Инкапсуляция

Java Core (/quests/QUEST_JAVA_CORE)
1 уровень (/quests/lectures/?quest=QUEST_JAVA_CORE&level=1), 7 лекция (/quests/lectures/questcore.level01.lecture07)

ОТКРЫТА

— Привет, Амиго! Хочу посвятить сегодняшнюю лекцию инкапсуляции. Ты уже знаешь в общих чертах, что это такое.

В чем же преимущества инкапсуляции? Их достаточно много, но я могу выделить четыре, на мой взгляд, основных:

1) Валидное внутреннее состояние.

В программах часто возникают ситуации, когда несколько классов, взаимодействуют с одним и тем же объектом. В результате их совместной работы нарушается целостность данных внутри объекта — объект уже не может продолжить нормально работать.

Поэтому объект должен следить за изменениями своих внутренних данных, а еще лучше – проводить их сам.

Если мы не хотим, чтобы какая-то переменная класса менялась другими классами, мы объявляем ее private, и тогда только методы её же класса смогут получить к ней доступ. Если мы хотим, чтобы значения переменных можно было только читать, но не изменять, тогда нужно добавить public getter для нужных переменных.

Например, мы хотим, чтобы все могли узнать количество элементов в нашей коллекции, но никто не мог его поменять без нашего разрешения. Тогда мы объявляем переменную private int count и метод public getCount().

Правильное использование инкапсуляции гарантирует, что ни один класс не может получить прямой доступ к внутренним данным нашего класса и, следовательно, изменить их без контроля с нашей стороны. Только через вызов методов того же класса, что и изменяемые переменные.

Лучше исходить из того, что другие программисты всегда будут использовать твои классы самым удобным для них образом, а не самым безопасным для тебя (для твоего класса). Отсюда и ошибки, и попытки заранее избавиться от них.

2) Контроль передаваемых аргументов.

Иногда нужно контролировать аргументы, передаваемые в методы нашего класса. Например, наш класс описывает объект «человек» и позволяет задать дату его рождения. Мы должны проверять все передаваемые данные на их соответствие логике программы и логике нашего класса. Например, не допускать 13-й месяц, дату рождения 30 февраля и так далее.

— А зачем кому-то указывать в дате рождения 30 февраля?

— Во-первых – это может быть ошибка ввода данных от пользователя.

Во-вторых, прежде чем программа будет работать как часы, в ней будет много ошибок. Например, возможна такая ситуация.

Программист пишет программу, которая определяет людей у кого день рождения послезавтра. Например, сегодня 3 марта. Программа добавляет к текущему дню месяца число 2 и ищет всех, кто родился 5 марта. Вроде бы все верно.

Вот только, когда наступит 30 марта программа не найдет никого, т.к. в календаре нет 32 марта. В программе становится гораздо меньше ошибок, когда в методы добавляют проверку переданных данных.

— Помню, когда мы изучали ArrayList, я смотрел его код, и там была проверка индекса в методах get и set: index больше или равен нулю и меньше длины массива. Там еще кидалось исключение, если в массиве нет элемента с таким индексом.

— Да, это классический пример проверки входных данных.

3) Минимизация ошибок при изменении кода классов.

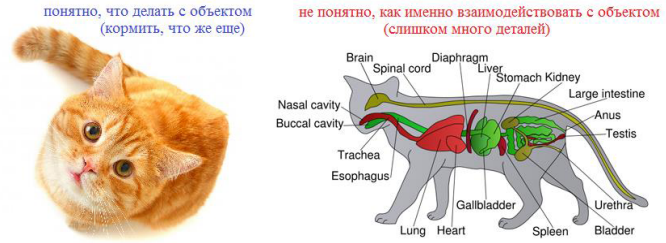
Представим, что мы написали один очень полезный класс, когда участвовали в большом проекте. Он так всем понравился, что другие программисты начали использовать его в сотнях мест в своем коде.

Класс оказался настолько полезен, что ты решил его улучшить. Но если ты удалишь какие-то методы этого класса, то код десятков людей перестанет компилироваться. Им придется срочно все переделывать. И чем больше переделок, тем больше ошибок. Ты поломаешь кучу сборок, и тебя будут ненавидеть.

А когда мы меняем методы, объявленные как private, мы знаем, что нигде нет ни одного класса, который вызывал бы эти методы. Мы можем их переделать, поменять количество параметров и их типы, и зависимый код будет работать дальше. Ну, или как минимум, компилироваться.

4) Задаем способ взаимодействия нашего объекта со сторонними объектами.

Мы можем ограничить некоторые действия, допустимые с нашим объектом. Например, мы хотим, чтобы объект можно было создать только в одном экземпляре. Даже если его создание происходит в нескольких местах проекта одновременно. И мы можем сделать это благодаря инкапсуляции.



Инкапсуляция позволяет добавлять дополнительные ограничения, которые можно превратить в дополнительные преимущества. Например, класс String реализован как immutable (неизменяемый) объект. Объект класса String неизменяем с момента создания и до момента смерти. Все методы класса String (remove, substiring, …), возвращают новую строку, абсолютно не изменяя объект, у которого они были вызваны.

— Ничего себе. Вот оно как, оказывается.

— Инкапсуляция очень интересная штука.

— Ага.

‹ (/quests/lectures/questcore.level01.lecture06)

×11 › (/quests/lectures/questcore.level01.lecture08)

0 0 0 0 0 +21

Комментарии (27) популярные новые старые

Никита

Введите текст комментария

King 16 уровень, Санкт-Петербург 6 марта, 18:10

"Все методы класса String (remove, substiring, …)"
В substring лишняя буква i. Звучит, как метод ускорения машины в стрит-рейсе)

Ответить 0

Джонни 22 уровень 23 октября 2017, 17:47

"*Мы можем ограничить некоторые действия, допустимые с нашим объектом. Например, мы хотим, чтобы объект можно было создать только в одном экземпляре.* "

Это КАК!?

Ответить 0

panvasyan 27 уровень, Мариуполь 23 октября 2017, 23:20

Поверь, через несколько лекций ты напишешь этот объект. Ну представь, что по логике функционирования должен существовать только один экземпляр некоторой сущности, например объект, представляющий твою видеокарту на компьютере и позволяющий её настраивать. Ну допустим, что у тебя в системе одна видеокарта. Он появляется в момент первого вызова (т. е. он кому-то понадобился), а потом все попытки создания объекта в других местах приводят к ссылке на один уже существующий.

Ответить +6

Джонни 22 уровень 24 октября 2017, 14:40

Верю, тут много чего ещё предстоит, я уверен) Спасибо за разъяснение.

Ответить 0

Kirill 27 уровень, Нижний Новгород 4 декабря 2017, 19:09

Реализовать паттерн Синглтон.

Ответить 0

Джонни 22 уровень 4 декабря 2017, 21:56

Да... уже 2 раза его писал)

Ответить 0

Вера Сургучёва 14 уровень 21 октября 2017, 14:16

— " Помню, когда мы изучали ArrayList, я смотрел его код, и там была проверка индекса в методах get и set: index больше или равен нулю и меньше длины массива. Там еще кидалось исключение, если в массиве нет элемента с таким индексом."

спасибо, опустили самооценку

Ответить +4

Алексей Свеженцев 23 уровень, Екатеринбург 10 сентября 2017, 22:34

Интересный ход .. получается все лекции по сути бесплатны, покупаешь только возможность решать задачи. Дополнительная мотивация решить больше задач, чтобы заплатить меньше денег)

Ответить 0

mila 39 уровень, Самара 9 сентября 2017, 21:25

щас с поднятием цен на подписку вобще тишина будет как в библиотеке)
всё интеллигентно и по делу)

Ответить 0

mila 39 уровень, Самара 15 сентября 2017, 11:35

ну это неудивительно, тем более в нашей стране

Ответить 0

Ivan Romanyuk 12 уровень, Минск 16 сентября 2017, 03:22

Можно ссылку в лс?

Ответить 0

Максим 13 уровень, Гомель 16 сентября 2017, 13:11

тоже бы не отказался от ссылки в лс

Ответить 0

Wh0 17 уровень 31 октября 2017, 21:39

хуя чаек набежало))))

Ответить +5

Anton Konkin 24 уровень 29 июля 2017, 22:58

Однозначно атмосфера изменилась)

Ответить +2

Elias Lezzz 24 уровень, Москва 7 мая 2017, 21:03

Скорее крыса на велосипеде
И электрогитара грифом бьёт её по голове

Ответить 0

Алексей Цыпленков 12 уровень 3 мая 2017, 22:48

Я один вижу в часах парня на велосипеде?

Ответить +1

Игорь 19 уровень, Новосибирск 11 мая 2017, 06:17

Там ещё гитара, Эйфелева башня и тарелка с вилками и ложками

Ответить 0

Александр Назаров 25 уровень, Москва 29 июня 2017, 13:13

Там еще и Эйфелева башня)

Ответить 0

Maxim Constantinov 18 уровень 17 июля 2017, 11:55

Хорош

Ответить 0

Roman Karimov 16 уровень, Москва 7 сентября 2017, 16:57

Парень едет бороться с чумой

Ответить 0

Даниил 35 уровень 19 апреля 2017, 12:52

Данил Богданов +

Ответить 0

Robert Lem 11 уровень, Киев 7 апреля 2017, 23:20

Ну наконец-то котэ вернули;)

Ответить 0

Загрузить еще