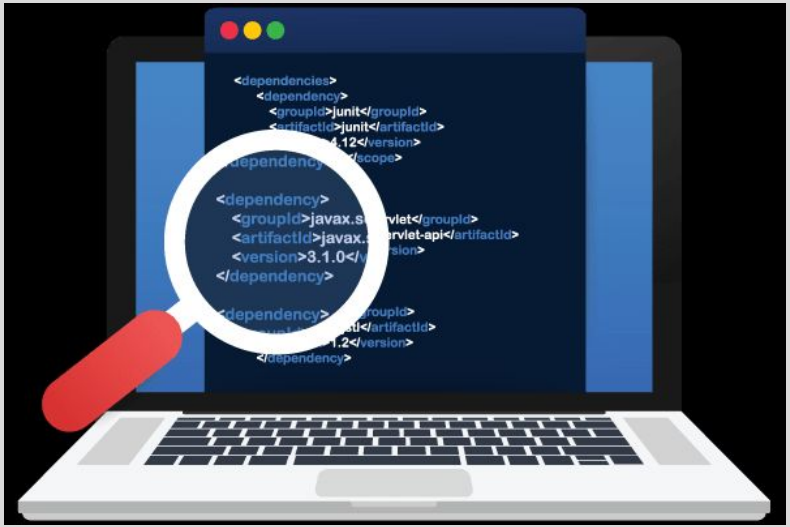
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO
Mestrado em Engenharia Informática e Computação

Static Testing

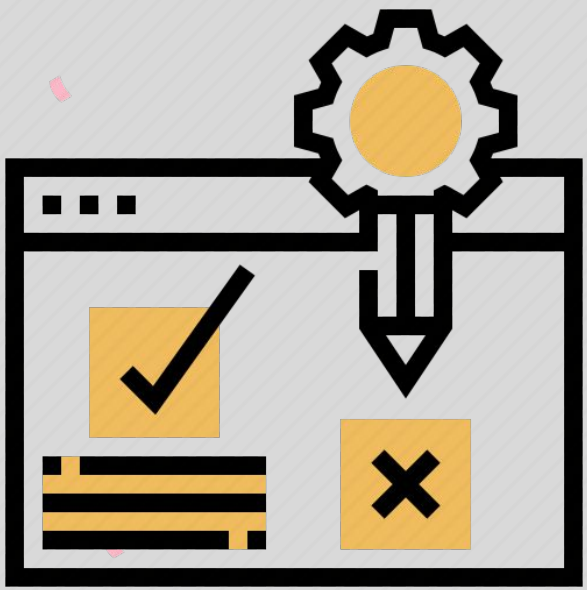


Test, Verification and Validation of Software

January 7th, 2022

Ana Silva - up201606703@edu.fe.up.pt
Maria Caldeira - up201704507@edu.fe.up.pt

Table of contents



1

Setup

Prepare yourself to learn about Static Testing

Requirements

1. Install IntelliJ IDEA in your machine.
    a. Inside IntelliJ, install SonarLint Plugin.
2. Make sure that you have Java SDK (JDK) installed.
3. Install Git.

You can access the complete Installation Guide here.

Some more steps

1. Clone the repository with the source code using git, or download it directly from GitHub.

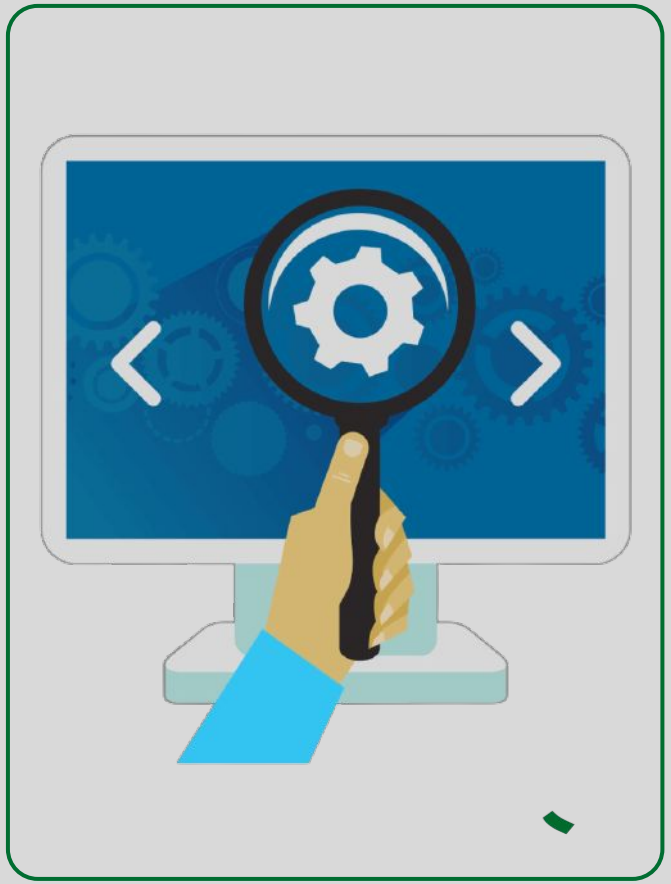
2. Open the project in IntelliJ IDEA:
   a. Open IntelliJ and make sure SonarLint plugin is installed and enabled;
   b. Open the project;
   c. When the project is loaded, try to build it. If you get the “Project JDK is not defined” error, go to File > Project Structure and define the SDK.

3. Verify if everything is working as expected:
   a. Try to build the project again. After that, try to analyze the project with SonarLint (*Ctrl+Shift+S* or right-click on the *snake_game* folder, go to the SonarLint menu and choose the option “Analyze with SonarLint”);
   b. Check the problems detected by SonarLint.

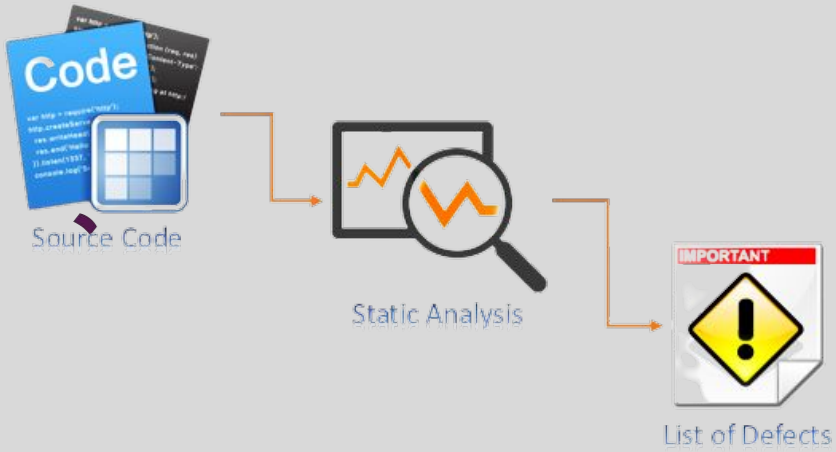
2

Static Testing

Learn more about Static Testing
and its techniques



What is Static Testing?



Software Testing Technique

Used to check defects in software without having to execute the code

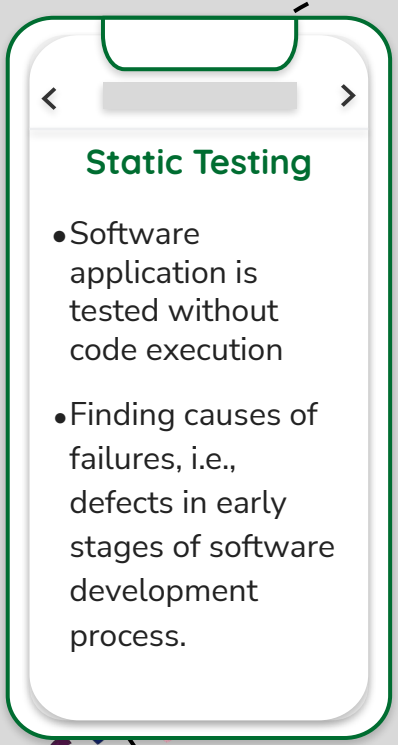
Early stage of development

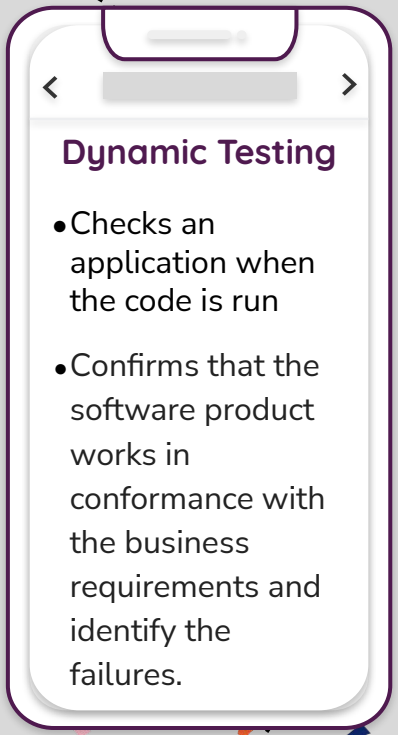
Easier to identify errors and solve them

Counterpart → Dynamic Testing

Errors that can't be found in Dynamic Testing, can be easily found with Static Testing

Static vs Dynamic





Which errors are typically found?

Security vulnerabilities

Undeclared variables

Inconsistent interface

Boundary violations

Syntax violations

Static Testing Techniques

Aim to **remove errors and ambiguities** found in the product, including supporting documents, such as software requirements specifications, design and test cases

Code analysis that may be conducted either **manually** or through **automation** with the use of various software testing tools

Types of reviews

Informal

Coworkers review documents and provide informal comments

Walkthrough

The author of document explains it to their team.

Technical/peer reviews

Technical specifications are reviewed by peers in order to detect any errors

Inspection

A moderator conducts a strict and formal review as a process to find defects.

Static Analysis Techniques

Control Flow Analysis

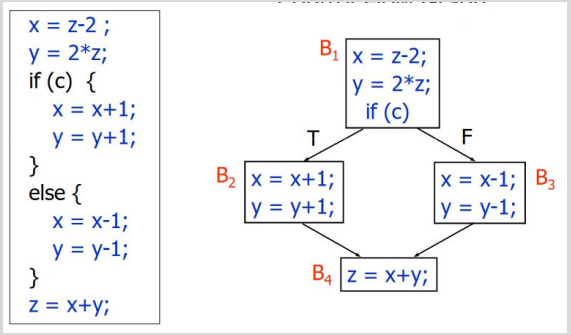
Technique for determining the control flow of the program.

Data Flow Analysis

Process that determines information regarding the definition and use of data in program

Lexical Analysis

Process of converting a sequence of characters into a sequence of tokens, in order to identify invalid tokens.



Example of CFG (Control Flow Graph)

```
#include <stdio.h>
    int maximum(int x, int y) {
        // This will compare 2 numbers
        if (x > y)
            return x;
        else {
            return y;
        }
    }
```

Lexeme	Token
int	Keyword
maximum	Identifier
(	Operator
x	identifier

Example of conversion to tokens

Data Flow Analysis Example

```java
int sum(int x, int y)
{
    int z = x * y;
    return x + y;
}
```

Unused variable

```java
int sum(int x, int y)
{
    return x + y;
    int z = x * y;
}
```

Unreachable definition

```java
int sum()
{
    int x = 1;
    int y;

    return x + y;
}
```

Uninitialized variable

```java
void function(){
    int sum = 0;
    for (int i=0; i<10; i++){
        sum += i;
        int temp = 2*i;
    }
    System.out.println(temp);
}
```

Live variables

```java
    int x = 10;
    int a = x-1;
    int b = x-2;

    while(x > 0) {
        System.out.println(a * b - x);
        x=x-1;
    }
    System.out.println(a * b);
```

```java
private static void busyExpression(){
    int x = 10;
    int a = x-1;
    int b = x-2;
    int c = a * b;

    while(x > 0) {
        System.out.println(c - x);
        x=x-1;
    }
    System.out.println(c);
}
```

Very busy expression

```perl
my $name = $cgi->param("name");  # Get the name from the browser
$dbh->{TaintIn} = 1;
$dbh->execute("SELECT * FROM users WHERE name = '$name';");
```

Taint Analysis

Static Testing Pros and Cons

Pros

- Helps identifying flaws in code in the early stages of software development cycle
- Fast and easy way to get feedback on clear coding violations
- Reduces cost of rework
- Offers increased awareness about the various quality issues in the software
- Highlight better approaches to junior developers
- Reinforce that the programmer has adopted a good coding approach (when no violations are reported)
- Improves communication, about critical and important information, among team members

Static Testing Pros and Cons

Cons

- Lack of fixes
- Configuration to ignore rules
- Adding context-specific rules
- Can be time consuming when it is done without support tools
- Doesn’t find vulnerabilities introduced during runtime
- Can lead to false positives or false negatives

False Positives and False Negatives

A false positive occurs when a static analysis tools incorrectly reports that a static analysis rule was violated.

```
int bar ();
void foo ()
{
  int i = 0;
  if (bar ())
  {
    i = 1;
    1 / i;
  }
}
```

```
void bar (int * p)
{

}

void foo ()
{
  int i = 0;
  bar (&i);
  1 / i;
}
```

A false negative is an issue that goes undetected. Happens when a rule violation exists, but no diagnostic is created.

Tools

- IDE Integration
- More focused on style
- Can report things that aren't relevant
- Java

- IDE Integration
- Static analysis of the Java bytecode
- Early feedback about potential errors in the code
- Can generate numerous false positives
- Java

Tools

- Detects unknown security issues and other complex defects
- Lack of false positives
- Paid
- PHP

- Ease of use and implementation as it's a SaaS
- Identifies vulnerabilities without to many false positives
- Code security
- SaaS based

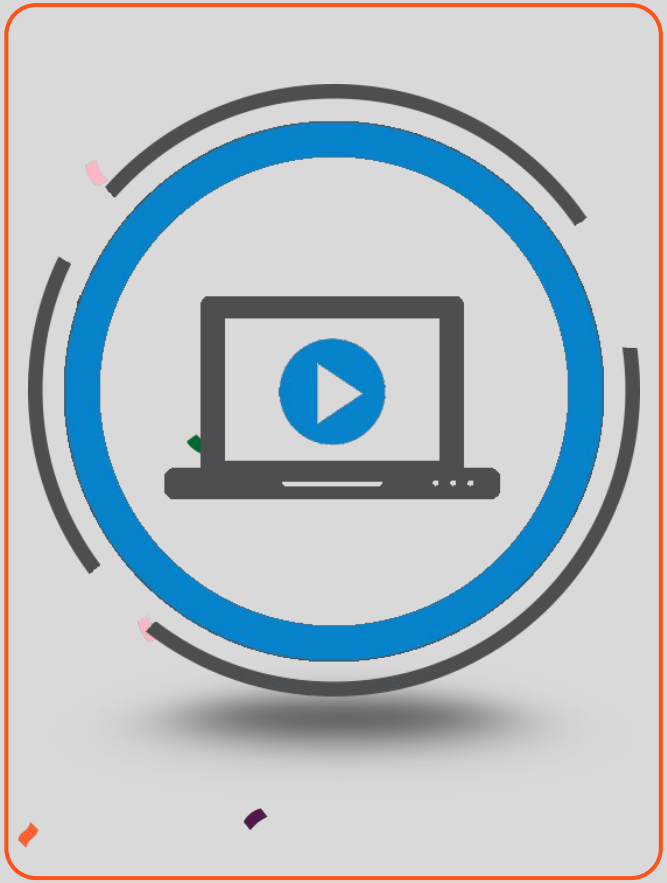
VERACODE

You change the world, we'll secure it.

Tools

- IDE Integration
- Bug detection
- Uncovers old issues
- Instant Feedback
- Java, PHP, Python, C++, etc

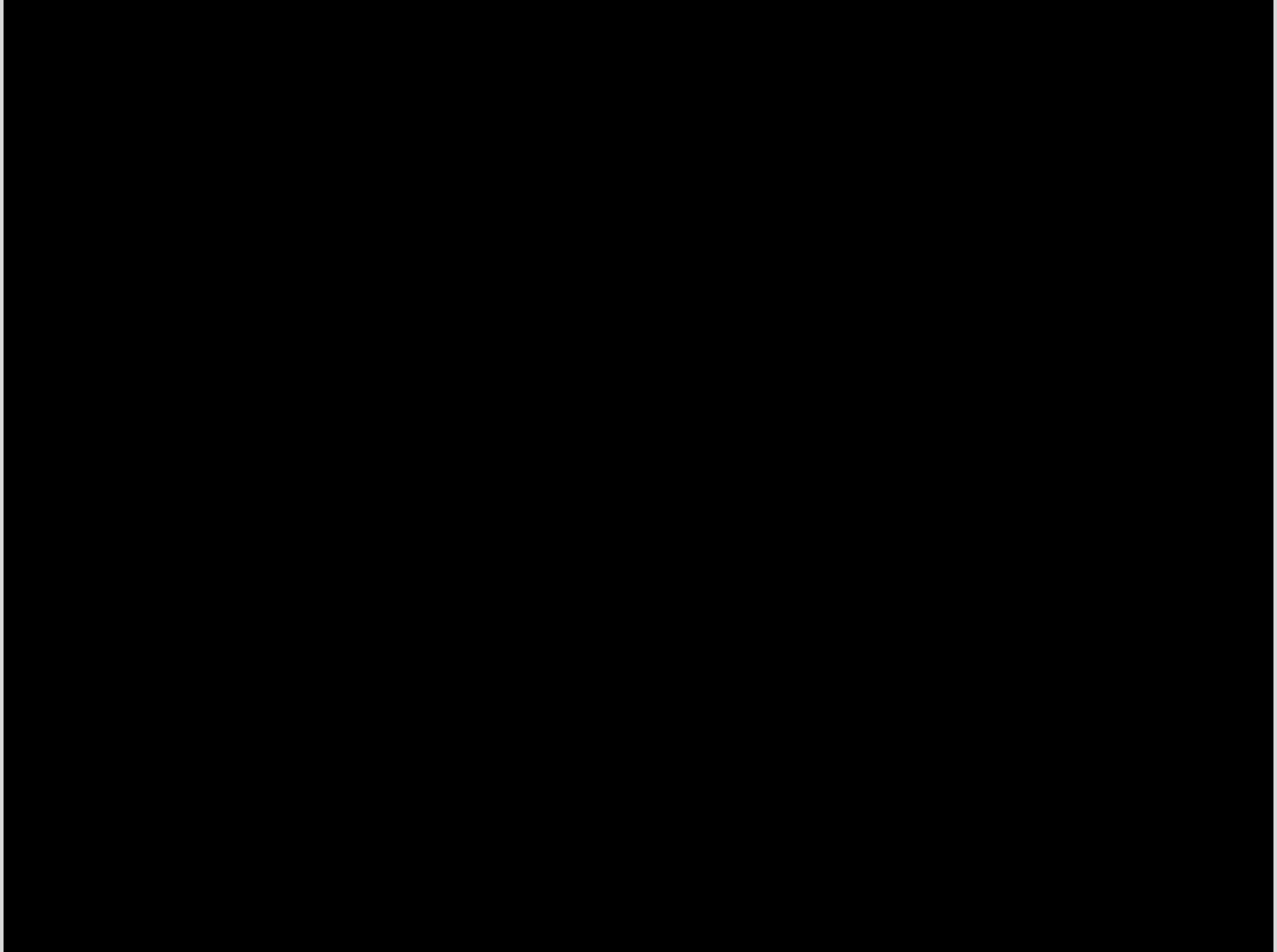
- IDE Integration
- Customizable
- Focuses code review on quality not style
- Javascript



3

Demo

Get ready for the exercises

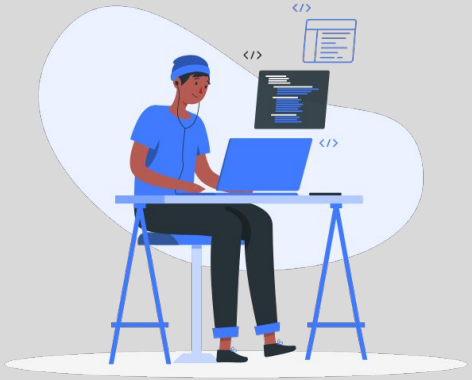


Available here

4

Practical exercises

Let’s practice!



You can find a more detailed guide here.

Issues Types and Severities in SonarLint

Bug

Vulnerability

Code Smell

Info Minor Major Critical Blocker

Severity

Snake Issues

- Program based on the well-known Snake Game
- **54 issues** in the base code

Board.java:

- 40 code smells:
  - Minor: 23
  - Major: 13
  - Critical: 4
- 9 bugs:
  - Minor: 1
  - Major: 6
  - Blocker: 2

Snake.java:

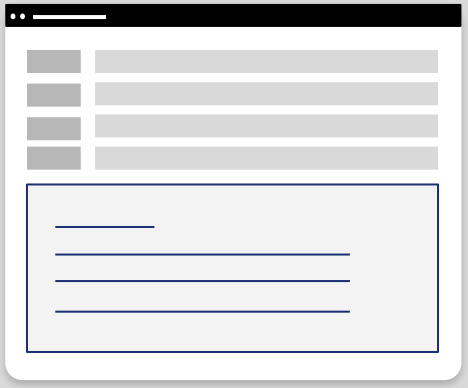
- Minor code smell: 2
- Major code smell: 3

The Exercise

- Single exercise divided into several points
- Some advices:
  - Try to understand all the issues and their possible implications on the program
  - Reading the title and the description of every issue can be very useful and, if you have any doubts, look at the example code available on SonarLint
  - If SonarLint does not automatically update when you solve an issue, you should do it manually

First Step

Start by solving the issue on *Board.js*, at line 23, already solved in the demonstration.



Second Step

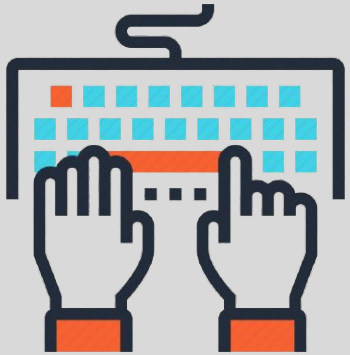
There is an issue common to both files: “*The default unnamed package should not be used.*”

Begin creating a package named “SnakeGame” and moving both files into it. How many errors do you get?

Third Step

Although the issues with a higher severity are a priority, we suggest that you start by solving all the **minor** bugs and code smells.

Then, try to fix the **major** issues. At last, solve the **critical** and **blocker** issues.



Try to play

Now that you are issue-free, test your ability to play the Snake Game and **have fun**!

Thank you!

Do you have any questions?

You can check an example of a solution here.