Как пользоваться абстрактными классами

Java Core (/quests/QUEST_JAVA_CORE)
3 уровень (/quests/lectures/?quest=QUEST_JAVA_CORE&level=3), 3 лекция (/quests/lectures/questcore.level03.lecture03)

ОТКРЫТА

— Привет, Амиго! Вчера ты уже слушал лекцию про абстрактные классы. Теперь пришло время углубить наши познания. Хочу научить тебя правильно пользоваться абстрактными классами.

Сложно представить аналогию абстрактного класса в реальной жизни. Обычно класс является моделью какой-нибудь сущности. Но абстрактный класс содержит не только реализованные методы, но и не реализованные. Что же это значит? Аналогом чего является абстрактный класс и есть ли у него аналоги в реальном мире?

На самом деле есть. Представь себе почти законченный кузов машины на конвейере. Туда могу поставить как спортивный двигатель, так и экономичный. Как кожаный салон, так и матерчатый. Конкретная реализация машины еще не определена. Более того, таких конкретных реализаций на основе этого кузова предполагается несколько. Но в таком виде машина никому не нужна. Это — классический абстрактный класс: его объекты не имеют смысла, поэтому их создание запрещено, класс имеет смысл, но только для его многочисленных полноценных наследников, которые будут созданы на его основе.

— Это не сложно.

— Но могут быть и более абстрактные аналогии. Больше похожие на интерфейсы, с несколькими реализованными методами. Например, профессия переводчик. Без уточнения, с какого, и на какой язык, получим «абстрактного переводчика в вакууме». Или телохранитель. Про него может быть известно, что он владеет восточными единоборствами и может защитить клиента. Но какими именно единоборствами, и каким способом защитить клиента – это уже «особенности реализации» каждого конкретного телохранителя.

Давай посмотрим пример:

Код на Java

Описание

```
1  abstract class BodyGuard
2  {
3   abstract void applyMartialArts(Attacker attacker);
4
5   void shoot(Attacker attacker)
6   {
7      gun.shoot(attacker);
8   }
9
10  void saveClientLife(Attacker attacker)
11  {
12   if (attacker.hasGun())
13      shoot(attacker);
14   else
15      applyMartialArts(attacker);
16  }
17 }
```

В классе «телохранитель» определено, как поступать в случае нападения: стрелять или применить восточные единоборства.

Но не определено, какие именно восточные единоборства, хотя точно известно, что этот навык есть.

Мы можем создать несколько разных телохранителей (унаследовав этот класс). Все они будут уметь защищать клиента и стрелять в нападающего.

— Действительно, по смыслу очень напоминает интерфейс с несколькими реализованными методами.

— Да, абстрактные классы такого типа мы будем часто встречать среди стандартных классов JavaSE.

‹ (/quests/lectures/questcore.level03.lecture02)

×13 › (/quests/lectures/questcore.level03.lecture04)

0 0 0 0 0 +20

Комментарии (5) популярные новые старые

Никита

Введите текст комментария

Hexogon 20 уровень 3 октября 2017, 22:01

Разница между таким абстрактным классом еще больше размывается в Java 8 в котором можно задавать default методы

Ответить 0

Orion 22 уровень 7 сентября 2017, 23:16

Сферический переводчик в вакууме

Ответить +3

karbofas 18 уровень 24 августа 2017, 12:43

Что-то я запутался..
gun.shoot(attacker); в первом случае, видимо, есть некий объект gun, (кстати, откуда она взялась - её не объявляли, не передавали. Получается, что есть ещё один класс, в который входит класс BodyGuard, где объявляется глобальная переменная gun? так можно?), у этого объекта есть метод shoot(Attacker attacker), мы его вызываем через объект.. Всё вроде бы сходится, за исключением того, что это и есть реализация этого метода. Это что, получается, рекурсивный бесконечный вызов метода shoot()?

А второй раз мы вызвали метод shoot(Attacker attacker) напрямую (shoot(attacker);), без упоминания названия класса, в котором он содержится (applyMartialArts.shoot(attacker);). Так что ли можно, если мы находимся в этом же классе?

Ответить 0

Иван Корсаков 20 уровень, Санкт-Петербург 24 августа 2017, 18:04

В примере просто вырванный кусок кода, как ниже написал Илья. Да, gun - это эбъект типа Gun (я так его назвал), у которого есть метод shoot(Attacker attacker). Да, мне пришлось создать другой класс Security, расширяющий BodyGuard, а также класс Attacker. Про gun.shoot(attacker) - я для себя понял так, что внутри класса BodyGuard без явного указания класса при вызове метода shoot будет вызываться метод текущего класса и случится рекурсия. Поэтому ссылаемся на объект gun класса Gun. Ну а в методе saveClientLife вызывается метод shoot текущего класса.

Ответить 0

Илья Иванов 22 уровень, Ярославль 12 марта 2017, 01:13

Это как бы кусок кода, который выдернули. Это в общем лирика, на которую обращать внимания не стоит. Тут объясняются общие принципы реализации абстрактного класса.

Ответить +3