPoolExecutor( corePoolSize: 0,
                            maximumPoolSize: 8,
                            keepAliveTime: 1, TimeUnit.SECONDS,
                            new LinkedBlockingQueue<>());

  void detectEncoding(Collection<VirtualFile> files)
    throws InterruptedException {
    EXECUTOR.invokeAll(files.stream()
          .map(file -> (Callable<Charset>) () -> detect(file))
          .collect(toList()));
  }
```

Tiny threads

Overhead

Parallelization: из лазера по мухам



```
class FileTypeManager {
  private final ExecutorService EXECUTOR
    = new BoundedTaskExecutor( name: "FileTypeManager redetect pool",
                                     PooledThreadExecutor.INSTANCE,
                                     maxSimultaneousTasks: 8, disposable);
```

```
  void detectEncoding(Collection<VirtualFile> files)
    throws InterruptedException {
    EXECUTOR.invokeAll(files.stream()
         .map(file -> (Callable<Charset>) () -> detect(file))
         .collect(toList()));
  }
```

Мораль

1. Cache!
   - Guava
   - WeakHashMap
   - CachedValue

Мораль

1. Cache!
2. Don’t cache!
   - Limit
   - Invalidation
   - Concurrency

Мораль

1. Cache!
2. Don’t cache!
3. Optimize!
   - Refactoring is hard
   - O(n^2) Pattern

Мораль

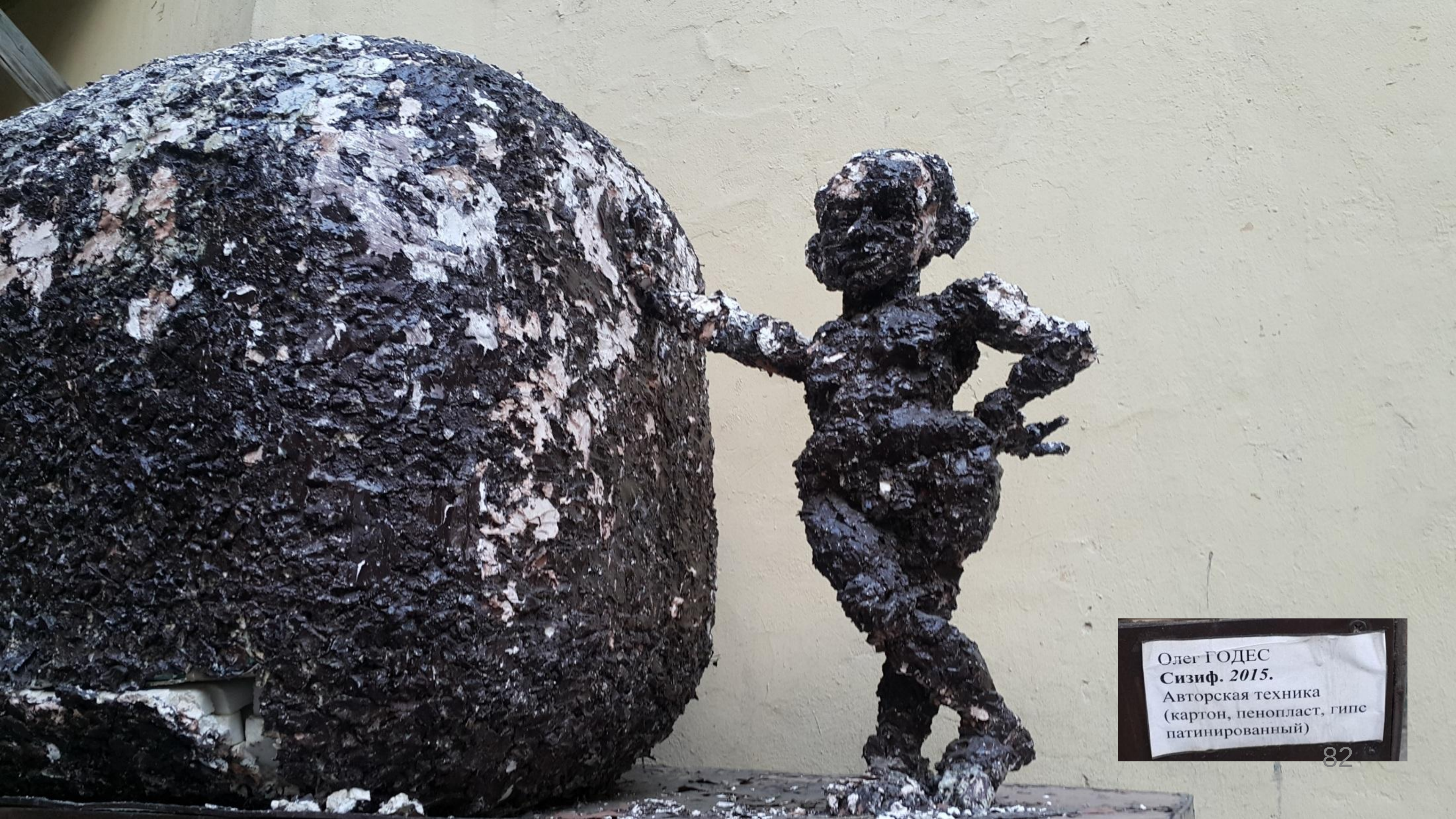
1. Cache!
2. Don’t cache!
3. Optimize!
4. Don’t optimize!
   - Profile

Мораль

1. Cache!
2. Don’t cache!
3. Optimize!
4. Don’t optimize!
5. Optimize for speed!
   - Memory is cheap
   - Example: Indices

Мораль

1. Cache!
2. Don’t cache!
3. Optimize!
4. Don’t optimize!
5. Optimize for speed!
6. Optimize for memory!
   - CPU is fast
   - Example: File type bit packing


Олег ГОДЕС
Сизиф. ***2015.***
Авторская техника
(картон, пенопласт, гипс
патинированный)

Alexey.Kudravtsev@jetbrains.com

github.com/JetBrains/intellij-community