



Причины существования интерфейсов — поддержка заявленного поведения

Java Core (/quests/QUEST_JAVA_CORE)
3 уровень (/quests/lectures/?quest=QUEST_JAVA_CORE&level=3), 1 лекция (/quests/lectures/questcore.level03.lecture01)

ОТКРЫТА

— Привет, Амиго! Хочу сегодня тебе рассказать о причинах существования интерфейсов. Тебе очень часто придется слышать, что такой-то класс, объект или сущность поддерживает определенный интерфейс. Что же это значит – поддерживать интерфейс?



В более широком смысле интерфейс какой-нибудь вещи – это механизм взаимодействия этой вещи с другими предметами. Например, пульт от телевизора – это дистанционный интерфейс. Собака понимает и исполняет команды — это значит, что собака поддерживает голосовой интерфейс (управления). Если все это подытожить, то можно сказать, что интерфейс – это стандартизированный способ взаимодействия двух вещей, и этот стандарт известен двум сторонам. Когда человек говорит собаке «сидеть», он отдает команду в соответствии с «голосовым интерфейсом управления собакой», и если собака выполняет эту команду, то мы говорим, что собака поддерживает этот интерфейс.

Так же и в программировании. Методы – это действия над объектом, над его данными. И если класс реализует определенные методы, то он «поддерживает исполнение» определенных команд. Какие же преимущества дает объединение методов в интерфейс?

- 1) Каждый interface, как и class, имеет уникальное имя. Обе стороны могут быть на 100% уверены, что вторая сторона поддерживает именно нужный (известный им) интерфейс, а не похожий.
- 2) Каждый интерфейс налагает определенные ограничения на тот класс, который собирается поддерживать его. Класс сам решает (его разработчик), что он будет делать в случае вызова его методов, которые он унаследовал от интерфейса, но результат должен находиться в пределах ожиданий. Если мы скомандовали собаке «сидеть», и она покрутилась 5 минут на месте и села, то это – поддержка интерфейса. А если она вместо этого вцепилась вам в ногу, то ни о какой поддержке тут не может быть и речи. Выполнение команды не привело к ожидаемым результатам.

Допустим, ты с друзьями участвуешь в написании компьютерной игры. И тебе досталась работа запрограммировать поведение одного персонажа. Один ваш коллега уже написал код по отображению всех персонажей на экран. Второй, отвечающий за сохранение игры на диск, написал код по сохранению всех объектов игры в файл. Каждый из них написал много кода и сделал интерфейс для взаимодействия с ним. Например, это может выглядеть так:

Код на Java

Описание

```
1 interface Saveable
2 {
3     void saveToMap(Map<String, Object> map);
4     void loadFromMap(Map<String, Object> map);
5 }
```

— интерфейс по сохранению/загрузке объекта из map'a.

```
1 interface Drawable
2 {
3     void draw(Screen screen);
4 }
```

— интерфейс по отрисовки объекта внутри переданного объекта screen.

```
1 class PacMan implements Saveable, Drawable
2 {
3     ...
4 }
```

— твой класс, реализующий поддержку двух интерфейсов.

Другими словами, чтобы поддержать реализацию какого-то интерфейса (группы интерфейсов) в своем классе нужно:

- 1) Унаследоваться от них
- 2) Реализовать объявленные в них методы
- 3) Методы должны делать то, для чего они предназначены.

Тогда остальной код программы, который ничего не знает о твоём классе и его объектах, сможет успешно работать с ним.

— А почему код может ничего не знать о моем классе?

— Допустим, ты взял код программы, который кто-то написал год назад. Или твои друзья купили/лицензировали движок игры у кого-то еще. Есть рабочий код игры. Тысячи объектов, которые взаимодействуют друг с другом. И они могут с легкостью правильно взаимодействовать с твоими объектами, если взаимодействие организовано через интерфейсы, и ты правильно реализовал эти интерфейсы в своих классах.

— Круто! Не знал что так можно.

— На этом принципе основаны все большие проекты. Уже давно никто ничего не пишет с нуля.

Люди тоже не изобретают математику и алфавит каждый раз заново, а изучают все то, что было придумано до них.

[< \(/quests/lectures/questcore.level03.lecture00\)](/quests/lectures/questcore.level03.lecture00)

[×13 > \(/quests/lectures/questcore.level03.lecture02\)](/quests/lectures/questcore.level03.lecture02)



0



0



0



0



0

+31

Комментарии (30)

популярные

новые

старые

Никита

Сергей 16 уровень, Москва

26 марта, 15:51



Ответить

0

Dinar 15 уровень, Уфа

3 марта, 09:53



Допустим, ты с друзьями участвуешь в написании компьютерной игры. И тебе досталась работа запрограммировать поведение одного персонажа. Один ваш коллега уже написал код по отображению всех персонажей на экран. Второй, отвечающий за сохранение игры на диск, написал код по сохранению всех объектов игры в файл. Каждый из них написал много кода и сделал интерфейс для взаимодействия с ним. Например, это может выглядеть так:

- интерфейс по сохранению/загрузке объекта из тар'а.
- интерфейс по отрисовки объекта внутри переданного объекта screen.

далее

- твой класс, реализующий поддержку двух интерфейсов.

Какие тонны кода у друзей, если реализацию методов из интерфейсов надо самому писать? Без наставника и пузыря не разобраться...

Ответить

+3

Роман 13 уровень, Новосибирск

позавчера, 15:19



например:

есть реализованный кем-то класс ShowScreen

Для отрисовки ему нужен (например) битмап твоего объекта и позиция объекта на карте

Если твой объект MyHero поддерживает (наследует и реализует) интерфейс Drawable, в котором есть getBitmap и getPosX, getPosY методы, о которых ShowScreen знает и знает что они отдают, то он может им пользоваться. И ему не важна вся остальная реализация твоего класса

Он просто в цикле обходит Drawable obj = myHero (angryEnemies...etc), берет .getBitmap() и рисует их (героев, врагов, деревья, камни etc). Все, что drawable

Как-то так

Ответить

0

Роман 13 уровень, Новосибирск

позавчера, 15:27



То же самое для Saveable и Loadable

Если твой класс MyHero реализует эти интерфейсы, и вы договорились с классом (написанным не вами), как он их реализует (отдает сырые данные или сериализованные или еще как), то уже не важно все остальное (какие это данные - жизнь, позиция или еще что-то)

Класс Saver может сохранить вашего героя, Loader загрузить

Ответить

0

Alexander Kuznetsov 14 уровень

11 февраля, 19:31



Ответить

+2

theBaldSoprano 16 уровень, Санкт-Петербург

4 февраля, 19:48



лол

"А если она вместо этого вцепилась вам в ногу, то ни о какой поддержке тут не может быть и речи. Выполнение команды не привело к ожидаемым результатам."

ну а если она укусила за ногу и потом села, то все ок и интерфейс она поддерживает))

Ответить

+12

Сергей 20 уровень

1 февраля, 19:01



Понял что такое интерфейс по такой фразе. Интерфейс это контракт(реализация набора методов), который надо выполнить при реализации того или иного интерфейса.

Ответить

+3

УАЗ 22 уровень

11 января, 09:12

На примитивном уровне я так понял:
классы(и абстрактные в том числе)- чей родственник?
интерфейсы - че умеешь делать?

Ответить

0

Иван Лаврентьев 20 уровень, Москва

18 ноября 2017, 15:41

очень надеюсь, что скоро будут примеры, где четко виден смысл применения интерфейсов...
перечитал кучу статей, понятно, что это такое, а какая от него польза - все равно не до конца понятно..
и походу я такой тут не один

Ответить

+4

Til Ulenshpigel 17 уровень, Минск

12 декабря 2017, 17:10

Поищите паттерн Стратегия. Там очень хорошо описано применение интерфейсов.

Ответить

0

Джонни 22 уровень

26 октября 2017, 06:18

Главное, ребята, это хорошо преподнесённая лекция вначале уровня. От этого многое зависит. Спасибо,
очень интересная и познавательная лекция.
А Степановна в комментариях снизу отжигает!
Всем читать эту безумную и понятную объяснялку! (=

Ответить

+1

Александр Назаров 18 уровень, Санкт-Петербург

14 октября 2017, 19:14

[Хорошее объяснение интерфейсов](#)

Ответить

+5

Andrew Lan 24 уровень

30 сентября 2017, 22:35

Авторы курсов сами-то читают, что написали? Походу, объяснить что-то - не судьба для них. Или переводчик не особо заморачивался смыслом переводимого..

Ответить

+1

[Загрузить еще](#)

[jsh.ru/](#)). [G+ \(https://plus.google.com/114772402300089087607\)](https://plus.google.com/114772402300089087607). [Twitter \(https://twitter.com/javarush_ru\)](https://twitter.com/javarush_ru). [Facebook \(https://www.facebook.com/javarush.ru\)](#)



Программистами не рождаются
© 2018