RETROFITTING JAVA OPTIONAL THROUGH AUTOMATED REFACTORING

Amanda Randombage

A dissertation submitted in partial fulfilment of the requirement for

Bachelor of Engineering (Honours) degree in Software Engineering

Department of Computing

Informatics Institute of Technology, Sri Lanka

in collaboration with

University of Westminster, UK

2021

Abstract

NullPointerExceptions (NPEs) are very common in Java programming. Tony Hoare, who introduced null references into the programing world described his own idea as a ‘billion-dollar mistake’. The reason is the confusion created among developers by the ambiguity of returning null to represent “empty” or “missing information” which can lead to NPEs and is considered a bad practice.

A research done using open-source codebases shows that 35% of the conditional statements in Java code include null checks and 71% of those null checks refer to values returned from methods. (Osman *et al.*, 2016) There are many researches done in the area, all concluding towards eliminating returning null references from methods to reduce the frequency of NPEs.

Introduction of Optional class in Java with Java 8 provides a mechanism to write better code that can avoid NPEs, but in general, the cost of code refactoring acts as a barrier to retrofit new code features to existing legacy code bases. This research presents a proof of concept of a tool to assist the developer to locate the code blocks where returning a null reference is a possibility and provide automated refactoring of the code to use Optional.

Subject Descriptors:

Programing Principals

Maintenance

Key words:

Software Maintenance, Static Analysis, Null Pointer Dereferences, Optional