

Private и final?

Java Q&A

Действительно ли необходимо метод, описанный как `private`, определять еще и как `final`?

Что дано:

- методы, описанные как `private`, не могут быть переопределены в подклассе;
- методы, описанные как `final`, также не могут быть переопределены в подклассе;
- `final` методы позволяют ускорить выполнение кода, при компиляции со включенной оптимизацией (`javac -O`)

Заданные мне вопросы:

1. Почему бы не описывать все `private` методы как `final`?
2. Интерпретируются ли большинством компиляторов `private` методы как `final`?

Вы задали интересные вопросы.

Как вы знаете, при разработке подклассы не могут переопределять `private` методы? Более того, ключевое слово `final` говорит компилятору, что подклассы не могут переопределить метод, согласно уровню доступа к нему. Смысл ключевого слова `private` уже подразумевает, что подкласс не может переопределить метод и описание `private` метода еще и как `final` — избыточно. Создание такого описания не вызовет проблем, но и не произведет никакого другого эффекта, так как `private` автоматически приравнивается к `final`.

Но практика описания `private` методов как `final`, будет иметь один сторонний эффект. Любой начинающий Java программист, столкнувшийся с вашим кодом, воспримет вашу технику использования `private final`, предполагая, что так и должно делаться. Таким образом, вы накажите и тех кто ознакомился с вашим кодом, и тех кто этого избежал. Это может оказаться интересным уроком.

Ответ на 1-й вопрос: нет надобности описывать `private` члены класса как `final`.

И ответ на 2-й вопрос: любой оптимизирующий компилятор и виртуальная машина Java (JVM) могут использовать выгоды и `private` методов и `final` методов. Так как подклассы не могут переопределять данные этих типов, во время выполнения кода нет необходимости производить динамическую привязку. Подклассы никогда не переопределяют метод, поэтому среда выполнения всегда знает что метод вызывается без просмотра иерархии наследования. После компиляции, оптимизирующий компилятор может даже произвести `inline` подстановку всех `private` и `final` методов для повышения производительности.

Таким образом ответ на второй вопрос положительный, все компиляторы интерпретируют `private` методы как `final`. Компилятор не позволит переопределить ни один `private` метод. Также, все компиляторы не допустят переопределения подклассом `final` метода.

Более интересный вопрос — будут ли компиляторы оптимизировать `final` и `private` методы путем `inline` подстановки? Если кратко, то нет. Поведение компиляции зависит от компилятора и его настроек.

1. Об авторе

[Tony Sintes](#) главный консультант в [BroadVision](#). Tony, сертифицированный Sun Java 1.1 программист и Java 2 разработчик, работает с Java начиная с 1997г.

Reprinted with permission from the September 2000 edition of JavaWorld magazine. Copyright © ITworld.com, Inc., an IDG Communications company.

View the original article at: <http://www.javaworld.com/javaworld/javaqa/2000-09/02-qa-0915-private.html>

[Перевод на русский © Александр Серебряков, 2000](#)