01

Extraindo comportamento e duck typing

Transcrição

[00:00] Então vamos dar uma olhada na nossa classe “livro”. Os livros podem ter uma sobrecapa. Vou adicionar o valor, sobrecapa, “possui_sobrecapa”, e atribuo como variável membro.

[00:15] Além disso, as revistas podem ter um número. Ela é número 1, número 3, Arquitetura número 17, número 21, mesma coisa. Recebe o número, e atribui. Agora vamos no nosso sistema, vamos atualizar os nossos livros e revistas. Por exemplo, esse livro “Programming Ruby”, ele possui sobrecapa 5. Mas ele não tem número, o que eu faço?

[00:45] E a revista? A revista de Ruby não tem sobrecapa, não faz sentido a questão da sobrecapa. Mas ela é de número 3, apresenta algo de estranho. Eu chamei os dois caras de livro, falei que livro é uma revista, revista é um livro. Tem algo de estranho nessa minha afirmação, quando eu uso objeto do tipo “livro” para representar tanto livros, quanto revista.

[01:04] Vou abrir a nossa classe “livro”, copiar tudo, criar um outro livro chamado “revista”, e colocar a nossa classe revista lá dentro. Vou mudar de livro para revista. Repara que no nosso livro, eu não preciso do número, então vou voltar atrás e manter só aquela história da sobrecapa: “possui_sobrecapa”.

[01:32] No caso da revista, vou deixar só o número, então eu coloco só o número: voltei atrás e coloquei só o número. E agora? Essa variável “tipo”, faz sentido ter uma string que define qual o tipo?

[01:50] O objeto me diz se ela é uma revista ou um livro. Eu vou remover essa história de tipo: tiro do construtor, tiro o reader, e tiro também a atribuição da variável. Vou no meu livro, e faço a mesma coisa. Tiro a atribuição, tiro o construtor, e tiro o reader. O tipo, é um objeto e não uma variável membro.

[02:13] Por fim, vou criar agora um ebook. Crio uma classe nova, e faço o mesmo processo: copio e colo na classe livro, renomeio para ebook, e tiro essa questão de sobrecapa. Ebook não tem sobrecapa, e também não tem a questão de possuir reimpressão. Removo isso também, reimpressão e sobrecapa.

[02:34] Removo o método que eu estava disponibilizando, e pronto. Volto no meu sistema, e assim como requeremos para o livro, vamos fazer o “require” da revista e do ebook. Agora vamos atualizar cada um desses objetos, de acordo com o construtor. Vamos tirar os tipos, que ficava passando: livro, revista, ebook... apago tudo isso.

[03:02] Troca a instância das revistas para revista, e o ebook para ebook. Agora vamos na classe livro e repara que precisamos passar se possui sobrecapa. No caso dos algoritmos, possui sobrecapa, eu vou colocar “true”.

[03:21] No caso de Arquitetura, Programming e Ruby, eu vou colocar “false”. São só exemplos. No caso da revista, o que ela recebe? O número. Então eu vou passar 3.

[03:30] A nossa revista é a terceira “Revista de Ruby”. E o ebook? O que o ebook recebe? Damos uma olhada e na verdade tiramos se ele possui reimpressão, então vamos tirar aqui do construtor também.

[03:39] Vamos rodar o programa e reparem que deu erro no método “que¬*mais*vendeu_por”. O método “tipo” não existe. Vamos lá procurar esse método, “que¬*mais*vendeu_por”. Reparem que usamos o método “tipo” do livro.

[03:57] Se lembram que tinha um método “tipo” antes? No “attr_reader”? Não tem mais. Então uma solução bem simples, primeiro seria implementar esse método. Vamos definir o método “tipo”. Na revista ela devolve, revista. No ebook ela vai devolver como ebook, e no livro vai devolver, livro.

[04:15] Pronto, se rodarmos tudo de novo funcionou. Mas repara que quando temos o método “respond_to” ele devolve um método muito estranho. Vamos dar uma olhada no método “respond_to” no nosso estoque?

[04:29] Ele faz um match, e o match devolve o nome do método. Por isso que fica muito estranho. Então eu extraio uma variável “matched”, que diz se foi ou não. Pegamos esse “matched” e vai devolver “true” ou “false”.

[04:41] Então eu quero transformar isso em um “boolean”. Ele me devolveu uma string e eu quero transformar isso em um “boolean”.

[04:47] Como é que eu transformo uma string ou um nulo em “boolean”? Eu vou fazer o “double bang” “!!”. Se for nulo, com a primeira exclamação, eu estou transformando em “true” e depois com a segunda exclamação, volta para falso.

[04:59] Se ele encontrou esse método com a primeira exclamação, ele transforma em false, e com a segunda, true. Você transforma um objeto nulo em true ou false.

[05:10] Rodamos a aplicação de novo, e o resultado é o que esperávamos: true. Mas é muito estranho, na classe revista devolver revista. Na outra classe devolver ebook, na outra classe devolver livro.

[05:24] Poxa, será que eu não tenho outra maneira ao invés de ficar trabalhando com esses tipos, ao invés de ficar devolvendo uma string? Reparem que “que_mais_vendeu_por” estávamos chamando de “l”, o nosso livro. Vamos chamar ele de produto agora, afinal ele pode ser um livro, uma revista, ele pode ser qualquer coisa.

[05:43] E pra parar de fazer essa comparação de igualdade, eu vou perguntar para o meu produto: “Você é um ebook?” “Você é um livro?” “Você é algum tipo de coisa?”. Então “matches”, algum tipo de coisa que eu estou interessado? Por exemplo, você é um livro que eu estou interessado? Você é uma revista que eu estou interessado?

[06:03] O ebook que mais vendeu por? Ou você é o impresso que mais vendeu por digital? Porque a revista é impressa, o livro também é impresso. O ebook, é digital. Então eu posso perguntar de diversas maneiras.

[06:15] Então eu pergunto para o meu produto, “você matcheia?” “você pertence a essa categoria que eu estou criando?” Sim ou não? Lembrando que o método de pergunta que devolve em ruby, é tradicional colocar ponto de interrogação.

[06:27] Então vamos renomear o método do nosso ebook pra “matches”. E no caso, se a query que você está perguntando for igual a ebook, eu vou falar beleza. Ou se a query que você está falando é por exemplo, digital, também beleza.

[06:40] Eu sou digital e eu sou um ebook. Posso implementar esse método de diversas maneiras. Vamos agora refatorar esse método e fazer de uma maneira um pouquinho diferente. Coloco tanto ebook, quanto digital dentro de uma “array”, e pergunto: a query pertence a esse tipo?

[07:02] Ela é ebook ou digital? É uma variação dessa implementação. Reparem que eu parei de me intrometer no ebook e passei a perguntar para ele. Novamente eu estou encapsulando este tipo de comportamento.

[07:18] E agora, vamos na revista, tiramos o método tipo, e colocamos o “matches”. Se for revista, ou se for impresso, sou eu. E no livro também. Se sua pergunta é de livro ou de impresso, sou eu.

[07:39] Então não me importa para quem eu estou perguntando, importa que você tem o método “matches”, e que o método “matches” me diga se você é dessa categoria ou não.

[07:51] E isso é o “duck typing”. Não me importa quem você é, me importa que você tenha esse comportamento e me diga o que eu quero saber.