Safe(r) Kotlin Code

Static Analysis Tools for Kotlin

Marvin Ramin

Static Analysis

```
public class kotlinconf {
  var YEAR = 2018

      fun Welcome(year: Int?) {
       println("Welcome to KotlinConf ${year!!}!")
  }
}
```

```
public class kotlinconf {
  var YEAR = 2018

      fun Welcome(year: Int?) {
       println("Welcome to KotlinConf ${year!!}!")
  }
}
```

Modern. Expressive. Safe.

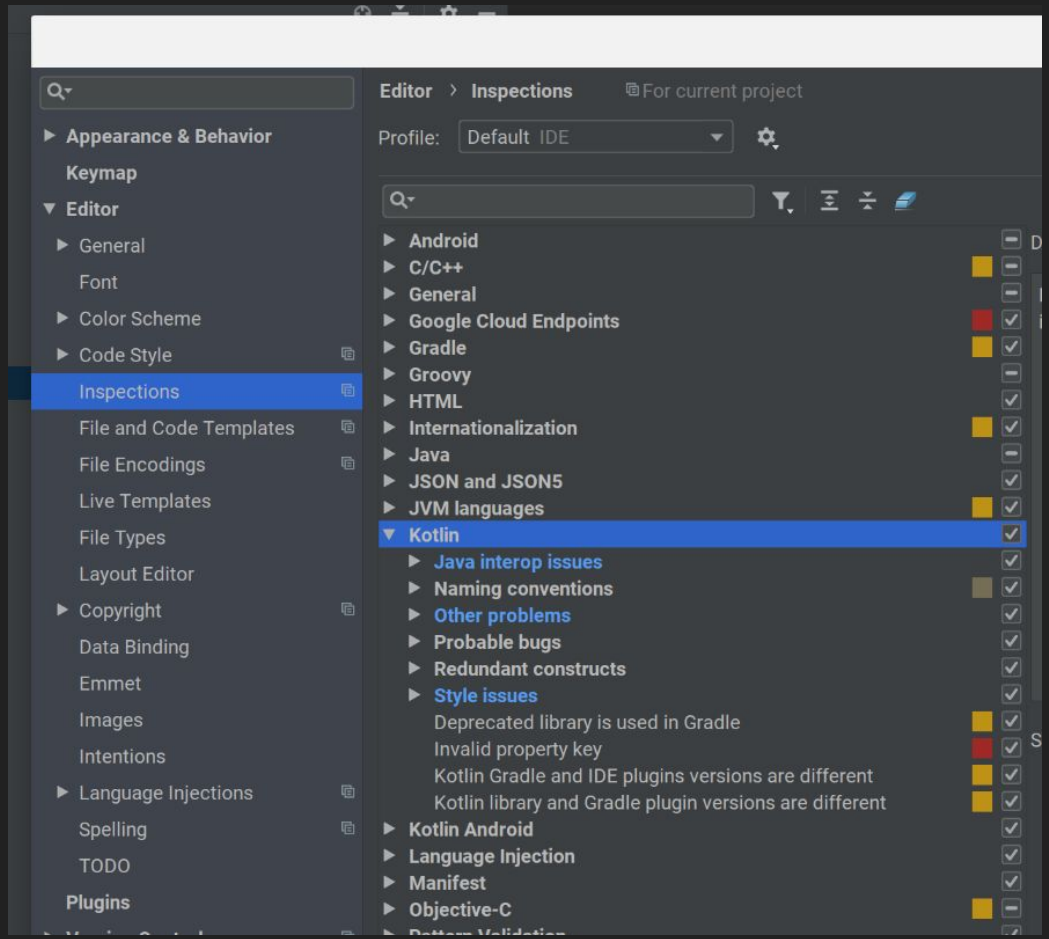
https://developer.android.com/kotlin/

"The introduction of Kotlin code in Android applications initially written in Java increases the quality of [...] 50% of the apps."

Mateus, B. G., & Martinez, M. (2018). An Empirical Study on Quality of Android Applications written in Kotlin language.

Kotlin Compiler

IntelliJ



Settings ➜ Editor ➜ Inspections

Hints during development

Hints during development

Support automatic fixing

Hints during development

Support automatic fixing

Can be run from the CLI

Bytecode Analysis

Android Lint

Most checks Android specific

Most checks Android specific

Checks run both on Java & Kotlin

Most checks Android specific

Checks run both on Java & Kotlin

Support for non-Android projects

Android Project `./gradlew lint`

Android Project

```
./gradlew lint
```

Kotlin

```
apply plugin com.android.lint
./gradlew lint
```

```
lintOptions {
    abortOnError true
    ignoreWarnings false
    lintConfig file("config.xml")
    ...
}
```

ktlint
detekt

ktlint

ktlint

No config required

ktlint

No config required

Focussed on code style

ktlint

No config required

Focussed on code style

Built in formatter

ktlint

```
ktlint
```

ktlint

```
ktlint "**/src/*.kt"
```

ktlint

```
ktlint "**/src/*.kt" --format
```

detekt

detekt

Rules can be configured in detail

detekt

Rules can be configured in detail

Many different rule sets

detekt

Rules can be configured in detail

Many different rule sets

Wraps ktlint

detekt

```
java -jar detekt.jar --input src/
```

detekt

```
java -jar detekt.jar --input src/
                     --config config.yml
```

detekt

```
plugins {
    id "io.gitlab.arturbosch.detekt" version "[version]"
}

detekt {
    input = file("src/main/kotlin")
}
```

Gradle Plugins

```
> ./gradlew check
```

```
> ./gradlew check
...
```

```
> ./gradlew check
...
...
```

```
> ./gradlew check
...
...

Static Analysis suite finished

Found 8572 issues
```

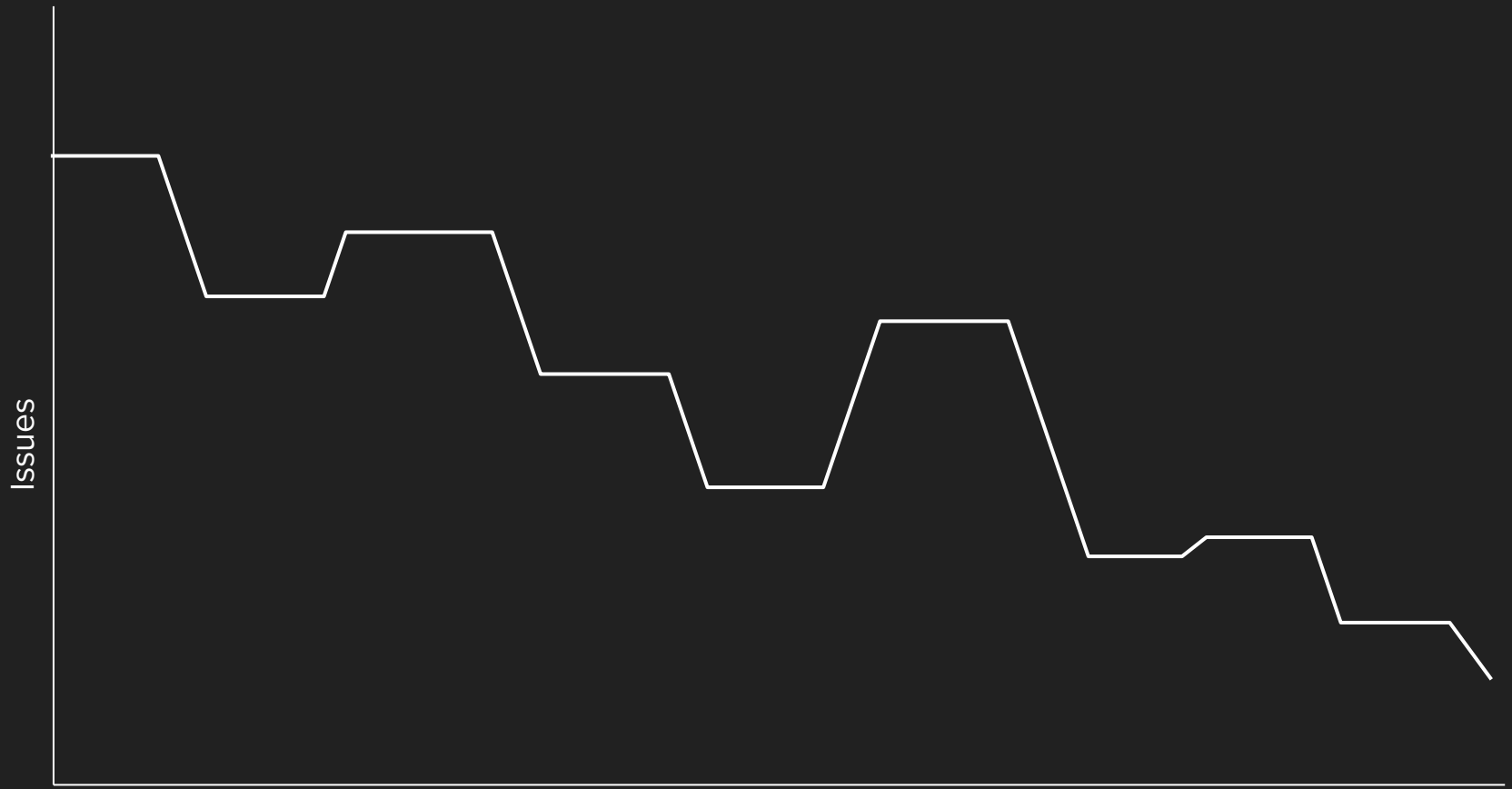
Using Static Analysis results

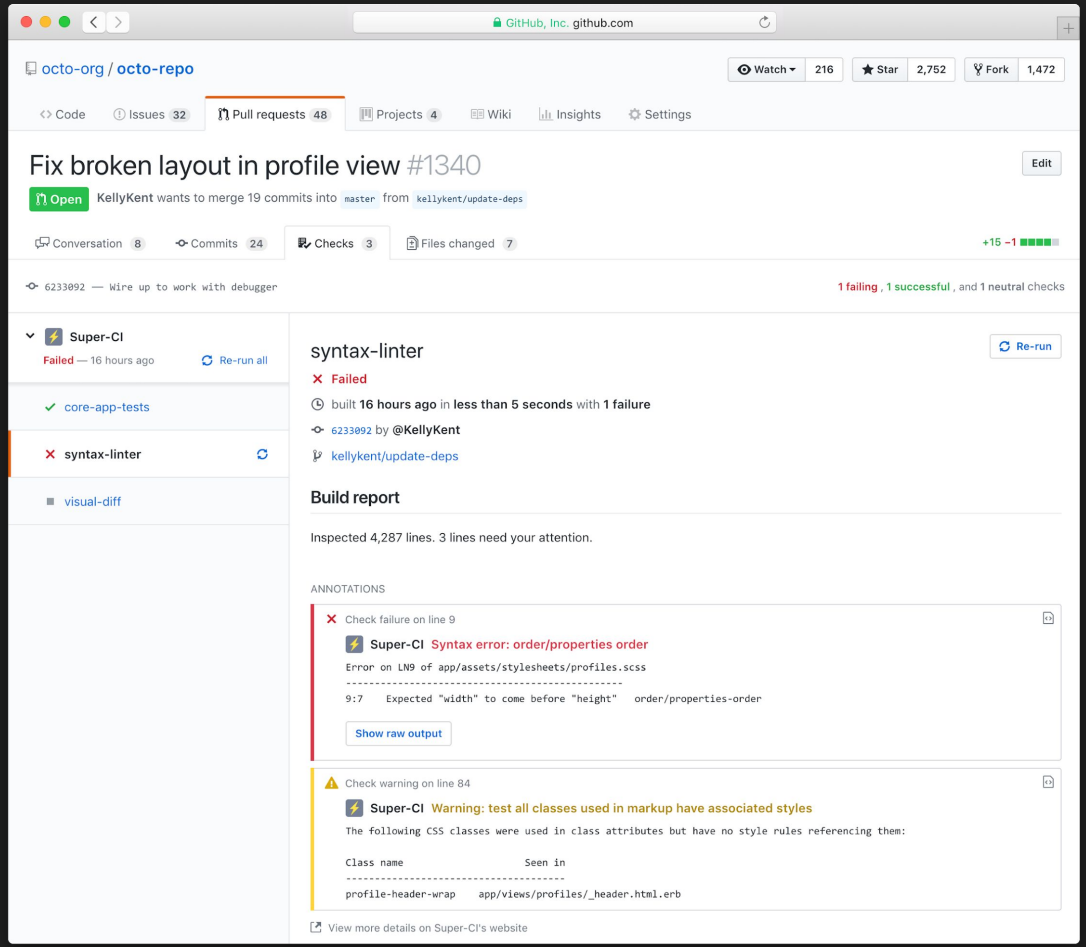
Run Static Analysis on every (CI) build

```
<issues format="5" by="lint 3.3.0-alpha12">
    <issue
        id="LambdaLast"
        severity="Error"
        message="Functional interface parameters should be last to improve Kotlin interoperability"
        category="Interoperability:Kotlin Interoperability"
        priority="6"
        summary="Lambda Parameters Last"
        explanation="To improve calling this code from Kotlin parameter types eligible for SAM conversion
should be last."
        url="https://android.github.io/kotlin-guides/interop.html#lambda-parameters-last"
        errorLine1="    public int doSomething(Action action, int count) {"
        errorLine2="                                          ~~~~~~~~~">
        <location
            file="/.../src/main/java/com/example/Test.java"
            line="4"
            column="43"/>
    </issue>
</issues>
```

```
<issues format="5" by="lint 3.3.0-alpha12">
    <issue
        id="LambdaLast"
        severity="Error"
        message="Functional interface parameters should be last to improve Kotlin interoperability"
        category="Interoperability:Kotlin Interoperability"
        priority="6"
        summary="Lambda Parameters Last"
        explanation="To improve calling this code from Kotlin parameter types eligible for SAM conversion should be last."
        url="https://android.github.io/kotlin-guides/interop.html#lambda-parameters-last"
        errorLine1="    public int doSomething(Action action, int count) {"
        errorLine2="                                           ~~~~~~~~~">
        <location
            file="/.../src/main/java/com/example/Test.java"
            line="4"
            column="43"/>
    </issue>
</issues>
```

```
<issues format="5" by="lint 3.3.0-alpha12">
    <issue
        id="LambdaLast"
        severity="Error"
        message="Functional interface parameters should be last to improve Kotlin interoperability"
        category="Interoperability:Kotlin Interoperability"
        priority="6"
        summary="Lambda Parameters Last"
        explanation="To improve calling this code from Kotlin parameter types eligible for SAM conversion
should be last."
        url="https://android.github.io/kotlin-guides/interop.html#lambda-parameters-last"
        errorLine1="    public int doSomething(Action action, int count) {"
        errorLine2="                                          ~~~~~~~~~~">
        <location
            file="/.../src/main/java/com/example/Test.java"
            line="4"
            column="43"/>
    </issue>
</issues>
```


Issues


GitHub, Inc. github.com
octo-org / octo-repo
Watch 216
Star 2,752
Fork 1,472
Code
Issues 32
Pull requests 48
Projects 4
Wiki
Insights
Settings
Fix broken layout in profile view #1340
Edit
Open
KellyKent wants to merge 19 commits into master from kellykent/update-deps
Conversation 8
Commits 24
Checks 3
Files changed 7
+15 −1
6233092 — Wire up to work with debugger
1 failing , 1 successful , and 1 neutral checks
Super-CI
Failed — 16 hours ago
Re-run all
core-app-tests
syntax-linter
visual-diff
syntax-linter
Re-run
Failed
built 16 hours ago in less than 5 seconds with 1 failure
6233092 by @KellyKent
kellykent/update-deps
Build report
Inspected 4,287 lines. 3 lines need your attention.
ANNOTATIONS
Check failure on line 9
Super-CI Syntax error: order/properties order
Error on LN9 of app/assets/stylesheets/profiles.scss
9:7 Expected "width" to come before "height" order/properties-order
Show raw output
Check warning on line 84
Super-CI Warning: test all classes used in markup have associated styles
The following CSS classes were used in class attributes but have no style rules referencing them:
Class name Seen in
profile-header-wrap app/views/profiles/_header.html.erb
View more details on Super-CI's website

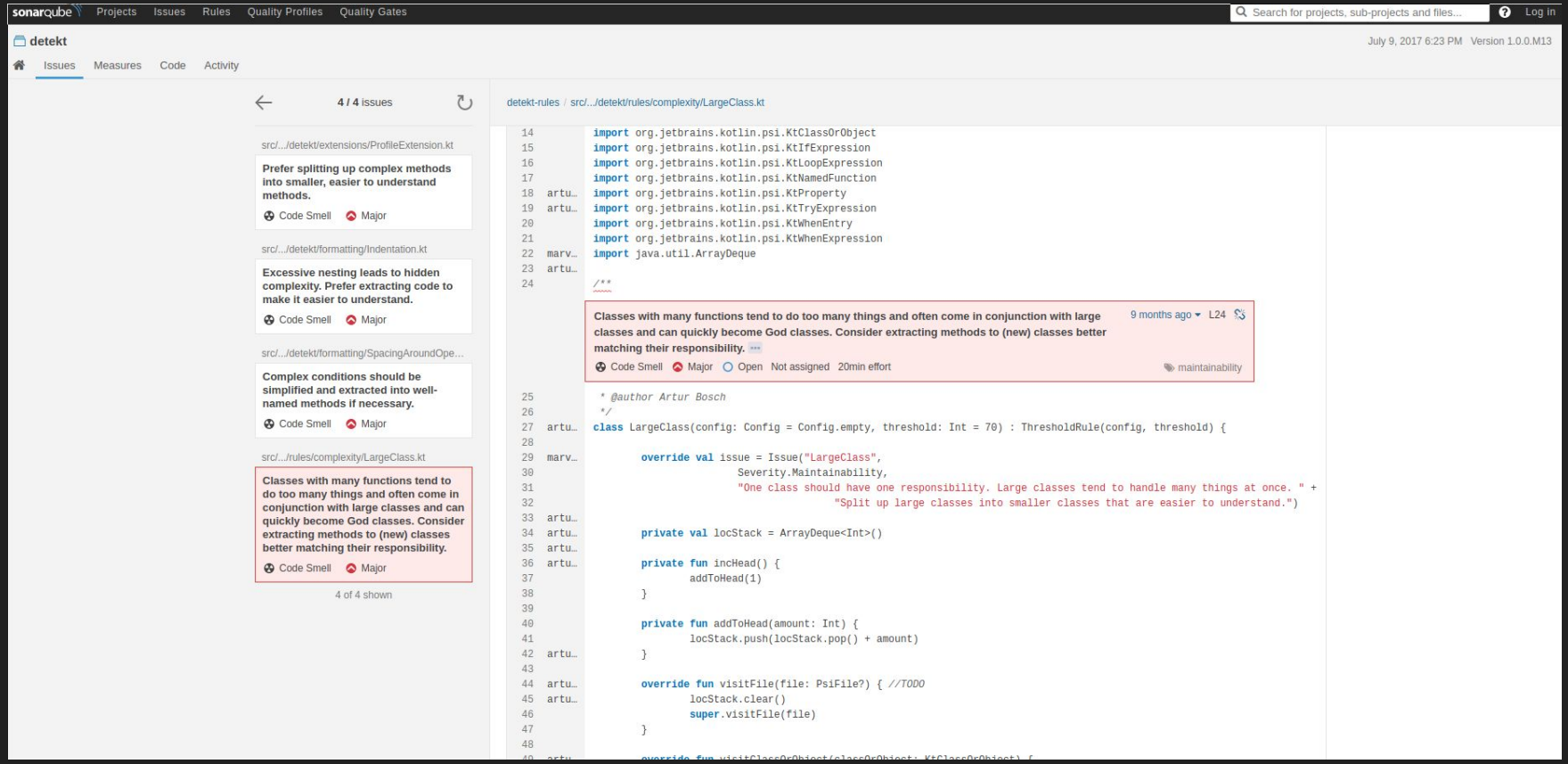
path/to/some/SampleFile.kt

```
58 +
59 +     println("Welcome to KotlinConf ${year!!}")
```

ReviewBot 11 days ago

Reporter: Detekt
Rule: UnsafeCallOnNullableType
Severity: Warning

If $year is null at runtime this expression will throw a NullPointerException.


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Search for projects, sub-projects and files...
Log in
detekt
July 9, 2017 6:23 PM Version 1.0.0.M13
Issues
Measures
Code
Activity
4 / 4 issues
src/.../detekt/extensions/ProfileExtension.kt
Prefer splitting up complex methods into smaller, easier to understand methods.
Code Smell
Major
src/.../detekt/formatting/Indentation.kt
Excessive nesting leads to hidden complexity. Prefer extracting code to make it easier to understand.
Code Smell
Major
src/.../detekt/formatting/SpacingAroundOpe...
Complex conditions should be simplified and extracted into well-named methods if necessary.
Code Smell
Major
src/.../rules/complexity/LargeClass.kt
Classes with many functions tend to do too many things and often come in conjunction with large classes and can quickly become God classes. Consider extracting methods to (new) classes better matching their responsibility.
Code Smell
Major
4 of 4 shown
detekt-rules / src/.../detekt/rules/complexity/LargeClass.kt
Classes with many functions tend to do too many things and often come in conjunction with large classes and can quickly become God classes. Consider extracting methods to (new) classes better matching their responsibility.
9 months ago
L24
Code Smell
Major
Open
Not assigned
20min effort
maintainability
import org.jetbrains.kotlin.psi.KtClassOrObject
import org.jetbrains.kotlin.psi.KtIfExpression
import org.jetbrains.kotlin.psi.KtLoopExpression
import org.jetbrains.kotlin.psi.KtNamedFunction
import org.jetbrains.kotlin.psi.KtProperty
import org.jetbrains.kotlin.psi.KtTryExpression
import org.jetbrains.kotlin.psi.KtWhenEntry
import org.jetbrains.kotlin.psi.KtWhenExpression
import java.util.ArrayDeque
/**
* @author Artur Bosch
*/
class LargeClass(config: Config = Config.empty, threshold: Int = 70) : ThresholdRule(config, threshold) {
override val issue = Issue("LargeClass",
Severity.Maintainability,
"One class should have one responsibility. Large classes tend to handle many things at once. " +
"Split up large classes into smaller classes that are easier to understand.")
private val locStack = ArrayDeque<Int>()
private fun incHead() {
addToHead(1)
}
private fun addToHead(amount: Int) {
locStack.push(locStack.pop() + amount)
}
override fun visitFile(file: PsiFile?) { //TODO
locStack.clear()
super.visitFile(file)
}

Warning vs. Error

Warning vs. Error

Baseline

Adapt tools to your best-practices

Enable/Disable Rules

Enable/Disable Rules

Configure rules

Enable/Disable Rules

Configure rules

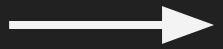
Baseline

```
@Suppress("RuleToSuppress")
```

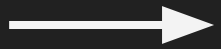
```
@SuppressLint("KotlinPropertyAccess")
```

```
// Ignore because of reason
@SuppressLint("KotlinPropertyAccess")
```

Inner workings of a Static Analysis tool



Tool



AST

UAST

```kotlin
class KotlinConf {
    const val YEAR = 2018

    fun welcome(year: Int) {
        println("Welcome to KotlinConf $year!")
    }
}
```

UFile (package =)

```
class KotlinConf {
    const val YEAR = 2018

    fun welcome(year: Int) {
        println("Welcome to KotlinConf $year!")
    }
}
```

```
UFile (package = )
  UClass (name = KotlinConf)
```

```
class KotlinConf {
    const val YEAR = 2018

    fun welcome(year: Int) {
        println("Welcome to KotlinConf $year!")
    }
}
```

```
class KotlinConf {
    const val YEAR = 2018

    fun welcome(year: Int) {
        println("Welcome to KotlinConf $year!")
    }
}
```

```
UFile (package = )
  UClass (name = KotlinConf)
    UField (name = YEAR)
      UAnnotation
      ULiteralExpression (value = 2018)
```

```
class KotlinConf {
    const val YEAR = 2018

    fun welcome(year: Int) {
        println("Welcome to KotlinConf $year!")
    }
}
```

```
UFile (package = )
  UClass (name = KotlinConf)
    UField (name = YEAR)
      UAnnotation
      ULiteralExpression (value = 2018)
    UAnnotationMethod (name = welcome)
      UParameter (name = year)
        UAnnotation
```

```
class KotlinConf {
    const val YEAR = 2018

    fun welcome(year: Int) {
        println("Welcome to KotlinConf $year!")
    }
}
```

```
UFile (package = )
  UClass (name = KotlinConf)
    UField (name = YEAR)
      UAnnotation
      ULiteralExpression (value = 2018)
    UAnnotationMethod (name = welcome)
      UParameter (name = year)
        UAnnotation
      UBlockExpression
        UCallExpression
          UIdentifier (Identifier (println))
          USimpleNameReferenceExpression
          ULiteralExpression
```

Lint → IssueRegistry

IssueRegistry ⟶ `List<Issue>`

```
public final class Issue ... {
    private final String mId;
    private final String mBriefDescription;
    private final String mExplanation;
    private final Category mCategory;
    private final int mPriority;
    private final Severity mSeverity;
    private Object mMoreInfoUrls;
    private boolean mEnabledByDefault = true;
    private Implementation mImplementation;

    ...

}
```

```
public final class Issue ... {
    private final String mId;
    private final String mBriefDescription;
    private final String mExplanation;
    private final Category mCategory;
    private final int mPriority;
    private final Severity mSeverity;
    private Object mMoreInfoUrls;
    private boolean mEnabledByDefault = true;
    private Implementation mImplementation;

    ...

}
```

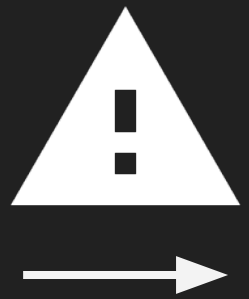
```
<issues format="5" by="lint 3.3.0-alpha12">
    <issue
        id="LambdaLast"
        summary="Lambda Parameters Last"
        message="..."
        explanation="..."
        category="Kotlin Interoperability"
        priority="6"
        severity="Error"
        url="..."
        errorLine1="..."
        <location
            file="Test.java"
            line="4"
            column="43"/>
    </issue>
</issues>
```


AST
Detector

AST

Detector

AST



Detector

```
import android.util.Log
import com.something.Else

class Something {
```

```
import android.util.Log
import com.something.Else

class Something {
```

```
import android.util.Log
import com.something.Else

class Something {
```

```
0: UFile
  1: UImportStatement
  2: UImportStatement

  4: UClass
```


import android.util.Log
import com.something.Else
class Something {

0: UFile
1: UImportStatement
2: UImportStatement
4: UClass


import android.util.Log
import com.something.Else
class Something {

0: UFile
1: UImportStatement
2: UImportStatement
4: UClass

```
class ImportVisitor(private val context: JavaContext) : UElementHandler() {
  override fun visitImportStatement(import: UImportStatement) {
    val resolved = import.resolve()
    if (resolved !is PsiClass) {
        return
    }

    if (resolved.qualifiedName == "android.util.Log") {
        context.report(ISSUE, import, context.getLocation(import), "...")
    }
  }
}
```

```
class ImportVisitor(private val context: JavaContext) : UElementHandler() {
  override fun visitImportStatement(import: UImportStatement) {
    val resolved = import.resolve()
    if (resolved !is PsiClass) {
        return
    }

    if (resolved.qualifiedName == "android.util.Log") {
        context.report(ISSUE, import, context.getLocation(import), "...")
    }
  }
}
```

```
class ImportVisitor(private val context: JavaContext) : UElementHandler() {
  override fun visitImportStatement(import: UImportStatement) {
    val resolved = import.resolve()
    if (resolved !is PsiClass) {
        return
    }

    if (resolved.qualifiedName == "android.util.Log") {
        context.report(ISSUE, import, context.getLocation(import), "...")
    }
  }
}
```

```
class ImportVisitor(private val context: JavaContext) : UElementHandler() {
  override fun visitImportStatement(import: UImportStatement) {
    val resolved = import.resolve()
    if (resolved !is PsiClass) {
        return
    }

    if (resolved.qualifiedName == "android.util.Log") {
        context.report(ISSUE, import, context.getLocation(import), "...")
    }
  }
}
```

```
class ImportVisitor(private val context: JavaContext) : UElementHandler() {
  override fun visitImportStatement(import: UImportStatement) {
    val resolved = import.resolve()
    if (resolved !is PsiClass) {
        return
    }

    if (resolved.qualifiedName == "android.util.Log") {
        context.report(ISSUE, import, context.getLocation(import), "...")
    }
  }
}
```

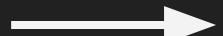
android.util.Log

```
android.util.Log

"android.util.Log"
```

```
class AndroidLogDetector : Detector(), Detector.UastScanner {

  override fun getApplicableUastTypes(): List<Class<out UElement>>? {
    return listOf(UImportStatement::class.java)
  }
}
```

```
class AndroidLogDetector : Detector(), Detector.UastScanner {

  override fun getApplicableUastTypes(): List<Class<out UElement>>? {
    return listOf(UImportStatement::class.java)
  }
}
```

```
class AndroidLogDetector : Detector(), Detector.UastScanner {

  override fun getApplicableUastTypes(): List<Class<out UElement>>? {
    return listOf(UImportStatement::class.java)
  }
}
```

Findings  Reports

Reports

HTML

XML

Extensibility

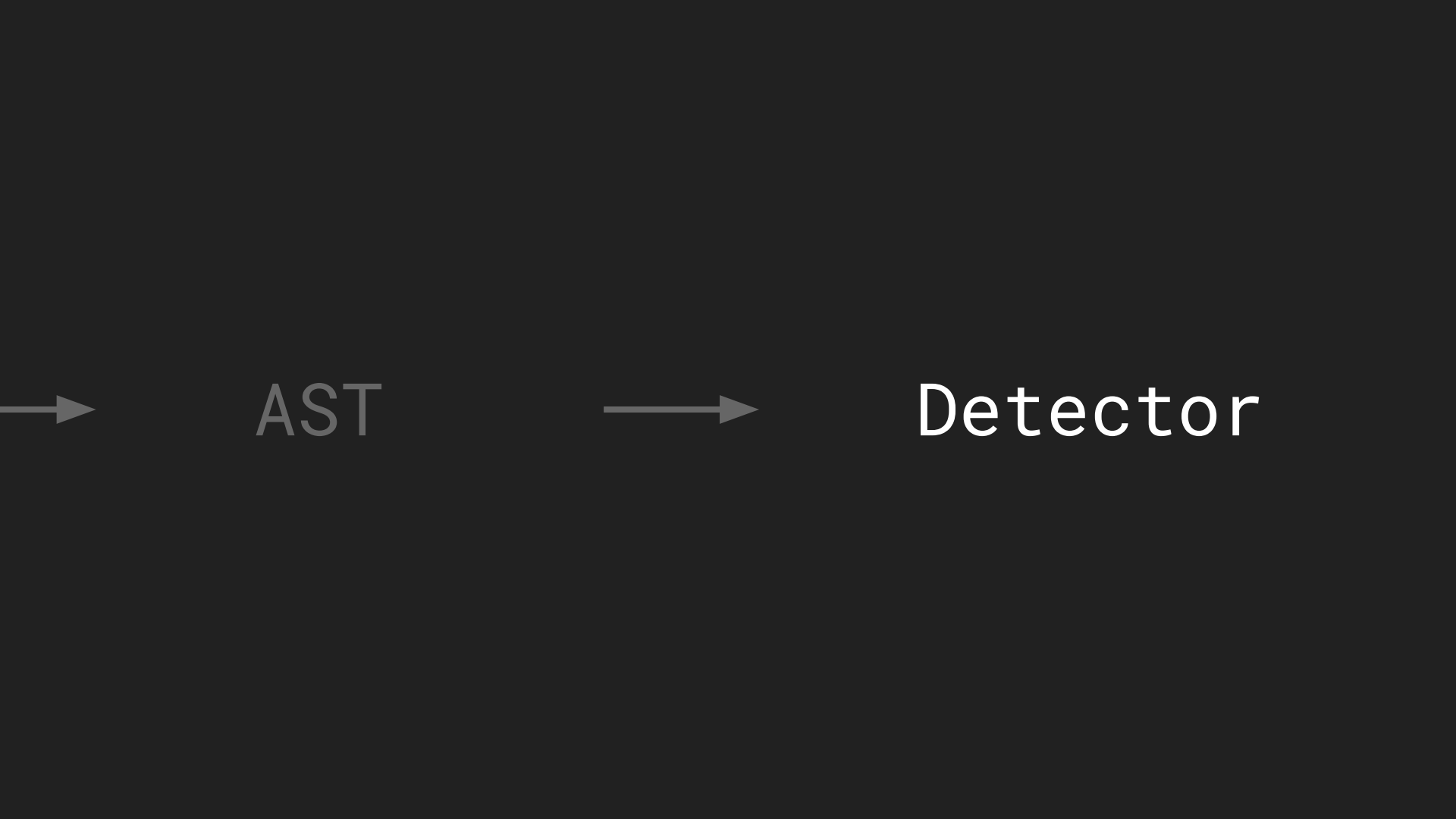
```
59  59
    60  +   import android.util.Log
```

Mauin 3 minutes ago

Please use our custom Log class instead.

AST

Detector


AST
Detector

```
package com.example
class CustomIssueRegistry : IssueRegistry() {
  override fun getIssues() = listOf(ISSUE_ANDROID_LOG)
}
```

```
jar {
  manifest {
    attributes("Lint-Registry-v2": "com.example.CustomIssueRegistry")
  }
}
```

Write your custom Issue & Detector

```
val kotlinFile = kt("""
            |import android.util.Log
            |import java.util.List
            |
            |class KotlinConf {
            |}
    """.trimMargin())

@Test fun testKotlinFile() {
    lint()
      .files(kotlinFile)
      .issues(ISSUE_ANDROID_LOG)
      .run()
      .expectCount(1)
}
```

```
val javaFile = java("""
            |import android.util.Log;
            |import java.util.List;
            |
            |public class KotlinConf {
            |}
    """.trimMargin())

@Test fun testJavaFile() {
    lint()
      .files(javaFile)
      .issues(ISSUE_ANDROID_LOG)
      .run()
      .expectCount(1)
}
```

```
dependencies {
  lintChecks project(":lint")
}
```

Create IssueRegistry

Create IssueRegistry

Manifest file

Create IssueRegistry

Manifest file

Implement Issue & Detector

Create IssueRegistry

Manifest file

Implement Issue & Detector

Test your Rule

Create IssueRegistry

Manifest file

Implement Issue & Detector

Test your Rule

Tell Lint about your custom Lint JAR

Libraries publishing Lint rules

Libraries publishing Lint rules

IntelliJ support

Libraries publishing Lint rules

IntelliJ support

Quick fixes

groups.google.com/forum/#!forum/lint-dev

Make your Kotlin code safer

Make your Kotlin code safer

Coherent codebase

Make your Kotlin code safer

Coherent codebase

Encode your own best-practices

Thank you!

twitter.com/**@Mauin**

github.com/**Mauin**