



99.95

Рейтинг

Productivity Inside

Для старательного нет ничего невозможного



ProductivityInside

5 июн 2020 в 10:26

Принципы SOLID в картинках



4 мин



266K

Блог компании Productivity Inside, Совершенный код*

[Перевод](#)

Автор оригинала: Ugonna Thelma



Если вы знакомы с объектно-ориентированным программированием, то наверняка слышали и о принципах SOLID. Эти пять правил разработки ПО задают траекторию, по которой нужно следовать, когда пишешь программы, чтобы их проще было масштабировать и поддерживать. Они получили известность благодаря программисту Роберту Мартину.

В Сети множество отличных статей, где рассказывается о принципах SOLID, но иллюстрированных среди них мне практически не попадалось. Из-за этого таким людям со склонностью к визуальному восприятию информации – таким, как я – бывает сложно схватывать суть и не отвлекаться.

Основная цель этой статьи – лучше усвоить принципы SOLID через отрисовку иллюстраций, а также определить назначение каждого принципа. Дело в том, что некоторые из принципов кажутся похожими, но функции выполняют разные. Может получиться так, что одному принципу следуешь, а другой при этом нарушаешь, хотя с виду особой разницы между ними нет.

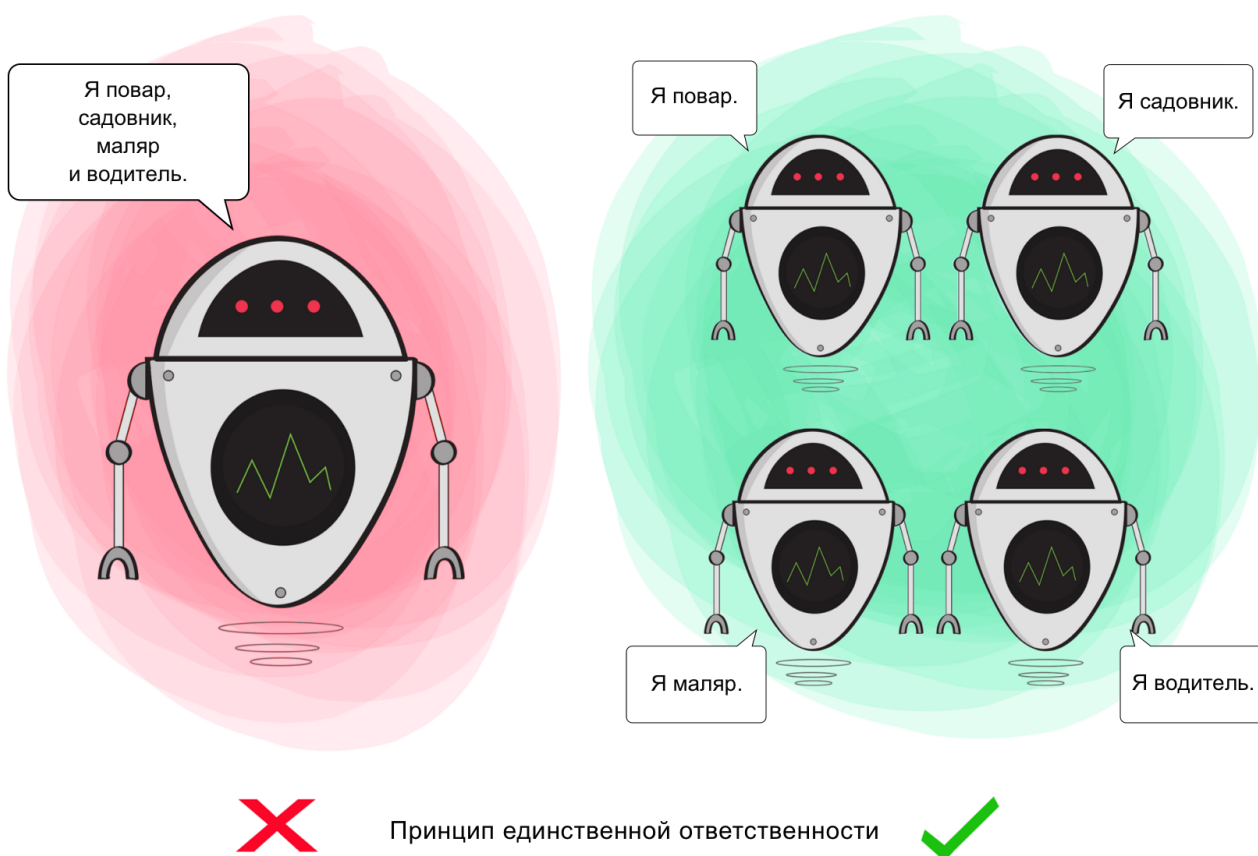
Чтобы проще читалось, я упоминаю здесь только классы, однако всё сказанное в статье применимо также к функциям, методам и модулям, так что имейте это в виду.

Ну, приступим.

Принципы SOLID

S – Single Responsibility (Принцип единственной ответственности)

Каждый класс должен отвечать только за одну операцию.



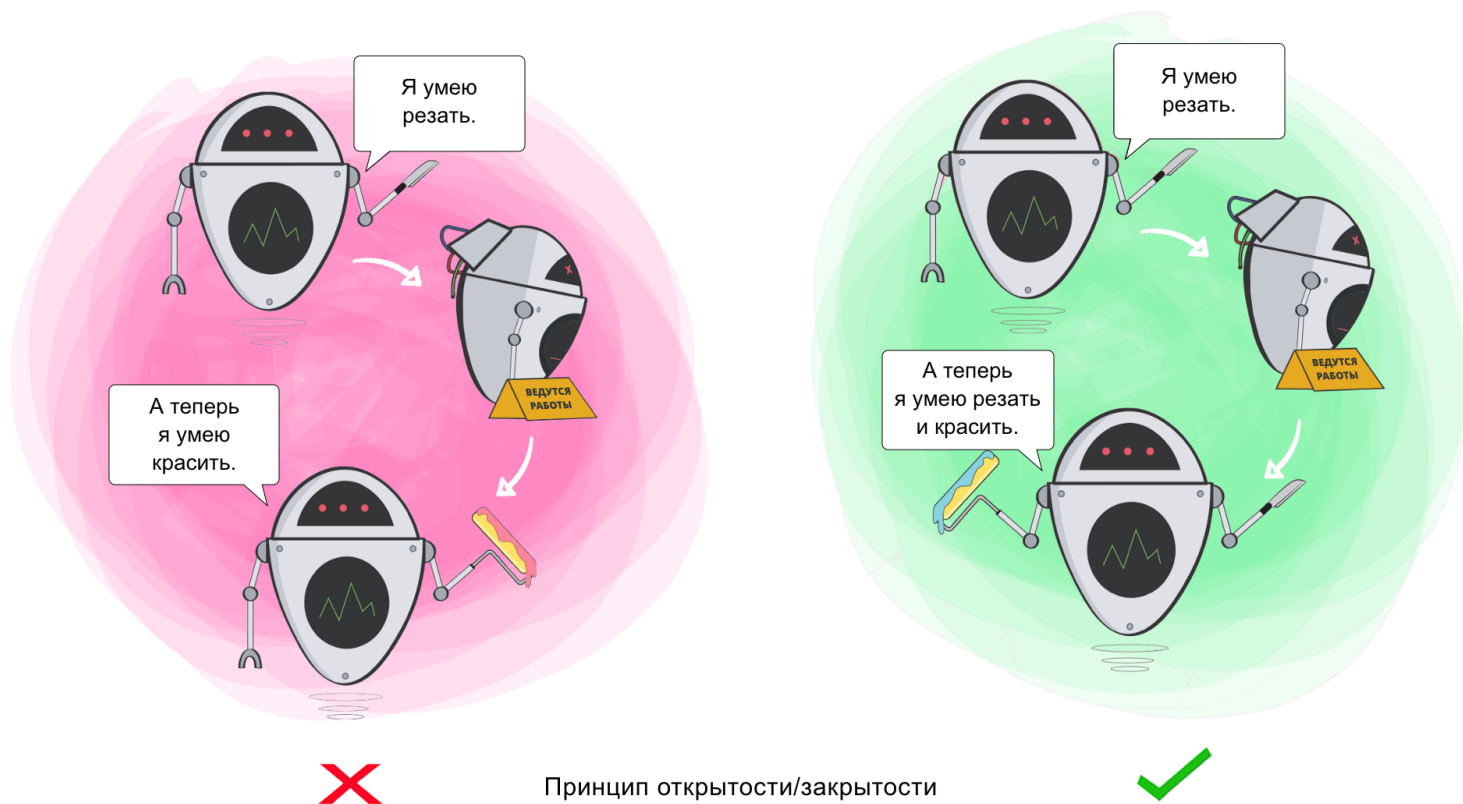
Если класс отвечает за несколько операций сразу, вероятность возникновения багов возрастает — внося изменения, касающиеся одной из операций вы, сами того не подозревая, можете затронуть и другие.

Назначение

Принцип служит для разделения типов поведения, благодаря которому ошибки, вызванные модификациями в одном поведении, не распространялись на прочие, не связанные с ним типы.

O — Open-Closed (Принцип открытости-закрытости)

Классы должны быть открыты для расширения, но закрыты для модификации.



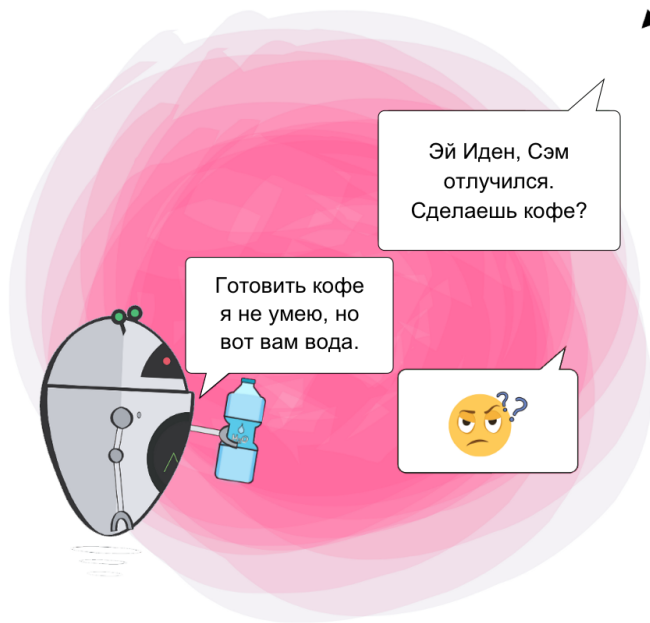
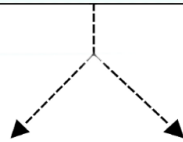
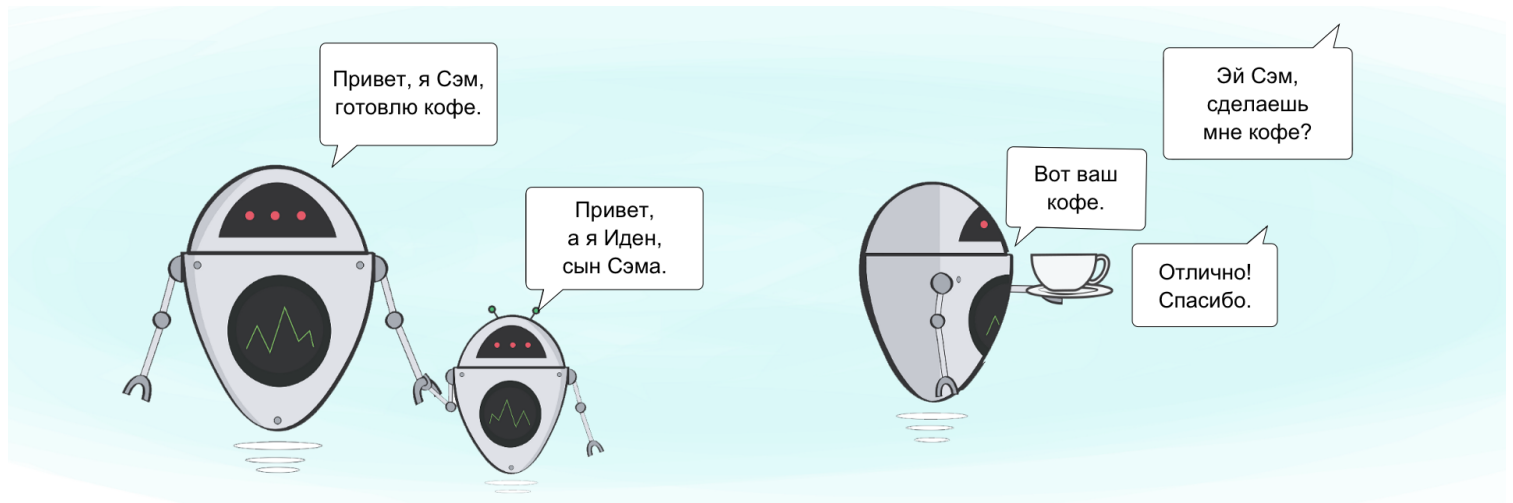
Когда вы меняете текущее поведение класса, эти изменения сказываются на всех системах, работающих с данным классом. Если хотите, чтобы класс выполнял больше операций, то идеальный вариант – не заменять старые на новые, а добавлять новые к уже существующим.

Назначение

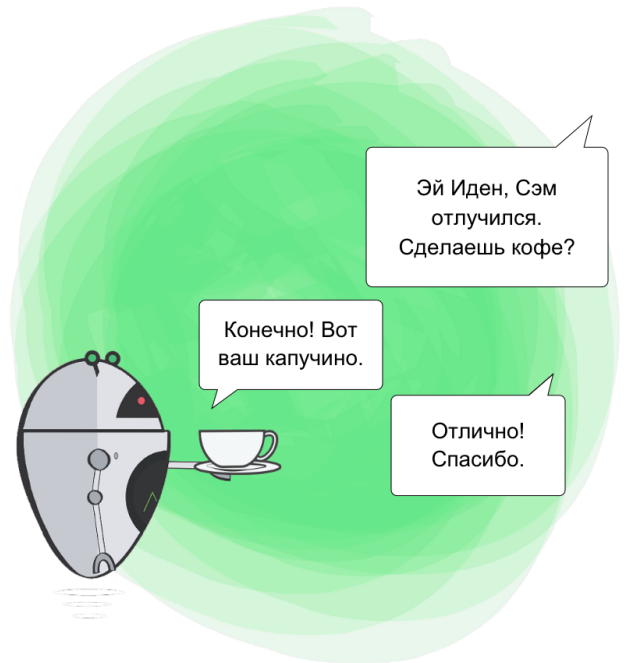
Принцип служит для того, чтобы делать поведение класса более разнообразным, не вмешиваясь в текущие операции, которые он выполняет. Благодаря этому вы избегаете ошибок в тех фрагментах кода, где задействован этот класс.

L — Liskov Substitution (Принцип подстановки Барбары Лисков)

Если P является подтипом T , то любые объекты типа T , присутствующие в программе, могут заменяться объектами типа P без негативных последствий для функциональности программы.



Принцип подстановки Барбары Лисков



В случаях, когда класс-потомок не способен выполнять те же действия, что и класс-родитель, возникает риск появления ошибок.

Если у вас имеется класс и вы создаете на его базе другой класс, исходный класс становится родителем, а новый – его потомком. Класс-потомок должен производить такие же операции, как и класс-родитель. Это называется наследственностью.

Необходимо, чтобы класс-потомок был способен обрабатывать те же запросы, что и родитель, и выдавать тот же результат. Или же результат может отличаться, но при этом относиться к тому же типу. На картинке это показано так: класс-родитель подаёт кофе (в любых видах), значит, для класса-потомка приемлемо подавать капучино (разновидность кофе), но неприемлемо подавать воду.

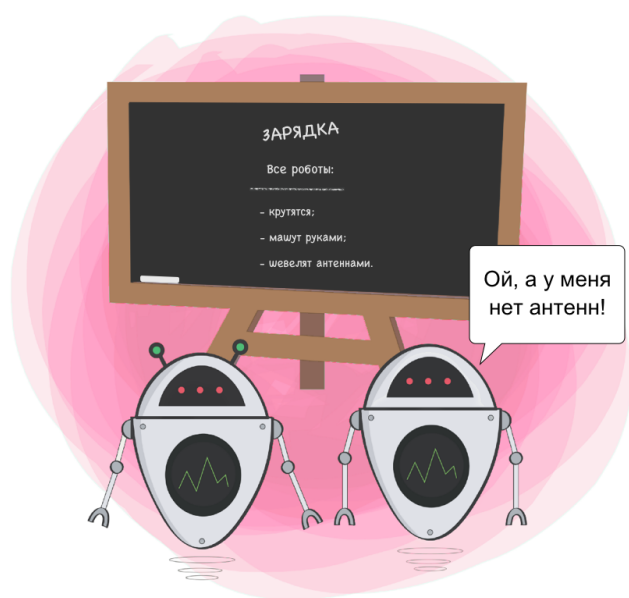
Если класс-потомок не удовлетворяет этим требованиям, значит, он слишком сильно отличается от родителя и нарушает принцип.

Назначение

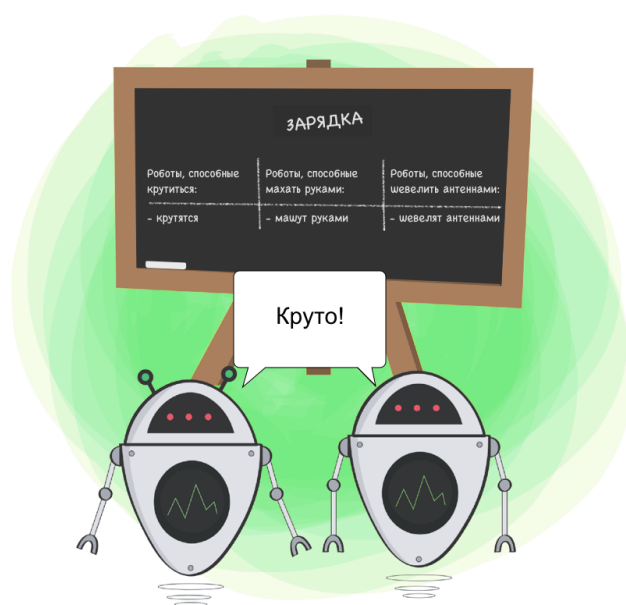
Принцип служит для того, чтобы обеспечить постоянство: класс-родитель и класс-потомок могут использоваться одинаковым образом без нарушения работы программы.

I — Interface Segregation (Принцип разделения интерфейсов)

Не следует ставить клиент в зависимость от методов, которые он не использует.



Принцип разделения интерфейсов



Когда классу приходится производить действия, не несущие никакой реальной пользы, это выливается в пустую трату ресурса, а в случае, если класс выполнять эти действия не способен, ведёт к возникновению багов.

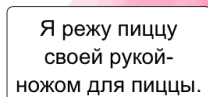
Класс должен производить только те операции, которые необходимы для осуществления его функций. Все другие действия следует либо удалить совсем, либо переместить, если есть вероятность, что они понадобятся другому классу в будущем.

Назначение

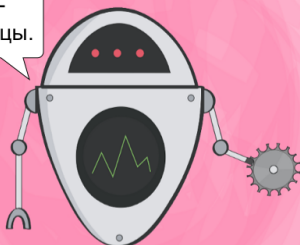
Принцип служит для того, чтобы раздробить единый набор действий на ряд наборов поменьше – таким образом, каждый класс делает то, что от него действительно требуется, и ничего больше.

D — Dependency Inversion (Принцип инверсии зависимостей)

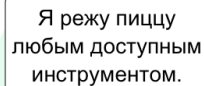
Модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те, и другие должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.



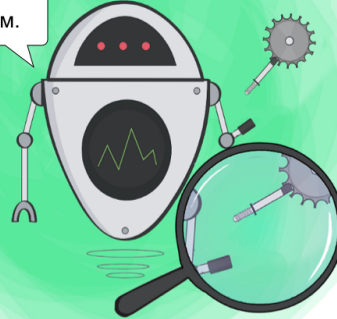
Я режу пиццу своей рукой-ножом для пиццы.



Принцип инверсии зависимостей



Я режу пиццу любым доступным инструментом.



Для начала объясню термины, которые здесь применяются, простыми словами.

Модули (или классы) верхнего уровня = классы, которые выполняют операцию при помощи инструмента

Модули (или классы) нижнего уровня = инструменты, которые нужны для выполнения операций

Абстракции – представляют интерфейс, соединяющий два класса

Детали = специфические характеристики работы инструмента

Согласно данному принципу, класс не должен соединяться с инструментом, который применяет для выполнения операции. Вместо этого он должен быть соединён с интерфейсом, который поможет установить связь между инструментом и классом.

Кроме того, принцип гласит, что ни интерфейс, ни класс, не обязаны вникать в специфику работы инструмента. Напротив, это инструмент должен подходить под требования интерфейса.

Назначение

Этот принцип служит для того, чтобы устранить зависимость классов верхнего уровня от классов нижнего уровня за счёт введения интерфейсов.

Обобщая сказанное

Мы разобрали все пять принципов и сформулировали для каждого назначение. Всё это призвано помочь вам писать код, который можно модифицировать, расширять и тестировать с минимумом проблем. Спасибо, что читали; надеюсь, вы получили не меньше удовольствия, чем я в процессе работы над статьёй.

Теги: объектно-ориентированное по, принципы разработки, solid

Редакторский дайджест



Присылаем лучшие статьи раз в месяц

Электронпочта



Productivity Inside

Для старательного нет ничего невозможного

Сайт



197

35

Карма

Рейтинг

Productivity Inside @ProductivityInside

Пользователь

Комментарии 25

Публикации

ЛУЧШИЕ ЗА СУТКИ

ПОХОЖИЕ



einhorn

15 часов назад

Сравнение нейросетей для перевода



Средний



13 мин



4.7K

Обзор



+41



33



34



Erwinmal

21 час назад

Не только тентакли: за что Интернеты полюбили Ктулху? Часть 2



Простой



10 мин



2.2K



+33



22



1



VladGoryachev

18 часов назад

О вычислительной природе реальности

🕒 12 мин 👁 6.8K

Из песочницы

💎 +32

🔖 47

💬 99



OldfagGamer

17 часов назад

Ностальгические игры: GTA Vice City (часть 1)

🎮 Простой 🕒 10 мин 👁 3.1K

Ретроспектива

💎 +30

🔖 16

💬 20



klimkovsky

15 часов назад

Астероид (488453) 1994 XD сблизится с Землей до расстояния 3 млн.км 12 июня 2023 года

💧 Средний 🕒 4 мин 👁 2K

Кейс

💎 +17

🔖 5

💬 7

Показать еще

Ваш аккаунт

Войти

Регистрация

Разделы

Статьи

Новости

Хабы

Компании

Авторы

Песочница

Информация

Устройство сайта

Для авторов

Для компаний

Документы

Соглашение

Конфиденциальность

Услуги

Корпоративный блог

Медийная реклама

Нативные проекты

Образовательные
программы

Стартапам

Спецпроекты



[Настройка языка](#)

[Техническая поддержка](#)

[Вернуться на старую версию](#)

© 2006–2023, Habr

ИНФОРМАЦИЯ

Сайт	productivityinside.com
Дата регистрации	24 октября 2016
Дата основания	10 апреля 2009
Численность	101–200 человек
Местоположение	Россия

ССЫЛКИ

Сайт Productivity Inside
productivityinside.com
Сайт MaCleaner
macleaner.com

ПРИЛОЖЕНИЯ



MaCleaner 11: Top Disk Cleaner

Новый MaCleaner с кастомизированным дизайном: приложение автоматически определяет модель Вашего Mac и предлагает особый интерфейс для каждого девайса.

Mac OS X



Globe 3D

Интерактивная модель Земли с детализацией вплоть до дома.

Mac OS X iOS



Fashion Design Sketches

Программа для всех дизайнеров одежды, от профессионалов до любителей, которые хотят воплотить свои творческие идеи в жизнь.

iOS Mac OS X



Master of Typing 3

Эффективный курс упражнений и экзаменов по освоению техники слепой печати.

Mac OS X



DJ Mix Pads 2

Приложение для создания музыки и целое музыкальное сообщество.

Android iOS Mac OS X

БЛОГ НА ХАБРЕ

26 мая в 11:03

Проверяем невероятные заявления разработчиков приложения-мессенджера с шифрованием



4.3K



5

21 мая в 11:04

Мобильная разработка за неделю #489 (15 — 21 мая)



1.8K



0

19 мая в 12:10

У нас была возможность удалить любой пост с LinkedIn



3.1K



3

14 мая в 18:26

Мобильная разработка за неделю #488 (8 — 14 мая)



2.1K



3

7 мая в 18:29

Мобильная разработка за неделю #487 (1 — 7 мая)



2.1K



0