Множественное наследование интерфейсов

Java Core (/quests/QUEST_JAVA_CORE)
3 уровень (/quests/lectures/?quest=QUEST_JAVA_CORE&level=3), 5 лекция (/quests/lectures/questcore.level03.lecture05)

ОТКРЫТА

— Привет, Амиго! Наконец-то мы добрались до очень интересной темы. Сегодня я расскажу тебе про множественное наследование. На самом деле множественное наследование очень интересный и мощный инструмент. И если бы не некоторые проблемы, то в Java было бы множественное наследование классов. Но т.к. его нет, придется довольствоваться множественным наследованием интерфейсов. Что тоже не мало.

Представь, что ты пишешь компьютерную игру. И ее герои – твои объекты – должны демонстрировать очень сложное поведение: ходить по карте, собирать предметы, выполнять квесты, общаться с другими героями, кого-то убивать, кого-то спасать. Допустим, ты смог разделить все объекты на 20 категорий. Это значит, что если тебе повезет, ты можешь обойтись всего 20-ю классами, для их описания. А теперь вопрос на засыпку: сколько всего уникальных видов взаимодействия у этих объектов. Объект каждого типа может иметь уникальные взаимодействия с 20-ю видами других объектов (себе подобных тоже считаем). Т.е. всего нужно запрограммировать 20 на 20 – 400 взаимодействий! А если уникальных видов объектов будет не 20, а 100, количество взаимодействий может достигнуть 10,000!

— Ничего себе! Теперь понимаю, почему программирование такая непростая работа.

— Она простая. Благодаря многим абстракциям. И в не последнюю очередь – множественному наследованию интерфейсов.

Очень часто можно упростить взаимодействие объектов, если взаимодействовать будут не объекты, а их роли и/или способности. А способности, как мы уже знаем, легко добавляются в класс, когда он реализует некоторый интерфейс.

Когда пишется большая программа, обычно с этого сразу и начинают:

1) Определяют все существующие способности/роли.

2) Затем описывают взаимодействие между этими ролями.

3) А потом просто наделяют все классы их ролями.

— А можно пример?

— Конечно. Давай рассмотрим роли, на основе героев мультика «Том и Джерри».

Код на Java

Описание

```
1  interface Moveable
2  {}
```

— роль/способность передвигаться.

```
1 interface Eatable
2 {}
```

— роль/способность быть съеденным.

```
1 interface Eat
2 {}
```

— роль/способность съесть кого-нибудь.

```
1 class Tom extends Cat implements Moveable, Eatable, Eat
2 {}
```

Tom – это кот, у которого есть три роли:
1) может передвигаться
2) может кого-то съесть
3) может быть съеденным кем-то (собакой)

```
1 class Jerry extends Mouse implements Moveable, Eatable
2 {}
```

Jerry – это мышь, у которого есть две роли:
1) может передвигаться
2) может быть съеденным кем-то

```
1 class Killer extends Dog implements Moveable, Eat
2 {}
```

Killer – это собака, у которого есть две роли: 1) может передвигаться 2) может кого-то съесть

Зная всего эти три роли (интерфейса) можно написать программу и описать корректное взаимодействие этих ролей. Например, объект будет гнаться (посредством интерфейса Moveable) за тем, «кого ты можешь съесть» и убегать от того, «кто может съесть тебя». И все это без знаний о конкретных объектах. Если в программу добавить еще объектов (классов), но оставить эти роли, она будет прекрасно работать – управлять поведением своих объектов.

‹ (/quests/lectures/questcore.level03.lecture04)

×13 › (/quests/lectures/questcore.level03.lecture06)

Комментарии (18) популярные новые старые

Никита

Введите текст комментария

Илья 15 уровень, Минск 27 марта, 02:46

короче интерфейс - это абстрактный класс, но с множественным наследованием. при этом для каждго класса, реализующего какойто интерфейс, нужно с нуля все методы писать.

в java 8 в интерфейсах добавили и такую функцию абстрактного класса, как реализация метода по умолчанию. в таком виде наконец интерфейсы стали удобным инструментом при программировании, но и разница с абстрактными классами почти нивелировалась полностью

Ответить +1

Юрий Кузнецов 17 уровень 25 марта, 17:28

Отличная лекция) Благодаря тому и джери лекция понятна xD

Ответить 0

Евгений 15 уровень 26 ноября 2017, 13:50

Плохо, что нет конкретный примеров работающей программы. Пусть даже это было бы в виде Единичка может съесть Нолик, а Двоечка - Единичку.

Ответить +10

Vaiki 18 уровень, Минск 20 августа 2017, 18:42

И вот 100 классов... и каждому классу писать реализацию методов интерфейсов, а в интерфейсе допустим по методов 5, а к классу применяешь по 2-3 интерфейса, ... почему нельзя прописывать реализации по умолчанию, как в абстрактных классах...

Ответить +3

LCI 18 уровень, Москва 27 августа 2017, 01:17

Ну я думаю, потому что благодаря интерфейсам можно добавить новые классы вне существующих иерархий, при этом объекты этих новых классов все равно будут вести себя правильно. Допустим роль "агрессор" в РПГ (атакует героя) может быть присвоена, например, сооружению или ловушке, однако такие объекты будет нелогично включать в цепочку наследования монстров.

Ответить +6

Egorro 40 уровень 14 сентября 2017, 11:37

Как-то пока все же вызывает определенное недоумение смысл этих интерфейсов. Вот если бы прописывалось в интерфейсе Eat что-то типа

```
1   void eat (Creature target) { target.dead = true; this.hungry = false; }
```

было бы понятно, для чего наследуется - реально классы получают общие методы, которые **могут быть**, а не обязательно **должны быть** переопределены для каждого.
А так - "у таких-то классов должны быть понаписаны такие-то методы с одинаковыми названиями (для каждого класса - с нуля)" - как-то ни то, ни се))

Ответить +1

Даниленко Виктор 40 уровень, Днепр 21 сентября 2017, 19:59

Как я понял - Смыслинтерфейсов - это не класс с реализациями методов, это "обязательство" классов использующих этот интерфейс иметь систематизированно одинаковые методы для взаимодействия в общей среде. Стандартизация.
Интерфейсы не помогают писать меньше кода, они стандартизируют взаимодействие между классами с минимумом кода. Чтобы одинаковые методы возникли ещё до появления классов с разными.

Все части команды пишущие код для общих\своих\новых\абстрактных классов, работать с объектами, не будут писать и выдумывать свои названия методов для одних и тех же действий. И будут понимать что обращаться к классам таких-то интерфейсов можно именно так, а не гадать или уточнять каждый раз.

В твоём примере ничего не мешает написать и интерфейс с обязательными методами или уже использовать существующий, и общий абстрактный класс их реализующий и который будет наследоваться дальше и экономить код для подклассов. И двойной работы не будет, как раз потому что интерфейс тел методов не требует.

Ответить +13

Roman Bogdanov 20 уровень, Днепр 26 сентября 2017, 02:03

Запрещено множественное наследование Классов. А в интерфейсе запрещена реализация методов(они абстрактны). Значит Eat был бы классом(видимо абстрактным, но это не важно). И если бы Tom его наследовал, то он бы не мог быть ещё и Котом(Cat).

Ответить 0

Эльдар Ахметов 20 уровень, Уфа 28 сентября 2017, 14:17

В Java 8 добавили возможность реализовывать методы в интерфейсах, (методы по умолчанию), перед методом нужно указать ключевое слово (default), а классы реализующие такой интерфейс не обязаны писать его реализацию. Если написать его реализацию в классе будет стандартное переопределение метода.
Пример: default void eat (Creature target) { target.dead = true; this.hungry = false; }

Ответить +20

Радхараман Скороход 16 уровень 28 марта, 02:15

Если они разные по сути то да, а вы как хотели? Но если есть типовые - их можно наследовать от абстрактного класса, реализующего эти интерфейсы.

Ответить 0

Alexashka 17 уровень, Новосибирск 9 августа 2017, 10:42

Том и Джерри - мультик детства)

Ответить +1

Olexandr Leonets 25 уровень, Киев 30 апреля 2017, 14:00

Комменты изучите. Помогут разобраться в теме . Спасибо всем.

Ответить +3

Alexey Finik 14 уровень, Москва 3 июня 2017, 22:46

Лол:)). Я поверил и пошел изучать:)))

Ответить 0

Eugene Palesika 13 уровень 14 июля 2017, 10:28

+1 :D

Ответить 0

LCI 18 уровень, Москва 27 августа 2017, 01:09

тянет на статью

Ответить +6

Psyh 2409 28 уровень, Киев 5 апреля 2017, 22:16

Спайк не ел Тома, а давал ему тумаков. Тут же авторы используют три интерфейса "съесть" , "быть съеденным" ну и двигаться. Так что Спайк не запятнан, а Киллер, вполне уместное имя. Разве что недостаточно прожорливое.

Ответить 0

Viktor Gartman 16 уровень 4 апреля 2017, 00:23

прямо клуб знатоков Тома и Джерри))

Ответить 0