Project 2 Proposal: Exception Handling Analyzer

Group Number: 12

Team Members: Matthew Wu, Samantha Tseng, Ray Huang, Calvin Zhang

A. Brief (1-3 sentence) Description of Proposed Program Analysis Tool:

We will be building a tool to help analyze exceptions in Java. The tool will help users identify and handle exceptions that might be thrown in their Java code.

B. Intended User-Group:

Java developers who want to improve their code quality. And group 12.

C. What language will you target? What features will you aim to support (or not support)?

We will support Java

Features to support:

- Identify empty catch exception blocks
- Preventing expensive exception-based control flow
    - Check if try-catch blocks in loops or any in-loop methods
- Assign risk scores to exception handling code based on the frequency of errors
- Identify if the thrown error is not declared in the method
- Identify any open resource that is not closed by the catch block
- Suggest adding a try-catch block to methods that potentially generate errors if the user didn't include it
- Identify if a catch block is still printing to System error
- Identify any catch blocks that are always triggered

D. Which questions about program behavior will users be able to answer with your tool?

- Are there areas where we are missing a try-catch block?
- Redundant try-catch blocks (e.g. we can use Files.exists() to check file existence rather than catching FileNotFoundException)
- Are there any catch blocks that are *always* triggered?
- Where are empty catch blocks?
- Which try-catch blocks still print errors?
- Where might resource leaks happen (eg. Unclosed file streams)
- Which try-catch blocks are in loops that may slow down execution?
- Build a call graph, and check how many functions might encounter this error

E. Which of the three criteria does your project satisfy? Justify why the criterion is satisfied.

1. ***Includes a static analysis component*** Yes

   Our tool can be run on Java code, without executing the Java code

2. ***Has a substantial visualisation component, to present resulting data in a useful / appealing way***

   We will highlight sections of the code that have been identified as potentially problematic. Users can hover over these sections, to display more information on what the potential issue is, as well as suggestions for potential fixes.

   We can also assign each risk a “risk level” depending on estimated severity, and use different colors to highlight each severity

   We might also build a call graph, to view how often we might encounter certain exceptions.

F. Which of the four impossible properties does your project not satisfy?

Our tool aims to be a static analysis over all executions. We will not be satisfying property 4 (always say no when it should say no). We will be taking an “optimistic” approach, to prevent overloading the user with too many errors/warnings. This means that if we do identify an error, it will for sure be an error. However, this means sometimes we will “pass” programs that should be failed.