A majority of modern software is constructed using languages that compute by producing side-effects such as reading/writing from/to files, throwing exceptions, acquiring locks, etc. To understand a piece of software, e.g. a class, it is important for a developer to understand its side-effects.

Similarly, to replace a class with another, it is important to understand whether the replacement is a safe substitution for the former in terms of its behavior, a property known as substitutability, because mismatch may lead to bugs. The problem is especially severe for superclass-subclass pairs since at runtime an instance of the subclass may be used in the client code where a superclass is mentioned. Despite the importance of this property, we do not yet know whether substitutability w.r.t. effects between subclass and superclass is preserved in the wild, and if not what sorts of substitutability violations are common and what is the impact of such violations.

This thesis conducts a large scale study on over 20 million Java classes, in order to compare the effects of the methods of subclasses and superclasses in practice. Our comprehensive study considers the exception, synchronization, I/O, and method call effects. It reveals several interesting findings and provides useful guidance for bug detection, testing, and code smell detection tool design.