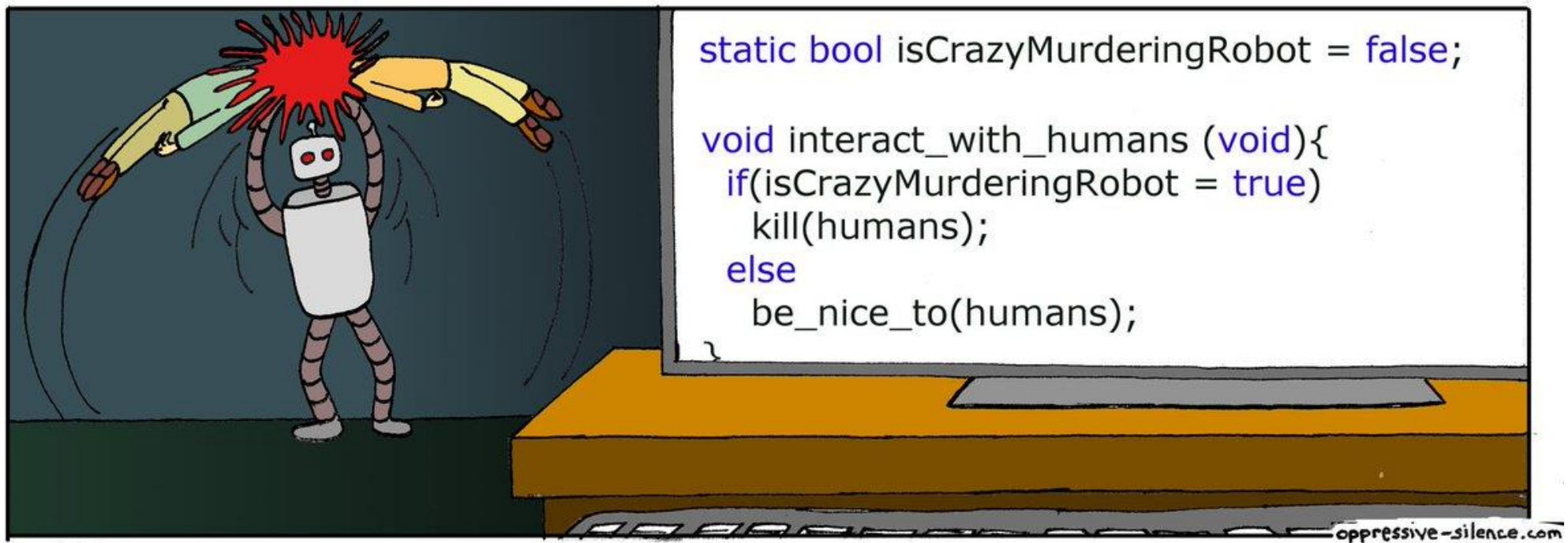
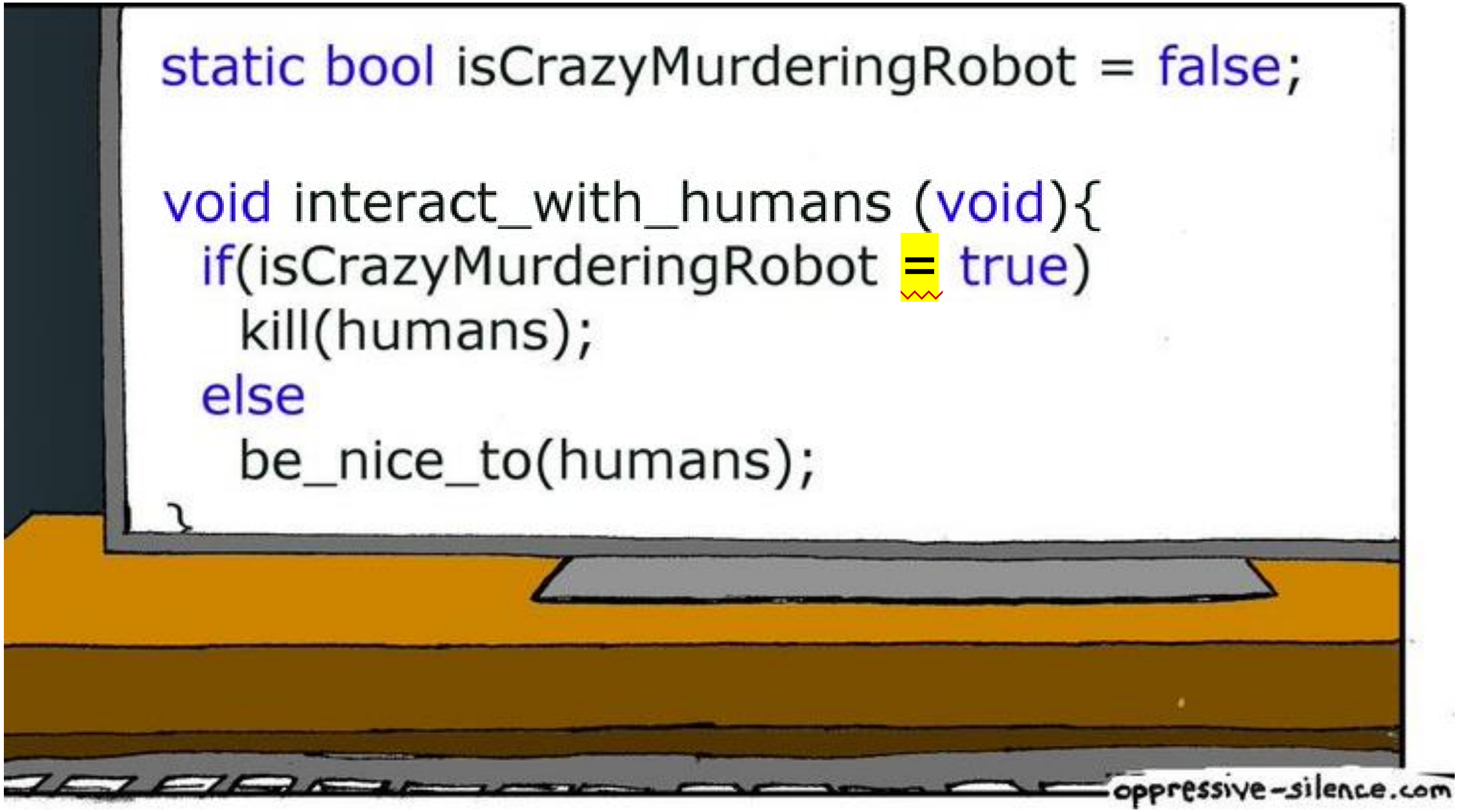
Java-инспекции в IntelliJ IDEA

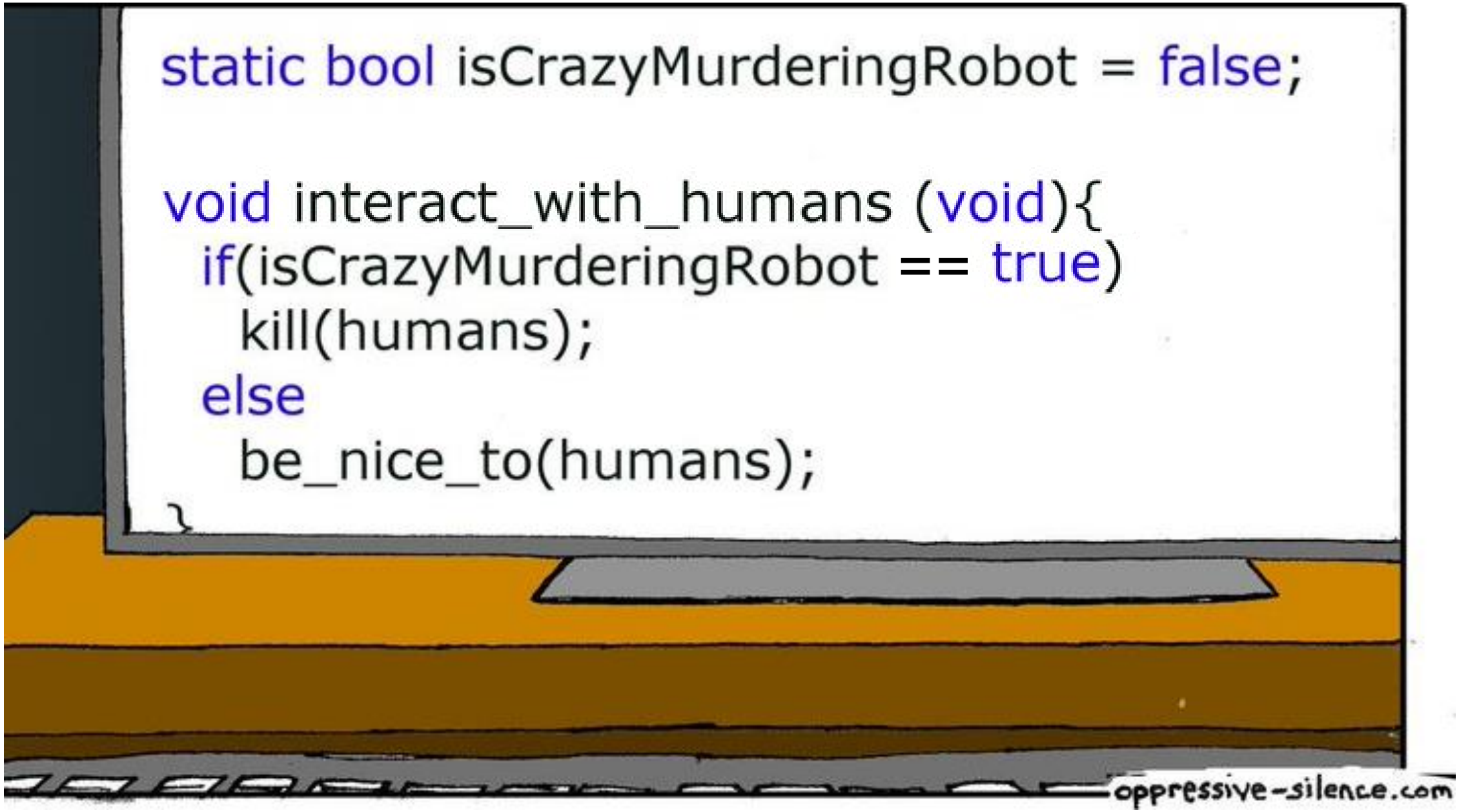
Что может пойти не так?

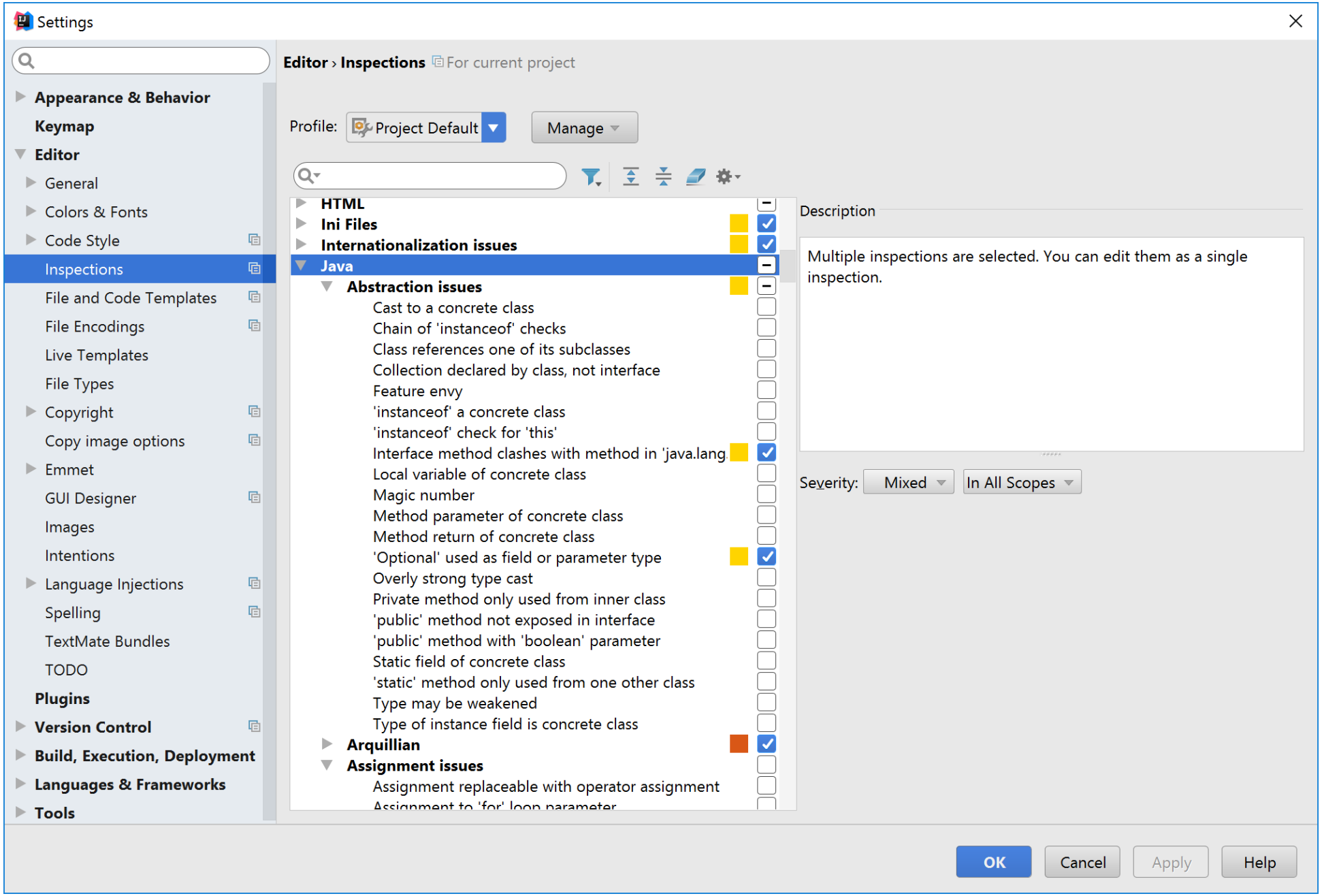
Тагир Валеев


OH NO! THE ROBOTS ARE KILLING US!!!
BUT WHY?!? WE NEVER PROGRAMMED THEM TO DO THIS!!!


static bool isCrazyMurderingRobot = false;
void interact_with_humans (void){
if(isCrazyMurderingRobot = true)
kill(humans);
else
be_nice_to(humans);
}
oppressive-silence.com


static bool isCrazyMurderingRobot = false;
void interact_with_humans (void){
if(isCrazyMurderingRobot = true)
kill(humans);
else
be_nice_to(humans);
}
oppressive-silence.com


static bool isCrazyMurderingRobot = false;
void interact_with_humans (void){
if(isCrazyMurderingRobot == true)
kill(humans);
else
be_nice_to(humans);
}
oppressive-silence.com


Settings
Editor › Inspections For current project
Appearance & Behavior
Keymap
Editor
General
Colors & Fonts
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Copyright
Copy image options
Emmet
GUI Designer
Images
Intentions
Language Injections
Spelling
TextMate Bundles
TODO
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Profile: Project Default
Manage
HTML
Ini Files
Internationalization issues
Java
Abstraction issues
Cast to a concrete class
Chain of 'instanceof' checks
Class references one of its subclasses
Collection declared by class, not interface
Feature envy
'instanceof' a concrete class
'instanceof' check for 'this'
Interface method clashes with method in 'java.lang.
Local variable of concrete class
Magic number
Method parameter of concrete class
Method return of concrete class
'Optional' used as field or parameter type
Overly strong type cast
Private method only used from inner class
'public' method not exposed in interface
'public' method with 'boolean' parameter
Static field of concrete class
'static' method only used from one other class
Type may be weakened
Type of instance field is concrete class
Arquillian
Assignment issues
Assignment replaceable with operator assignment
Description
Multiple inspections are selected. You can edit them as a single inspection.
Severity: Mixed
In All Scopes
OK
Cancel
Apply
Help

Поиск

—

The Simpsons
@Simpsons_tweets

Читать

Yep, here's your problem. Someone set this thing to "Evil." #TreehouseOfHorror

Язык твита: английский

РЕТВИТОВ 172

ОТМЕТКИ «НРАВИТСЯ» 252

Created by Vladimir Dolzhenko 16 Oct 2016 12:04 Updated by Tagir Valeev 02 Feb 2017 09:18 visible to: All Users

~~IDEA-162662~~ inspection: simplify obvious stream collect transformation to direct java.util equivalent

Relates to: × IDEA-163833

follow code snippet

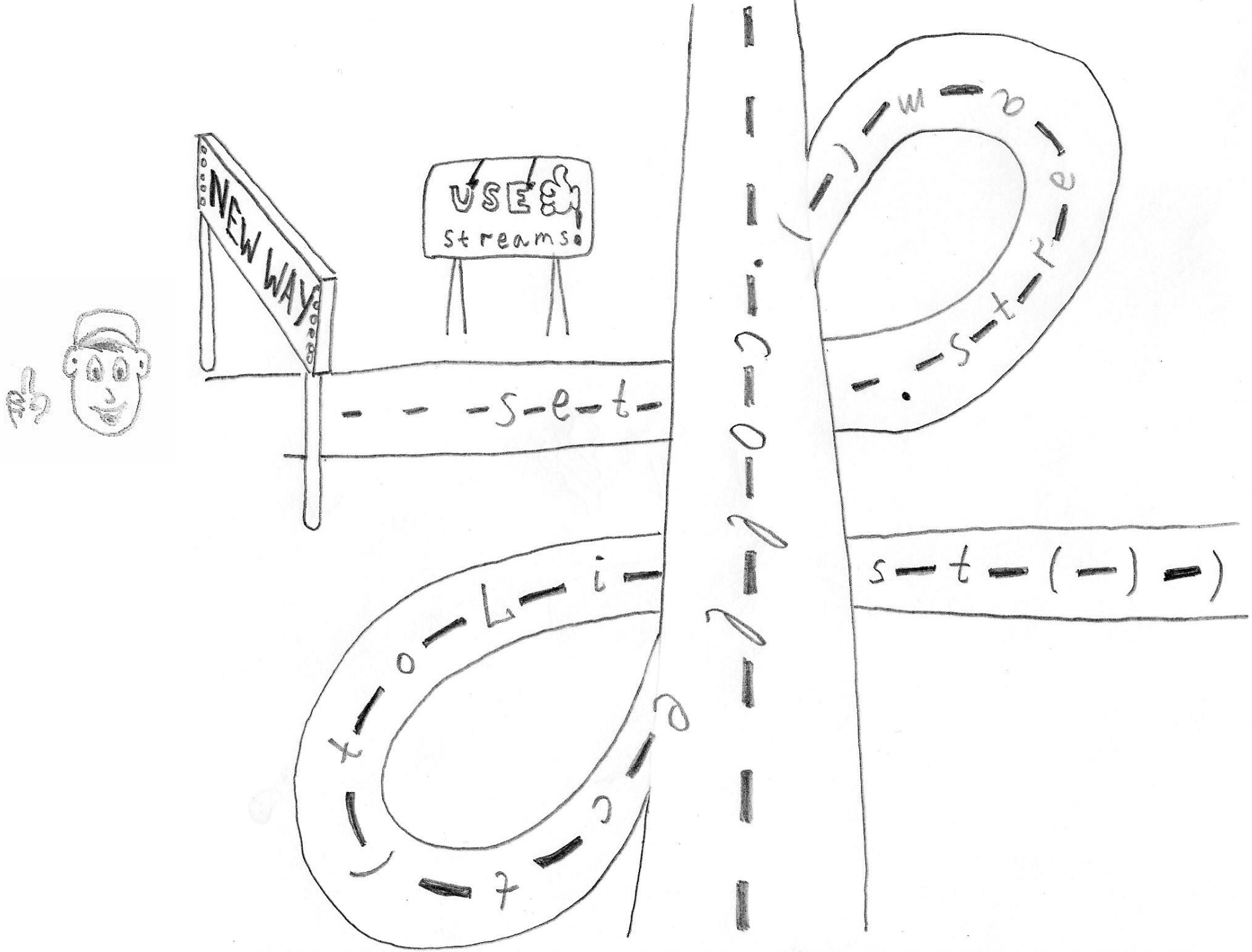
```
set.stream().collect(Collectors.toList())
```

could be replaced with

```
new ArrayList<>(set)
```

the reason for the simplification: avoid unnecessary intermediate stream objects and transformation


OLD WAY
new ArrayList<>(set)


NEW WAY
USE
streams

Created by Vladimir Dolzhenko 16 Oct 2016 12:04 Updated by Tagir Valeev 02 Feb 2017 09:18 visible to: All Users

~~IDEA-162662~~ inspection: simplify obvious stream collect transformation to direct java.util equivalent

Relates to: × IDEA-163833

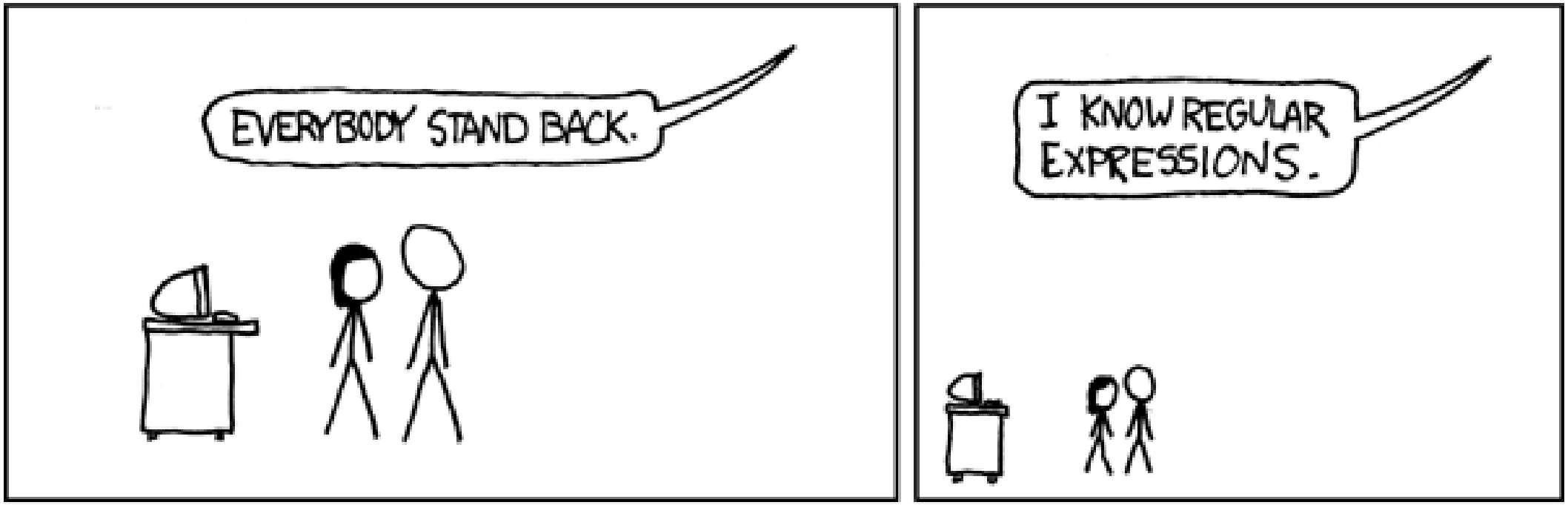
follow code snippet

```
set.stream().collect(Collectors.toList())
```

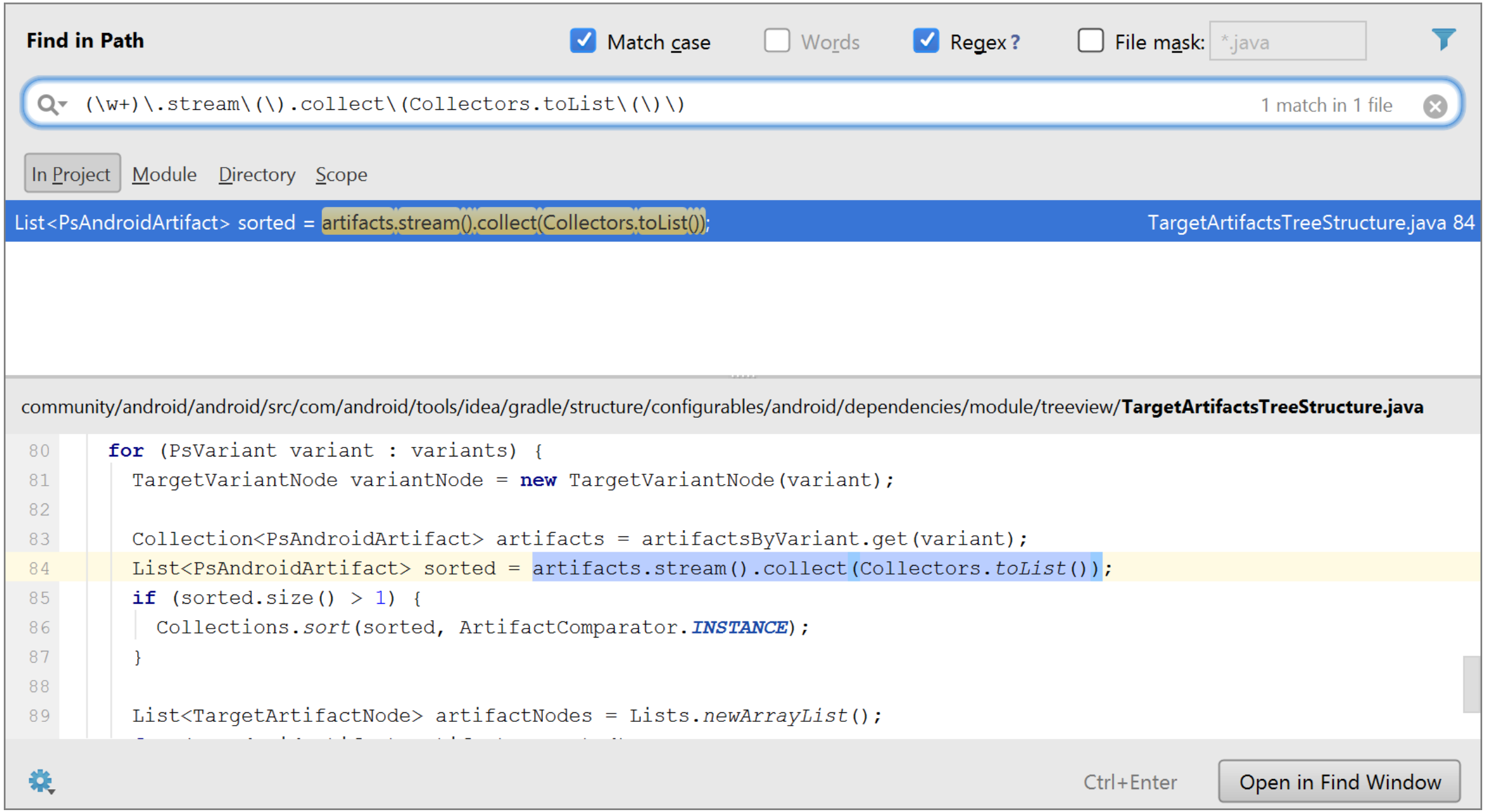
could be replaced with

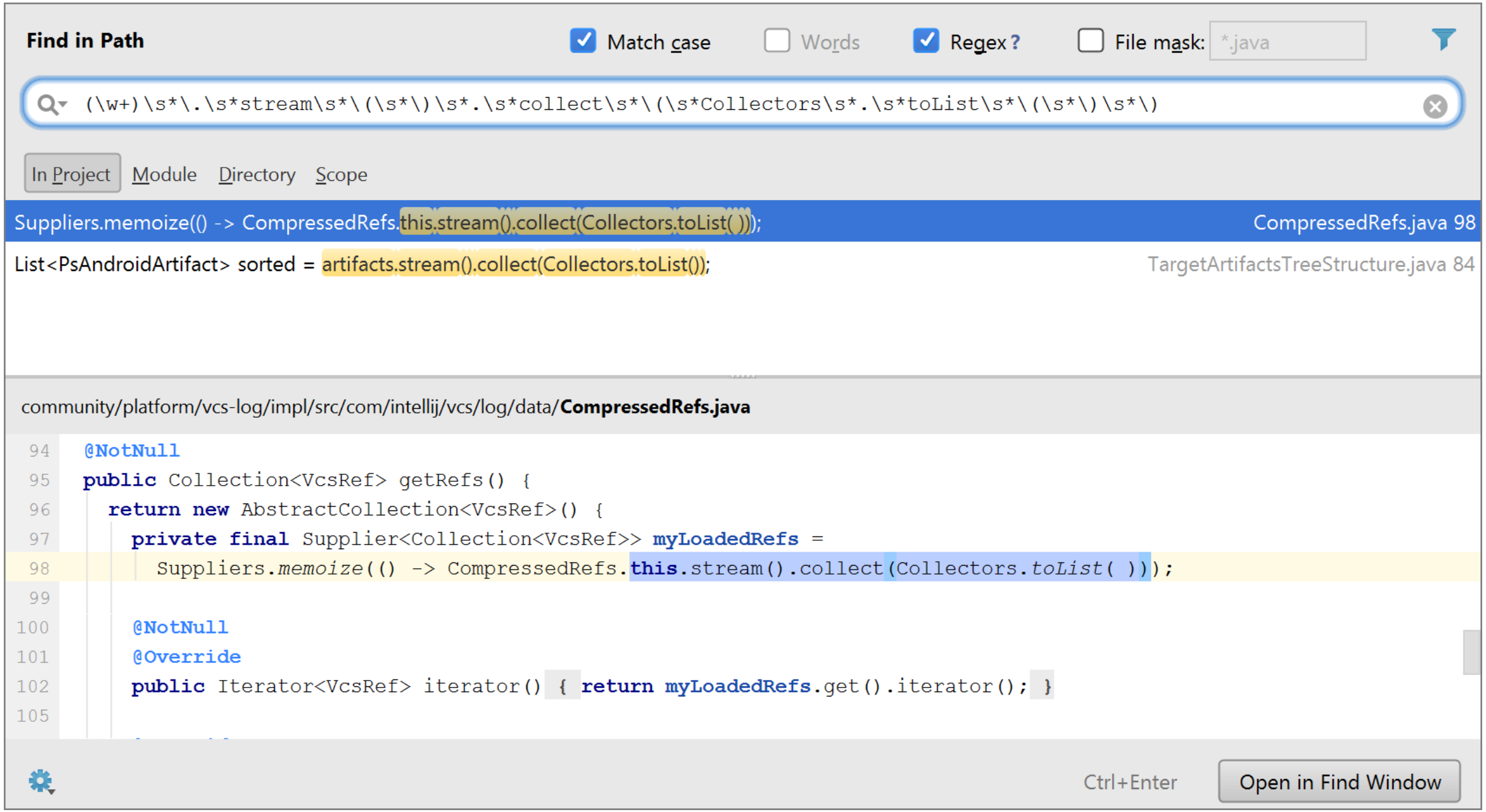
```
new ArrayList<>(set)
```

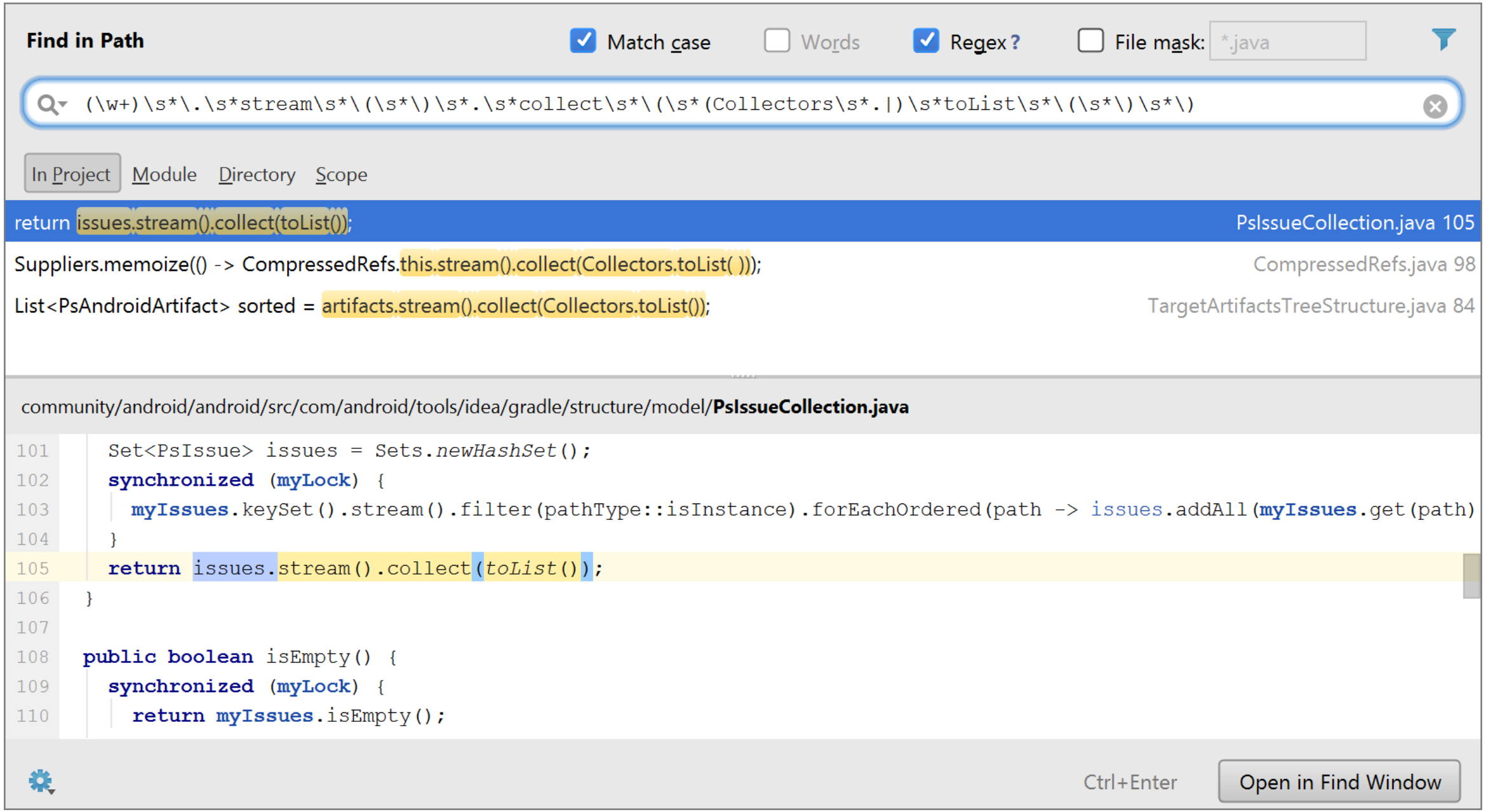
the reason for the simplification: avoid unnecessary intermediate stream objects and transformation



https://xkcd.com/208/

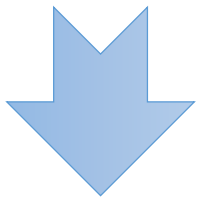

Find in Path
Match case
Words
Regex?
File mask: *.java
(\w+)\.stream\(\).collect\(Collectors.toList\(\)\)
1 match in 1 file
In Project
Module
Directory
Scope
List<PsAndroidArtifact> sorted = artifacts.stream().collect(Collectors.toList());
TargetArtifactsTreeStructure.java 84
community/android/android/src/com/android/tools/idea/gradle/structure/configurables/android/dependencies/module/treeview/TargetArtifactsTreeStructure.java
80 for (PsVariant variant : variants) {
81 TargetVariantNode variantNode = new TargetVariantNode(variant);
82
83 Collection<PsAndroidArtifact> artifacts = artifactsByVariant.get(variant);
84 List<PsAndroidArtifact> sorted = artifacts.stream().collect(Collectors.toList());
85 if (sorted.size() > 1) {
86 Collections.sort(sorted, ArtifactComparator.INSTANCE);
87 }
88
89 List<TargetArtifactNode> artifactNodes = Lists.newArrayList();
Ctrl+Enter
Open in Find Window


Find in Path
Match case
Words
Regex?
File mask: *.java
(\w+)\s*\.\s*stream\s*\(\s*\)\s*.\s*collect\s*\(\s*Collectors\s*.\s*toList\s*\(\s*\)\s*\)
In Project
Module
Directory
Scope
Suppliers.memoize(() -> CompressedRefs.this.stream().collect(Collectors.toList( ))); CompressedRefs.java 98
List<PsAndroidArtifact> sorted = artifacts.stream().collect(Collectors.toList()); TargetArtifactsTreeStructure.java 84
community/platform/vcs-log/impl/src/com/intellij/vcs/log/data/CompressedRefs.java
94 @NotNull
95 public Collection<VcsRef> getRefs() {
96 return new AbstractCollection<VcsRef>() {
97 private final Supplier<Collection<VcsRef>> myLoadedRefs =
98 Suppliers.memoize(() -> CompressedRefs.this.stream().collect(Collectors.toList( )));
99
100 @NotNull
101 @Override
102 public Iterator<VcsRef> iterator() { return myLoadedRefs.get().iterator(); }
105
Ctrl+Enter
Open in Find Window


Find in Path
Match case
Words
Regex?
File mask: *.java
(\w+)\s*\.\s*stream\s*\(\s*\)\s*.\s*collect\s*\(\s*(Collectors\s*.|)\s*toList\s*\(\s*\)\s*\)
In Project
Module
Directory
Scope
return issues.stream().collect(toList()); PsIssueCollection.java 105
Suppliers.memoize(() -> CompressedRefs.this.stream().collect(Collectors.toList( ))); CompressedRefs.java 98
List<PsAndroidArtifact> sorted = artifacts.stream().collect(Collectors.toList()); TargetArtifactsTreeStructure.java 84
community/android/android/src/com/android/tools/idea/gradle/structure/model/PsIssueCollection.java
101 Set<PsIssue> issues = Sets.newHashSet();
102 synchronized (myLock) {
103 myIssues.keySet().stream().filter(pathType::isInstance).forEachOrdered(path -> issues.addAll(myIssues.get(path)
104 }
105 return issues.stream().collect(toList());
106 }
107
108 public boolean isEmpty() {
109 synchronized (myLock) {
110 return myIssues.isEmpty();
Ctrl+Enter
Open in Find Window

```
List<String> list = this.set.stream().collect(Collectors.toList());
```

```
List<String> list = this.set.stream().collect(Collectors.toList());
```



```
List<String> list = this.new ArrayList<>(set);
```

```
public class CompressedRefs {
  ...

  @NotNull
  public Stream<VcsRef> stream() {
    return Stream.concat(streamBranches(), streamTags());
  }

  @NotNull
  public Collection<VcsRef> getRefs() {
    return new AbstractCollection<VcsRef>() {
      private final Supplier<Collection<VcsRef>> myLoadedRefs =
        Suppliers.memoize(() -> CompressedRefs.this.stream()
                                                   .collect(Collectors.toList()));
      ...
    }
    ...
  }
  ...
}
```

```
PsiMethod m = methodCall.resolveMethod();
if(m != null && m.getName().equals("toList") &&
   m.getParameterList().getParametersCount() == 0) {
  PsiClass c = m.getContainingClass();
  if(c != null &&
     "java.util.stream.Collectors".equals(c.getQualifiedName())) {
    // Наш коллектор!
  }
}
```

```
PsiMethod m = methodCall.resolveMethod();
if(m != null && m.getName().equals("toList") &&
   m.getParameterList().getParametersCount() == 0) {
  PsiClass c = m.getContainingClass();
  if(c != null &&
     "java.util.stream.Collectors".equals(c.getQualifiedName())) {
    // Наш коллектор!
  }
}
```

```
list.stream().collect(Collectors.toList());

list.stream().collect(java.util.stream.Collectors.toList());


import static java.util.stream.Collectors.toList;
list.stream().collect(toList());
```

```
PsiMethod m = methodCall.resolveMethod();
if(m != null && m.getName().equals("toList") &&
   m.getParameterList().getParametersCount() == 0) {
  PsiClass c = m.getContainingClass();
  if(c != null &&
     "java.util.stream.Collectors".equals(c.getQualifiedName())) {
    // Наш коллектор!
  }
}
```

```
import com.example.Collectors;
list.stream().collect(Collectors.toList());
```

```
import static com.example.MyCollectors.toList;
list.stream().collect(toList());
```

Замена

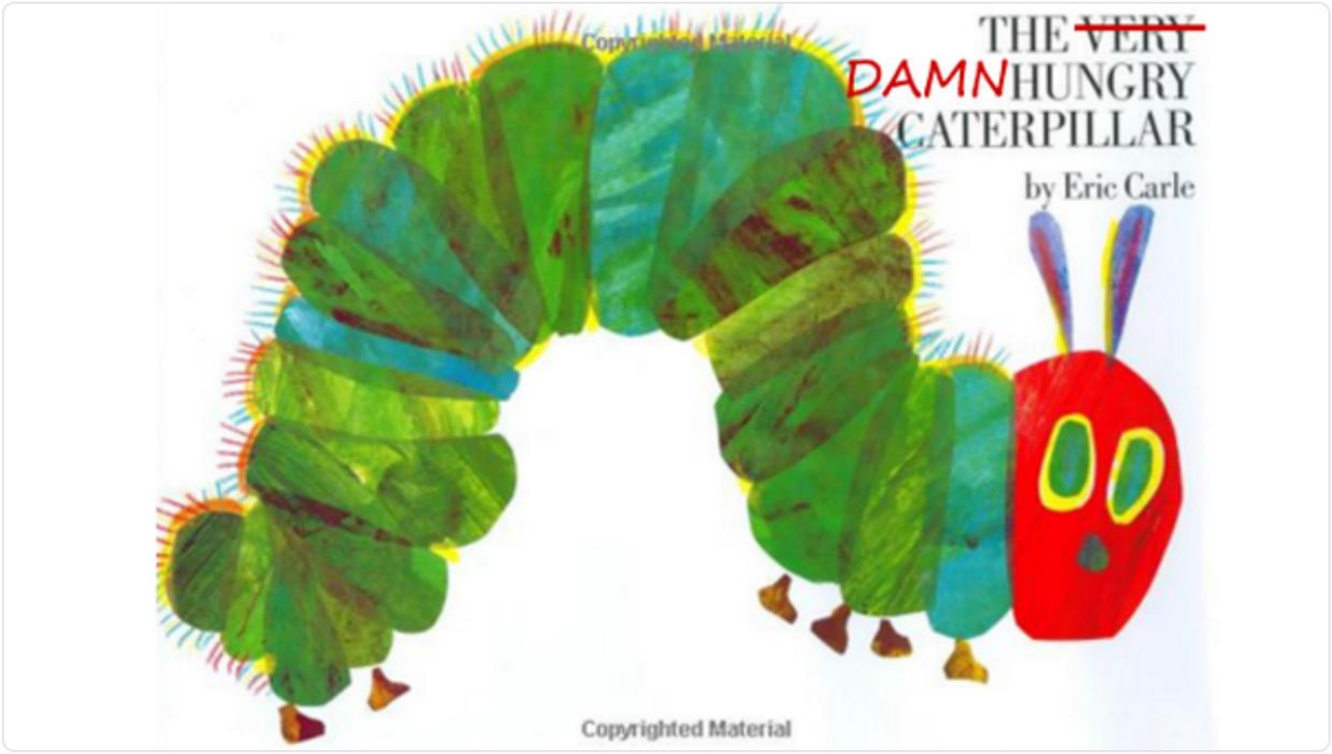
—

Lifehacker
@lifehacker

Читать

Replace "very" with "damn" to improve your writing: lifehac.kr/hxQzTnE

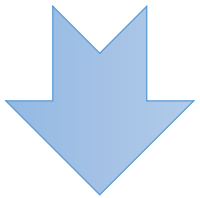
Язык твита: английский



РЕТВИТОВ 139

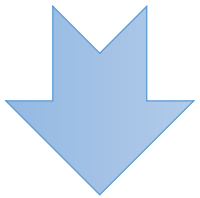
ОТМЕТОК «НРАВИТСЯ» 197

```
List<String> list = this.set.stream().collect(Collectors.toList());
```



```
List<String> list = new ArrayList<>(this.set);
```

```
List<String> list = this.set.stream().collect(Collectors.toList());
```



```
List<String> list = new ArrayList<>(this.set);
```

Cannot resolve symbol 'ArrayList'

```
PsiExpression replacement = factory
  .createExpressionFromText("new java.util.ArrayList<>(" +
                            collectionExpression.getText() + ")", collectCall);

PsiElement result = collectCall.replace(replacement);

JavaCodeStyleManager.getInstance(project).shortenClassReferences(result);
```

Нормализация кода:

—

- Сокращение квалифицированных имён классов
- Удаление ненужных квалификаторов
- Удаление ненужных приведений типов
- Удаление ненужных дженерик-аргументов
- Превращение лямбд в ссылки на методы, где возможно
- Форматирование замены в соответствии с выбранным стилем

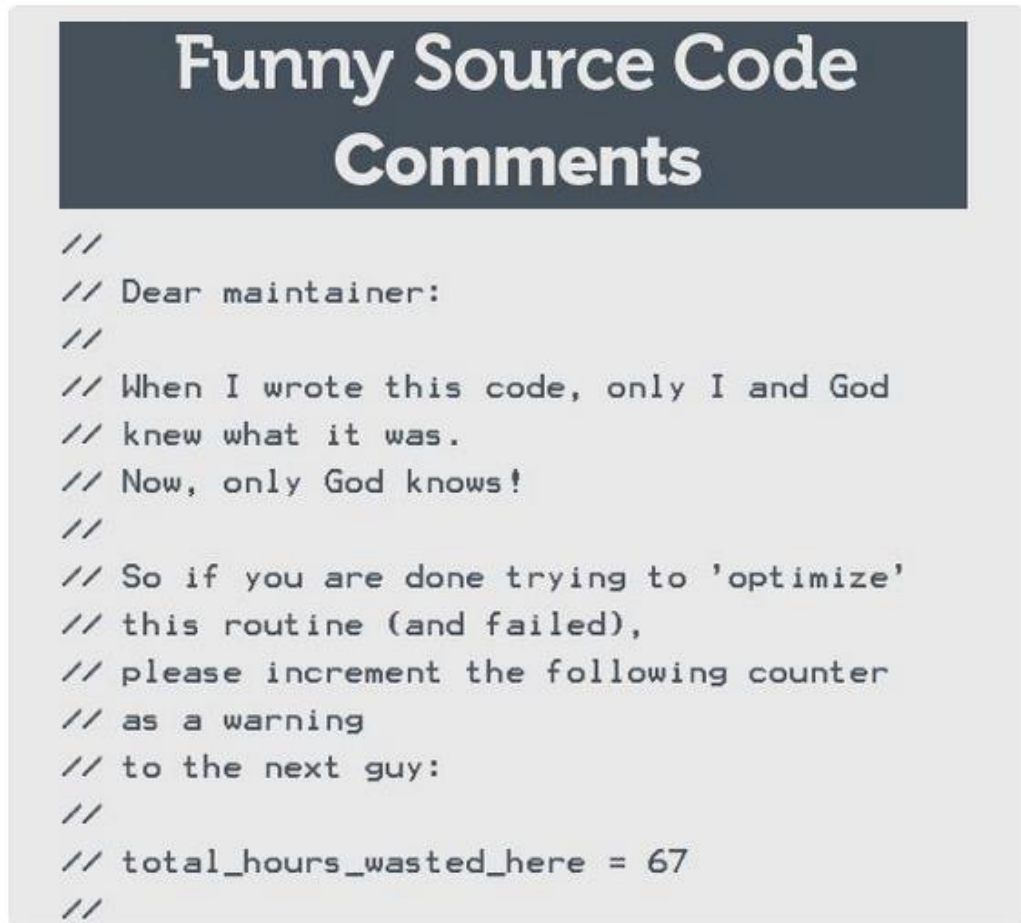
/* Комментарии */

Nevercode
@nevercodeHQ

Читать

What is the funniest comment you have found or written in the code? Care to share? 😉
#funfriday

Язык твита: английский



РЕТВИТОВ 5

ОТМЕТОК «НРАВИТСЯ» 6

```
List<String> list = Arrays.asList( ...a: "foo");
```

```
List<String> list = Collections.singletonList( o: "foo");
```

```
Arrays.asList("foo")
```

```
footprint:
     COUNT       AVG       SUM   DESCRIPTION
         1        24        24   [Ljava.lang.String;
         1        24        24   java.util.Arrays$ArrayList
         2                  48   (total)
```

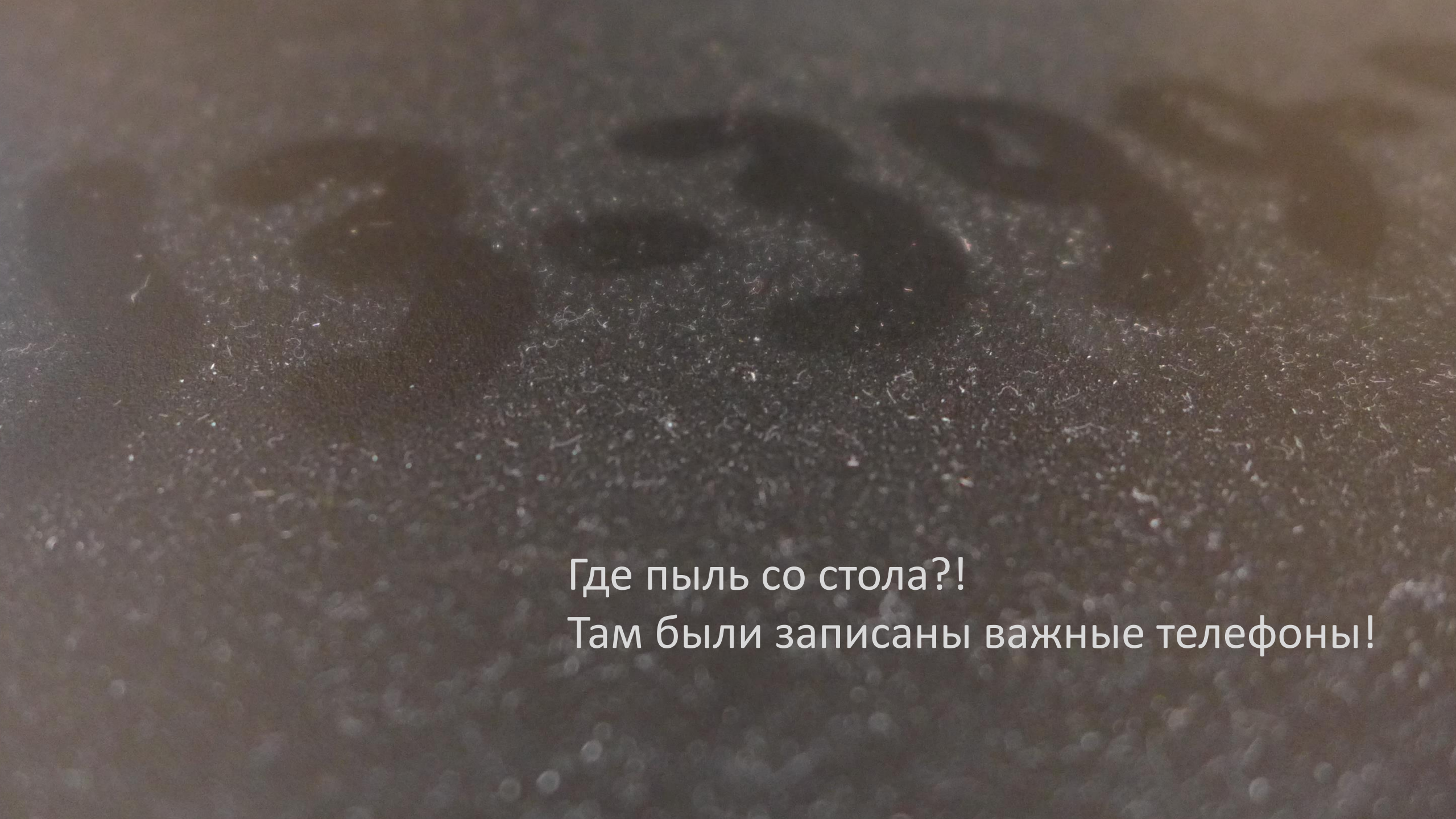
```
Collections.singletonList("foo")
```

```
footprint:
     COUNT       AVG       SUM   DESCRIPTION
         1        24        24   java.util.Collections$SingletonList
         1                  24   (total)
```

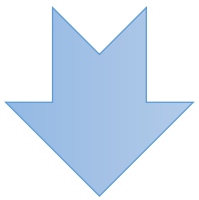
via JOL: http://openjdk.java.net/projects/code-tools/jol/

```
List<String> list = Arrays./* valuable comment */asList( ...a: "foo");
```

```
List<String> list = Collections.singletonList( o: "foo");
```


Где пыль со стола?!
Там были записаны важные телефоны!

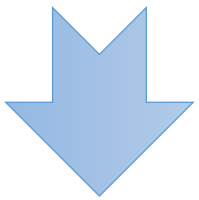
```java
public void removeEmpty(List<String> list) {
    Iterator<String> it = list.iterator();
    while(it.hasNext()) {
        String s = it.next();
        if(s == null || s.trim().isEmpty()) {
            it.remove();
        }
    }
}
```



```java
public void removeEmpty(List<String> list) {
    list.removeIf(s -> s == null || s.trim().isEmpty());
}
```

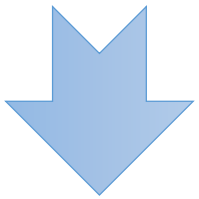
```
public void removeEmpty(List<String> list) {
    Iterator<String> it = list.iterator();
    while(it.hasNext()) {
        String s = it.next();
        if(s == null || // trimming is important here
                s.trim().isEmpty()) {
            it.remove(); // remove from list
        }
    }
}
```

```
public void removeEmpty(List<String> list) {
    Iterator<String> it = list.iterator();
    while(it.hasNext()) {
        String s = it.next();
        if(s == null || // trimming is important here
                s.trim().isEmpty()) {
            it.remove(); // remove from list
        }
    }
}
```



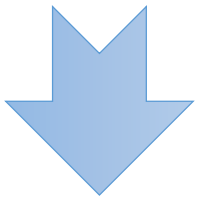
```
public void removeEmpty(List<String> list) {
    // remove from list
    // trimming is important here
    list.removeIf(s -> s == null || // trimming is important here
            s.trim().isEmpty());
}
```

```
public void removeEmpty(List<String> list) {
    Iterator<String> it = list.iterator();
    while(it.hasNext()) {
        String s = it.next();
        if(s == null || // trimming is important here
                s.trim().isEmpty()) {
            it.remove(); // remove from list
        }
    }
}
```



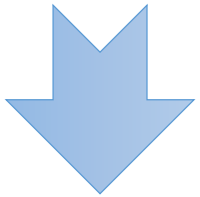
```
public void removeEmpty(List<String> list) {
    // remove from list
    // trimming is important here
    list.removeIf(s -> s == null ||
            s.trim().isEmpty());
}
```

```
public void removeEmpty(List<String> list) {
    Iterator<String> it = list.iterator();
    while(it.hasNext()) {
        String s = it.next();
        if(s == null || // trimming is important here
                s.trim().isEmpty()) {
            it.remove(); // remove from list
        }
    }
}
```



```
public void removeEmpty(List<String> list) {
    // remove from list
    list.removeIf(s -> s == null || // trimming is important here
            s.trim().isEmpty());
}
```

```
public String test() {
    try { return run(); }
    catch (IllegalArgumentException ex) {
        return null;
    }
    catch (IllegalStateException ex) {
        return null;
    }
}
```

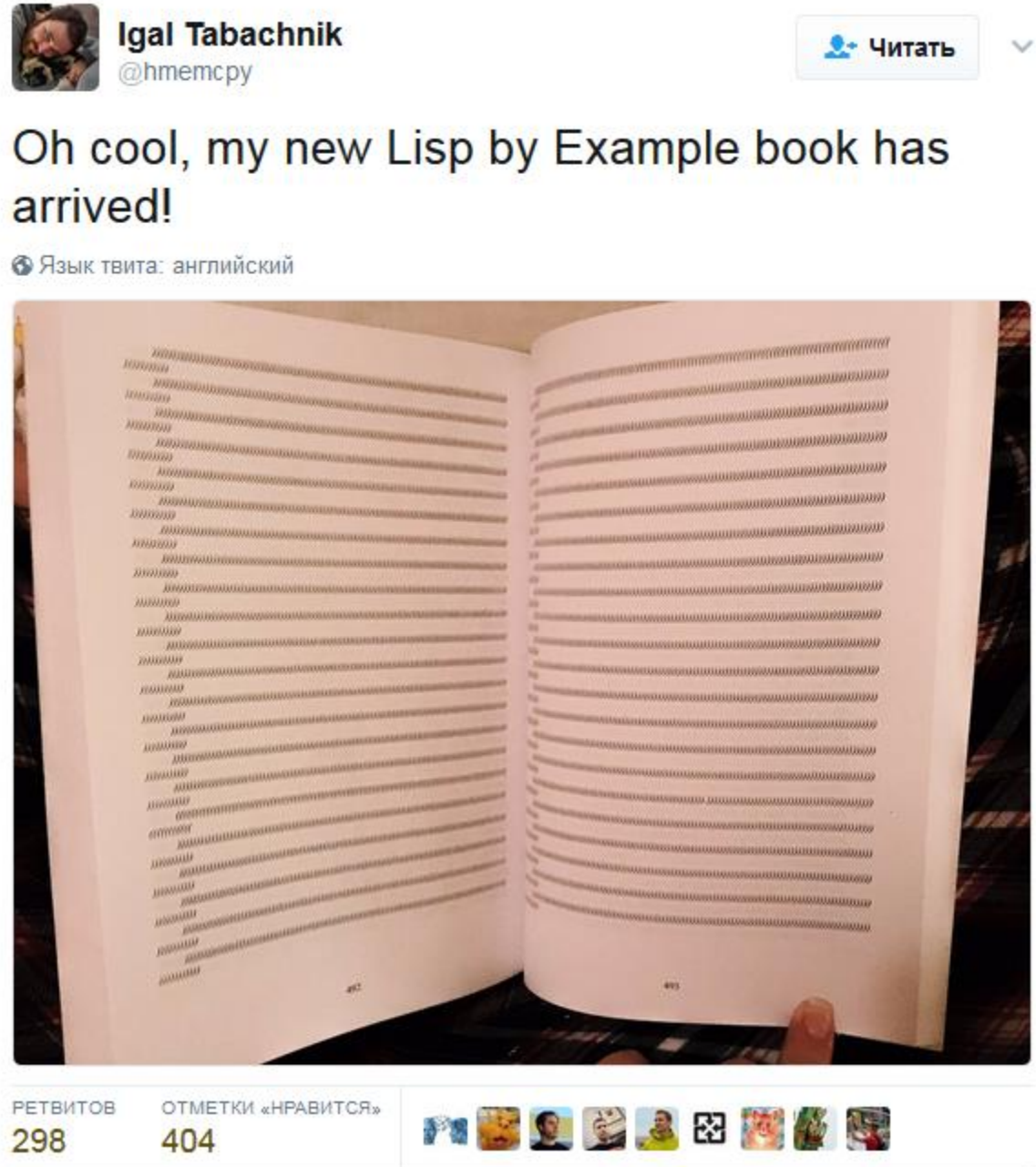


```
public String test() {
    try { return run(); }
    catch (IllegalArgumentException | IllegalStateException ex) {
        return null;
    }
}
```

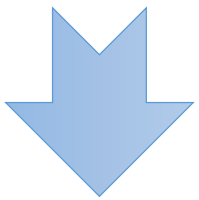
```
try {...}
catch (VcsLogRefreshNotEnoughDataException e) {
  // valid case: e.g. another developer merged a long-developed branch,
  // or we just didn't pull for a long time
  LOG.info(e);
}
catch (IllegalStateException e) {
  // it happens from time to time,
  // but we don't know why, and can hardly debug it.
  LOG.info(e);
}
```

(скобочки)

—


Igal Tabachnik
@hmemcpy
Читать
Oh cool, my new Lisp by Example book has arrived!
Язык твита: английский
РЕТВИТОВ
298
ОТМЕТКИ «НРАВИТСЯ»
404

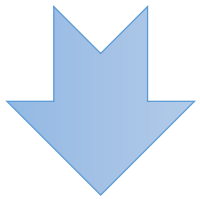
```
for(int i=0; i<strings.size(); i++) {
    System.out.println(strings.get(i));
}
```



```
for (String string : strings) {
    System.out.println(string);
}
```

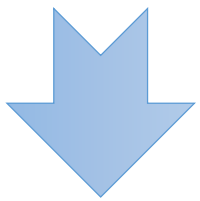
```
for(int i=(0); (i)<((strings).size()); (i)++) {
    System.out.println(strings.get((i)));
}
```

```
String foo = ((Supplier<String>) () -> "bar").get();

String foo = ((Supplier<String>) (() -> "bar")).get();
```



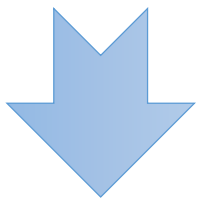
```
String foo = "bar";
```

```
boolean hasEmpty(List<String> list) {
    for(String s : list) {
        if(s.isEmpty()) {
            return true;
        }
    }
    return false;
}
```



```
boolean hasEmpty(List<String> list) {
    return list.stream().anyMatch(String::isEmpty);
}
```

```
boolean hasEmpty(Object list) {
    for(String s : (List<String>)list) {
        if(s.isEmpty()) {
            return true;
        }
    }
    return false;
}
```



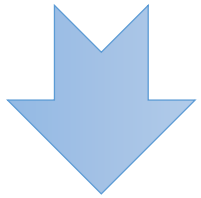
```
boolean hasEmpty(Object list) {
    return ((List<String>) list).stream()
            .anyMatch(String::isEmpty);
}
```

Семантика

—


Gunnar Bittersmann
@g16n
Читать
“If you want them to RTFM, make a better FM”
—Kathy Sierra quoted by @brnnbrn
#FrontTrends
Язык твита: английский
“If you want them to RTFM, make a better FM”
Kathy Sierra
РЕТВИТОВ
10
ОТМЕТОК «НРАВИТСЯ»
20

```
List<String> list = Arrays.asList( ...a: "foo");
```



```
List<String> list = Collections.singletonList( o: "foo");
```

singletonList

```
public static <T> List<T> singletonList(T o)
```

Returns an immutable list containing only the specified object. The returned list is serializable.

asList

```
@SafeVarargs public static <T> List<T> asList(T... a)
```

Returns a fixed-size list backed by the specified array. (Changes to the returned list "write through" to the array.) This method acts as bridge between array-based and collection-based APIs, in combination with Collection.toArray(). The returned list is serializable and implements RandomAccess.

This method also provides a convenient way to create a fixed-size list initialized to contain several elements:

```
List<String> stooges = Arrays.asList("Larry", "Moe", "Curly");
```

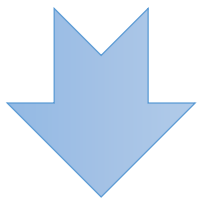
```java
public static void main(String[] args) {
    List<String> list = Arrays.asList( ...a: "foo");
    list.set(0, "bar");
    System.out.println(list);
}
```

```
"C:\Program Files\Java\jdk1.8.0_101\bin\java" ...
[bar]

Process finished with exit code 0
```

```java
public static void main(String[] args) {
    List<String> list = Collections.singletonList( o: "foo");
    list.set(0, "bar");
    System.out.println(list);
}
```

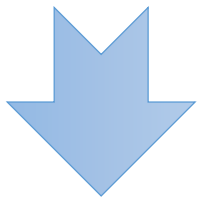
```
"C:\Program Files\Java\jdk1.8.0_101\bin\java" ...
Exception in thread "main" java.lang.UnsupportedOperationException
    at java.util.AbstractList.set(AbstractList.java:132)
    at test.Main.main(Main.java:14) <5 internal calls>

Process finished with exit code 1
```

```
void putTest(Map<String, String> map,
             String key) {
    String val = map.get(key);
    if(val == null) {
        map.put(key, "");
    }
}
```



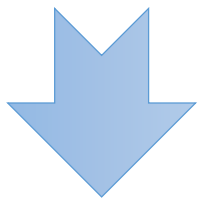
```
void putTest(Map<String, String> map,
             String key) {
    map.putIfAbsent(key, "");
}
```

```
void putTest(Map<String, String> map,
             String key) {
    String val = map.get(key);
    if(val == null) {
        map.put(key, "");
    }
}
```



```
void putTest(Map<String, String> map,
             String key) {
    map.putIfAbsent(key, "");
}
```

```
String getTest(Map<String, String> map,
               String key) {
    String val = map.get(key);
    if(val == null) {
        val = "";
    }
    return val;
}
```



```
String getTest(Map<String, String> map,
               String key) {
    return map.getOrDefault(key, "");
}
```

getOrDefault

`default V getOrDefault(Object key, V defaultValue)`

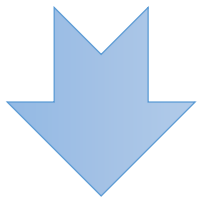
Returns the value to which the specified key is mapped, or `defaultValue` if this map contains no mapping for the key.

putIfAbsent

`default V putIfAbsent(K key, V value)`

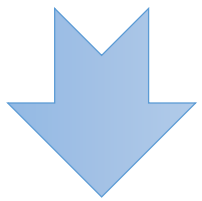
If the specified key is not already associated with a value (or is mapped to `null`) associates it with the given value and returns `null`, else returns the current value.

```
void putTest(Map<String, String> map,
             String key) {
    String val = map.get(key);
    if(val == null) {
        map.put(key, "");
    }
}
```



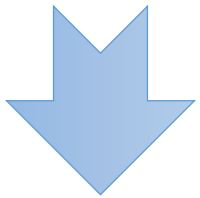
```
void putTest(Map<String, String> map,
             String key) {
    map.putIfAbsent(key, "");
}
```

```
String getTest(Map<String, String> map,
               String key) {
    String val = map.get(key);
    if(val == null) {
        val = "";
    }
    return val;
}
```



```
String getTest(Map<String, String> map,
               String key) {
    return map.getOrDefault(key, "");
}
```

```
void jpa(JpaRepository<Entity, Long> jpaRepository, List<Long> ids) {
    List<Entity> entities = ids.stream()
            .map(id -> jpaRepository.findOne(id))
            .collect(Collectors.toList());
    System.out.println(entities);
}
```



```
void jpa(JpaRepository<Entity, Long> jpaRepository, List<Long> ids) {
    List<Entity> entities = jpaRepository.findAll(ids);
    System.out.println(entities);
}
```

Interface JpaRepository<T,ID extends Serializable>

findAll

`List<T> findAll(Iterable<ID> ids)`

Specified by:

findAll in interface CrudRepository<T,ID extends Serializable>

Interface JpaRepository<T,ID extends Serializable>

findAll

`List<T> findAll(Iterable<ID> ids)`

Specified by:

findAll in interface CrudRepository<T,ID extends Serializable>

Interface CrudRepository<T,ID extends Serializable>

findAll

`Iterable<T> findAll(Iterable<ID> ids)`

Returns all instances of the type with the given IDs.

Parameters:

`ids` -

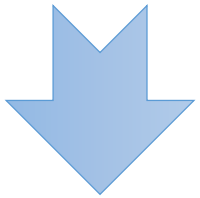
Returns:

Дженерики<?>

—

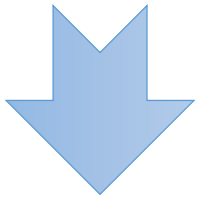

Charles Nutter
@headius
Читать
All you Java devs out there wish you had this neato generic belt. Parametric polymorphism for trousers! #jokerconf
Язык твита: английский
РЕТВИТОВ
8
ОТМЕТКИ «НРАВИТСЯ»
34

```java
public Number get(Map<String, Number> map) {
    return map.containsKey("foo") ? map.get("foo") : 0.0;
}
```



```java
public Number get(Map<String, Number> map) {
    return map.getOrDefault( key: "foo",  defaultValue: 0.0);
}
```

```java
public Number get(Map<String, Integer> map) {
    return map.containsKey("foo") ? map.get("foo") : 0.0;
}
```

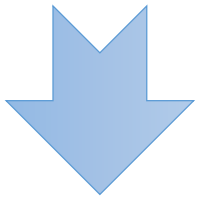


```java
public Number get(Map<String, Integer> map) {
    return map.getOrDefault( key: "foo", 0.0);
}
```

Wrong 2nd argument type. Found: 'double', required: 'java.lang.Integer' less...

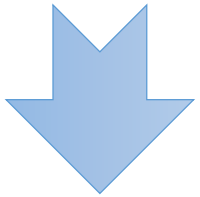
getOrDefault (Object, java.lang.Integer) in **Map** cannot be applied to **(String, double)**

```
Iterator<Integer> getIterator(Collection<Integer> input) {
    return input.stream().collect(Collectors.toList())
            .iterator();
}
```



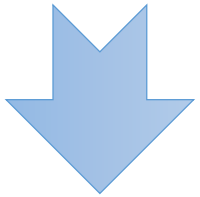
```
Iterator<Integer> getIterator(Collection<Integer> input) {
    return new ArrayList<>(input).iterator();
}
```

```
Iterator<Number> getIterator(Collection<Integer> input) {
    return input.stream().collect(Collectors.<Number>toList())
            .iterator();
}
```



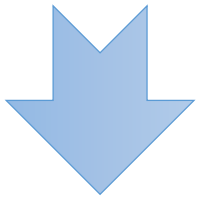
```
Iterator<Number> getIterator(Collection<Integer> input) {
    return new ArrayList<>(input).iterator();
}
```

```
Iterator<Number> getIterator(Collection<Integer> input) {
    return input.stream().collect(Collectors.<Number>toList())
            .iterator();
}
```



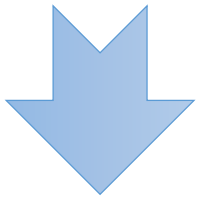
```
Iterator<Number> getIterator(Collection<Integer> input) {
    return new ArrayList<Number>(input).iterator();
}
```

```
List<Number> test(List<Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toList());
}
```



```
List<Number> test(List<Number> list) {
    List<Number> result = new ArrayList<>();
    for (Number number : list) {
        if (number != null) {
            result.add(number);
        }
    }
    return result;
}
```

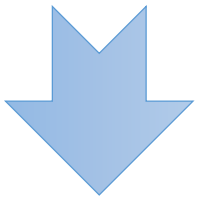
```
List<Number> test(List<Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toList());
}
```



```
List<Number> test(List<Number> list) {
    List<Number> result = new ArrayList<>();
    for (Number number : list) {
        if (number != null) {
            result.add(number);
        }
    }
    return result;
}
```

```
List<? extends Number> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toList());
}
```

```
List<? extends Number> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toList());
}
```



```
List<? extends Number> test(List<? extends Number> list) {
    List<? extends Number> result = new ArrayList<>();
    for (Number number : list) {
        if (number != null) {
            result.add(number);
        }
    }
    return result;
}
```

```
List<? extends Number> makeList(Number n) { return singletonList(n); }

List<? extends List<? extends Number>> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .map(this::makeList).collect(Collectors.toList());
}
```

```
List<? extends Number> makeList(Number n) { return singletonList(n); }

List<? extends List<? extends Number>> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .map(this::makeList).collect(Collectors.toList());
}
```

```
List<? extends List<? extends Number>> test(List<? extends Number> list) {
    List<List<Number>> result = new ArrayList<>();
    for (Number number : list) {
        if (number != null) {
            List<? extends Number> numbers = makeList(number);
            result.add(numbers);
        }
    }
    return result;
}
```

```
Map<Long, ? extends List<? extends Number>> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.groupingBy(n -> n.longValue() % 10));
}
```

```
Map<Long, ? extends List<? extends Number>> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.groupingBy(n -> n.longValue() % 10));
}
```

```
Map<Long, ? extends List<? extends Number>> test(List<? extends Number> list) {
    Map<Long, List<? extends Number>> map = new HashMap<>();
    for (Number n : list) {
        if (n != null) {
            map.computeIfAbsent( key: n.longValue() % 10, k -> new ArrayList<>())
               .add(n);
        }
    }
    return map;
}
```

```
List<? extends Number> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toList());
}
```

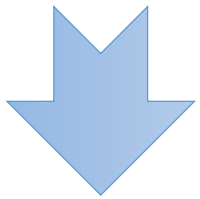
```
List<? extends Number> test(List<? extends Number> list) {
    List<Number> result = new ArrayList<>();
    for (Number number : list) {
        if (number != null) {
            result.add(number);
        }
    }
    return result;
}
```

```
List<? extends Number> makeList(Number n) { return singletonList(n); }

List<? extends List<? extends Number>> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .map(this::makeList).collect(Collectors.toList());
}
```

```
List<? extends List<? extends Number>> test(List<? extends Number> list) {
    List<List<? extends Number>> result = new ArrayList<>();
    for (Number number : list) {
        if (number != null) {
            List<? extends Number> numbers = makeList(number);
            result.add(numbers);
        }
    }
    return result;
}
```

```
List<Object> test(List<Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toList());
}
```

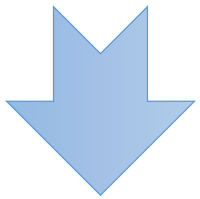
```
List<Object> test(List<Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toList());
}
```



```
List<Object> test(List<Number> list) {
    List<Number> result = new ArrayList<>();
    for (Number number : list) {
        if (number != null) {
            result.add(number);
        }
    }
    return result;
}
```

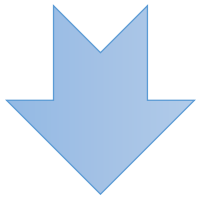
```
class MyList<T> extends ArrayList<T> {}
MyList<Number> create() { return new MyList<>(); }

MyList<? extends Number> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
            .collect(Collectors.toCollection(this::create));
}
```

```
class MyList<T> extends ArrayList<T> {}
MyList<Number> create() { return new MyList<>(); }

MyList<? extends Number> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toCollection(this::create));
}
```



```
MyList<? extends Number> test(List<? extends Number> list) {
    MyList<Number> numbers = create();
    for (Number number : list) {
        if (number != null) {
            numbers.add(number);
        }
    }
    return numbers;
}
```

```
class MyList<T> extends ArrayList<Number> {}
MyList<? extends Number> create() { return new MyList<>(); }

MyList<? extends Number> test(List<? extends Number> list) {
    return list.stream().filter(Objects::nonNull)
               .collect(Collectors.toCollection(this::create));
}
```



```
MyList<? extends Number> test(List<? extends Number> list) {
    MyList<Number> numbers = create();
    for (Number number : list) {
        if (number != null) {
            numbers.add(number);
        }
    }
    return numbers;
}
```

(Приведение) типов

—

```
System.out.println(Math.random() > 0.5 ? "yes" : "no");
```

```
if (Math.random() > 0.5) {
    System.out.println("yes");
} else {
    System.out.println("no");
}
```

```
System.out.println(Math.random() > 0.5 ? "yes" : null);
```

```
if (Math.random() > 0.5) {
    System.out.println("yes");
} else {
    System.out.println( x: null);
}
```

Ambiguous method call. Both

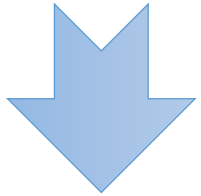
println (char[]) in **PrintStream** and

println (String) in **PrintStream** match

```
System.out.println(Math.random() > 0.5 ? "yes" : null);
```

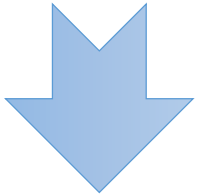
```
if (Math.random() > 0.5) {
    System.out.println("yes");
} else {
    System.out.println((String) null);
}
```

```
return streamOfStrings.map(String::isEmpty)
        .anyMatch(Boolean::booleanValue);
```



```
return streamOfStrings.anyMatch(String::isEmpty);
```

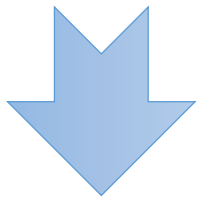
```
Function<String, Boolean> isEmpty = String::isEmpty;
return streamOfStrings.map(isEmpty)
        .anyMatch(Boolean::booleanValue);
```



```
Function<String, Boolean> isEmpty = String::isEmpty;
return streamOfStrings.anyMatch(isEmpty);
```

anyMatch (java.util.function.Predicate<? super java.lang.String>) in **Stream** cannot be applied to **(java.util.function.Function<java.lang.String,java.lang.Boolean>)**

```
Function<String, Boolean> isEmpty = String::isEmpty;
return streamOfStrings.map(isEmpty)
        .anyMatch(Boolean::booleanValue);
```



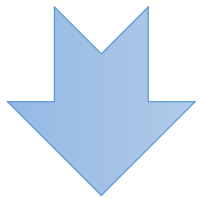
```
Function<String, Boolean> isEmpty = String::isEmpty;
return streamOfStrings.anyMatch(isEmpty::apply);
```

```
return streamOfStrings.map(flag ? "foo"::equals : "bar"::equals)
        .anyMatch(Boolean::booleanValue);
```

```
return streamOfStrings
        .anyMatch((flag ? "foo"::equals : "bar"::equals)::apply);
```

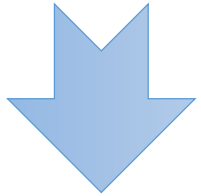
Method reference expression is not expected here

```
return streamOfStrings.map(flag ? "foo"::equals : "bar"::equals)
        .anyMatch(Boolean::booleanValue);
```



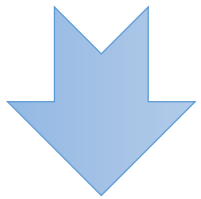
```
return streamOfStrings
        .anyMatch(flag ? "foo"::equals : "bar"::equals);
```

```
Function<String, Boolean> fn = "foo"::equals;
return streamOfStrings.map(flag ? fn : "bar"::equals)
        .anyMatch(Boolean::booleanValue);
```



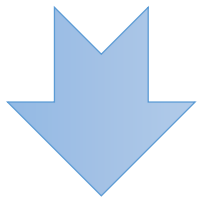
???

```
Function<String, Boolean> fn = "foo"::equals;
return streamOfStrings.map(flag ? fn : "bar"::equals)
        .anyMatch(Boolean::booleanValue);
```



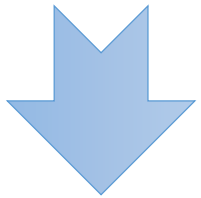
```
Function<String, Boolean> fn = "foo"::equals;
return streamOfStrings
        .anyMatch(flag ? fn::apply : "bar"::equals);
```

```
interface MyFunction extends Function<Object, Boolean> {
    default boolean apply(String s) {return false;}
}

MyFunction fn = "xyz"::equals;
System.out.println(
        Stream.of("xyz").map(fn).anyMatch(Boolean::booleanValue));
// prints "true"
```



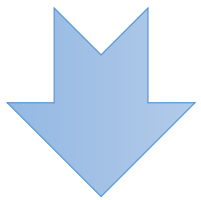
```
System.out.println(
        Stream.of("xyz").anyMatch(fn::apply));
// prints "false"
```

```
boolean x = !!str.isEmpty();
```



```
boolean x = str.isEmpty();
```

```
void run(Consumer<String> cons) {}
void run(Predicate<String> pred) {}

void test() {
    run(str -> !!str.isEmpty());
}
```



```
void run(Consumer<String> cons) {}
void run(Predicate<String> pred) {}

void test() {
    run(str -> str.isEmpty());
}
```

{ некорректный код

—

`com.intellij.psi.PsiConditionalExpression`

`@Nullable PsiExpression getThenExpression()`

Returns the expression which is the result used when the condition is true.

Returns:

the true result expression, or `null` if the conditional expression is incomplete.

```
return a > 0 ?:0;
```

`com.intellij.psi.PsiAssignmentExpression`

`@Nullable PsiExpression getRExpression()`

Returns the expression on the right side of the assignment.

Returns:

the right side expression, or `null` if the assignment expression is incomplete.

```
a += ;
```

```
// returns "abcdefghijklmnopqrstuvwxyz"
static String letters() {
    return IntStream.rangeClosed('a', 'z')
            .collect(StringBuilder::new,
                    StringBuilder::appendCodePoint,
                    combiner: null)
            .toString();
}
```

```
public void visitMethodCallExpression(PsiMethodCallExpression call) {

}
```

```
public void visitMethodCallExpression(PsiMethodCallExpression call) {
  PsiMethod method = call.resolveMethod();
  // Проверяем, что имя метода - collect
  if (method == null || !"collect".equals(method.getName())) return;
}
```

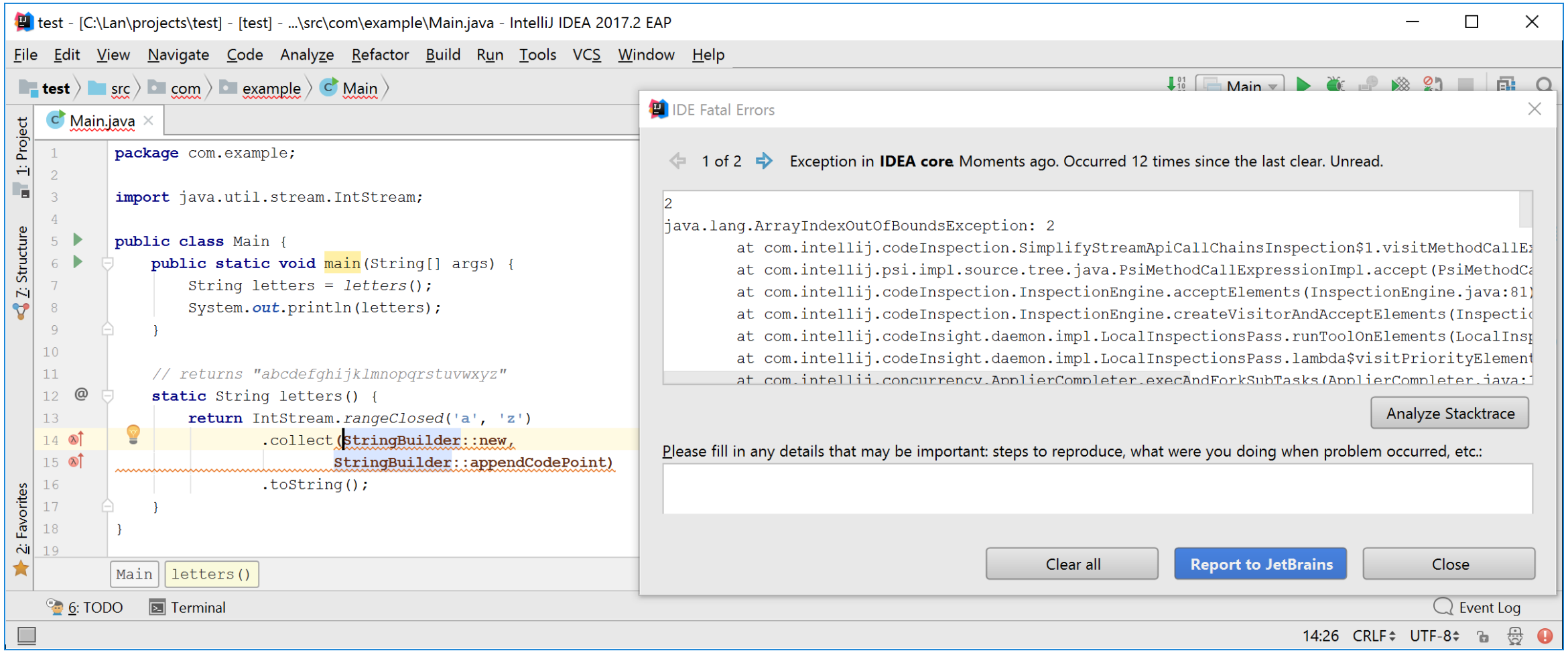
```
public void visitMethodCallExpression(PsiMethodCallExpression call) {
  PsiMethod method = call.resolveMethod();
  // Проверяем, что имя метода - collect
  if (method == null || !"collect".equals(method.getName())) return;
  PsiClass aClass = method.getContainingClass();
  // Проверяем, что имя класса - IntStream
  if (aClass == null || !"java.util.stream.IntStream"
    .equals(aClass.getQualifiedName())) return;




}
```

```
public void visitMethodCallExpression(PsiMethodCallExpression call) {
  PsiMethod method = call.resolveMethod();
  // Проверяем, что имя метода - collect
  if (method == null || !"collect".equals(method.getName())) return;
  PsiClass aClass = method.getContainingClass();
  // Проверяем, что имя класса - IntStream
  if (aClass == null || !"java.util.stream.IntStream"
    .equals(aClass.getQualifiedName())) return;
  // На всякий случай проверяем, что у метода три параметра
  if (method.getParameterList().getParametersCount() != 3) return;
}
```

```
public void visitMethodCallExpression(PsiMethodCallExpression call) {
  PsiMethod method = call.resolveMethod();
  // Проверяем, что имя метода - collect
  if (method == null || !"collect".equals(method.getName())) return;
  PsiClass aClass = method.getContainingClass();
  // Проверяем, что имя класса - IntStream
  if (aClass == null || !"java.util.stream.IntStream"
    .equals(aClass.getQualifiedName())) return;
  // На всякий случай проверяем, что у метода три параметра
  if (method.getParameterList().getParametersCount() != 3) return;
  // Берём третий фактический аргумент и проверяем на null
  PsiExpression thirdArgument = call.getArgumentList().getExpressions()[2];
  if (ExpressionUtils.isNullLiteral(thirdArgument)) {
    holder.registerProblem(thirdArgument,
                           descriptionTemplate: "In Java 9 you will DIE!!!");
  }
}
```

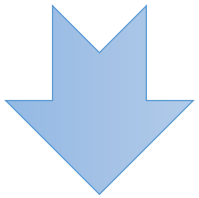
```
// returns "abcdefghijklmnopqrstuvwxyz"
static String letters() {
    return IntStream.rangeClosed('a', 'z')
            .collect(StringBuilder::new,
                    StringBuilder::appendCodePoint,
                    combiner: null)
            .to
}
```

In Java 9 you will DIE!!! more... (Ctrl+F1)


test - [C:\Lan\projects\test] - [test] - ...\src\com\example\Main.java - IntelliJ IDEA 2017.2 EAP
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
test src com example Main
Main.java
1: Project
7: Structure
2: Favorites
1 package com.example;
2
3 import java.util.stream.IntStream;
4
5 public class Main {
6 public static void main(String[] args) {
7 String letters = letters();
8 System.out.println(letters);
9 }
10
11 // returns "abcdefghijklmnopqrstuvwxyz"
12 static String letters() {
13 return IntStream.rangeClosed('a', 'z')
14 .collect(StringBuilder::new,
15 StringBuilder::appendCodePoint)
16 .toString();
17 }
18 }
19
Main letters()
IDE Fatal Errors
1 of 2 Exception in IDEA core Moments ago. Occurred 12 times since the last clear. Unread.
2
java.lang.ArrayIndexOutOfBoundsException: 2
at com.intellij.codeInspection.SimplifyStreamApiCallChainsInspection$1.visitMethodCallEx
at com.intellij.psi.impl.source.tree.java.PsiMethodCallExpressionImpl.accept(PsiMethodCa
at com.intellij.codeInspection.InspectionEngine.acceptElements(InspectionEngine.java:81)
at com.intellij.codeInspection.InspectionEngine.createVisitorAndAcceptElements(Inspectio
at com.intellij.codeInsight.daemon.impl.LocalInspectionsPass.runToolOnElements(LocalInsp
at com.intellij.codeInsight.daemon.impl.LocalInspectionsPass.lambda$visitPriorityElement
Analyze Stacktrace
Please fill in any details that may be important: steps to reproduce, what were you doing when problem occurred, etc.:
Clear all
Report to JetBrains
Close
6: TODO
Terminal
Event Log
14:26 CRLF UTF-8

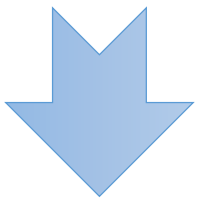
```
public void visitMethodCallExpression(PsiMethodCallExpression call) {
  PsiMethod method = call.resolveMethod();
  // Проверяем, что имя метода - collect
  if (method == null || !"collect".equals(method.getName())) return;
  PsiClass aClass = method.getContainingClass();
  // Проверяем, что имя класса - IntStream
  if (aClass == null || !"java.util.stream.IntStream"
    .equals(aClass.getQualifiedName())) return;
  // На всякий случай проверяем, что у метода три параметра
  if (method.getParameterList().getParametersCount() != 3) return;
  // Берём третий фактический аргумент и проверяем на null
  PsiExpression thirdArgument = call.getArgumentList().getExpressions()[2];
  if (ExpressionUtils.isNullLiteral(thirdArgument)) {
    holder.registerProblem(thirdArgument,
                           descriptionTemplate: "In Java 9 you will DIE!!!");
  }
}
```

```
static class X { long longField; }

Comparator<X> getComparator() {
    return (x1, x2) -> x1.longField - x2.longField;
}
```



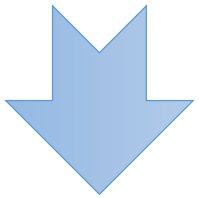
```
Comparator<X> getComparator() {
    return Comparator.comparingInt(x -> x.longField);
}
```

```
static class X { long longField; }

Comparator<X> getComparator() {
    return (x1, x2) -> x1.longField - x2.longField;
}
```



```
Comparator<X> getComparator() {
    return Comparator.comparingInt(x -> x.longField);
}
```

```
static class X { long longField; }

Comparator<X> getComparator() {
    return (x1, x2) -> x1.longField - x2.longField;
}
```



```
Comparator<X> getComparator() {
    return Comparator.comparingLong(x -> x.longField);
}
```

- Поиск
- Замена
- /* Комментарии */
- (скобочки)
- Семантика
- Дженерики <?>
- (Приведение) типов
- { некорректный код

Спасибо
за внимание

—

https://twitter.com/tagir_valeev
https://habrahabr.ru/users/lany
https://github.com/amaembo