1. Introduction

Static code analysis is a type of program analysis that is performed without actually executing programs but rather examining the source or object code. There are many commercial and open-source tools designed to inspect code and report found issues. In this lab, students will use one of them, FindBugs, to find bugs in a given Java project.

The objective of this lab is to let students know that tools like FindBugs exist and to get some hands-on experience in how to use them and how to interpret reported issues so that it can help improve the quality of the code.

Analysis tool:

The tool we will be using in this lab is FindBugs and it works as a plug-in for IntelliJ IDEA and Eclipse. FindBugs is a Static code analysis tool that analyses Java bytecode and detects a wide range of problems based on the concept of bug patterns. FindBugs requires minimum Java 7 as runtime environment.

System under test (SUT):

The project that we will be testing in this lab is HospitalSystem 1.0 which is a simple Java project for checking in and out patients and doctors and assigning patients to the right doctors. FindBugs requires minimum Java 8 as runtime environment.

More information from the document “HospitalSystem Overview and Tool Setup”

2. Tasks

Task 1: Manual code inspection

Take a look at document “First Task” and try to find as many faulty parts as possible within 5 minutes.

Task 2: Tool setup

Set up Eclipse IDE, install the FindBugs plugin, import HospitalSystem and run FindBugs on it. Instructions can be found in the document “HospitalSystem Overview and Tool Setup”.

Task 3: Analyzing issues

To understand what FindBugs issues mean and how to analyze them, have a look at the document “Analyzing an issue”.

3. Deliverables and Grading

D1. Analyze bugs (7 points)

Your first task is to analyze 10 bugs of your choice found by the FindBugs tool in the HospitalSystem project. Choose the bugs from all different ranks (Of Concern, Troubling, Scary, Scariest) and confidences (Low, Normal, High) and try to decide whether the given bug is an actual fault or a false-
positive (instructions for that can be found from the document “Analyzing an issue”). In case of an actual fault, try to think of a solution to the problem.

For each bug fill out a row in the following table, the first one is given as an example:

<table>
<thead>
<tr>
<th>Bug description and location</th>
<th>What seems to be the problem?</th>
<th>Is this a false or a true positive? Should this bug be fixed? If yes, then how, otherwise why not?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Exception created and dropped rather than thrown Person.java:15</td>
<td>This code creates an exception (or error) object, but doesn't do anything with it.</td>
<td>This is a true positive and should be fixed. Besides simply creating the instance of the exception, it should also be thrown. The fixed line would be: throw new Exception(&quot;Birthyear cannot be in the future!&quot;);</td>
</tr>
</tbody>
</table>

D2. Compare methods (2 points)

Your second task is to briefly compare the benefits of using static code analysis tools versus dynamic testing. State at least one concrete benefit for both methods.