

Волшебство

Андрей Терешко

Волшебство — это то, что не вписывается в правила, что вызывает удивление и желание познать. В этой статье я приведу немного волшебства из мира Java. Постарайтесь сами, уважаемый Читатель, найти ответ на эти "загадки". А я "раскрою карты" позже, в следующий раз.

1. Модификатор поля (переменной класса) "final" — "объявляет, что значение переменной присваивается всего один раз, при ее инициализации" (К.Арнольд и Д.Гослинг "Язык программирования Java").

Приведите пример, когда волшебным образом, можно менять поле с модификатором "final" столько раз, сколько душе угодно. Подсказка — это поле вы, уважаемый Читатель, наверняка знаете и довольно часто используете в своих программах на Java.

2. Если, при создании класса, приписать методу атрибут доступа "private", то доступ к такому методу может быть осуществлен только из самого класса.

Приведите пример, когда разработчик класса должен создавать методы класса с атрибутом доступа "private", но нигде(!) в своем классе, не использовать вызов этих методов. Вот как удивлялся этому волшебству Д. Флэнэген в своей книге "Java in a Nutshell": "... Как ни странно, но эти методы не определены ни в одном из интерфейсов. Их следует объявлять как "private", что также вызывает некоторое удивление, поскольку они вызываются извне...". Кому же нужны такие методы? Кто их вызывает извне и как?

3. Конструктору класса нельзя приписать атрибут "synchronized", ибо, как пишут К.Арнольд и Д.Гослинг ("Язык программирования Java") — "Конструктор не обязан быть synchronized, поскольку он выполняется только при создании объекта, а это может происходить только в одном потоке для каждого вновь создаваемого объекта". Замечу, что не просто "не обязан", а просто и не может иметь атрибут synchronized,

иначе при компиляции сразу вылетает ошибка: "Constructors can't be native, abstract, static, synchronized, or final".

Приведите пример, возможно и гипотетического, но вполне работоспособного кода, когда конструктор просто обязан(!) обладать свойствами, обычно даруемыми атрибутом "synchronized". И как же этого достичь?

4. При вызове конструктора оператором "new", конструктор всегда возвращает **НОВЫЙ** объект — то есть ссылку на новый объект. Заставить конструктор вернуть вместо ссылки на новый объект ссылку на уже существующий объект нельзя. А если очень хочется? Как "обмануть" конструктор?

Приведите пример кода, когда при "создании" объекта — то есть при присвоении новой ссылке значения, эта новая ссылка получает значение ссылки на уже существующий объект.

Секрет большинства из этого волшебства, как и любых других, находится, конечно, "за сценой". И надо сказать, что чем меньше такого волшебства "за сценой", тем лучше. Как и обещал, в следующей статье я "раскрою карты".

Если Читатель знает еще, подобные вышеописанным, волшебства, то сообщите мне, я расширю список, указав источник, то есть Читателя.

Я надеюсь, что Читатель не испугается такому волшебству в языке Java, а только немного удивится, ведь главное, чтобы Вы понимали.