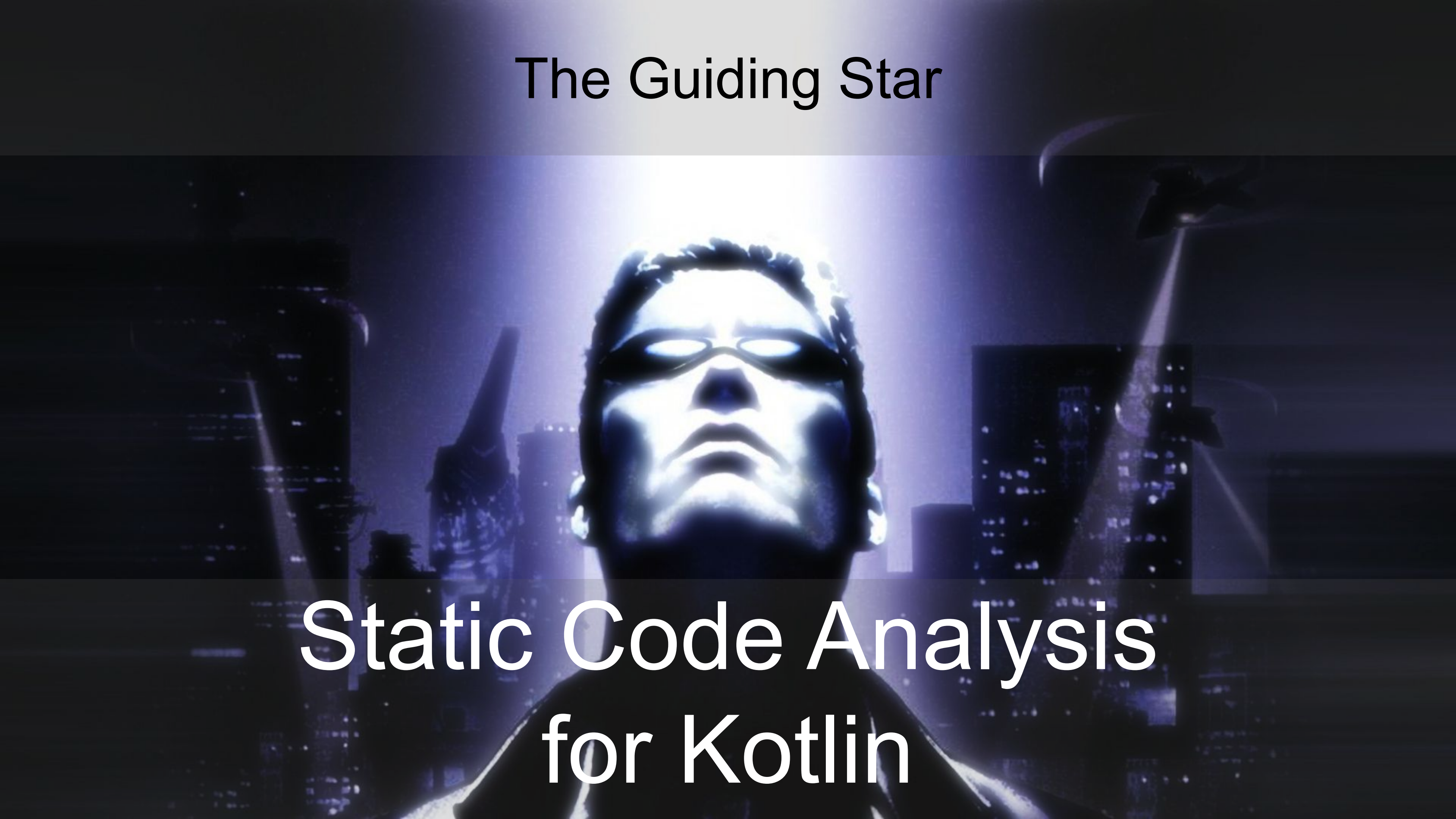


The Guiding Star

Static Code Analysis for Kotlin

- Flo Health UAB
- 11+ years of experience
- Not Guru, but in Love with Linters

Agenda

- Briefly review the most popular status code analysis tools for Kotlin
- Look into their internals to understand better how they process the code
- Find out the commonalities in their workflow and customization process
- Implement custom rule for the reviewed tools

Heroes of the day

ktlint

- Analyze Kotlin code
- Offer number of built-in rule sets
- Support multiple reporting formats
- Support custom rules and reporters
- Allow suppressing issues
- Support baseline files
- Offer CLI

ktlint

ktlint

- Checks for code style
- Focuses on simplicity

- Checks for many types of code smell
- Highly configurable

Which one to choose?

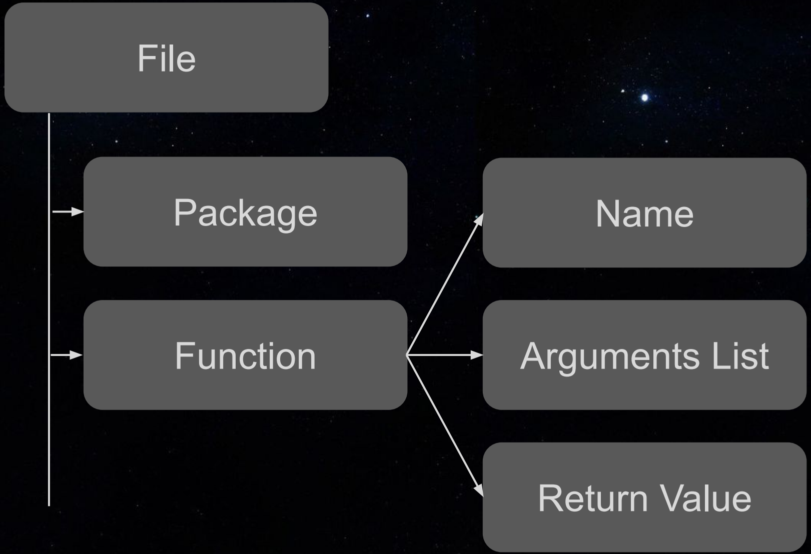
WE NEED TO GO
DEEPER

What is static code analysis?

Static code analysis (SCA) is the analysis of programs done without executing them.

In a nutshell, before or while a project is being built, a tool is **“reading”** through its code, and recognizes certain bad patterns.

How do they “read” code?



File

Package

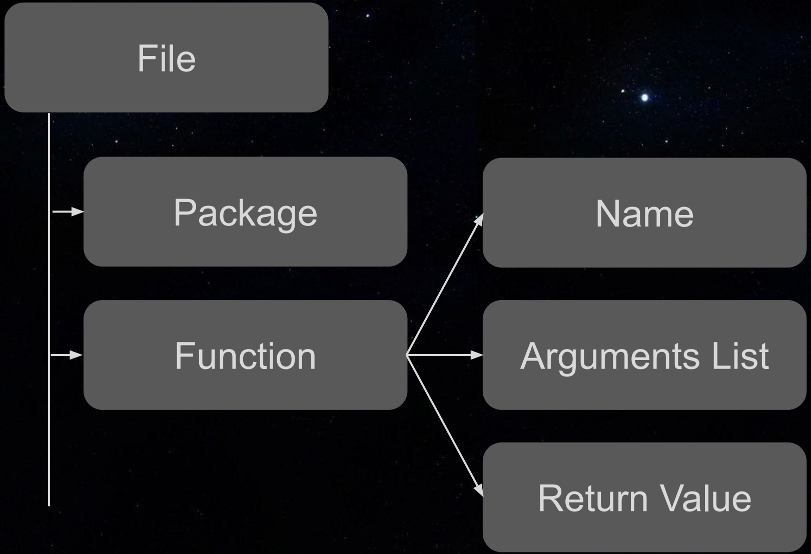
Function

Name

Arguments List

Return Value

```
fun main() {
    println("Hello world")
}
```



File

Package

Function

Name

Arguments List

Return Value

- The same file can be represented as different types of AST
- Different AST implementations provide different capabilities

- AST parsing relies on the official Kotlin grammar;
- AST is not built by the analyzers themselves
    - Kotlin Compiler
    - kotlinx.ast

Lets grow few trees

Abstractness

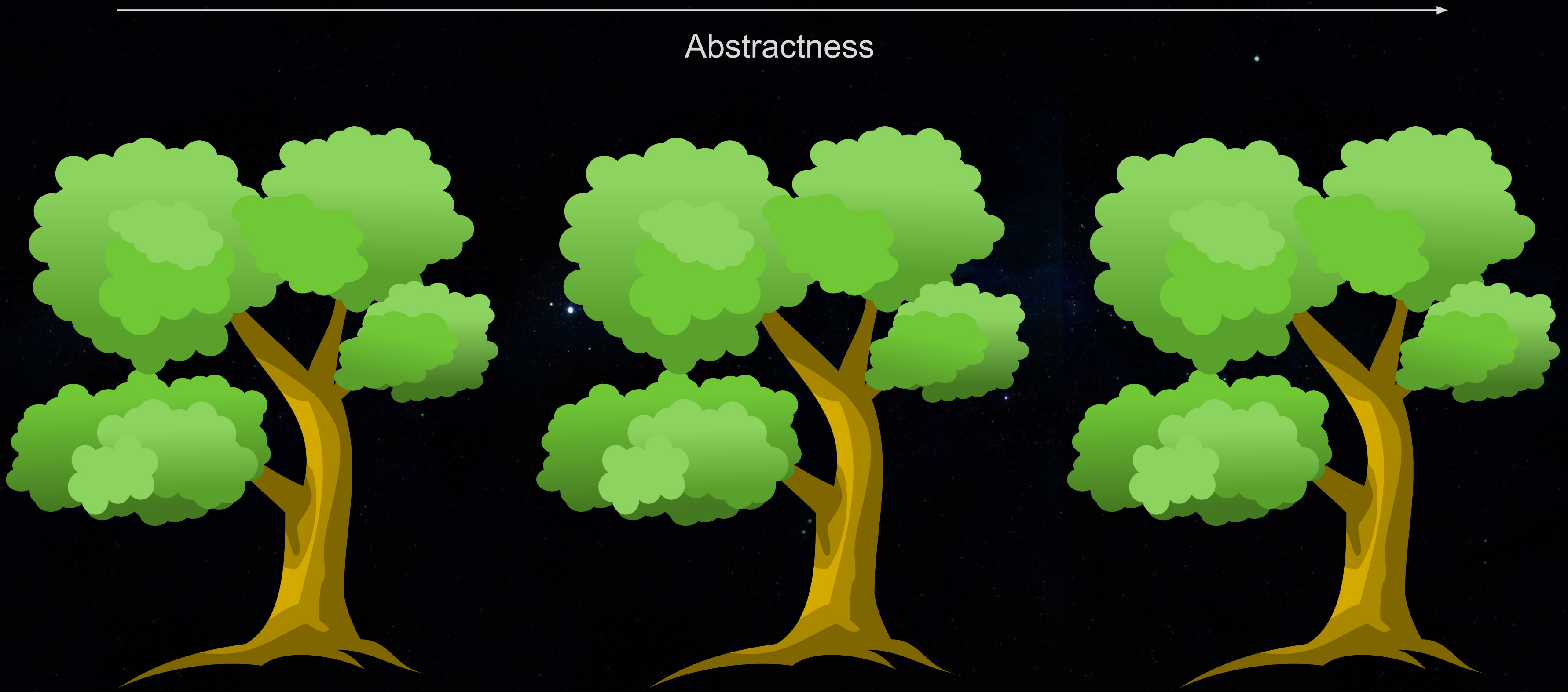
Abstractness

File AST

Abstractness

File AST

PSI
(Program Structure Interface)



Abstractness

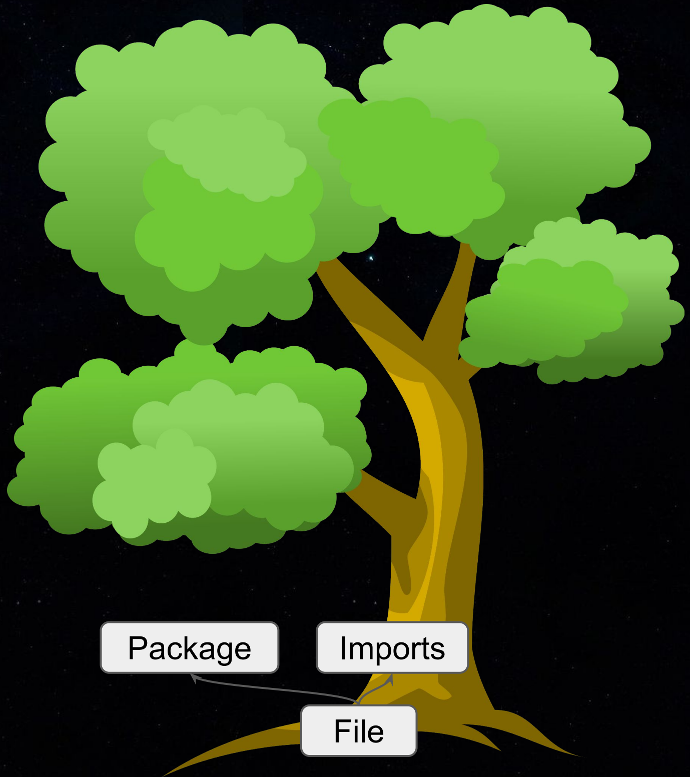
File AST

PSI
(Program Structure Interface)

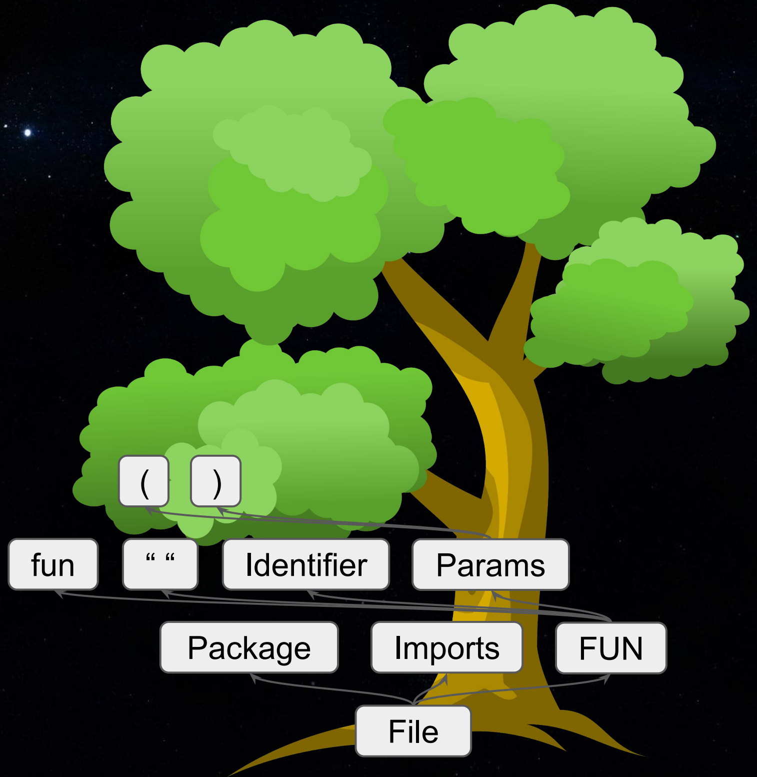
UAST
(Universal Abstract Syntax Tree)

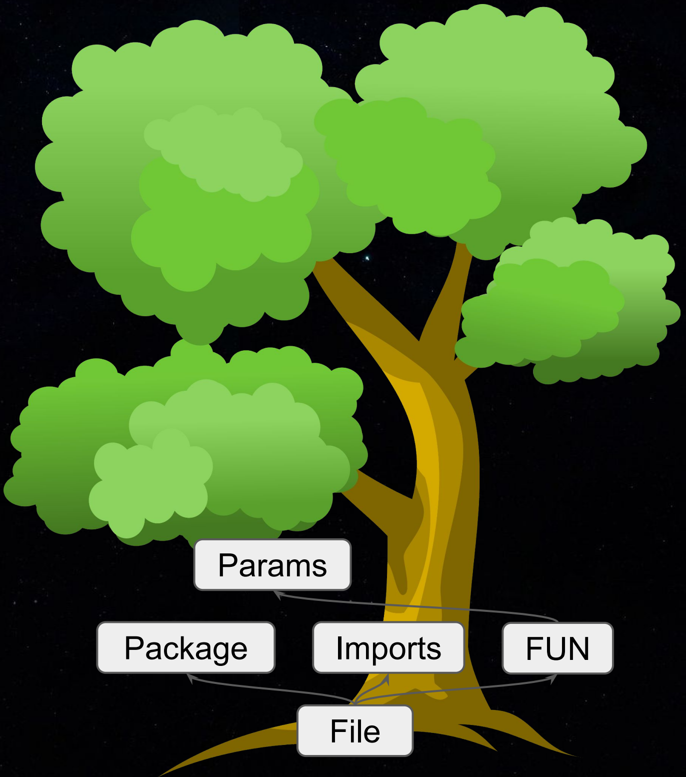
File
File

Package
Imports
File

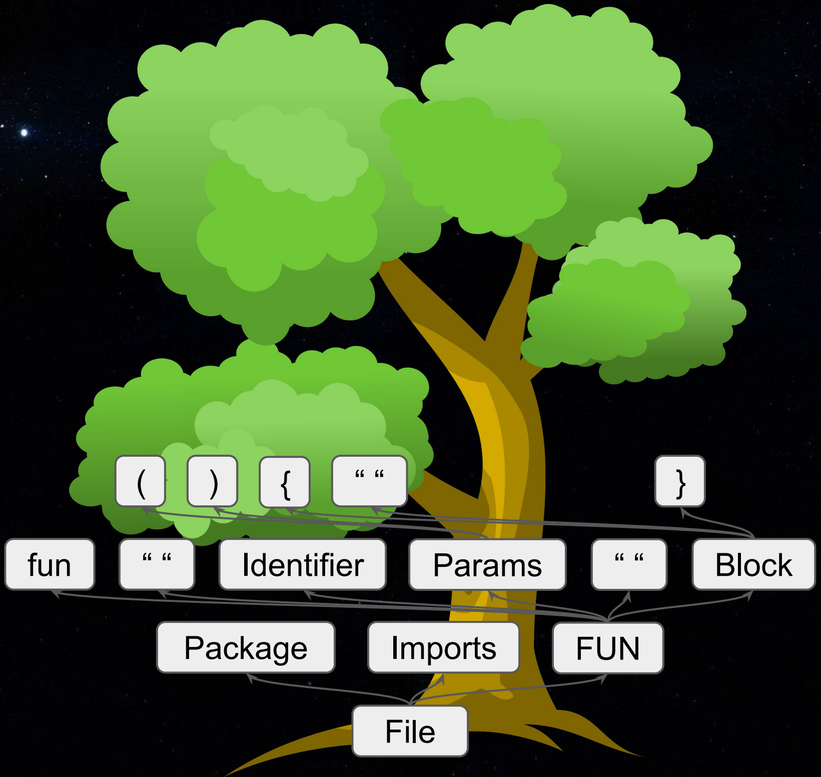

Package
Imports
File

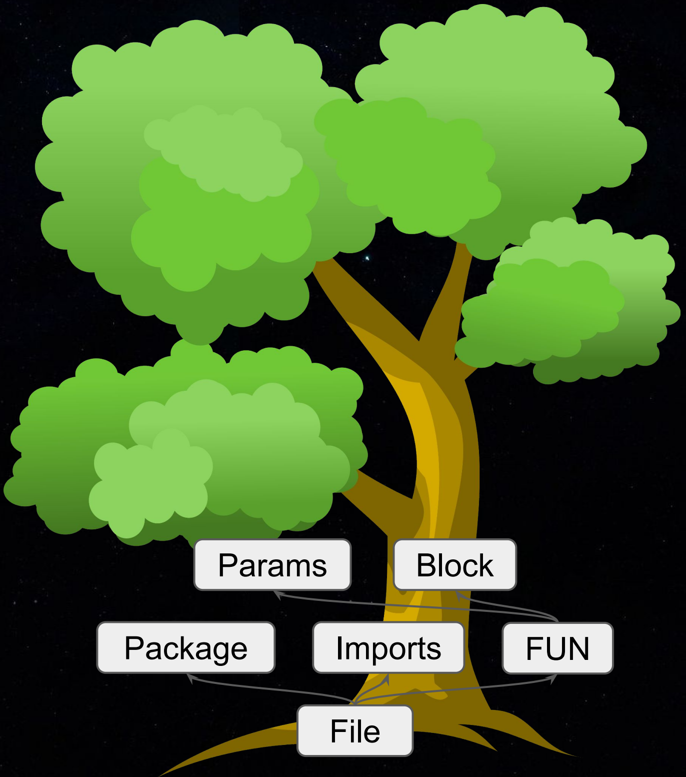
```
fun main()
```


(
)
fun
" "
Identifier
Params
Package
Imports
FUN
File

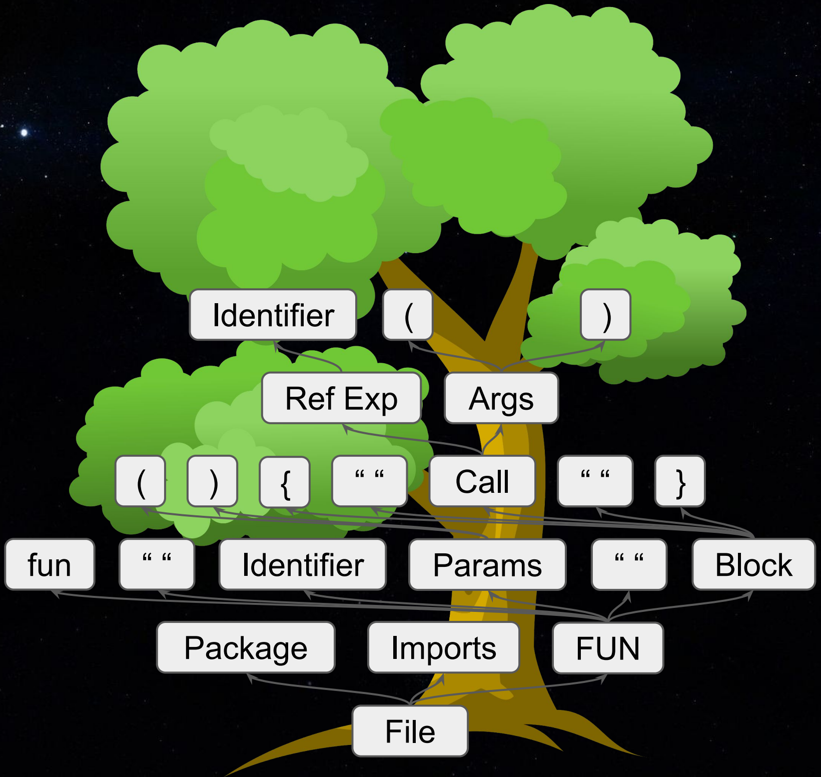

Params
Package
Imports
FUN
File

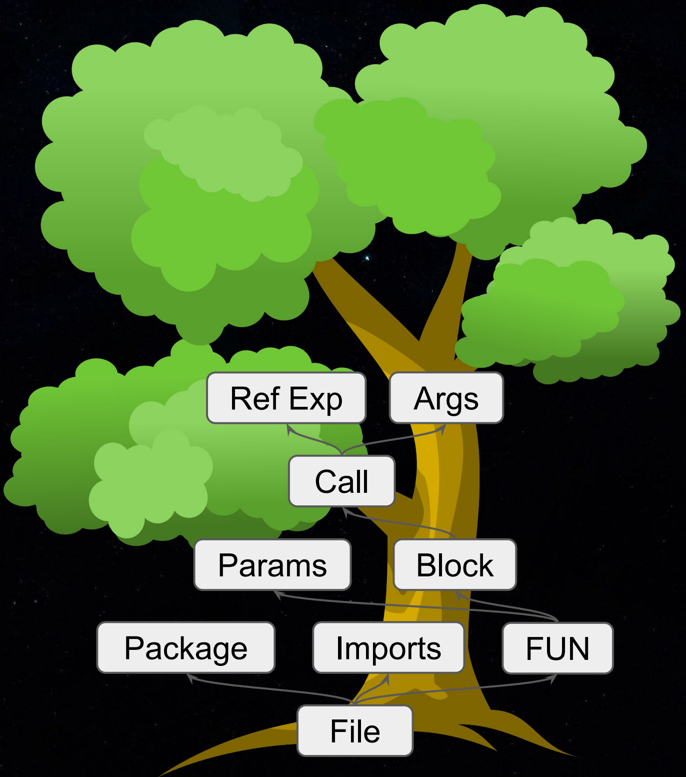
```
fun main() {
}
```


(
)
{
" "
}
fun
" "
Identifier
Params
" "
Block
Package
Imports
FUN
File

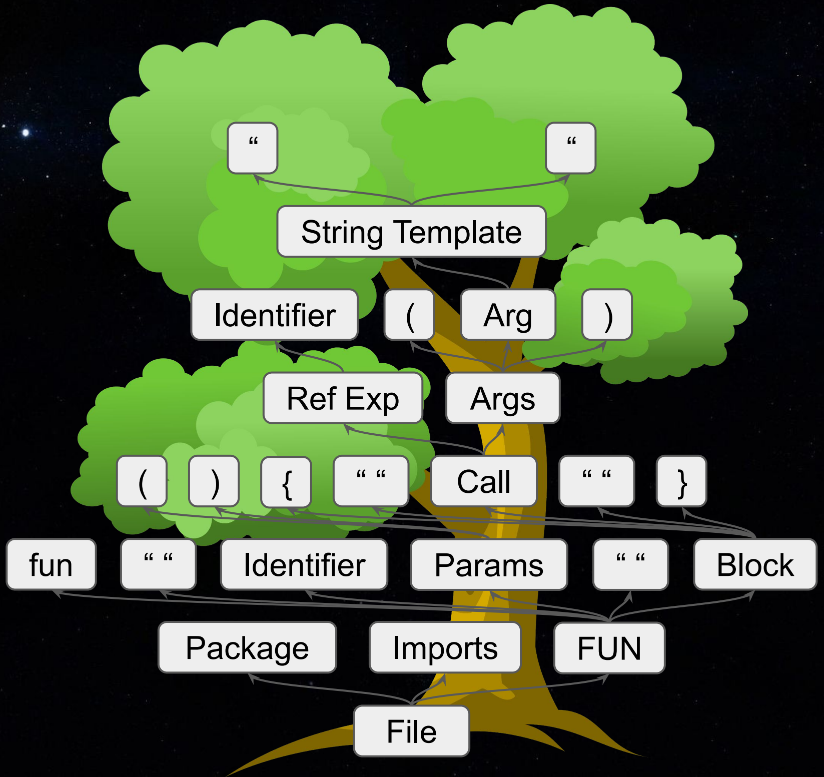

Params
Block
Package
Imports
FUN
File

```
fun main() {
    println()
}
```





```
fun main() {
    println("")
}
```



“ “

String Template

Identifier (Arg)

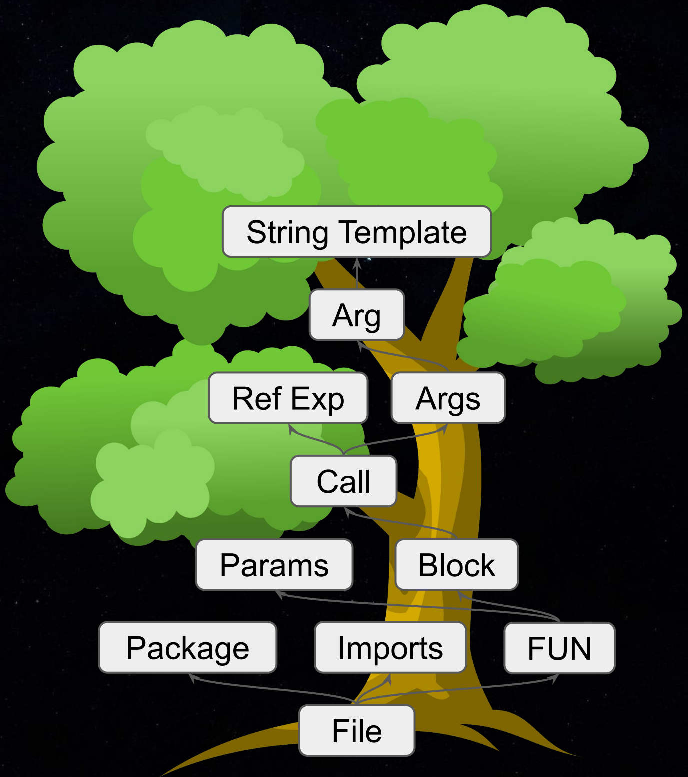
Ref Exp Args

( ) { “ “ Call “ “ }

fun “ “ Identifier Params “ “ Block

Package Imports FUN

File



String Template

Arg

Ref Exp Args

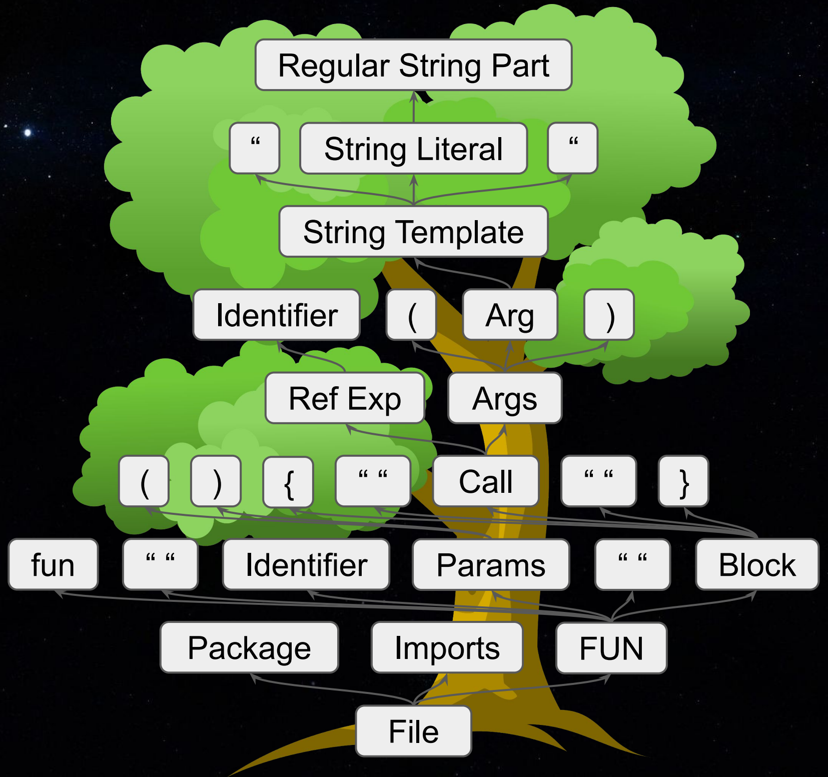
Call

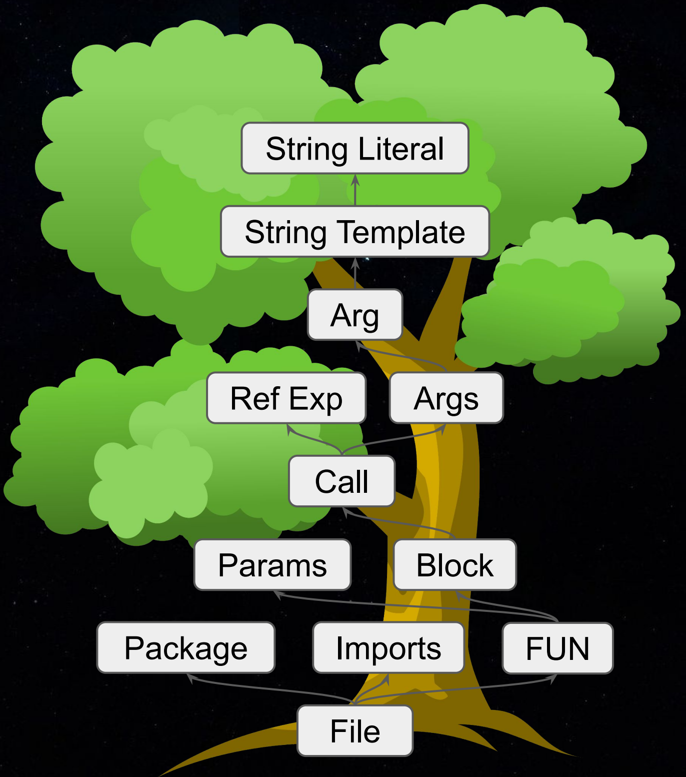
Params Block

Package Imports FUN

File

```
fun main() {
    println("Hello world")
}
```





PsiViewer Plugin

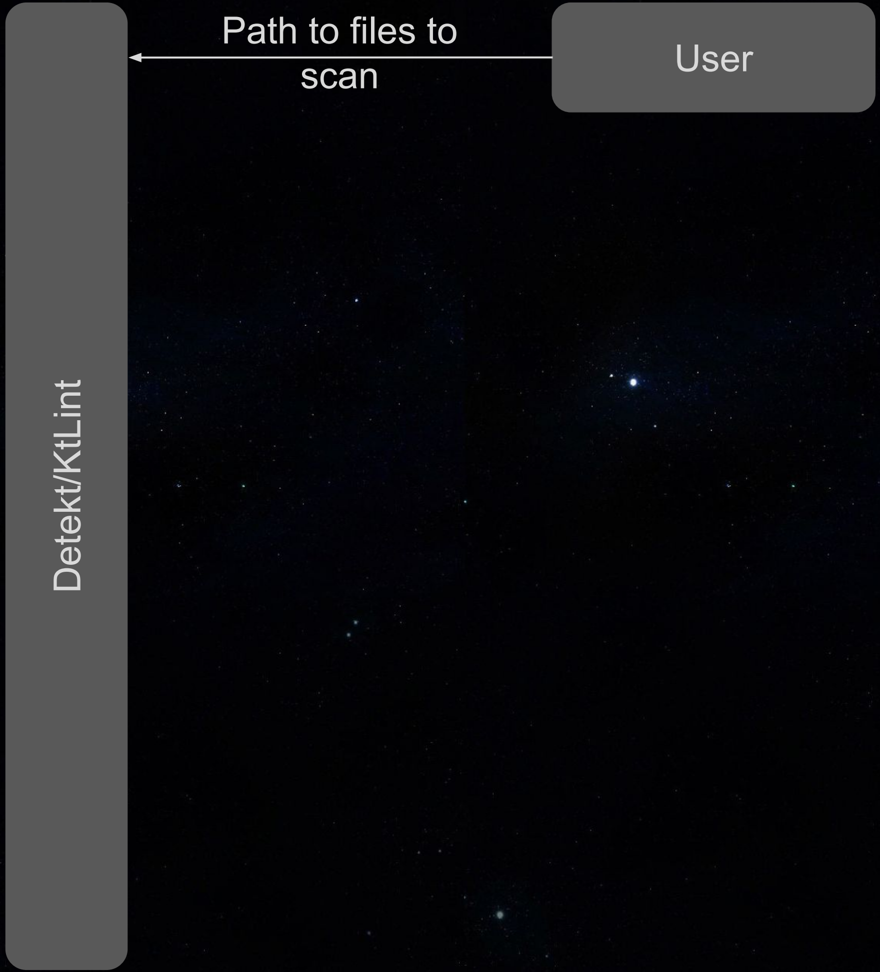
- Source Code
- Download

Psi Printer Project

- Source Code

Architecture

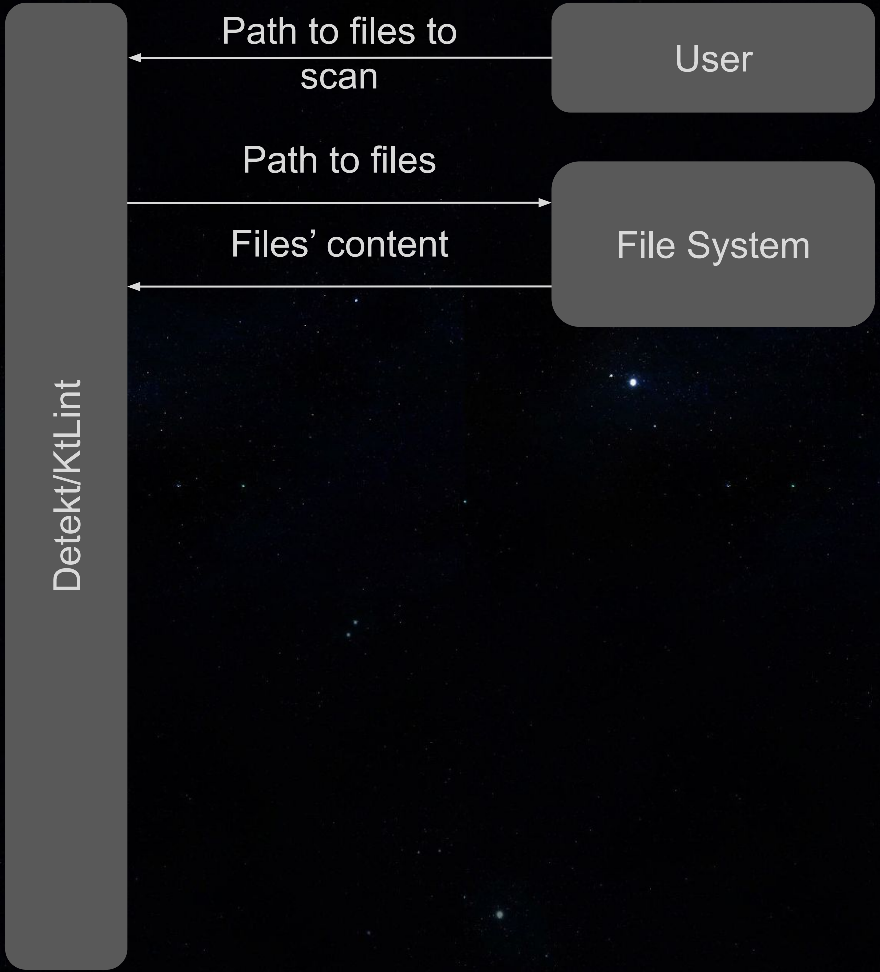
Detekt/KtLint



User

Path to files to scan

Detekt/KtLint



Detekt/KtLint

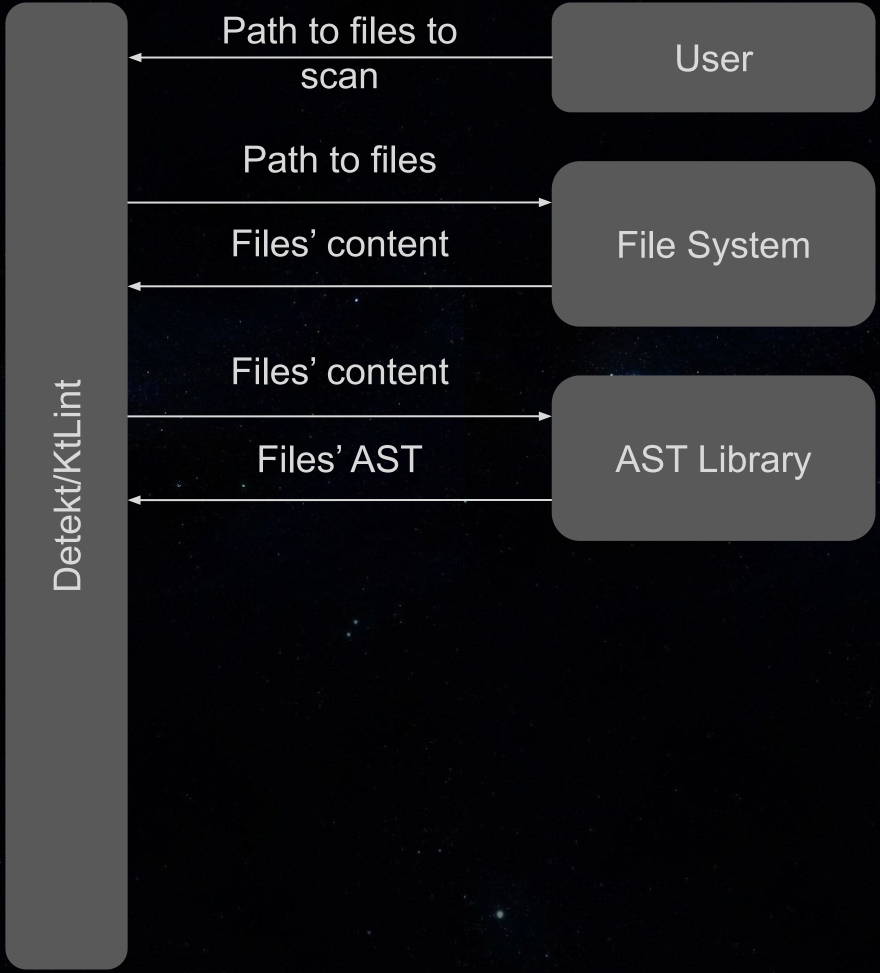
Path to files to scan

User

Path to files

Files’ content

File System



Detekt/KtLint

User

Path to files to scan

File System

Path to files

Files’ content

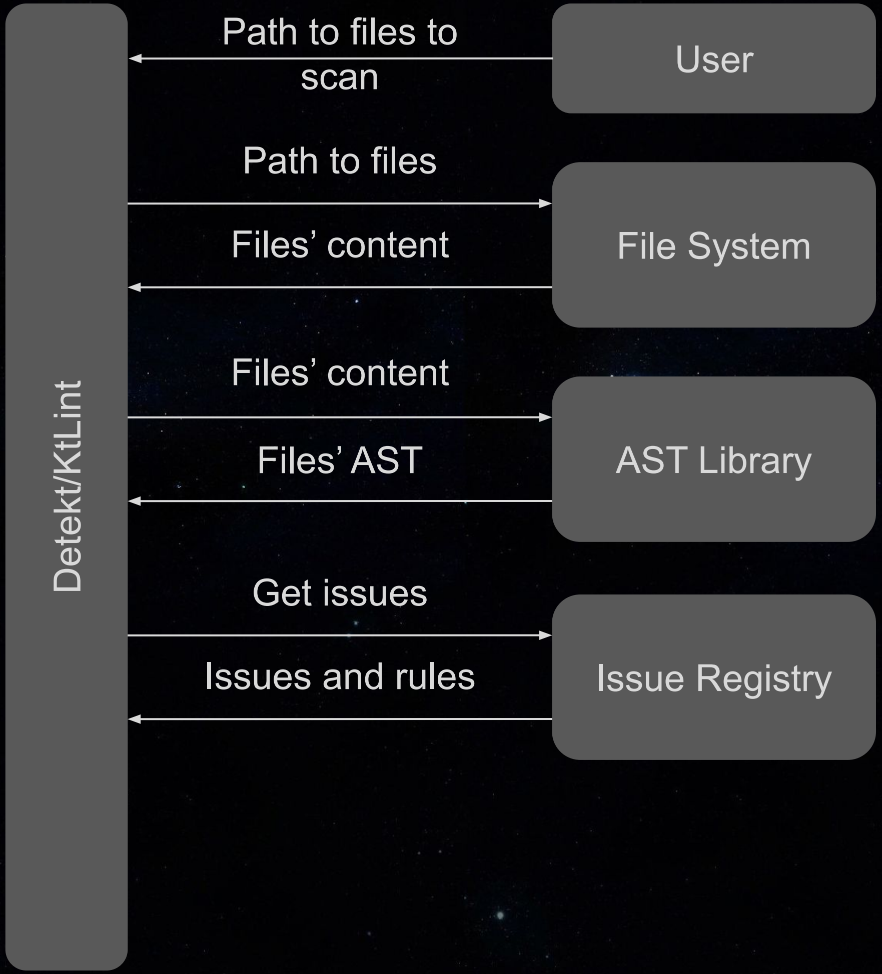
AST Library

Files’ content

Files’ AST

Detekt and KtLint use Kotlin Compiler to build ASTs of the scanned files.

KtLint uses File AST
Detekt uses Kotlin PSI



Detekt/KtLint

User

Path to files to scan

File System

Path to files

Files’ content

AST Library

Files’ content

Files’ AST

Issue Registry

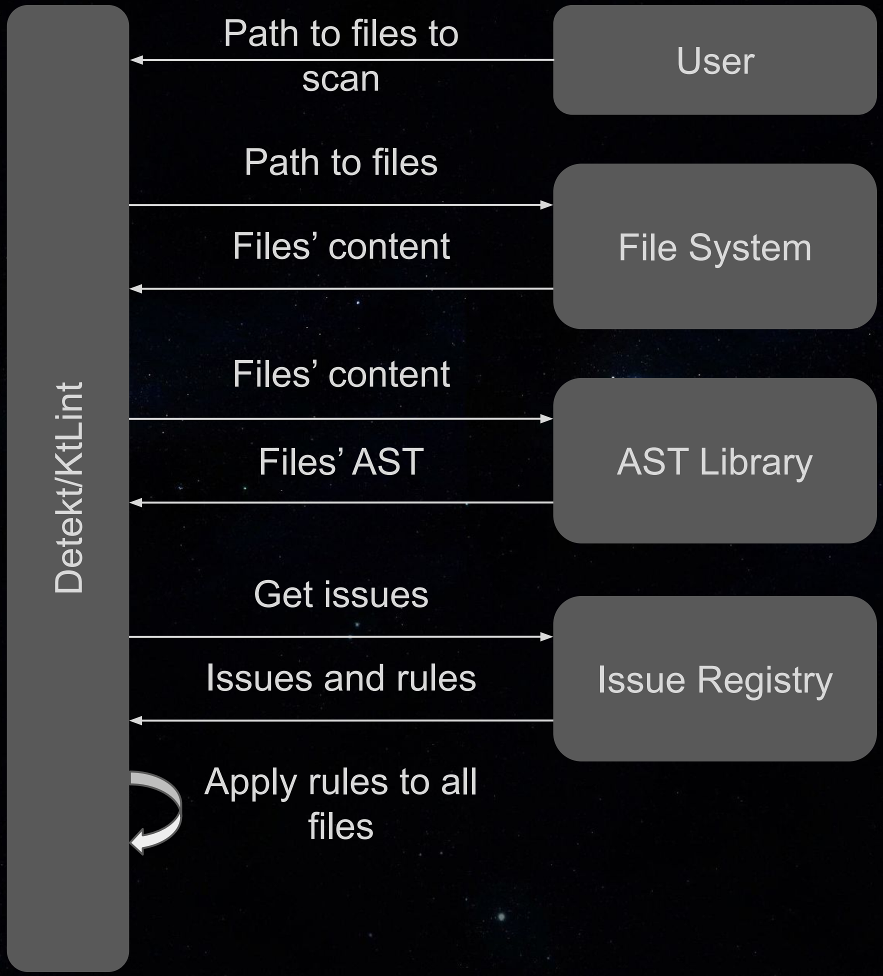
Get issues

Issues and rules

Issue is an entity, which holds metadata about a problem.

Rule (*Detector* or *Scanner*) encapsulates the logic of a single issue lookup.

Issue Registry provides collection of Rules’ to apply during scanning. Java tools rely on the ServiceLoader mechanism to get the registries.



Detekt/KtLint

Path to files to scan

User

Path to files

Files’ content

File System

Files’ content

Files’ AST

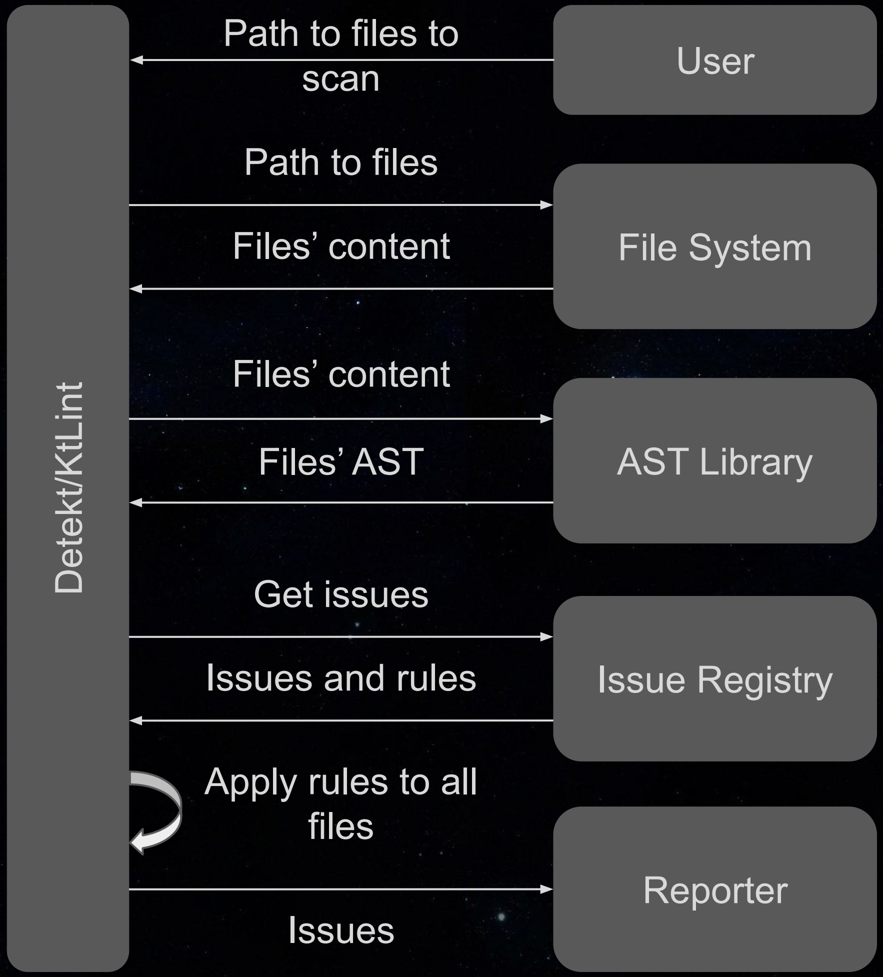
AST Library

Get issues

Issues and rules

Issue Registry

Apply rules to all files



Detekt/KtLint

Path to files to scan

User

Path to files

Files’ content

File System

Files’ content

Files’ AST

AST Library

Get issues

Issues and rules

Issue Registry

Apply rules to all files

Issues

Reporter

When *Rule* detects a problem within the scanned code, it reports an *Issue* via *Reporter*, which is responsible for presenting issues.

Issue Registry

Rule

Issue

Final word

- Static code analysis tools do not produce ASTs by themselves
- Modern Kotlin static code analysis tools have very similar architecture
- Thanks to that when you know how to customize one - you know how to customize any