
Why extends is not evil

A refutation of the article presented by Allen Holub, titled 'Why extends is evil' [<http://www.javaworld.com/javaworld/jw-08-2003/jw-0801-toolbox.html>]

Tony Morris <http://tmorris.net/>

Copyright © 2006 Tony Morris

Abstract

The Java `extends` keyword is not evil. It has never been evil; it will never be evil. What is evil, within a well defined axiom, is concrete behaviour inheritance, which is also known as 'implementation inheritance' or just 'concrete inheritance'. Holub sets out to prove this, but inadvertently implies a correlation between the Java `extends` keyword and implementation inheritance. The fact is, this correlation does not exist. This assertion would be a mere issue of semantics and would probably not warrant this article, if it weren't for a nasty, underlying issue.

This article does not set out to prove that concrete inheritance is evil. The emphasis on this fact is deferred for the sake of simplicity, since it is not simple to prove to a potential audience who has been following the doctrine that 'concrete inheritance is useful' for decades and therefore, requires more than a mere article to cover the topic appropriately. Allen Holub attempts to point out the facts, but fails to back up his assertion that the `extends` keyword is evil by portraying untruth, which unfortunately, undermines the real facts of the issue at hand.

Introduction

If I were to ask the average software developer for the meaning of 'inheritance', the typical response might be 'when a class extends another class'. This answer is a result of the modelled preconceptions that are being taught in such places as tertiary institutions. The fact is that this kind of inheritance implies that your software design is flawed. The only legitimate, and often overlooked type of inheritance is interface inheritance; 'when an interface extends an interface'. One can only dream of what we would have today if only this concept was taught from the very beginning.

Inheritance

The `extends` keyword is used in interface inheritance and this is one reason that the keyword is not evil. It is unfortunate that Holub has become a victim of his own preachings by associating the `extends` keyword with 'inheritance' and associating 'inheritance' with 'concrete inheritance'. These misconceptions should be abandoned - `extends` is not evil and inheritance is, if anything, interface inheritance. The only keyword that is exclusively associated with concrete inheritance is the `protected` keyword. The use of this keyword should be avoided indiscriminately. Also, the `abstract` keyword should be avoided. This is because the only legitimate abstract methods are those that are specified in an interface, and explicit specification of redundant modifiers is generally, and appropriately, considered poor form (but a much lesser evil with far less serious consequences than concrete inheritance). That is, interface methods are implicitly abstract and there is no good reason to explicitly specify them in this way, but this is a side issue to the one at hand.

Java 2 Standard Edition 1.5 introduces other uses for the `extends` keyword. The keyword may be used in generics [<http://java.sun.com/j2se/1.5.0/docs/guide/language/generics.html>], and it also may be used for interfaces that inherit from annotations [<http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html>]. Neither of these situations should be considered evil.

As a practical example of this concept, consider the source code to the JTiger Unit Testing Framework for J2SE 1.5 [<http://jtiger.org/>] project and the ContractualJ [<http://contractualj.com/>] project. In this source code, you will find:

- No occurrences of the `protected` keyword.

- No occurrences of the `abstract` keyword.
- Several occurrences of the `extends` keyword.
- All classes declared with the `final` modifier (explicitly preventing concrete inheritance).

Conclusion

Holub has good intentions with this article, and makes some very valid points regarding concrete inheritance. Unfortunately, these points are undermined by blatantly false information and incomplete (by far) reasoning. As a firm believer in the notion that 'concrete inheritance is evil', I acknowledge that I am a member of a minority group of 'believers', and presenting our opinion should be performed accurately, precisely, and completely in order to convince even the most ignorant audience by providing *all* of the facts. To reiterate, this article does not intend to present those facts, only refute those that have been falsely portrayed. Discussion and further proof that concrete inheritance is unreservedly indicative of a flawed software design is deferred to a future publication.