

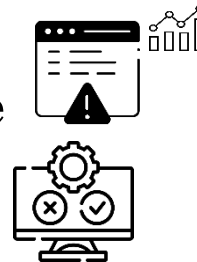


# **LABORATORIO 2 - ANALISI STATICA DEL SOFTWARE**

**Fondamenti di Ingegneria del Software 2024-2025**

# PROGRAM ANALYSIS

- Molti progetti falliscono a causa di “**poca qualità nel codice**”
  - Il controllo della qualità del codice non deve essere lasciato alla fine dello sviluppo
  - Aderenza ai buoni principi di design e sviluppo → il controllo è affidato a tool per l'analisi automatica del codice
- Due approcci
  - **Analisi statica:** Svolta senza eseguire il codice
  - **Analisi dinamica:** Svolta eseguendo il codice



# STATIC ANALYSIS

*“The process of automatically analyzing the computer programs without executing them”*

- Analisi del codice orientata a individuazione di problemi nel codice → code smell, codice duplicato, vulnerabilità di sicurezza, bug pattern, etc
- Svolta da un tool che può produrre reportistica, metriche di qualità, viste, etc

# STATIC ANALYSIS

*“The process of automatically analyzing the computer programs without executing them”*

- Analisi del codice orientata a individuazione di problemi nel codice → **code smell**<sup>1</sup>, codice duplicato, **vulnerabilità di sicurezza**<sup>2</sup>, **bug pattern**<sup>3</sup>, etc





<sup>1</sup>**Code smell:** codice che "puzza", ossia che può nascondere problemi (es. un metodo molto lungo che è difficile da comprendere e mantenere nel tempo)

<sup>2</sup>**Vulnerabilità di sicurezza:** una debolezza nel codice che può essere sfruttata per danneggiare il sistema (es. SQL Injection per accedere a dati senza autorizzazione)

<sup>3</sup>**Bug pattern:** un pattern erroneamente abusato durante la scrittura di codice (es. non gestire mai parametri *null*)

# STATIC ANALYSIS TOOLS<sup>1</sup>



Tool	Source
	<a href="https://github.com/checkstyle/checkstyle">https://github.com/checkstyle/checkstyle</a>
	<a href="https://github.com/pmd/pmd">https://github.com/pmd/pmd</a>
	<a href="https://github.com/spotbugs/spotbugs">https://github.com/spotbugs/spotbugs</a>
	<a href="https://www.sonarsource.com/">https://www.sonarsource.com/</a>
...	...

1: [https://en.wikipedia.org/wiki/List\\_of\\_tools\\_for\\_static\\_code\\_analysis](https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis)



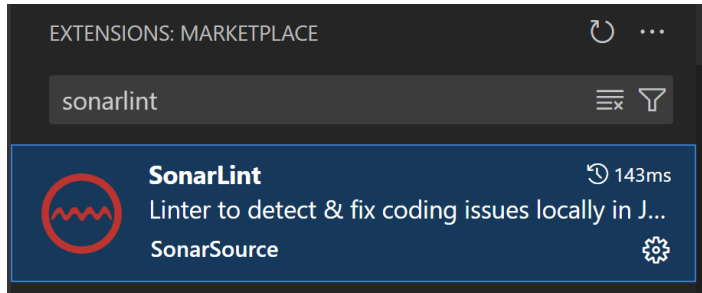


# SONARLINT & SONARQUBE

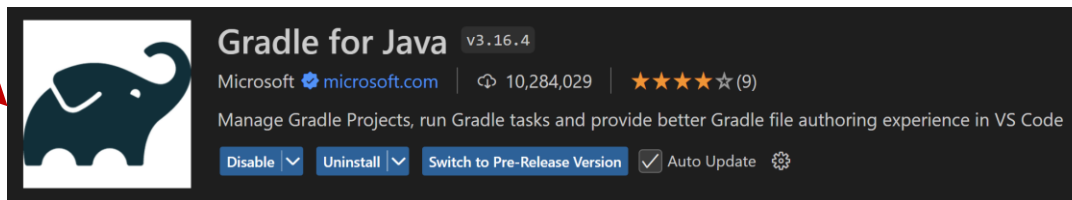
# SONARLINT

- Adatto per progetti locali
- Applicabile
  - Su più ambienti: **VS Code**, IntelliJ, Eclipse, ...
  - Su più linguaggi: **Java**, C#, Python, JavaScript, ...
- Oltre a individuare problemi nel codice, permette di
  - Approfondire problematiche mediante esempi
  - Applicare soluzioni in modo automatico
  - Includere/Escludere regole di interesse

# SONARLINT: INSTALLAZIONE IN VS CODE



*richiede*



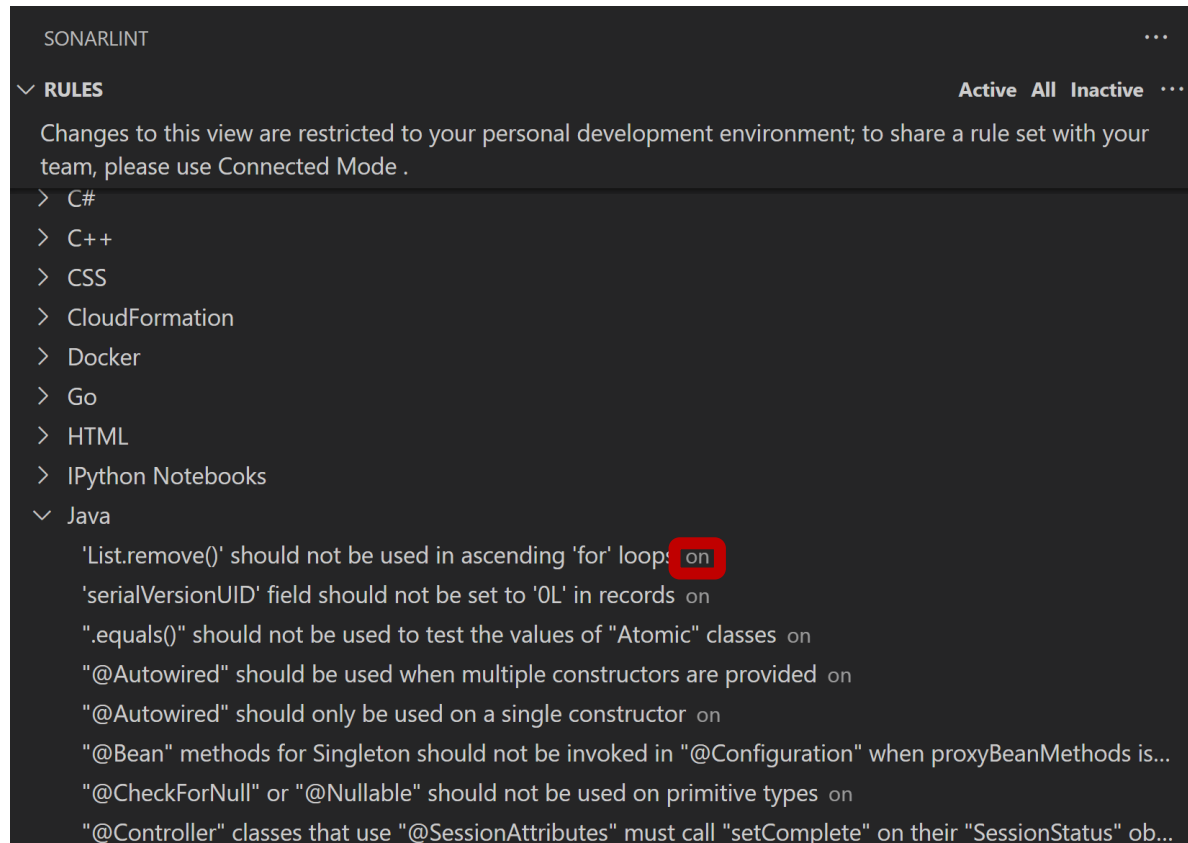


# SONARLINT: REGOLE PER JAVA<sup>1</sup>

- Il tool supporta oltre 700 regole per Java, consultabili e attivabili attraverso il plugin



*dalla scheda  
in VS Code...*



1: <https://rules.sonarsource.com/java/>

# SONARLINT: ESEMPI ISSUE

```
System.out.println(x:"Strings are equal");
```

Replace this use of System.out by a logger. sonarlint(java:S106)

Strings and Boxed types should be compared using "equals()". sonarlint(java:S4973)

[View Problem \(Alt+F8\)](#) [Quick Fix... \(Ctrl+.\)](#)

```
if (a == b) {
```

Method has 8 parameters, which is greater than 7 authorized. sonarlint(java:S107)

```
void demo.userdata.PriceCalculator.setParams(int a, int b, int c, int d, int e, int f, int g, int h)
```

[View Problem \(Alt+F8\)](#) [Quick Fix... \(Ctrl+.\)](#)

```
void setParams(int a, int b, int c, int d, int e, int f, int g, int h) {  
    // ...  
}
```

# SONARLINT: GESTIONE ISSUE

The diagram illustrates the SonarLint workflow for managing an issue. It starts with a code snippet where a variable is compared using the equals operator (`if (a == b) {`). A SonarLint tooltip appears, offering three quick fixes: "Replace with boxed comparison", "Open description of rule 'java:S4973'", and "Deactivate rule 'java:S4973'". Red arrows show the flow from the tooltip to the corrected code (`if (Objects.equals(a, b)) {`) and from the "Open description" option to the rule description panel. The panel shows the rule title "Strings and Boxed types should be compared using 'equals()' (java:S4973)", categorizes it as an "Intentionality issue", and shows a "Reliability" rating of "High". It also includes links for "What is clean code?", "Why is this an issue?", and "More Info".

```
if (a == b) {
```

Quick Fix

- SonarLint: Replace with boxed comparison
- SonarLint: Open description of rule 'java:S4973'
- SonarLint: Deactivate rule 'java:S4973'

```
if (Objects.equals(a, b)) {
```

```
if (a == b) {
```

SonarLint Rule Description X

**Strings and Boxed types should be compared using "equals()" (java:S4973)**

Intentionality issue | Not logical | Reliability

What is clean code?

Why is this an issue? More Info

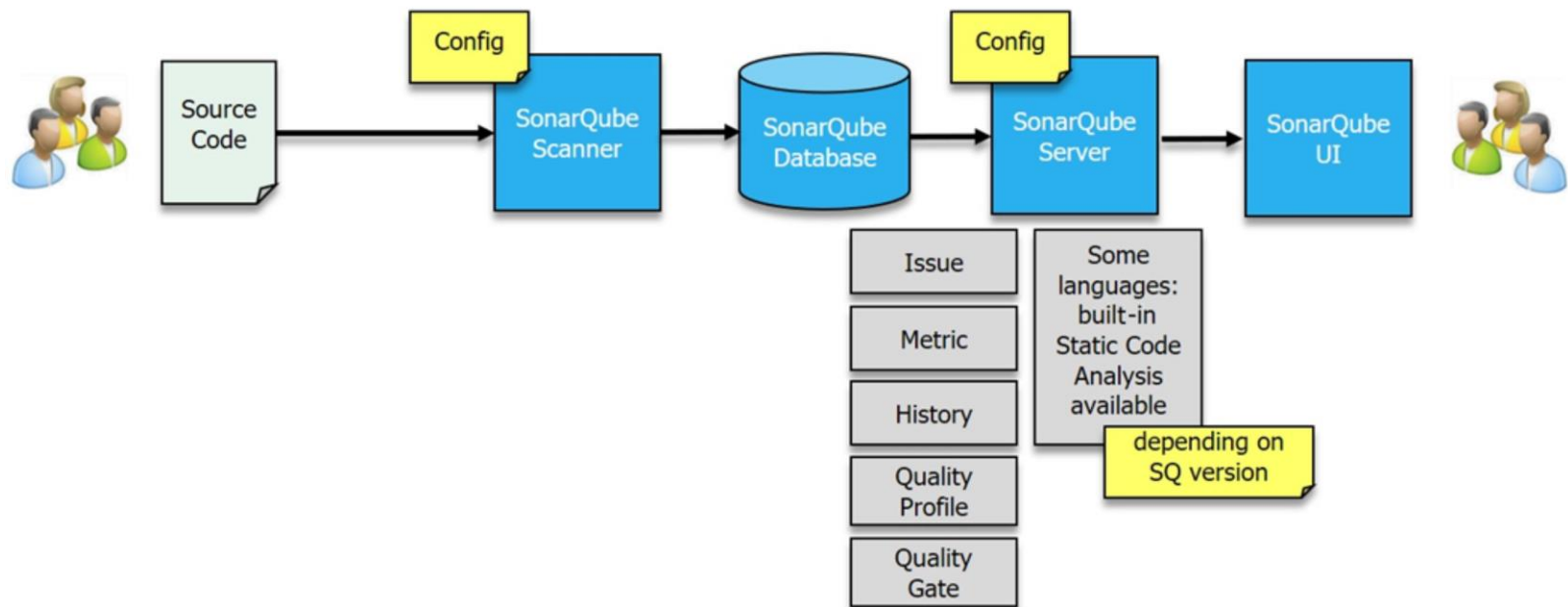
# SONARLINT: TIPOLOGIE DI PROBLEMI

- **Reliability:** rileva potenziali bug e problemi di natura logica che possono causare crash o comportamenti imprevisti (es. mancanza di gestione di eccezioni, loop infiniti, etc)
- **Maintainability:** rileva codice di difficile comprensione e manutenzione, nonché problemi di stile (es. variabili non utilizzate, metodi molto complessi, etc)
- **Security:** rileva possibili vulnerabilità del codice (es. credenziali hard-coded, mancanza di controlli negli input, etc) → I «security hotspot» (es. SQL injection) sono i problemi più difficili da individuare per SonarLint

```
String query = "SELECT * FROM users WHERE userId = '" + userId + "'";
```

# SONARQUBE

- Sonar "al cubo" → SonarLint + dashboard + analisi dettagliata (metriche di qualità, debito tecnico, codice storico, test coverage) + supporto gestione di issue, etc
- Adatto per grandi progetti e lavoro in collaborazione, nell'ottica CI/CD e DevOps

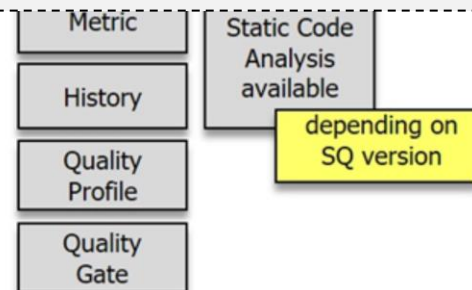


# SONARQUBE

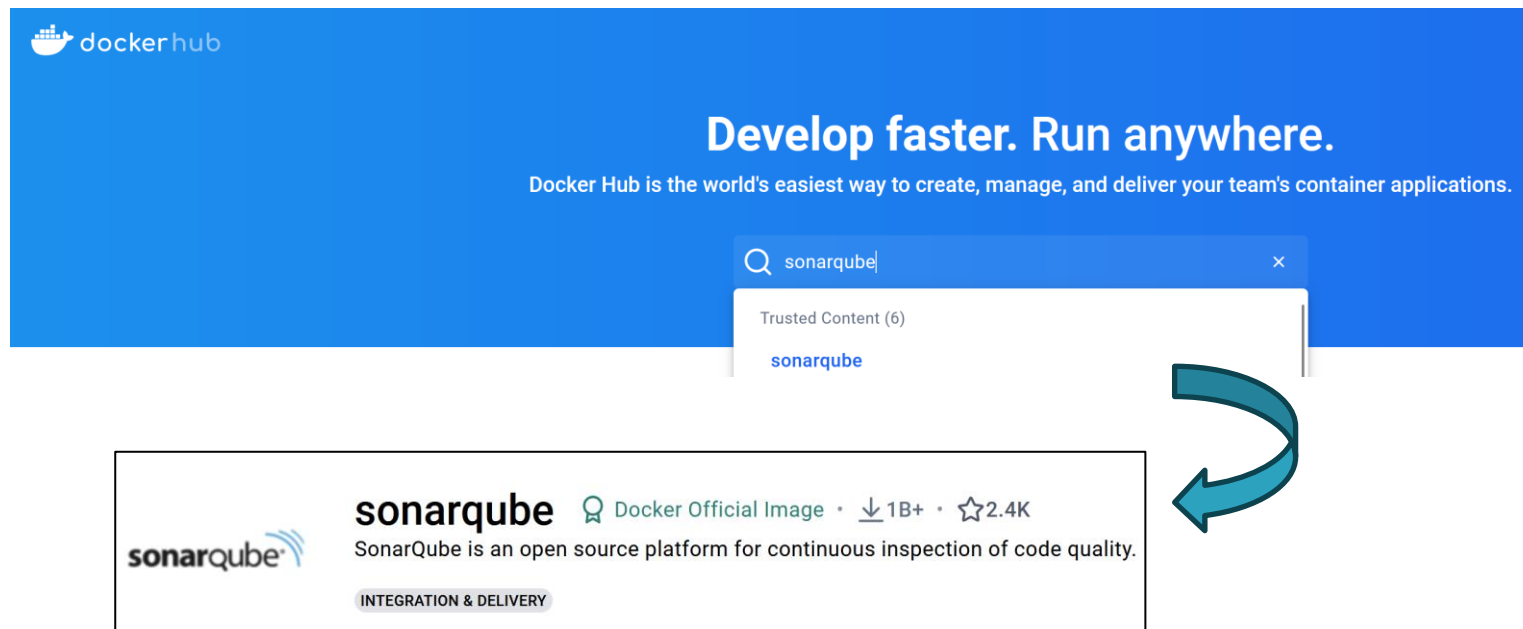
- Sonar "al cubo" → SonarLint + dashboard + analisi dettagliata (metriche di qualità<sup>1</sup>, debito tecnico<sup>2</sup>, codice storico, test coverage) + supporto gestione di issue, etc

<sup>1</sup>**Metrica di qualità:** misure stimate per valutare aspetti della qualità del codice, quali **dimensione** (es. LOC), **complessità**, **grado di instabilità** dovuta a eccessive dipendenze esterne, utili a capire quanto il codice sia "buono" (es. codice facilmente comprensibile, testabile, mantenibile, etc)

<sup>2</sup>**Debito tecnico:** il risultato di scelte di sviluppo rapide che riducono la qualità del codice e aumentano il lavoro futuro necessario per mantenere il sistema (es. non inserire logger per tracciare i problemi, non scrivere casi di test, applicare soluzioni deprecated ma più comode/rapide)



# SONARQUBE: INSTALLAZIONE (1)



```
docker pull sonarqube
```

```
docker run -d --name sonarqube-db -e POSTGRES_USER=sonar -e  
POSTGRES_PASSWORD=sonar -e POSTGRES_DB=sonarqube postgres:alpine
```

```
docker run -d --name sonarqube -p 9000:9000 --link sonarqube-db:db -e  
SONAR_JDBC_URL=jdbc:postgresql://db:5432/sonarqube -e  
SONAR_JDBC_USERNAME=sonar -e SONAR_JDBC_PASSWORD=sonar  
sonarqube
```

# SONARQUBE: INSTALLAZIONE (2)

Localhost 9000

## How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)?  
Create your project from your favorite DevOps platform.

First, you need to set up a DevOps platform configuration.

Import from Azure DevOps

Setup

Import from Bitbucket Cloud

Setup

Import from Bitbucket Server

Setup

Import from GitHub

Setup

Import from GitLab

Setup

Are you just testing or have an advanced use-case? Create a local project.

Create a local project

Si può creare un progetto locale oppure associarlo a un repo esistente, definendo una chiave di progetto e un token da usare come collegamento a IDE esterne

## Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

### 1 Provide a token

Generate a project token

Use existing token

Token name ?

Expires in

Analyze "Demo"

30 days

Generate



Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your [user account](#). See the [documentation](#) for more information.

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

1 of 2

## Create a local project

Project display name \*

Demo



Project key \*

Demo



Main branch name \*

main

The name of your project's default branch [Learn More](#)

Cancel

Next



# SONARQUBE: DASHBOARD (1)

```
./gradlew sonar
```



☐ [Revoke and change this password, as it is compromised.](#)

Security 🔴

☐ [Strings and Boxed types should be compared using "equals\(\)".](#)

Reliability 🟡

☐ [Replace this use of System.out by a logger.](#)

Maintainability 🟢

# SONARQUBE: DASHBOARD (2)

demo PUBLIC

Last analysis: 14 seconds ago • 49 Lines of Code • Java

1	1	17	0.0%	0.0%	0.0%
Security	Reliability	Maintainability	Hotspots Reviewed	Coverage	Duplications

SQL Injection 1

Make sure using a dynamically formatted SQL query is safe here.



```
13 public class BuggyClass {
14
15     public String USERID = "user001";
16     // Logger logger = Logger.getLogger(getClass().getName());
17
18     public void query(String user/* , String pwd */) throws SQLException {
19         try (
20             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydatabase",
21                 "user",
22                 "password");
23             Statement stmt = conn.createStatement()) {
24             ResultSet rs = stmt.executeQuery("select FNAME, LNAME, SSN " + "from USERS where UNAME=" +
25                 user);
26
27             Make sure using a dynamically formatted SQL query is safe here.
28         }
29     }
30 }
```

# SONARQUBE: ISSUE STATUS

Per tracciare lo stato del problema

Per assegnare il compito di gestire l'issue a un partecipante al progetto

Per visualizzare il codice relativo all'issue

Per ottenere informazioni sul motivo dell'issue

Per puntare al codice con l'issue all'interno del progetto in Vs Code

The screenshot shows the SonarQube interface for an issue. At the top, there are filters for 'Open' (selected) and 'Not assigned'. Below these are tabs for 'Where is the issue?', 'Why is this an issue?', 'Activity', and 'More info'. The 'Where is the issue?' tab is active, showing the file path 'demo > src/main/java/demo/BuggyClass.java'. The code snippet is as follows:

```
60     }
61
62     }
63
64     public boolean areEquals(String a, String b) {
65         return a == b;
```

Below the code, the issue description reads: 'Strings and Boxed types should be compared using "equals()'. A red box labeled 'Open in IDE' points to the file path. Red lines connect the annotations to their corresponding UI elements: 'Per tracciare lo stato del problema' to the 'Open' filter, 'Per assegnare il compito di gestire l'issue a un partecipante al progetto' to the 'Not assigned' filter, 'Per visualizzare il codice relativo all'issue' to the 'Where is the issue?' tab, 'Per ottenere informazioni sul motivo dell'issue' to the 'Why is this an issue?' tab, and 'Per puntare al codice con l'issue all'interno del progetto in Vs Code' to the 'Open in IDE' button.

# SONARQUBE: ALTRE FEATURES

- Lo strumento fornisce anche metriche e conteggi di dimensione, personalizzazione di criteri di qualità (es. soglia su # di code smells), stime su debito tecnico, etc

		Complexity ?	▼
Lines	127	Cyclomatic Complexity	22
Statements	53		
Functions		Debt	2h 32min
Classes		Debt Ratio	6.8%

Misura quanto è complessa una porzione di codice.

**$CC = \pi + 1$** , dove  $\pi$  è il numero di punti decisionali (es. IF, FOR, WHILE etc.)  
**[McCabe, 1976]**

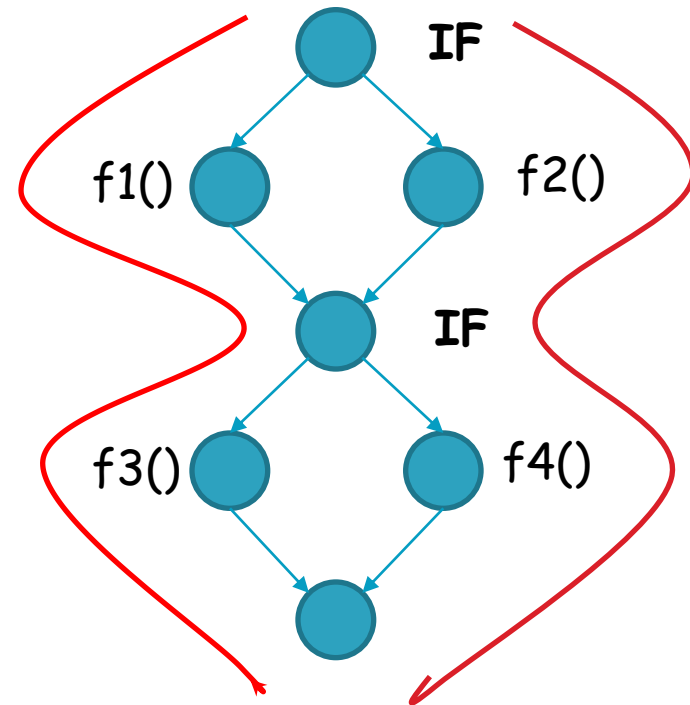
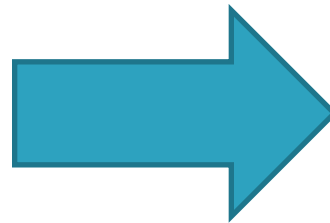
Costo totale stimato per risolvere issues (es. maintainability)

[Remove this useless assignment to local variable "c".](#) 1min effort

# CC E TESTING: SOLO INTUIZIONE

```
if( c1() )  
    f1();  
else  
    f2();  
  
if( c2() )  
    f3();  
else  
    f4();
```

**CC = 3**



**Control Flow Graph**  
grafo che rappresenta tutti i  
possibili cammini che possono  
essere presi durante l'esecuzione

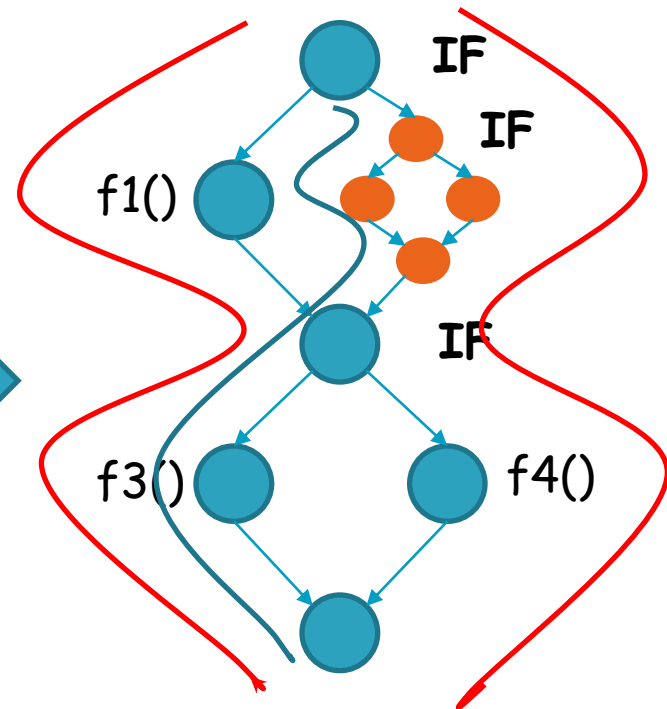
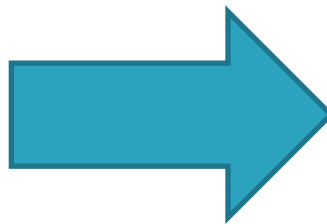
**Servono almeno due casi di test per coprire tutte le funzioni**

# CC E TESTING: SOLO INTUIZIONE

```
if( c1() )  
    f1();  
else  
    f2();  
  
if( c2() )  
    f3();  
else  
    f4();
```

IF(...)

CC = 4



Control Flow Graph

**All'aumentare della CC aumentano anche i casi di test!**

Servono almeno **tre** casi di test per coprire tutte le funzioni



# **DEMO:** **SONARLINT & SONARQUBE**



VISTE



# CLASS VISUALIZER<sup>1</sup>

- SonarLint + SonarQube producono reportistica su code smell/vulnerabilità/metriche di qualità, ma **non producono viste del codice**
- Visualizzare graficamente un progetto attraverso un diagramma permette di produrre una documentazione più chiara e di individuare problemi quali classi troppo complesse, elevate dipendenze, cicli, etc

Socomo <https://github.com/gdela/socomo>

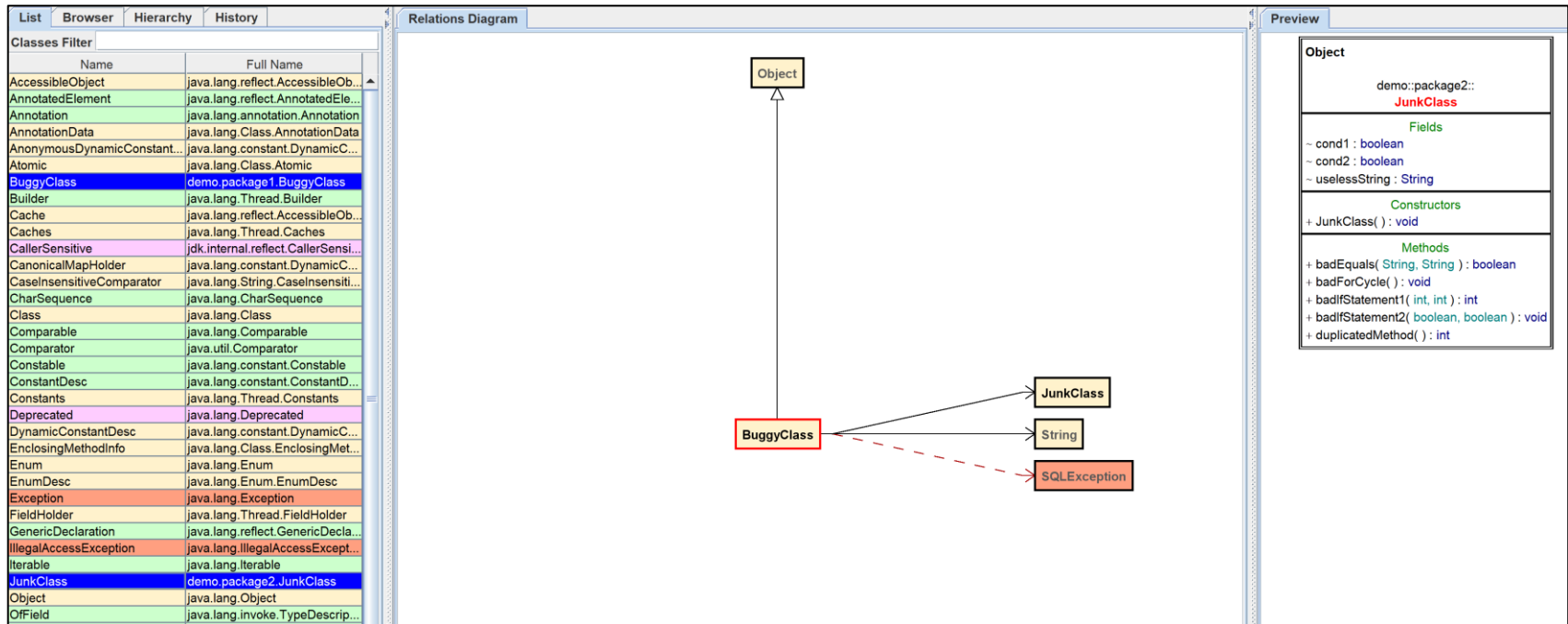
Stan4J <http://stan4j.com/>

Class Visualizer <http://www.class-visualizer.net/index.html>

Non supportano versioni  
più recenti di Java

# CLASS VISUALIZER<sup>1</sup>

Generatore automatico di class diagram  
a partire da bytecode di progetti Java



1: <http://www.class-visualizer.net/index.html>

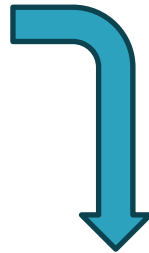
# CLASS VISUALIZER: CONFIGURAZIONE

Dopo aver scaricato e avviato il tool, per usarlo su un progetto sarà necessario aggiungere le librerie/dipendenze in base al build automation tool (es. Maven, Gradle, ...), quindi aggiungere i JAR del progetto di interesse

## Download

### Current release

[clsvis-21.0.0.zip](#) released on 28.12.2023



Unzip e run del JAR, oppure:

- Windows: clsvis.bat
- Linux/MacOS/Unix: clsvis.sh

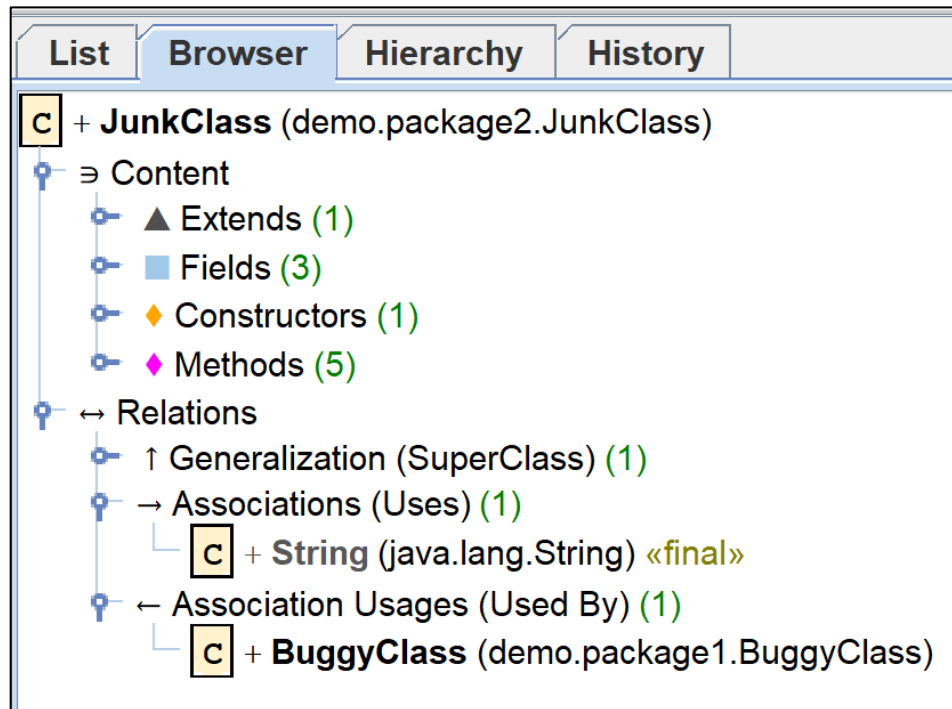
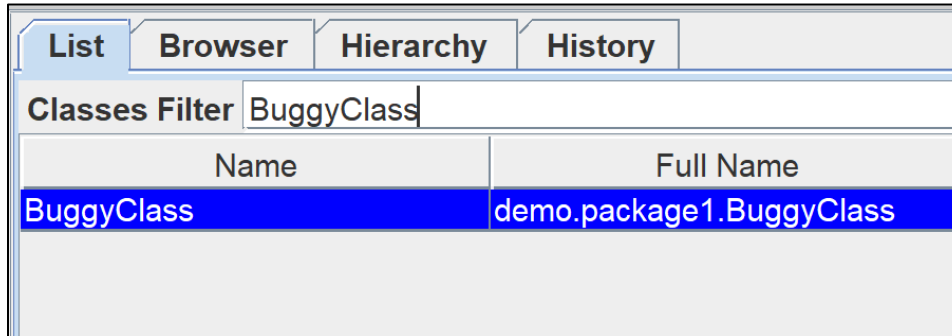
## How can I load a Gradle-based application?

In order to achieve that, perform the following steps:

- build the application
- choose **File -> Add Required Libraries** from the menu of Class Visualizer
- choose the Gradle modules cache (i.e. `~/ .gradle/caches/modules-2`)
- choose **File -> Load Classes -> from JAR files...** from the menu of Class Visualizer
- choose all relevant artifacts (JAR files) from the `build/libs` subdirectory of the application

<https://www.youtube.com/watch?v=WYfFU0AS0Jo>  
[https://www.youtube.com/watch?v=qGLPHAnW\\_58](https://www.youtube.com/watch?v=qGLPHAnW_58)

# CLASS VISUALIZER: UI (1)

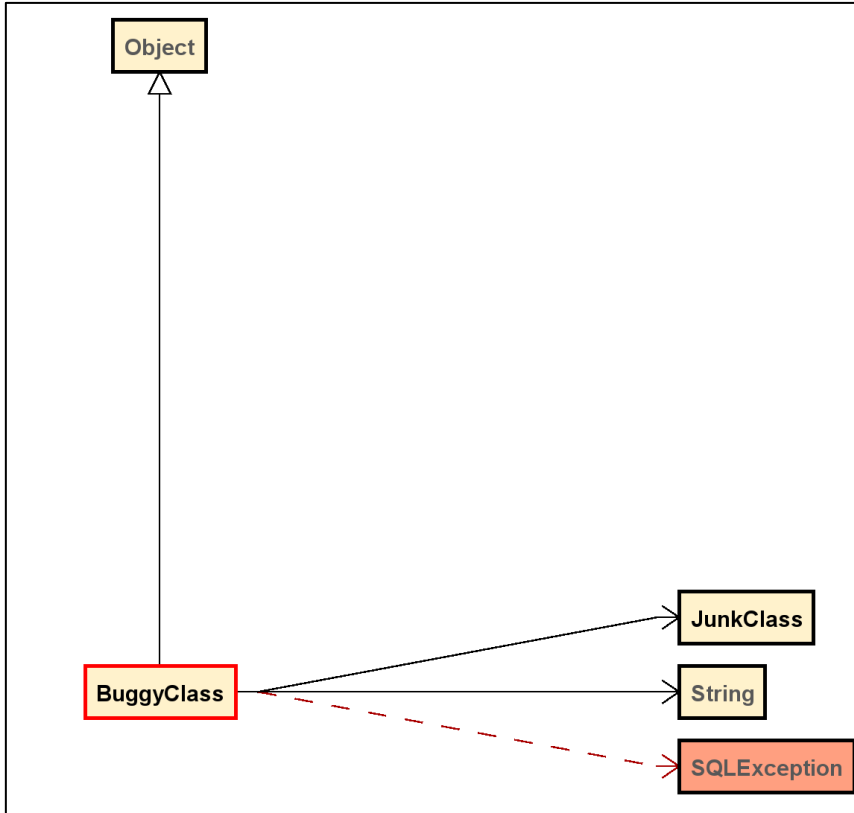


Il pannello a sinistra nella GUI elenca le classi di progetto

È possibile navigare nella struttura del diagramma, visualizzando i campi e metodi associati a una classe, associazioni (usa/usato da), gerarchia di classi, dipendenze, etc

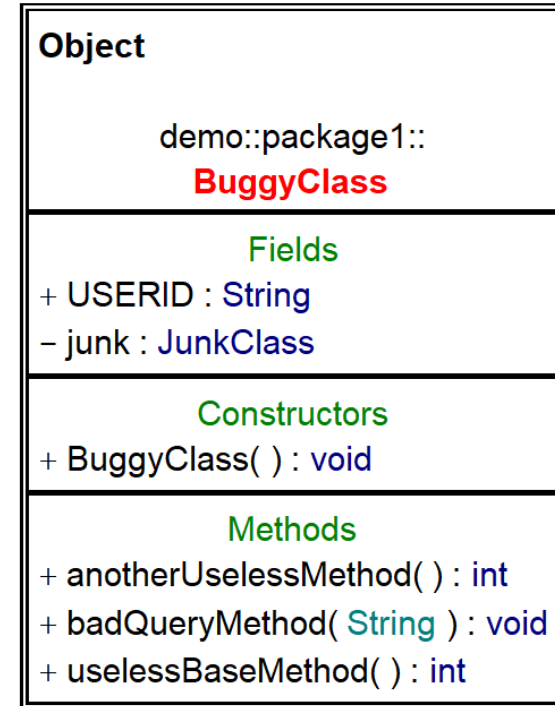
# CLASS VISUALIZER: UI (2)

## Relations Diagram Panel



Il pannello centrale visualizza la porzione di class diagram in cui la classe di interesse è coinvolta, mostrando associazioni/dipendenze/etc

## Preview Panel



Il pannello a destra visualizza la classe in un formato UML-like



# **DEMO:** **CLASS VISUALIZER**