**Композиция vs Наследование в Java**

В чём отличие между абстрактным классом и интерфейсом?  
В чём отличие между композицией и наследованием?

Так получилось, что эти вопросы я задал довольно большому количеству людей на собеседованиях. И, как мне кажется, есть определённое непонимание этих базовых концепций, вернее, расхождение между теорией и практикой. Данная статья призвана внести ясность и улучшить  код.

Немного теории

С композицией всё просто. Большой объект состоит из меньших и выполняет (делегирует) какую-то работу с их помощью. Например, автомобиль состоит из кузова, двигателя, колёс и т.д. А метод ехать() реализован что-то вроде: двигатель.работает(), колёса.крутятся() и поэтому кузов.перемещается().

С наследованием, на мой взгляд, сложнее. То есть определение «механизм языка, позволяющий описать новый класс на основе уже существующего» — это, конечно, хорошо, а когда его использовать?

Есть критерий, что композиция — это отношение has-a, тогда как наследование — is-a. Есть принцип Лисков, третья буква в абревиатуре SOLID, который утверждает, что наследуемый класс должен дополнять, а не замещать поведение базового класса. Об этом, кстати, прямо намекает ключевое слово extends в Java. Есть Джошуа Блох, который в Effective Java говорит, что наследование — это сильная связь.

На мой взгляд, даже сам термин наследование не очень удачный, не отображает суть, и правильнее было бы использовать «дополнение», но традицию не изменить. И вообще, было бы неплохо различать наследование классов (дополнение реализации?!) и наследование интерфейсов (дополнение контракта — это пожалуйста).

Переходим к практике

А теперь от игры словами давайте перейдём к написанию кода.

Итак, задача. Нужно разработать модуль генерации отчётов для банка. Каждый отчёт состоит из трёх частей — заголовка (header), собственно тела отчёта (body) и колонтитула (footer). Формируют их некие методы. Формирование header и footer для всех отчётов одинаково и меняться не будет. Поэтому их код разумно переиспользовать. Body для каждого отчёта, естественно, специфично. Начинаем с двух отчётов, дальнейшие пока согласовываются с заказчиком.

Собственно, вопрос сводится к такому: есть четыре блока кода — header(), footer(), body1() и body2(). Как разложить их по классам?

И тут происходит переломный момент. Несмотря на все правильные определения вначале, многие почему-то предлагают такое решение.

**class** **BaseReport** {

**void** **printHeader**() {

// 100 lines of header code

}

**void** **printFooter**() {

// 50 lines of footer code

}

**abstract** **void** **printBody**() {}

**void** **print**() {

printHeader();

printBody();

printFooter();

}

}

сlass Report1 extends BaseReport {

@Overrride

**void** **printBody**() {

// specific body of Report1

}

}

BaseReport report1 = **new** Report1();

report1.print();

Аналогично для второго отчёта. Вроде как хороший вариант, применён шаблон проектирования Template Method, но есть нюанс.

Почему так делает большинство кандидатов, я не знаю. Думаю, дело в литературе, где наследование объясняется на неудачных примерах, в результате формируется убеждение, что главное в ООП — это наличие иерархии классов, а о Лисков и Блохе сразу не упоминают.

Если вы видите проблему, поздравьте себя, уровень вашего мастерства явно выше Junior, если не видите — сейчас проблема будет :-)

По этой схеме пишутся второй, третий и так далее отчёты, а, допустим, в пятом появляется уточнение — колонтитул (footer) не нужен. Окей, ломать не строить, его можно убрать.

**class** **Report5** **extends** **BaseReport** {

@Overrride

**void** **printBody**() {

// specific body of Report 5

}

@Overrride

**void** **printFooter**() {

}

}

Пустой метод выглядит немного странно, но задачу свою выполняет.

Идём дальше (в следующих примерах я буду опускать body). В шестом отчёте нужно после колонтитула добавить ещё какой-то блок, там, список использованной литературы (appendix). Есть два варианта — либо по аналогии с printBody() добавить абстрактный метод printAppendix() в базовый класс, в этом классе его переопределить, а во всех предыдущих отчётах добавить его пустым, либо исхитриться так:

**class** **Report6** **extends** **BaseReport** {

@Override

**void** **printFooter**() {

**super**.printFooter();

printAppendix();

}

**void** **printAppendix**() {

// 50 lines of appendix code

}

}

Коряво, но пусть будет. Дальше, в седьмом отчёте нужно сделать полностью другой заголовок. Тут (или даже раньше) можно, конечно, начать возмущаться. Как же так? Ведь в условиях задачи было недвусмысленно сказано, заголовок и колонтитул изменяться никогда не будут. Да, небольшой подвох. Даже не подвох, а обычная рабочая ситуация. Ну, изменились требования, бывает. Что же теперь делать?

**class** **Report7** **extends** **BaseReport** {

@Override

**void** **printHeader**() {

// 30 lines of completely another header code

}

}

Окей, ещё один отчёт с таким же другим заголовком. Так, что ли?

**class** **Report8** **extends** **Report7** {

// using printHeader() from the parent class

@Overrride

**void** **printBody**() {

// specific body of Report 8

}

}

А если заголовок как в 7-ом отчёте, а колонтитул как в 10-ом? От кого наследоваться?

Или так. Восьмой отчёт практически соответствует начальным условиям. Только в стандартном заголовке выводится текущая дата, а здесь её не нужно. Это же совсем маленькое (показывает пальцами) изменение, правда? Т.е. у нас есть 100 строк кода, которые формируют заголовок, в 78-ой выводится эта дата, как её убрать?

Скопипастить и оставить 99 строк? Плохо!

Снова модифицировать базовый класс (и заодно всю кучу его наследников) и добавлять булевый метод, типа, нужно ли выводить дату с return true по умолчанию, а здесь его переопределять? Тоже нездоровое решение.

Очевидно, наш дизайн зашёл в тупик, и виной этому, увы, неправильное применение наследования. Мы вовсю противоречим принципу Лисков и только тем и занимаемся, что переопределяем поведение.

Как исправить ситуацию

Давайте посмотрим на задачу с самого начала. От «отчёты состоят из трёх блоков, два из которых неизменны» она превратилась в «разные отчёты состоят из неких частей, какие-то из них общие или похожи, но не всегда». Требования эволюционировали (так обычно и бывает), а наследование вроде как и помогло нам избежать дублирования кода в первых двух классах, но мешало дальше, причём всё больше. Вот и проявилась сильная связность — все последующие отчёты мы буквально запихивали в «схему».

Как быть? Собственно, слово «состоит» в условии задачи уже намекает, что композиция будет более подходящим решением в этом случае.

Итак, у нас будет не базовый класс, а интерфейс.

**interface** **Report** {

**void** **print**();

}

Далее, у нас будут, допустим, классы DefaultHeader и DefaultFooter тоже с методами print(), что наводит на мысль и его вынести в интерфейс ReportSection, а может Report и ReportSection будут одним и тем же.

**class** **Report1** **implements** **Report** {

DefaultHeader header;

Body body;

DefaultFooter footer;

@Override

void print() {

header.print();

body.print();

footer.print();

}

// nested **class**

**class** **Body** {

}

}

И ответы на все заданные выше вопросы — отчёт без колонтитула, с списком после колонтитула, с другим заголовком, снова с другим заголовком — решаются элементарно.

А, у нас был стандартный заголовок, совсем нестандартный и стандартный, но чуть-чуть другой (без даты)? Значит нам нужна фабрика заголовков, которая по параметру будет возращать нужный.

**class** **Report8** **implements** **Report** {

HeaderFactory headers;

Body body;

DefaultFooter footer;

@Override

void **print**() {

ReportSection header = headers.create(WITHOUT\_DATE);

header.**print**();

body.**print**();

footer.**print**();

}

}

Главное, что композиция даёт нам полную гибкость. Очередной отчёт, какой бы странный он ни был (может заголовок и колонтитул надо местами поменять), никак не связан с предыдущими и нет риска их сломать.

Также хотелось бы обратить внимание, что код стал объекто-ориентированным header.print(), а не процедурным printHeader(), как раньше.

Резюме

Итак, нельзя сказать, что какие-то подходы правильные или неправильные, всё зависит от ситуации, но

1) если общая функциональность выносится в родительский класс;

2) появляются слова Base и abstract;

3) в наследниках переопределяются или используются методы родительского класса

... скорее всего, что-то пошло не так и проблемы не за горами.

Можно сформулировать даже проще. Наследование — один из базовых принципов ООП. Не используйте наследование! (имеется в виду наследование классов; дополнять интерфейсы можно).

Методом исключения получается, что «хорошее» наследование — это добавление новых методов, которые используют исключительно вновь добавленные поля этого класса, но никак не родительские методы.

Тут бы надо привести пример этого самого правильного наследования, но поскольку абсолютное большинство задач гораздо проще решаются композицией и наследованием интерфейсов(!), я, честно говоря, затрудняюсь это сделать.

Вернёмся к примеру с автомобилем. Допустим, есть класс Автомобиль с методами ездить(), сигналить() и т.д. Как бы могло выглядеть его наследование? Что значит дополнение автомобиля не с точки зрения программирования, а в обычном житейском понимании?

Предположим, мы хотим сделать Боевой Автомобиль, который умеет всё то же самое, что и обычный (отношение is-a), но кроме того на нём будет установлен пулемёт с методом стрелять().

Наследование? Выше рассмотрено, к чему это может привести, особенно, когда окажется, что есть и другие транспортные средства, и различное вооружение, и их всевозможные комбинации. Нет, Боевой Автомобиль — это всё таки композиция Автомобиля и Пулемёта с реализацией двух интерфейсов — Транспорт и Оружие.

Можно было бы обратиться к биологии с её, на первый взгляд, незыблемой иерархией классов, но даже там всякие рыбообразные дельфины и нелетающие пингвины подпортят концепцию :-)

Рекомендую также почитать статьи:  
— [Интерфейс vs. Классы](https://habrahabr.ru/post/30444/);  
— [Принцип подстановки Барбары Лисков](https://habrahabr.ru/post/83269/);  
— [Я не знаю ООП](https://habrahabr.ru/post/147927/).

**P.S.** Для усвоения материала напомню классическую задачу про наследование.

*Для геометрических фигур есть методы подсчёта периметра и площади. Рассмотрим прямоугольник со сторонами a и b, т. е. класс с двумя сеттерами. Его периметр определяется по формуле P = 2a + 2b, а площадь S = ab. Квадрат — это частный случай прямоугольника, его единственная характеристика — длина стороны a (достаточно одного сеттера), а формулы периметра и площади можно переиспользовать, полагая b = a.*

*Вопрос. Как, учитывая эти факты, построить иерархию классов: унаследовать Прямоугольник от Квадрата или Квадрат от Прямоугольника?*

Авторский ответ на задачу

**Теми:** [Java](https://dou.ua/lenta/tags/Java/), [композиция](https://dou.ua/lenta/tags/%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%B7%D0%B8%D1%86%D0%B8%D1%8F/), [наследование](https://dou.ua/lenta/tags/%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5/)

Facebook98

Twitter

**Найкращі коментарі**[пропустити](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/" \l "comments)

[**[https://s.dou.ua/img/avatars/40x40_6233.png](https://dou.ua/users/silverwolf/)Bogdan Shyiak**](https://dou.ua/users/silverwolf/)[04.08.2016 22:24](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965593)

Также хотелось бы обратить внимание, что код стал объекто-ориентированным header.print(), а не процедурным printHeader(), как раньше.

Эта фраза — квинтэссенция всей статьи: ООП — это когда выражения с точечкой. :)

ReportSection header = headers.create(WITHOUT\_DATE);

Главное, что композиция даёт нам полную гибкость.

Ну круто, супер гибкость — добавим вытягивание зависимости прямо в методе вывода. Своим опусом вы породили не меньше инвалидов от программирование чем университетские сказки про три кита.

Вопрос. Как, учитывая эти факты, построить иерархию классов: унаследовать Прямоугольник от Квадрата или Квадрат от Прямоугольника?

Просто экскурс в историю:  
Задача про прямоугольник и квадрат для демонстрации ООП появилась как часть задачи про графический редактор, а не про геометрические фигуры. И демонстрировала в основном трейдоф между производительность и переиспользованием кода.  
.  
P.S. Посыл про то что наследованием классов надо пользоваться очень аккуратно правильный, но это не отменяет всех тех глупостей которые есть в статье.

[**Перейти до дискусії**](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965593)

**152 коментарі**

[Підписатись на коментарі](javascript:;) Коментарі можуть залишати тільки користувачі з [підтвердженими аккаунтами](https://dou.ua/verified-accounts/" \t "_blank).

[**[https://s.dou.ua/img/avatars/40x40_default-user-avatar.jpg](https://dou.ua/users/maksim-r/)Maks Rybalchenko**](https://dou.ua/users/maksim-r/) супер пупер кодер[22.08.2016 23:37](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#976656)

Ну, що тут сказати, фанатичність (одержимість) якимось одним підходом (точкою зору і тд) це не є здорово, професіонал це той хто вміє вдало підібрати необхідний інструмент (принцип) для задачи. Ось так і з’являються хейтери статичних методів, наслідування, пхп і будь-чого ще.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[22.08.2016 23:52](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#976659)

Згоден з вами. Не зважайте, стаття написана дещо гротексно. Тим не менш, хочу зауважити, що у статичних методів, наслідування і може й в пхп (не знайомий, не можу дискусутувати) дійсно є вагомі недоліки. Ви слушно зауважили про професіоналізм й інструменти, але я, нажаль, чимало стикався з підходом «спільний код виносимо в базовий клас» і от, як вмію, висвітлюю альтернативу — композицію.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_default-user-avatar.jpg [Ще 1 коментар](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_145216.jpg](https://dou.ua/users/nikolai-ponomarchuck/)Nikolai Ponomarchuck**](https://dou.ua/users/nikolai-ponomarchuck/) Chief Executive Officer[09.08.2016 15:56](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#967817)

От себя могу добавить, что наследование это «конституция» приложения, а композиция (использование подклассво) это сопутствующие законы этого приложения. Они могут не пеесекаться, но обязаны четко следовать логике наследования методов. Абстрактные классы обижены зря — это статут того, как будут вести себя наследники (завещание). Интерфейсы — нотариусы. //и нет, я не юрист

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_anon.png](https://dou.ua/users/ivanonyshchenko/)Ivan Onyshchenko**](https://dou.ua/users/ivanonyshchenko/) SSE[09.08.2016 09:34](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#967563)

Возник ряд вопросов собственно по собеседованию:  
О собеседованиях на какую позицию/роль идет речь?  
С какой целью на собеседовании вы задаете вопросы об отличии абстрактного класса от интерфейса, наследования от композиции?   
С какой целью даете задание на проектирование?   
Как по ответам вы решаете, принять кандидата на работу или нет?   
Как и кем определяется правильность ответа или решения?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[09.08.2016 10:32](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#967586)

Спасибо за вопросы.

1. Middle java developer.  
2. Вводные вопросы для общей оценки знаний кандидата в ООП. Например, кандидат может не знать понятия интерфейса.  
3. Практически проверить понимание проектирования на простом примере.  
4. Решение принимает менеджер с учётом в том числе технического фидбека. Если кандидат даёт правильные ответы, фидбек позитивный.  
5. Если кандидат предлагает дизайн, позволяющий без копи-паста скомпоновать отчёты, ответ засчитывается. Мною. Если говорит, что не знает, как решить задачу, ответ не засчитывается.

Во избежание недосказанности, и в этом комментарии я повторюсь, что никаких вариантов ответа, особенно ложных, на собеседовании не предлагается. Единственный нюанс, который я считаю не подвохом, а вполне рабочей ситуацией, это то, что меняются требования. И опять же, абсолютно допустимо пересмотреть дизайн с самого начала.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_6233.png https://s.dou.ua/img/avatars/25x25_anon.png https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg [Ще 7 коментарів](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_profile-edited.jpg](https://dou.ua/users/vitaliy-yakovchuk/)Vitaliy Yakovchuk**](https://dou.ua/users/vitaliy-yakovchuk/) Senior Backend Developer в [MyHeritage](https://jobs.dou.ua/companies/myheritage/)[08.08.2016 18:14](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#967278)

В чём отличие между композицией и наследованием?

Вопрос из разряда почему люки круглые или как измерить расстояние с помощью градусника. А в чем отличие между теплым и мягким?

Также хотелось бы обратить внимание, что код стал объекто-ориентированным header.print(), а не процедурным printHeader(), как раньше.

И композиция и наследование это часть концепции ООП.

Вопрос. Как, учитывая эти факты, построить иерархию классов: унаследовать Прямоугольник от Квадрата или Квадрат от Прямоугольника?

Это Вы лихо придумали: задать вопрос с двумя неправильными вариантами ответа. Как и в версии «В чём отличие между композицией и наследованием?» вопрос задан не верно.

Можно сформулировать даже проще. Наследование — один из базовых принципов ООП. Не используйте наследование! (имеется в виду наследование классов; дополнять интерфейсы можно).

Я чуть смузи не подавился прочитав такие выводы :) Думаю проблема в том что как раз нужно понять композиция и наследования вещи разные, а не выпиливать одно в угоду другого.

Конкретно для вашего примера таки да не подходит тот вариант наследования, но зачем плодить разные класы для

DefaultHeader

и

DefaultFooter

, создали бы один интерфейс для хранения списка компонентов внутри отчета и хранили бы там вашы Header, Footer, Appendix, Body и все что душа пожелает.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[08.08.2016 20:32](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#967352)

Вопрос из разряда почему люки круглые или как измерить расстояние с помощью градусника. А в чем отличие между теплым и мягким?

Извините, вы, по-моему, разные вещи в одну кучу смешали.

Вопрос по люки призывает задуматься о привычных вещах и имеет чёткий инженерный ответ.  
Измерение расстояния градусником — как я понимаю, о нетипичном использовании инструмента.  
Тёплое и мягкое — классический пример несравнимых категорий.

По вашему, композиция и наследование — это тоже несравнимые категории? На мой взгляд обе эти техники позволяют переиспользовать код и, поэтому, могут противопоставляться

Это Вы лихо придумали: задать вопрос с двумя неправильными вариантами ответа. Как и в версии «В чём отличие между композицией и наследованием?» вопрос задан не верно.

Не буду спорить, вопрос задан некоректно, с подвохом и ложной дихотомией. Но это же статья, а не собеседование. Не разделяю возмущения :-) Если что, на собеседовании вопросы в таком стиле я не задаю.

Думаю проблема в том что как раз нужно понять композиция и наследования вещи разные, а не выпиливать одно в угоду другого.

Конечно. Вот я и пытаюсь, как умею, это и объяснить.

Конкретно для вашего примера таки да не подходит тот вариант наследования

Рад, что для вас это очевидно. Если бы я не сталкивался с противоположными примерами, вряд ли бы написал эту статью.

но зачем плодить разные класы для DefaultHeader и DefaultFooter, создали бы один интерфейс для хранения списка компонентов внутри отчета и хранили бы там вашы Header, Footer, Appendix, Body и все что душа пожелает.

Вроде как именно так и написано. Разные блоки наверное будут реализовывать один интерфейс. Тогда их действительно можно хранить в списке. Всё же, главный посыл был о неприменимости наследования. Как строить композицию — тоже важный вопрос, но выходит за рамки статьи.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_profile-edited.jpg [Ще 1 коментар](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_57196.jpg](https://dou.ua/users/aliaksandr-valialkin/)Aliaksandr Valialkin**](https://dou.ua/users/aliaksandr-valialkin/) программист[08.08.2016 01:05](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#966688)

Вот [еще одна статья про вред наследования в ООП](https://m.habrahabr.ru/company/mailru/blog/307168/).

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_85453.jpg](https://dou.ua/users/igor-lytvynenko/)Igor Lytvynenko**](https://dou.ua/users/igor-lytvynenko/) Java Developer[06.08.2016 07:56](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#966216)

Ось вам приклад на наслідування:  
Згадайте класичний Composite в системах, де треба робити формочки, кнопочки.  
Там зазвичай класи Node, Group, Button, Panel, ... наслідуються один від одного.  
І тут не обійтися лише одними інтерфейсами. Та ж сама Node не може обійтися без стану, скажем, без X та Y.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[06.08.2016 19:22](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#966321)

Дякую за приклад. Згода, є системи об’єктів, які утворюють настільки чітку ієрархію, і до дедублювання кода яких дійсно можна застусування наслідування. Але вони радше виняток.

І тут не обійтися лише одними інтерфейсами.

Ось тут не певен.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_57196.jpg](https://dou.ua/users/aliaksandr-valialkin/)Aliaksandr Valialkin**](https://dou.ua/users/aliaksandr-valialkin/) программист[05.08.2016 23:08](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#966163)

Хорошая статья. Для тех, кто начал понимать бредовость наследования, есть отличная замена java — go. Там [есть только композиция](https://golang.org/doc/faq#inheritance) плюс расширение интерфейсов. Также в go [не нужно перечислять многочисленные интерфейсы, реализуемые классом](https://golang.org/doc/faq#implements_interface).

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_me21_moUEx5F.jpg](https://dou.ua/users/sergey-morenets/)Sergey Morenets**](https://dou.ua/users/sergey-morenets/) Java evangelist[07.08.2016 12:54](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#966452)

А что же делать, если у меня у классов общие поля/методы ?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_82427.jpg https://s.dou.ua/img/avatars/25x25_me21_moUEx5F.jpg https://s.dou.ua/img/avatars/25x25_57196.jpg [Ще 6 коментарів](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_3147.jpg](https://dou.ua/users/Ultras/)Serhiy Serbin**](https://dou.ua/users/Ultras/) [05.08.2016 14:00](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965943)

Так получилось, что эти вопросы я задал довольно большому количеству людей на собеседованиях. И, как мне кажется, есть определённое непонимание этих базовых концепций, вернее, расхождение между теорией и практикой. Данная статья призвана внести ясность и улучшить мир код.

Да перестаньте просто эти бесполезные вопросы задавать на собеседовании.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[05.08.2016 14:04](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965948)

Да перестаньте просто эти бесполезные вопросы задавать на собеседовании.

Предложите, пожалуйста, свой вариант вопроса или задачи, позволяющий оценить понимание кандидатом базовых вещей ООП.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_CHU106.jpg https://s.dou.ua/img/avatars/25x25_3147.jpg https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg [Ще 9 коментарів](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_96b14aac31b3bc04f2455a1a31fc2c44.jpg](https://dou.ua/users/vitalik-r/)Vitalii Rastvorov**](https://dou.ua/users/vitalik-r/) Software Engineer в [SPD-Ukraine](https://jobs.dou.ua/companies/spd-ukraine/)[05.08.2016 13:37](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965930)

я ещё только учусь, не судите строго

Композиция vs Наследование в Java

Luxoft — Java Tech Lead

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_MZEyNx2TyPQ.jpg](https://dou.ua/users/rinat-veliahmedov/)Ринат Велиахмедов**](https://dou.ua/users/rinat-veliahmedov/) C++ Developer – GlobalLogic[05.08.2016 12:50](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965890)

Нам достаточно интерфейса Фигура и класса Прямоугольник

Дать задачу с двумя вариантами ответа. Оба из который неправильные. Аплодисменты.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[05.08.2016 12:59](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965897)

Извините, Ринат. Провокативно, не отрицаю. Но разве на работе и в жизни не бывает таких ситуаций, когда нужно выходить за рамки поставленных условий?

[ru.wikipedia.org/wiki/Ложная\_дилемма](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B6%D0%BD%D0%B0%D1%8F_%D0%B4%D0%B8%D0%BB%D0%B5%D0%BC%D0%BC%D0%B0)

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg https://s.dou.ua/img/avatars/25x25_MZEyNx2TyPQ.jpg [Ще 2 коментарі](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/25x25_CHU106.jpg](https://dou.ua/users/oleksandr-shpak/)Олександр Шпак**](https://dou.ua/users/oleksandr-shpak/) Oarsman[05.08.2016 14:02](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965945)

А правильна відповідь яка?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_CHU106.jpg https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg [Ще 2 коментарі](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_acid-house.jpg](https://dou.ua/users/andrey-bobrov-1/)Andrey Bobrov**](https://dou.ua/users/andrey-bobrov-1/) [05.08.2016 07:12](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965704)

а как же принцип never repeat yourself? ваш код выглядит как сплошной копипаст, однако, он дает долю flexibility, но количество кода растет пропорционально   
кстати, репорт не должен сам себя генерировать/печатать — в этом изначальная ошибка вашего подхода

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[05.08.2016 10:30](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965759)

а как же принцип never repeat yourself? ваш код выглядит как сплошной копипаст, однако, он дает долю flexibility, но количество кода растет пропорционально

DRY — это хороший принцип. Но в приведённом примере DRY с помощью наследования не справился с изменившимися требованиями. Поэтому предлагается перейти к композиции и, если необходимо, снова вспомнить о DRY, но уже другими способами.

кстати, репорт не должен сам себя генерировать/печатать — в этом изначальная ошибка вашего подхода

С одной стороны, это просто пример вынесения общего кода в базовый класс, может не самый удачный, но я к нему привык. В комментариях на это указывали, что должен быть ReportPrinter.  
С другой — как вам такое мнение [www.yegor256.com/.../objects-end-with-er.html](http://www.yegor256.com/2015/03/09/objects-end-with-er.html) ?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_59133.jpg https://s.dou.ua/img/avatars/25x25_acid-house.jpg [Ще 3 коментарі](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[05.08.2016 00:10](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965656)

Спасибо за обсуждение задачи, в конце статьи добавлен авторский ответ

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_6233.png](https://dou.ua/users/silverwolf/)Bogdan Shyiak**](https://dou.ua/users/silverwolf/) SE or Dev[05.08.2016 00:30](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965668)

в конце статьи добавлен авторский ответ

Ответ правильный, но он не на заданый вопрос.  
От отвечает на задачу:  
Опишите объектную структуру необходимую для расчета периметра и площади квадрата и прямоугольника.

Как, учитывая эти факты, построить иерархию классов: унаследовать Прямоугольник от Квадрата или Квадрат от Прямоугольника?

Ваш вопрос явно требует «построить иерархию классов», при этом еще и, по факту, не посторить, а выбрать одну из предложенных.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_6233.png https://s.dou.ua/img/avatars/25x25_56246_EZgrBGV.jpg https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg [Ще 9 коментарів](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[[https://s.dou.ua/img/avatars/40x40_56246_EZgrBGV.jpg](https://dou.ua/users/andrej-chubarev/)And riy](https://dou.ua/users/andrej-chubarev/)[04.08.2016 23:37](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965640)

Такое чувство, что люксофт платит бонусы за статьи на доу. Просто не знаю, что еще может побудить накатывать столь унылые опусы.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[05.08.2016 00:00](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965653)

Может, раз такое чувство, подскажите к кому конкретно в Люксофте за деньгами обратиться?

Побудили собеседования, я же написал об этом.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_IMG_9202.JPG](https://dou.ua/users/karpolan/)Anton Karpenko**](https://dou.ua/users/karpolan/) I am[04.08.2016 23:04](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965614)

Отличный бред... Посмотри любую библиотеку компонентов/контролов и убейсибяапстену © :)

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 23:21](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965628)

Давайте, пока я не добежал до стены и ещё жив, уточним: мы про библиотеку на каком языке? Есть ли в нём понятие интерфейсов? Если что, я ещё раз подчеркну, что расширять интерфейсы можно.

Я не утверждаю, что наследование категорически неприменимо. Тот же принцип Лисков не отрицает наследование, а лишь позволяет проверить, не ошибаемся ли мы сходу.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_Oleg_Kariakin.jpg](https://dou.ua/users/oleg-kariakin/)Oleg Kariakin**](https://dou.ua/users/oleg-kariakin/) Senior Java Developer[04.08.2016 22:54](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965606)

Вот авторитетное мнение, эксперта что наследование — г#вно. И я на 146% согласен. Когда в примерах идет речь о кружочках, фигурках и прочих, оторванных от реальности сущностей, это хорошая концепция.   
[youtu.be/G6LJkWwZGuc?t=587](https://youtu.be/G6LJkWwZGuc?t=587)  
смотреть с 9:47

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_6233.png](https://dou.ua/users/silverwolf/)Bogdan Shyiak**](https://dou.ua/users/silverwolf/) SE or Dev[04.08.2016 22:24](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965593)

Также хотелось бы обратить внимание, что код стал объекто-ориентированным header.print(), а не процедурным printHeader(), как раньше.

Эта фраза — квинтэссенция всей статьи: ООП — это когда выражения с точечкой. :)

ReportSection header = headers.create(WITHOUT\_DATE);

Главное, что композиция даёт нам полную гибкость.

Ну круто, супер гибкость — добавим вытягивание зависимости прямо в методе вывода. Своим опусом вы породили не меньше инвалидов от программирование чем университетские сказки про три кита.

Вопрос. Как, учитывая эти факты, построить иерархию классов: унаследовать Прямоугольник от Квадрата или Квадрат от Прямоугольника?

Просто экскурс в историю:  
Задача про прямоугольник и квадрат для демонстрации ООП появилась как часть задачи про графический редактор, а не про геометрические фигуры. И демонстрировала в основном трейдоф между производительность и переиспользованием кода.  
.  
P.S. Посыл про то что наследованием классов надо пользоваться очень аккуратно правильный, но это не отменяет всех тех глупостей которые есть в статье.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 22:47](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965601)

Богдан, я ещё только учусь, не судите строго. Как вы считатете, будет ли полезно читателям и мне, если вы не только укажете на глупости, но и исправите их?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_6233.png https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg https://s.dou.ua/img/avatars/25x25_Vol.jpg [Ще 7 коментарів](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[05.08.2016 23:27](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#966179)

Попробую ответить ещё раз.

Эта фраза — квинтэссенция всей статьи: ООП — это когда выражения с точечкой. :)

Я не утверждал, что ООП — это когда выражения с точечкой. Вы приписали мне утверждение и высмеяли его. Кажется, это приём Pugna.

Ну круто, супер гибкость — добавим вытягивание зависимости прямо в методе вывода.

Целью статьи было показать ограничения применения наследования на примере Template Pattern. Как организовать композицию — это следующий вопрос.

Своим опусом вы породили не меньше инвалидов от программирование чем университетские сказки про три кита.

Сильно сомневаюсь, что вы можете подтвердить это утверждение ссылкой на статистику инвалидов. Скорее, это всё же ваше субъективное мнение.

Просто экскурс в историю:Задача про прямоугольник и квадрат для демонстрации ООП появилась как часть задачи про графический редактор, а не про геометрические фигуры.

Познавательно. А я по её мотивам хотел напомнить, что отношение is-a реальных объектов не означает наследования.

P.S. Посыл про то что наследованием классов надо пользоваться очень аккуратно правильный

Это единственный посыл статьи. Рад, что тут мы сходимся во мнениях и, как я понимаю, дискутируем о том, на каком примере его можно показать нагляднее?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_100x100_111398_SkkaU5q.jpg](https://dou.ua/users/anton-hristiansen/)Anton Khristiansen**](https://dou.ua/users/anton-hristiansen/) web back end developer[04.08.2016 18:06](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965510)

И еще, по поводу вынесения какого — то основного функционала в базовый класс. Такой вопрос, и я буду благодарен если вы меня поправите \ дополните.

Есть цсв файлы, содержащие инфу про товар в БД — сео теги, описание, тайтл, цену, и т д . Есть ряд файлов, отличающийся структурой, и задачами — один содержит всю инфу, для большинства товаров и ведется уже лет 7 и весит гигабайт 2, второй заведен для прописывания описания в товарах, что недавно появились, третий имеет лишь часть полей первого документа и имеет доп. фильтрацию инфы.

Соответственно, разумно под каждый файл запилить по одному классу. Сразу отметим, что один метод у них будет общий — принять ссылку на файл, прочитать его, проверяя, были ли изменены строки по сравнению с пред. чтением, нет ли строк дублей по отношению к одному и тому же айтему в бд, и по окончанию процесса, снести файл. Чем плохо вынести этот метод в отдельный класс, котгорый будут наследовать остальные, с учетом что он будет для всех классов один и то же — читать, фильтровать файл, передавая дату в другой метод? Да, проверка на изменение строк может быть разной ( хотя думаю тут будет целесообразно брать хеш строки и хранить в бд вместе с типом файла), и перегружаться в наследниках.. Буду рад если вы предложите лучший вариант ( все выдумал из головы, не думайте что спрашиваю аля для рабочей задачи :) )

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 22:19](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965591)

Антон, не воспринимайте, пожалуйста как личный выпад, но вся эта статья призвана бороться с позицией

«Я, конечно, знаю, что джунам запрещают наследование. И на ДОУ вот недавно опять писали. Но, во-первых, я далеко не джун, а, во-вторых, у этих классов есть общий код. Надо вынести его в базовый класс!!!»

Не надо. Лучше сделать отдельный класс, который может обрабатывать СSV — читать, фильтровать и т.д. В итоге возвращать данные. И использовать (а не наследовать), создавая экземпляр этого класса в других. Первый выигрыш от такого подхода — его можно замокать для тестов.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_100x100_111398_SkkaU5q.jpg https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg [Ще 2 коментарі](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_100x100_111398_SkkaU5q.jpg](https://dou.ua/users/anton-hristiansen/)Anton Khristiansen**](https://dou.ua/users/anton-hristiansen/) web back end developer[04.08.2016 17:56](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965503)

Спасибо за статью, на фоне соседних домашних посиделок на кухне — как глоток свежего воздуха. По поводу вашего первого примера, согласно которому так делать не нужно, и почему большинство кандидатов решают эту задачу именно таким способом — потому что перед ними поставлена конкретная задача, которую нужно решить. И в данном контексте, предложенный вариант полностью решает поставленную задачу. Использование более гибкого подхода, а именно композиции, имхо, будет предложено, когда  
а). Вы явно укажете в задаче, что архитектура этого модуля будет дополняться \ изменяться весьма извилистыми путями, так как заказчик str\_replace(’м’, ’ч’, ’чудак’) - хотя это сплошь и рядом, и при планировании это надо учесть.  
б). Тот, кому задали задачу при собеседовании, сам, исходя из своего опыта, схватится, и предложит композицию наследованию, хотя это должен быть человек либо который постоянно работал в коллективе, у которого SOLID и чистота \ крепкость архитектуры на первом месте, либо же это серьезный сеньор, с действительно солидным опытом, который не сидел на легаси проектах.

А так — если в рамках собеса и тем более в рамках работы, если задача поставлена

Каждый отчёт состоит из трёх частей — заголовка (header), собственно тела отчёта (body) и колонтитула (footer). Формируют их некие методы. **Формирование header и footer для всех отчётов одинаково и меняться не будет**.

то поверьте, в рамках поставленных дедлайнов, 70% сделают как в первом варианте, а второй сделают, отрефакторив первый, когда придет соотв. задача, это требующая, хотя это и нарушает принцип расширения, а не изменения :)

ПЫ СЫ :) Пишите Еще, подобные статьи весьма интересны и полезны!

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 22:08](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965590)

Спасибо, такой комментарий мотивирует писать ещё.

Касательно собеседования:

а). Заказчик платит деньги и может менять требования. Это, считаю, нормально. Явно делать акцент — дать подсказку.  
б). Вот кандидатов, которые «спохватятся и предложат композицию» я и стараюсь брать. И, хотелось чтобы разницу между наследованием и композицией понимали не только серьёзные сеньоры, а, извините, все.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_6233.png](https://dou.ua/users/silverwolf/)Bogdan Shyiak**](https://dou.ua/users/silverwolf/) SE or Dev[04.08.2016 22:27](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965596)

ПЫ СЫ :) Пишите Еще, подобные статьи весьма интересны и полезны!

Интересны — может быть, явно не хуже чем очередной высер про ХРов или ЗП.  
Но вот явно не полезны:  
— люди, которые имеют представление про ООП, скорее посмеются или разозлятся очередному бреду;  
— а вот людей без опыта и знаний, такая статья только подтолкнет к «инвалидности» (не хуже большинства универских программ по ООП)

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_88712.jpg](https://dou.ua/users/victoria-globa/)Victoria Globa**](https://dou.ua/users/victoria-globa/) Software Architect[04.08.2016 16:33](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965421)

унаследовать Прямоугольник от Квадрата или Квадрат от Прямоугольника

Это задача вообще не на наследование, и не на ООП. Это пример, демонстрирующий, что близость ООП к реальному миру — условна, и оно не призвано зеркалить в коде реальный мир, а призвано описывать взаимосвязи между абстракциями конкретной программной системы, для удобства и эффективности этой самой системы. Текстовка к этой задаче идет типа: вот смотрите, в реальной жизни квадрат «is a» прямоугольник, а в программе такая реализация будет избыточной.

Дополнительно не могу не отметить, что показывать различие композиции и наследования примером про машинки-пулеметики — это днище какое-то. Вы читали вообще, как сама Лисков демонстрировала Liskov Substitution Principle, какими словами он сформулирован? Это о программировании, а не о машинках. Кого от чего мы там отнаследуем и отнаследуем ли, зависит только от того, какие именно аспекты пулемета и машины нам нужны в нашей программе. Может никакие, а только строка с названием модели. Вот потом дети начитаются таких примеров и бред на собеседованиях несут, и написать что-то простейшее неспособны.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 16:52](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965454)

Конечно там не нужно наследование! Элементарный подвох на усвоение материала.

Я, собственно, два раза и пытался сказать, что іs-a ни разу не означает наследование.

С ним, на мой взгляд, не только у детей проблема. Повторяюсь, я затрудняюсь привести пример правильного. Может вы?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_88712.jpg https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg [Ще 7 коментарів](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_82427.jpg](https://dou.ua/users/serge-kukharev/)Сергей Кухарев**](https://dou.ua/users/serge-kukharev/) Team Lead в [Carmudi GmbH](http://jobs.dou.ua/companies/carmudi-gmbh/" \t "_blank)[04.08.2016 16:30](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965417)

Увы не тянет на хорошую статью.

— Имя класса крайне неудачное. Это скорее ReportPrinter.   
— Композиция превыше наследование. Я лично считаю что джунам вообще надо запрещать наследовать, пока не научатся правильно составлять объекты. Поэтому с утверждением что

С композицией всё просто.

крайне не согласен, это то место которое большинство девелоперов понимают очень плохо, а зря. Вообще в конце статьи правильные выводы сделаны, но как-то сумбурно и недостаточно акцентов на правильных вещах.  
—

если общая функциональность выносится в родительский класс

общая функциональность в 95 случаев должа выноситься в отдельный тип, т.к. это скорее всего нарушение SOLID в том или ином виде. Опять же — наследование хуже композиции почти во всех случаях, за исключением «шаблонных».  
— Насчет квадратов-прямоугольников — оба типа будут implements Figure. Никакого наследования. Тот факт, что у них есть занимательные геометрические сходства не означает что квадрат может замещаться только прямоугольником, скорее любая геометрическая фигура может замещаться другой. Когда думаете о наследование, думайте в первую очередь о полиморфизме. Худшее что может сделать разработичк — ввести иерархию классов только ради устранение дублирования кода.  
— На кой черт в геометрических фигурах, которые типичные ValueObject, сеттеры?

Вообщем, правильные выводы в статье не видно за слегка сумбурным текстом. У меня в целом те же проблемы с изложением мыслей, поэтому и пишу Вам.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 16:47](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965448)

Спасибо за коментарий. Это моя первая статья, стараюсь как могу.

Я тоже считаю, что джунам нужно запретить наследование и именно это излагаю как умею.  
Про квадрат и сеттеры — естественно. И достаточно одного типа. Уже два коментария с правильным ответом, боюсь интрига раскрыта.

Пример правильного наследования сможете подсказать?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_82427.jpg [Ще 1 коментар](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/25x25_151906.jpg](https://dou.ua/users/andrej-popov-2/)Andrii Popov**](https://dou.ua/users/andrej-popov-2/) [04.08.2016 18:16](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965517)

Худшее что может сделать разработчик — ввести иерархию классов только ради устранение дублирования кода.

. А для чего правильно вводить иерархию объектов? Если можно поподробнее. Спасибо.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_82427.jpg https://s.dou.ua/img/avatars/25x25_15032270_207028116404678_7812941592273916947_n.jpg [Ще 2 коментарі](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/25x25_15032270_207028116404678_7812941592273916947_n.jpg](https://dou.ua/users/evgeniykhist/)Evgeniy Khyst**](https://dou.ua/users/evgeniykhist/) Java Solution Architect[04.08.2016 22:36](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965599)

Я лично считаю что джунам вообще надо запрещать наследовать

А когда разрешать наследование? Что будет признаком, что разработчику можно разрешить использовать наследование? На какие вопросы и что должны ответить разработчики, чтобы им разрешили использовать наследование?

На кой черт в геометрических фигурах, которые типичные ValueObject, сеттеры?

Этот пример фигурировал еще в статье про [The Liskov Substitution Principle](http://docs.google.com/a/cleancoder.com/viewer?a=v&pid=explorer&chrome=true&srcid=0BwhCYaYDn8EgNzAzZjA5ZmItNjU3NS00MzQ5LTkwYjMtMDJhNDU5ZTM0MTlh&hl=en" \t "_blank) за автором [Robert C. Martin](http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod" \t "_blank), который и описал SOLID принципы в начале 2000-х.

Также в книге Domain-Driven Design: Tackling Complexity in the Heart of Software автор Eric Evans пишет, что часто удобно иметь immutable Value Objects, но я нигде не нашел утверждения, что неизменяемость — обязательное условия для Value Object. Статья про Value Object на [Wikipedia](https://en.wikipedia.org/wiki/Value_object) говорит тоже самое:

*It is also useful for value objects to be immutable*, as client code cannot put the value object in an invalid state or introduce buggy behaviour after instantiation.

Но не вижу причин, почему в отдельных случаях объект-значение не может быть изменяемым.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg [Ще 1 коментар](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_CHU106.jpg](https://dou.ua/users/oleksandr-shpak/)Олександр Шпак**](https://dou.ua/users/oleksandr-shpak/) Oarsman[04.08.2016 15:46](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965362)

Каждый отчёт состоит из трёх частей — заголовка (header)

Далі можна не читати. Скільки можна орієнтуватися на візуальщину?  
Звіт візуально складається із чого завгодно. Правильно розділити цю функціональність на дві. Перша частина отримує фільтр та повертає дані в xml або json форматі, а інша вже перетворює ці дані на якийсь контент (темплейтування). Тому базовий абстрактний клас звіта буде складатися лише із одного метода (який можна й в конструктор засунути) — повернути дані по фільтру. Все, ніяких футерів, хедерів та іншого.  
Декомпозицію треба вчитися робити до спадкування. Це основа сталої системи.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 15:56](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965372)

Такий у нас візуальний ентерпрайз :-) Я вже звик до цього прикладу на співбесідах, але він не про темплейти. Хай буде замість звіта — будинок, header-a — дах, footer-a — фундамент. Стаття не зміниться.

Декомпозицію треба вчитися робити до спадкування.

От я і намагаюсь донести це.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_CHU106.jpg https://s.dou.ua/img/avatars/25x25_49801.png https://s.dou.ua/img/avatars/25x25_42625.png [Ще 9 коментарів](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/40x40_151906.jpg](https://dou.ua/users/andrej-popov-2/)Andrii Popov**](https://dou.ua/users/andrej-popov-2/) [04.08.2016 14:52](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965282)

Расскажите, как выбранный Вами подход разрулит ситуацию, когда в документе (вдруг внезапно) появились другие составные части, например логотип, печать, поля с отметками и прочее. Или, далее, появляются документы без footer или без header или без и того и того. А потом появляются документы с любым из 100+ составных частей и эти составные части могут бесконечно добавляться. И всё это нужно продумать чтобы ни в коем случае не противоречить принципам SOLID.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 15:00](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965289)

Так я вроде об этом и написал. Композиция. Новый отчёт (или модификация старого) никак не связан с предыдущими. Если существующие части (header, footer и т.д.) подходят — инжектим и используем их, если нет — дописываем нужные. Вы статью до конца прочитали?

[Відповісти](javascript:;)

[Підтримати](javascript:;)

https://s.dou.ua/img/avatars/25x25_CHU106.jpg https://s.dou.ua/img/avatars/25x25_88712.jpg https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg [Ще 7 коментарів](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/)

[**[https://s.dou.ua/img/avatars/25x25_MZEyNx2TyPQ.jpg](https://dou.ua/users/rinat-veliahmedov/)Ринат Велиахмедов**](https://dou.ua/users/rinat-veliahmedov/) C++ Developer – GlobalLogic[04.08.2016 15:03](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965293)

Думаю, такой подход как раз разрулит нормально. Сделать composite с одим абстрактным методом — напечать себя. Ну а его наследники пусть уже печатают что хотят, в том числе и другие вложенные элементы.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/40x40_50860.png](https://dou.ua/users/alexanderzvonov/)Alexander Zvonov**](https://dou.ua/users/alexanderzvonov/) Test Lead в [Luxoft](https://jobs.dou.ua/companies/luxoft/)[04.08.2016 13:46](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965212)

У вас там бага в предпоследнем куске кода. Печатается двухголовый репорт. Заказчик шоке, но сказал что оригинальность оценил.

[Відповісти](javascript:;)

[Підтримати](javascript:;)

[**[https://s.dou.ua/img/avatars/25x25_32_o8McEnA.jpg](https://dou.ua/users/andriy-slobodyanyk/)Andriy Slobodyanyk**](https://dou.ua/users/andriy-slobodyanyk/) Java Developer[04.08.2016 13:51](https://dou.ua/lenta/articles/composition-vs-inheritance-in-java/#965215)

Глаз — алмаз!