EqualsVerifier, ErrorProne
и все, все, все …

Андрей Сатарин
@asatarin

О чем я сегодня расскажу

- EqualsVerifier
    - Зачем он нужен
    - Как он может вам помочь
- ErrorProne
    - Что это
    - Какие проблемы решает

Зачем все это нужно?

- Есть куча низкоуровневых ошибок, которые сложно найти
- Эти ошибки могут очень сильно влиять на видимое поведение
- Так же эти ошибки приводят к общему “гниению” кода

```java
@Test
public void setSizeIsTwo() {
    final Set<Value> set = new HashSet<>();
    final Value a = new Value(0, 0);
    final ChildValue b = new ChildValue(0, 0, -29791);
    set.add(a);
    set.add(b);
    assertEquals(2, set.size());
}
```

```java
@Test
public void setSizeIsTwo() {
    final Set<Value> set = new HashSet<>();
    final Value a = new Value(0, 0);
    final ChildValue b = new ChildValue(0, 0, -29791);
    set.add(a);
    set.add(b);
    assertEquals(2, set.size());
}
```

```java
@Test
public void setSizeIsTwo() {
    final Set<Value> set = new HashSet<>();
    final Value a = new Value(0, 0);
    final ChildValue b = new ChildValue(0, 0, -29791);
    set.add(b);
    set.add(a);
    assertEquals(2, set.size());
}
```

Часть I
EqualsVerifier

Object::equals

```java
public boolean equals(Object obj) {
    return (this == obj);
}
```

```java
final Object a = new Object();
final Object b = new Object();

assertTrue(a == a);
assertTrue(a.equals(a));

assertFalse(a == b);
assertFalse(a.equals(b));
```

Integer::equals

```java
public boolean equals(Object obj) {
    if (obj instanceof Integer) {
        return value == ((Integer)obj).intValue();
    }
    return false;
}
```

```java
final Integer x = new Integer(43534);
final Integer y = new Integer(43534);

assertTrue(x == x);
assertTrue(x.equals(x));

assertFalse(x == y);
assertTrue(x.equals(y));
```

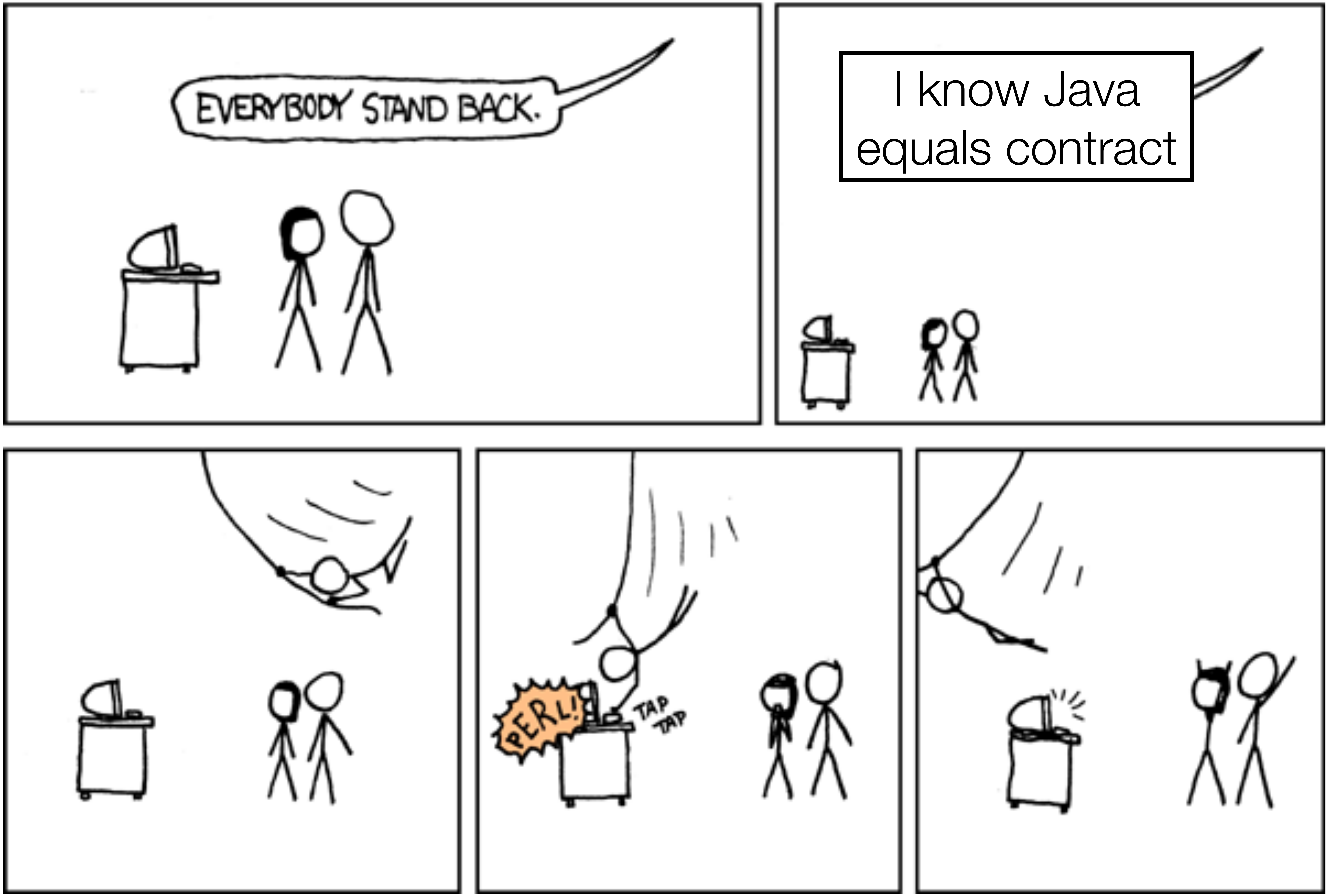
```java
final Integer x = new Integer(43534);
final Integer y = new Integer(43534);

assertTrue(x == x);
assertTrue(x.equals(x));

assertFalse(x == y);
assertTrue(x.equals(y));
```

- Иногда в нашем коде приходится реализовывать equals()
- Где есть реализация, там нужны и тесты

Кто помнит три свойства, которым должна удовлетворять корректная реализация equals()?


EVERYBODY STAND BACK.
I know Java equals contract
PERL!
TAP TAP

~~Три~~ Четыре свойства

- Рефлексия (reflexive)
- Симметрия (symmetric)
- Транзитивность (transitive)
- Консистентность (consistent)

Рефлексия

Для любого `a`, всегда верно `a.equals(a)`

$$\boxed{a} = \boxed{a}$$

Симметрия

Если верно `a.equals(b)`

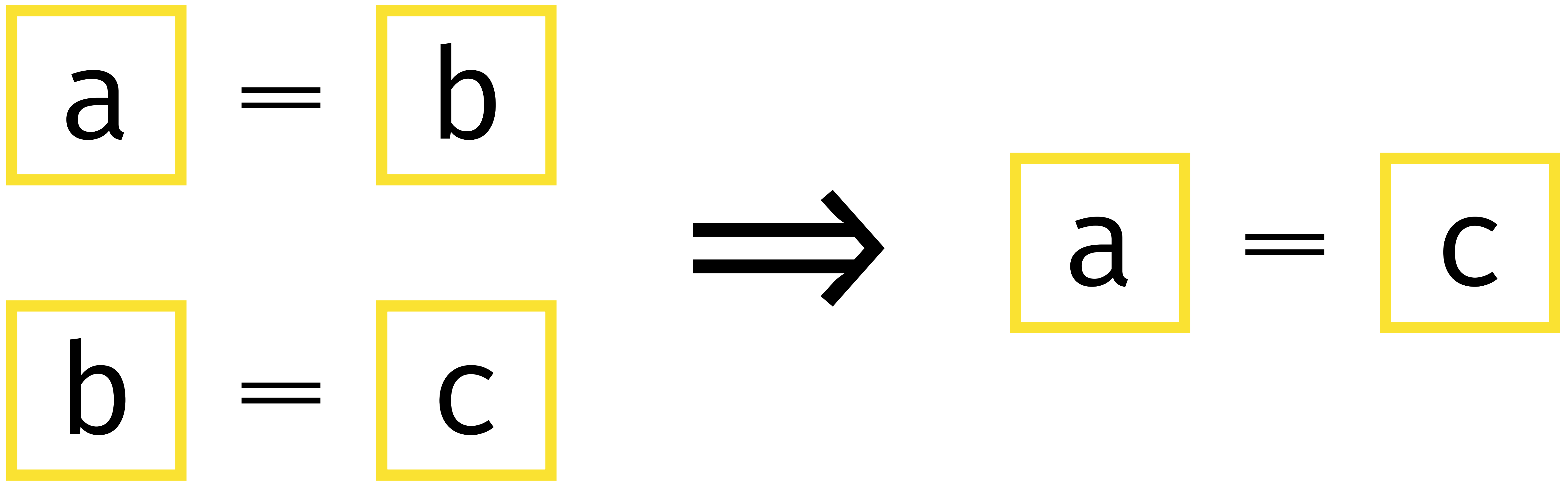
$\Rightarrow$ тогда верно `b.equals(a)`

$$\boxed{a} = \boxed{b} \quad \Rightarrow \quad \boxed{b} = \boxed{a}$$

Транзитивность

Если верно `a.equals(b)` и `b.equals(c)`

⇒ тогда верно `a.equals(c)`



Консистентность

Если `a.equals(b)` сейчас

$\Rightarrow$ тогда `a.equals(b)` в будущем

$$\boxed{a} = \boxed{b} \Rightarrow \boxed{a} = \boxed{b}$$

“A common source of bugs is the failure to override the hashCode method. You must override hashCode in every class that overrides equals. ”

Josh Bloch, Effective Java

equals() + hashCode()

Поэтому дальше я говорю про оба метода

- Object::equals
- Object::hashCode

Пример класса

```
public class ParsedURL {
    private final Protocol protocol;
    private final String hostname;
    private final int port;
    private final String path;
    private final Map<String, String> parameters;
    …
}
```

Как все это тестировать?

Вариант 1

Никак не тестировать:

- Просто и быстро
- Не надежно

Вариант 2

Очень простые тесты:

- Относительно просто и быстро
- Все еще не надежно

Примеры “простых” тестов

- 1 на равенство двух не идентичных объектов
- 5 на неравенство двух разных объектов
- 1 на рефлексию объекта с самим собой
- 1 на симметрию двух равных объектов
- 1 на консистентность (как это проверить вообще)?

“Простой” тест

```
@Test
public void testSimple() {
    final ParsedURL u1 = new ParsedURL(:));
    final ParsedURL u2 = new ParsedURL(:D);
    assertNotEquals(u1, u2);
}
```

- Получилось примерно 10 тестов
- Достаточно ли этого?

Таких тестов недостаточно

- Что с классами-потомками и классами-предками?
- Что с double/float полями?
- Что с nullable полями?
- Что будет при изменении класса?

Таких тестов недостаточно

- Что с классами-потомками и классами-предками?
- Что с double/float полями?
- Что с nullable полями?
- Что будет при изменении класса?
- Мы забыли про Object::hashCode

Вариант 3

Тесты с EqualsVerifier:

- Просто и быстро
- Очень надежно

Пример теста с EqualsVerifier

```
@Test
public void testEqualsContract() {
    EqualsVerifier
        .forClass(Value.class)
        .verify();
}
```

- Один (!) тест с EqualsVerifier дает 100% покрытие по строкам кода
- Если это не так — EqualsVerifier выдаст ошибку

Пример 1
Хороший equals при плохом классе

```java
public final class Value {
    private int x;
    private int y;
    public Value(int x, int y) {
        this.x = x;
        this.y = y;
    }
…
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Value)) return false;
        Value value = (Value) o;
        return x == value.x && y == value.y;
    }
}
```

```
java.lang.AssertionError:
Mutability: equals depends on mutable field x.
```

Чем это опасно?

```
@Test
public void testValueStaysInSet() {
    final Set<Value> set = new HashSet<>();
    final Value a = new Value(123, 789);
    set.add(a);
    assertTrue(set.contains(a));
    a.setX(0);
    assertFalse(set.contains(a));
}
```

Как починить?

1. Починить код
2. Рассказать тестам, что так и надо

```
public final class Value {
    private final int x;
    private final int y;
    public Value(int x, int y) {
        this.x = x;
        this.y = y;
    }
…
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Value)) return false;
        Value value = (Value) o;
        return x == value.x && y == value.y;
    }
}
```

2. Рассказать тестам, что так и надо

```
@Test
public void testLooseEqualsContract() {
    EqualsVerifier
        .forClass(Value.class)
        .suppress(Warning.NONFINAL_FIELDS)
        .verify();
}
```

Пример 2
Сын за отца в ответе

```java
public class ChildValue extends Value {
    private int z;
    public ChildValue(int x, int y, int z) {
        super(x, y);
        this.z = z;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof ChildValue))
            return false;
        if (!super.equals(o)) return false;
        ChildValue that = (ChildValue) o;
        return z == that.z;
    }
```

```
java.lang.AssertionError: Symmetry:
  ChildValue{z=1, x=1, y=1}
does not equal superclass instance
  Value{x=1, y=1}
```

```
java.lang.AssertionError: Symmetry:
  ChildValue{z=1, x=1, y=1}
does not equal superclass instance
  Value{x=1, y=1}
```

Тест, который провел EqualsVerifier

```java
final Value a = new Value(1, 1);
final ChildValue b = new ChildValue(1, 1, 1);

assertTrue("a, b", a.equals(b));
assertFalse("b, a", b.equals(a));
```

Чем это опасно?

```
@Test
public void setSizeIsTwo() {
    final Set<Value> set = new HashSet<>();
    final Value a = new Value(0, 0);
    final ChildValue b = new ChildValue(0, 0, -29791);
    set.add(a);
    set.add(b);
    assertEquals(2, set.size());
}
```

Чем это опасно?

```java
@Test
public void setSizeIsTwo() {
    final Set<Value> set = new HashSet<>();
    final Value a = new Value(0, 0);
    final ChildValue b = new ChildValue(0, 0, -29791);
    set.add(b);
    set.add(a);
    assertEquals(1, set.size());
}
```

Как чинить асимметрию?

```
public boolean canEqual(Object other) {
    …
}
```

https://www.artima.com/lejava/articles/equality.html

Типичная ошибка — реализуем equals() специально для тестов

Реализуем equals() для тестов

- Очень хотим вот так писать
  assertEquals(expected, actual);
- Это плохой equals():
  - Не несет смысл, важный для доменной области
  - Будет незаметно использован в ПРОД коде

Вот так хорошо

```
assertTrue(
    EqualsBuilder.reflectionEquals(
        expected, actual
    )
);
```

А вот так лучше

```
assertThat(
    actual,
    Matchers.reflectionEquals(expected)
);
```

Типичные возражения

Типичные возражения

- Я использую EqualsBuilder и HashCodeBuilder
- Я использую кодогенерацию из IDEA

Часть проблем остается

- Проблемы с мутабельными полями остаются
- Проблемы с наследниками остаются
- Возможные ошибки при добавлении полей остаются
- Возможно ошибок нет сейчас, но они могут появится в будущем

Альтернативные решения

- Google Auto Value
- @EqualsAndHashCode из Lombok
- Дождаться появления value классов в Java

EqualsVerifier выводы

- 100% покрытие одной строчкой кода
- Не надо писать новые тесты при изменении класса
- (Почти) Не надо помнить детали контракта equals()
- Все это гарантируется не только сегодня, но и в будущем
- Бесплатно как “бесплатное пиво”

Отлично, есть относительно простой способ искать ошибки в очень небольшой части кода.

А хочется очень простой способ искать ошибки по всему коду.

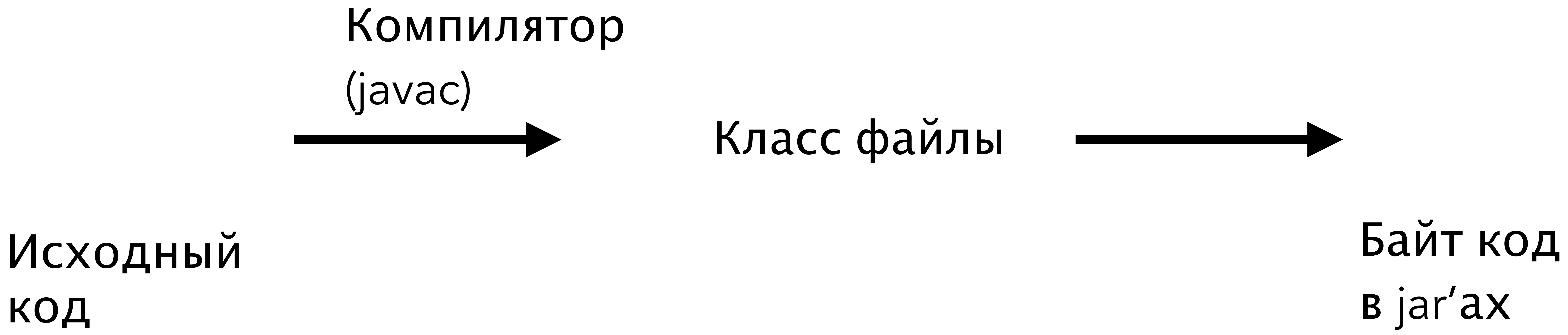
Часть II
ErrorProne

Статические анализаторы

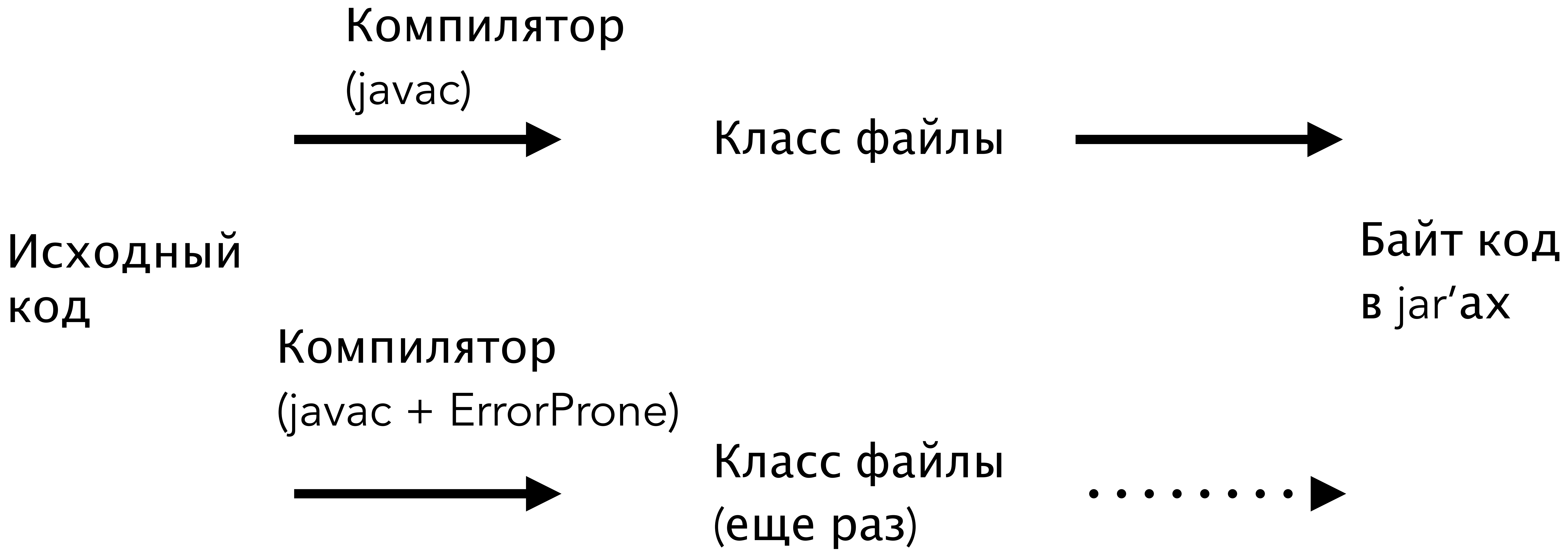
- FindBugs
- SonarQube
- PMD

ErrorProne — компилятор на стероидах

Как работает ErrorProne



Как работает ErrorProne



Интеграция

- IntelliJ IDEA
- Ant
- Maven
- Gradle
- Bazel

“Using Error Prone to augment the compiler’s type analysis, you can catch more mistakes before they cost you time, or end up as bugs in production.
We use Error Prone in Google’s Java build system
to eliminate classes of serious bugs from entering our code,
and we’ve open-sourced it, so you can too!”

http://errorprone.info/

“Using Error Prone to augment the compiler’s type analysis, you can catch more mistakes before they cost you time, or end up as bugs in production.
We use Error Prone in Google’s Java build system to eliminate classes of serious bugs from entering our code, and we’ve open-sourced it, so you can too!”

http://errorprone.info/

“Definitely embarrassing. I guess I’m back to liking Error Prone even though it sometimes annoys me :-)”

Doug Lea

Уровень 1
Подключил, починил и забыл

AndroidInjectionBeforeSuper
AndroidInjection.inject() should always be invoked before calling super.lifecycleMethod()

ArrayEquals
Reference equality used to compare arrays

ArrayFillIncompatibleType
Arrays.fill(Object[], Object) called with incompatible types.

ArrayHashCode
hashcode method on array does not hash array contents

ArrayToString
Calling toString on an array does not provide useful information

ArraysAsListPrimitiveArray
Arrays.asList does not autobox primitive arrays, as one might expect.

AsyncCallableReturnsNull
AsyncCallable should not return a null Future, only a Future whose result is null.

AsyncFunctionReturnsNull
AsyncFunction should not return a null Future, only a Future whose result is null.

AutoValueConstructorOrderChecker
Arguments to AutoValue constructor are in the wrong order

BadShiftAmount
Shift by an amount that is out of range

BundleDeserializationCast
Object serialized in Bundle may have been flattened to base type.

ChainingConstructorIgnoresParameter
The called constructor accepts a parameter with the same name and type as one of its caller's parameters, but its caller doesn't pass that parameter to it. It's likely that it was intended to.

CheckReturnValue
Ignored return value of method that is annotated with @CheckReturnValue

CollectionIncompatibleType
Incompatible type as argument to Object-accepting Java collections method

ComparableType
Implementing 'Comparable<T>' where T is not compatible with the implementing class.

ComparisonOutOfRange
Comparison to value that is out of range for the compared type

CompatibleWithAnnotationMisuse
@CompatibleWith's value is not a type argument.

CompileTimeConstant
Non-compile-time constant expression passed to parameter with @CompileTimeConstant type annotation.

ComplexBooleanConstant
Non-trivial compile time constant boolean expressions shouldn't be used.

ConditionalExpressionNumericPromotion
A conditional expression with numeric operands of differing types will perform binary numeric promotion of the operands; when these operands are of reference types, the expression's result may not be of the expected type.

ConstantOverflow
Compile-time constant expression overflows

DaggerProvidesNull
Dagger @Provides methods may not return null unless annotated with @Nullable

DeadException
Exception created but not thrown

DeadThread
Thread created but not started

DoNotCall
This method should not be called.

EqualsNaN
== NaN always returns false; use the isNaN methods instead

EqualsReference
== must be used in equals method to check equality to itself or an infinite loop will occur.

ForOverride
Method annotated @ForOverride must be protected or package-private and only invoked from declaring class, or from an override of the method

FormatString
Invalid printf-style format string

FormatStringAnnotation
Invalid format string passed to formatting method.

FunctionalInterfaceMethodChanged
Casting a lambda to this @FunctionalInterface can cause a behavior change from casting to a functional superinterface, which is surprising to users. Prefer decorator methods to this surprising behavior.

FuturesGetCheckedIllegalExceptionType
Futures.getChecked requires a checked exception type with a standard constructor.

GetClassOnAnnotation
Calling getClass() on an annotation may return a proxy class

GetClassOnClass
Calling getClass() on an object of type Class returns the Class object for java.lang.Class; you probably meant to operate on the object directly

GuardedBy
Checks for unguarded accesses to fields and methods with @GuardedBy annotations

GuiceAssistedInjectScoping
Scope annotation on implementation class of AssistedInject factory is not allowed

GuiceAssistedParameters
A constructor cannot have two @Assisted parameters of the same type unless they are disambiguated with named @Assisted annotations.

GuiceInjectOnFinalField
Although Guice allows injecting final fields, doing so is disallowed because the injected value may not be visible to other threads.

HashtableContains
contains() is a legacy method that is equivalent to containsValue()

IdentityBinaryExpression
A binary expression where both operands are the same is usually incorrect.

Immutable
Type declaration annotated with @Immutable is not immutable

ImmutableModification
Modifying an immutable collection is guaranteed to throw an exception and leave the collection unmodified

IncompatibleArgumentType
Passing argument to a generic method with an incompatible type.

IndexOfChar
The first argument to indexOf is a Unicode code point, and the second is the index to start the search from

InexactVarargsConditional
Conditional expression in varargs call contains array and non-array arguments

InfiniteRecursion
This method always recurses, and will cause a StackOverflowError

InjectMoreThanOneScopeAnnotationOnClass
A class can be annotated with at most one scope annotation.

InvalidPatternSyntax
Invalid syntax used for a regular expression

InvalidTimeZoneID
Invalid time zone identifier. TimeZone.getTimeZone(String) will silently return GMT instead of the time zone you intended.

IsInstanceOfClass
The argument to Class#isInstance(Object) should not be a Class

IsLoggableTagLength
Log tag too long, cannot exceed 23 characters.

JUnit3TestNotRun
Test method will not be run; please correct method signature (Should be public, non-static, and method name should begin with "test").

JUnit4ClassAnnotationNonStatic
This method should be static

JUnit4SetUpNotRun
setUp() method will not be run; please add JUnit's @Before annotation

JUnit4TearDownNotRun
tearDown() method will not be run; please add JUnit's @After annotation

JUnit4TestNotRun
This looks like a test method but is not run; please add @Test and @Ignore, or, if this is a helper method, reduce its visibility.

JUnitAssertSameCheck
An object is tested for reference equality to itself using JUnit library.

JavaxInjectOnAbstractMethod
Abstract and default methods are not injectable with javax.inject.Inject

LiteByteStringUtf8
This pattern will silently corrupt certain byte sequences from the serialized protocol message. Use ByteString or byte[] directly

LoopConditionChecker
Loop condition is never modified in loop body.

MislabeledAndroidString
Certain resources in `android.R.string` have names that do not match their content

MissingSuperCall
Overriding method is missing a call to overridden super method

MisusedWeekYear
Use of "YYYY" (week year) in a date pattern without "ww" (week in year). You probably meant to use "yyyy" (year) instead.

MockitoCast
A bug in Mockito will cause this test to fail at runtime with a ClassCastException

MockitoUsage
Missing method call for verify(mock) here

ModifyingCollectionWithItself
Using a collection function with itself as the argument.

MoreThanOneInjectableConstructor
This class has more than one @Inject-annotated constructor. Please remove the @Inject annotation from all but one of them.

MustBeClosedChecker
The result of this method must be closed.

NCopiesOfChar
The first argument to nCopies is the number of copies, and the second is the item to copy

NonCanonicalStaticImport
Static import of type uses non-canonical name

NonFinalCompileTimeConstant
@CompileTimeConstant parameters should be final or effectively final

NonRuntimeAnnotation
Calling getAnnotation on an annotation that is not retained at runtime.

NullTernary
This conditional expression may evaluate to null, which will result in an NPE when the result is unboxed.

OptionalEquality
Comparison using reference equality instead of value equality

OverlappingQualifierAndScopeAnnotation
Annotations cannot be both Scope annotations and Qualifier annotations: this causes confusion when trying to use them.

OverridesJavaxInjectableMethod
This method is not annotated with @Inject, but it overrides a method that is annotated with @javax.inject.Inject. The method will not be Injected.

PackageInfo
Declaring types inside package-info.java files is very bad form

PreconditionsCheckNotNull
Literal passed as first argument to Preconditions.checkNotNull() can never be null

PreconditionsCheckNotNullPrimitive
First argument to `Preconditions.checkNotNull()` is a primitive rather than an object reference

PredicateIncompatibleType
Using ::equals as an incompatible Predicate; the predicate will always return false

PrivateSecurityContractProtoAccess
Access to a private protocol buffer field is forbidden. This protocol buffer carries a security contract, and can only be created using an approved library. Direct access to the fields is forbidden.

ProtoFieldNullComparison
Protobuf fields cannot be null

ProtocolBufferOrdinal
To get the tag number of a protocol buffer enum, use getNumber() instead.

ProvidesMethodOutsideOfModule
@Provides methods need to be declared in a Module to have any effect.

RandomCast
Casting a random number in the range [0.0, 1.0) to an integer or long always results in 0.

RandomModInteger
Use Random.nextInt(int). Random.nextInt() % n can have negative results

RectIntersectReturnValueIgnored
Return value of android.graphics.Rect.intersect() must be checked

RestrictedApiChecker
Check for non-whitelisted callers to RestrictedApiChecker.

ReturnValueIgnored
Return value of this method must be used

SelfAssignment
Variable assigned to itself

SelfComparison
An object is compared to itself

SelfEquals
Testing an object for equality with itself will always be true.

ShouldHaveEvenArgs
This method must be called with an even number of arguments.

SizeGreaterThanOrEqualsZero
Comparison of a size >= 0 is always true, did you intend to check for non-emptiness?

StreamToString
Calling toString on a Stream does not provide useful information

StringBuilderInitWithChar
StringBuilder does not have a char constructor; this invokes the int constructor.

SuppressWarningsDeprecated
Suppressing "deprecated" is probably a typo for "deprecation"

ThrowIfUncheckedKnownChecked
throwIfUnchecked(knownCheckedException) is a no-op.

ThrowNull
Throwing 'null' always results in a NullPointerException being thrown.

TruthSelfEquals
isEqualTo should not be used to test an object for equality with itself; the assertion will never fail.

TryFailThrowable
Catching Throwable/Error masks failures from fail() or assert*() in the try block

TypeParameterQualifier
Type parameter used as type qualifier

UnnecessaryTypeArgument
Non-generic methods should not be invoked with type arguments

UnusedAnonymousClass
Instance created but never used

UnusedCollectionModifiedInPlace
Collection is modified in place, but the result is not used

Список проверок
Не надо читать

AndroidInjectionBeforeSuper
AndroidInjection.inject() should always be invoked before calling super.lifecycleMethod()

ArrayEquals
Reference equality used to compare arrays

ArrayFillIncompatibleType
Arrays.fill(Object[], Object) called with incompatible types.

ArrayHashCode
hashcode method on array does not hash array contents

ArrayToString
Calling toString on an array does not provide useful information

ArraysAsListPrimitiveArray
Arrays.asList does not autobox primitive arrays, as one might expect.

AsyncCallableReturnsNull
AsyncCallable should not return a null Future, only a Future whose result is null.

AsyncFunctionReturnsNull
AsyncFunction should not return a null Future, only a Future whose result is null.

AutoValueConstructorOrderChecker
Arguments to AutoValue constructor are in the wrong order

BadShiftAmount
Shift by an amount that is out of range

BundleDeserializationCast
Object serialized in Bundle may have been flattened to base type.

ChainingConstructorIgnoresParameter
The called constructor accepts a parameter with the same name and type as one of its caller's parameters, but its caller doesn't pass that parameter to it. It's likely that it was intended to.

CheckReturnValue
Ignored return value of method that is annotated with @CheckReturnValue

CollectionIncompatibleType
Incompatible type as argument to Object-accepting Java collections method

ComparableType
Implementing 'Comparable<T>' where T is not compatible with the implementing class.

ComparisonOutOfRange
Comparison to value that is out of range for the compared type

CompatibleWithAnnotationMisuse
@CompatibleWith's value is not a type argument.

CompileTimeConstant
Non-compile-time constant expression passed to parameter with @CompileTimeConstant type annotation.

ComplexBooleanConstant
Non-trivial compile time constant boolean expressions shouldn't be used.

ConditionalExpressionNumericPromotion
A conditional expression with numeric operands of differing types will perform binary numeric promotion of the operands; when these operands are of reference types, the expression's result may not be of the expected type.

ConstantOverflow
Compile-time constant expression overflows

DaggerProvidesNull
Dagger @Provides methods may not return null unless annotated with @Nullable

DeadException
Exception created but not thrown

DeadThread
Thread created but not started

DoNotCall
This method should not be called.

EqualsNaN
== NaN always returns false; use the isNaN methods instead

EqualsReference
== must be used in equals method to check equality to itself or an infinite loop will occur.

ForOverride
Method annotated @ForOverride must be protected or package-private and only invoked from declaring class, or from an override of the method

FormatString
Invalid printf-style format string

FormatStringAnnotation
Invalid format string passed to formatting method.

FunctionalInterfaceMethodChanged
Casting a lambda to this @FunctionalInterface can cause a behavior change from casting to a functional superinterface, whic[illegible] surprising to users[illegible] Prefer decorator methods to this surprising behavior.

FuturesGetCheckedIllegalExceptionType
Futures.getChecked requires a checked exception type with a standard constructor.

GetClassOnAnnotation
Calling getClass() on an annotation may return a proxy class

GetClassOnClass
Calling getClass() on an object of type Class returns the Class object for java.lang.Class; you probably meant to operate on the object directly

GuardedBy
Checks for unguarded accesses to fields and methods with @GuardedBy annotations

GuiceAssistedInjectScoping
Scope annotation on implementation class of AssistedInject factory is not allowed

GuiceAssistedParameters
A constructor cannot have two @Assisted parameters of the same type unless they are disambiguated with named @Assisted annotations.

GuiceInjectOnFinalField
Although Guice allows injecting final fields, doing so is disallowed because the injected value may not be visible to other threads.

HashtableContains
contains() is a legacy method that is equivalent to containsValue()

IdentityBinaryExpression
A binary expression where both operands are the same is usually incorrect.

Immutable
Type declaration annotated with @Immutable is not immutable

ImmutableModification
Modifying an immutable collection is guaranteed to throw an exception and leave the collection unmodified

IncompatibleArgumentType
Passing argument to a generic method with an incompatible type.

IndexOfChar
The first argument to indexOf is a Unicode code point, and the second is the index to start the search from

InexactVarargsConditional
Conditional expression in varargs call contains array and non-array arguments

InfiniteRecursion
This method always recurses, and will cause a StackOverflowError

InjectMoreThanOneScopeAnnotationOnClass
A class can be annotated with at most one scope annotation.

InvalidPatternSyntax
Invalid syntax used for a regular expression

InvalidTimeZoneID
Invalid time zone identifier. TimeZone.getTimeZone(String) will silently return GMT instead of the time zone you intended.

IsInstanceOfClass
The argument to Class#isInstance(Object) should not be a Class

IsLoggableTagLength
Log tag too long, cannot exceed 23 characters.

JUnit3TestNotRun
Test method will not be run; please correct method signature (Should be public, non-static, and method name should begin with "test").

JUnit4ClassAnnotationNonStatic
This method should be static

JUnit4SetUpNotRun
setUp() method will not be run; please add JUnit's @Before annotation

JUnit4TearDownNotRun
tearDown() method will not be run; please add JUnit's @After annotation

JUnit4TestNotRun
This looks like a test method but is not run; please add @Test and @Ignore, or, if this is a helper method, reduce its visibility.

JUnitAssertSameCheck
An object is tested for reference equality to itself using JUnit library.

JavaxInjectOnAbstractMethod
Abstract and default methods are not injectable with javax.inject.Inject

LiteByteStringUtf8
This pattern will silently corrupt certain byte sequences from the serialized protocol message. Use ByteString or byte[] directly

LoopConditionChecker
Loop condition is never modified in loop body.

MislabeledAndroidString
Certain resources in `android.R.string` have names that do not match their content

MissingSuperCall
Overriding method is missing a call to overridden super method

MisusedWeekYear
Use of "YYYY" (week year) in a date pattern without "ww" (week in year). You probably meant to use "yyyy" ([illegible]ar) instead.

MockitoCast
A bug in Mockito will cause this test to fail at runtime with a ClassCastException

MockitoUsage
Missing method call for verify(mock) here

ModifyingCollectionWithItself
Using a collection function with itself as the argument.

MoreThanOneInjectableConstructor
This class has more than one @Inject-ann[illegible] co[illegible]uctor. [illegible]ease [illegible]ove the @[illegible]ject a[illegible]om all but one of them.

MustBeClosedChecker
The result of this method must be clo[illegible].

NCopiesOfChar
The first argume[illegible] to nCopies [illegible]e num[illegible] of copies, and [illegible]e second is the item to copy

NonCanonica[illegible]taticImport
[illegible] import [illegible]ype uses non-can[illegible]cal na[illegible]

NonFi[illegible]Comp[illegible]TimeConstant
@Compi[illegible]me[illegible]stant paramete[illegible] should be final or effectively final

NonRunti[illegible]Anno[illegible]
Calling get[illegible]notation on an annotation that is not retained at runtime.

NullTerna[illegible]
This con[illegible]onal expression may evaluate to null, which will result in an NPE when the result is [illegible]boxed.

[illegible]alEquality
Comparison using reference equality instead of value equality

[illegible]verlappingQualifierAndScopeAnnotation
[illegible]nnotations cannot be both Scope annotations and Qualifier an[illegible]ta[illegible]ons: this [illegible]ses confusion [illegible]en trying to [illegible]m.

OverridesJavaxInjectableMethod
This method is not annotated with @Inject, but it overrides a [illegible]th[illegible] that is annota[illegible]d with @java[illegible]ct. The method will not be Injected.

PackageInfo
Declaring types inside package-info.jav[illegible]es is v[illegible] bad for[illegible]

PreconditionsCheckNotNull
Literal passed as first arg[illegible]ent to Precond[illegible]ns.che[illegible]Not[illegible]ll() can ne[illegible] be null

PreconditionsCheckNotN[illegible]Primitive
First argument to `Precondit[illegible]s.c[illegible]otN[illegible]()` is a[illegible]itive rather than an object reference

[illegible]eIncompatibleType
Using :[illegible]als as an incompatible Predicate; the predicate will always return false

PrivateSe[illegible]tyContractProtoAccess
Access to a[illegible]vate protocol buffer field is forbidden. This protocol buffer carries a security contract, and can only be created using an approved library. Direct access to the fields is forbidden.

ProtoFieldN[illegible]Comparison
Protobuf fie[illegible] cannot be null

Protoco[illegible]ferOrdinal
[illegible]e tag number of a protocol buffer enum, use getNumber() instead.

ProvidesMethodOutsideOfModule
@Provides methods need to be declared in a Module to have any effect.

RandomCast
Casting a random number in the range [0.0, 1.0) to an integer or long always results in 0.

RandomModInteger
Use Random.nextInt(int). Random.nextInt() % n can have negative results

RectIntersectReturnValueIgnored
Return value of android.graphics.Rect.intersect() must be checked

RestrictedApiChecker
Check for non-whitelisted callers to RestrictedApiChecker.

ReturnValueIgnored
Return value of this method must be used

SelfAssignment
Variable assigned to itself

SelfComparison
An object is compared to itself

SelfEquals
Testing an object for equality with itself will always be true.

ShouldHaveEvenArgs
This method must be called with an even number of arguments.

SizeGreaterThanOrEqualsZero
Comparison of a size >= 0 is always true, did you intend to check for non-emptiness?

StreamToString
Calling toString on a Stream does not provide useful information

StringBuilderInitWithChar
StringBuilder does not have a char constructor; this invokes the int constructor.

SuppressWarningsDeprecated
Suppressing "deprecated" is probably a typo for "deprecation"

ThrowIfUncheckedKnownChecked
throwIfUnchecked(knownCheckedException) is a no-op.

ThrowNull
Throwing 'null' always results in a NullPointerException being thrown.

TruthSelfEquals
isEqualTo should not be used to test an object for equality with itself; the assertion will never fail.

TryFailThrowable
Catching Throwable/Error masks failures from fail() or assert*() in the try block

TypeParameterQualifier
Type parameter used as type qualifier

UnnecessaryTypeArgument
Non-generic methods should not be invoked with type arguments

UnusedAnonymousClass
Instance created but never used

UnusedCollectionModifiedInPlace
Collection is modified in place, but the result is not used

Пример 3
Все животные равны

```java
Object[] a = new Object[]{1, 2, 3};
Object[] b = new Object[]{1, 2, 3};

if (a.equals(b)) {
    System.out.println("arrays are equal!");
}
```

```java
Object[] a = new Object[]{1, 2, 3};
Object[] b = new Object[]{1, 2, 3};

if (a.equals(b)) {
    System.out.println("arrays are equal!");
}
```

```
Error:(14, 27) java: [ArrayEquals] Reference equality used
to compare arrays
    (see http://errorprone.info/bugpattern/ArrayEquals)
  Did you mean 'if (Arrays.equals(a, b)) {'?
```

```
Object[] a = new Object[]{1, 2, 3};
Object[] b = new Object[]{1, 2, 3};

if (a.equals(b)) {
    System.out.println("arrays are equal!");
}
```

```
Error:(14, 27) java: [ArrayEquals] Reference equality used
to compare arrays
    (see http://errorprone.info/bugpattern/ArrayEquals)
  Did you mean 'if (Arrays.equals(a, b)) {'?
```

Пример 4

No toString() attached

```java
int[] a = {1, 2, 3};
System.out.println("array a = " + a);
```

```java
int[] a = {1, 2, 3};
System.out.println("array a = " + a);
```

```
Error:(23, 43) java: [ArrayToString] Calling toString
on an array does not provide useful information
    (see http://errorprone.info/bugpattern/
ArrayToString)
  Did you mean 'System.out.println("array a = " +
Arrays.toString(a));’?
```

```
int[] a = {1, 2, 3};
System.out.println("array a = " + a);
```

```
Error:(23, 43) java: [ArrayToString] Calling toString
on an array does not provide useful information
    (see http://errorprone.info/bugpattern/
ArrayToString)
  Did you mean 'System.out.println("array a = " +
Arrays.toString(a));'?
```

Пример 5
Форматировал, форматировал, да не отфармотировал

```
PrintStream out = System.out;
out.println(String.format("Hello, %s%s", "World!"));
out.println(String.format("Hello, $s", “World!"));
```

```
PrintStream out = System.out;
out.println(String.format("Hello, %s%s", "World!"));
out.println(String.format("Hello, $s", “World!"));
```

```
Error:(14, 39) java: [FormatString] missing argument
for format specifier '%s'
    (see http://errorprone.info/bugpattern/
FormatString)
Error:(15, 39) java: [FormatString] extra format
arguments: used 0, provided 1
    (see http://errorprone.info/bugpattern/
FormatString)
```

```
PrintStream out = System.out;
out.println(String.format("Hello, %s%s", "World!"));
out.println(String.format("Hello, $s", “World!"));
```

```
Error:(14, 39) java: [FormatString] missing argument
for format specifier '%s'
    (see http://errorprone.info/bugpattern/
FormatString)
Error:(15, 39) java: [FormatString] extra format
arguments: used 0, provided 1
    (see http://errorprone.info/bugpattern/
FormatString)
```

Уровень 2
Аннотировал, починил и забыл

Уровень 2

- Придется немного поработать
- Результаты того стоят

Пример 6

```
public class SelfSynchronized {


    private final List<Integer> lst = new ArrayList<>();

    public synchronized void add(final int x) {
        lst.add(x);
    }

    public int size() {
        return lst.size();
    }
...
```

```
public class SelfSynchronized {

    @GuardedBy(“this")
    private final List<Integer> lst = new ArrayList<>();

    public synchronized void add(final int x) {
        lst.add(x);
    }

    public int size() {
        return lst.size();
    }
 …

Error:(19, 16) java: [GuardedBy] This access should be guarded
by 'this', which is not currently held
    (see http://errorprone.info/bugpattern/GuardedBy)
```

```java
public class SelfSynchronized {

    @GuardedBy(“this")
    private final List<Integer> lst = new ArrayList<>();

    public synchronized void add(final int x) {
        lst.add(x);
    }

    public synchronized int size() {
        return lst.size();
    }
...
```

Пример 7
Пожалуйста, закрывайте двери

```java
private static class Resource implements AutoCloseable {

    public Resource() {}

    @Override
    public void close() throws Exception {}
    …
}

public void doWorkWithResource() {
    final Resource r = new Resource();
    r.doWork();
}
```

```
private static class Resource implements AutoCloseable {
    @MustBeClosed
    public Resource() {}

    @Override
    public void close() throws Exception {}
    …
}

public void doWorkWithResource() {
    final Resource r = new Resource();
    r.doWork();
}
Error:(39, 28) java: [MustBeClosedChecker] The result of this
method must be closed.
    (see http://errorprone.info/bugpattern/MustBeClosedChecker)
```

Уровень 3

WARNING => ERROR

Преимущества ErrorProne

- Его нельзя проигнорировать
- Очень низкий уровень false positive
- Можно постепенно “закручивать” гайки
- Бесплатный как “бесплатное пиво”

Заключение

Что с этим делать?

1. Обсудить доклад с разработчиками

2. Использовать EqualsVerifier для проверки ваших equals() и hashCode()

3. Включить ErrorProne для поиска гадких ошибок

4. Аннотировать ваш код для усиления проверок ErrorProne

Контакты

https://www.linkedin.com/in/asatarin

https://twitter.com/asatarin

asatarin@yandex.ru

Ссылки

- http://jqno.nl/equalsverifier/
- “Not all equals methods are created equal” by Jan Ouwens https://youtu.be/pNJ_O10XaoM
- https://www.artima.com/lejava/articles/equality.html
- http://www.drdobbs.com/jvm/java-qa-how-do-i-correctly-implement-th/184405053

Ссылки

- http://errorprone.info/
- https://github.com/google/auto/blob/master/value/userguide/index.md