

Perdeu preiboi, agora é OO – Parte III

09/09/2013 10:00

Reflexão do dia: "Mas vale um teste bem feito do que dois DUMPS em PRD".

Herança

(ou generalização) é o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos).

Quando você diz, "Poo quero fazer exatamente o que essa classe faz, apenas mudando um método", então você copia a classe, altera apenas o método que você precisa, amanhã um outro programador altera outro método da classe principal, problema, sua classe não sofreu a alteração, esse outro programador nem sabia da existência da sua classe, resultado:

CAOS NA TERRA

Quando usamos essa tal de herança?!

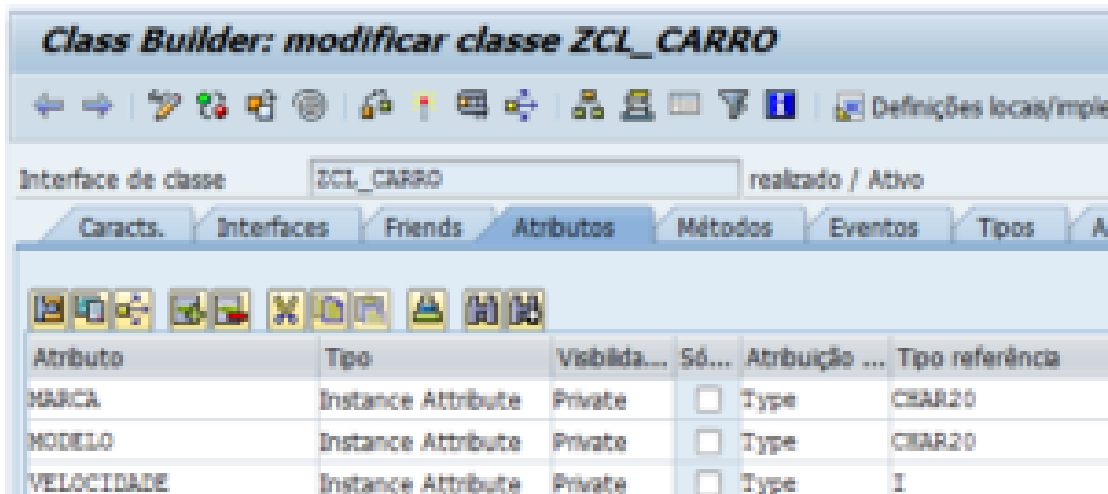
Vamos supor este cenário, temos a seguinte classe para controlar um carro:

Classe Carro

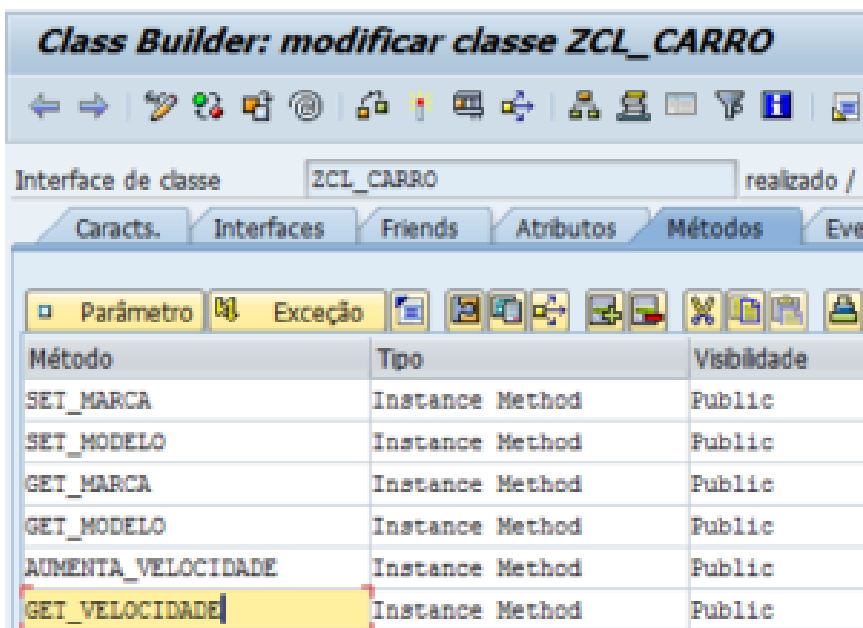
Atributos: Modelo, Marca, Velocidade

Métodos: set_modelo, get_modelo, set_marca, get_marca, aumenta_velocidade, get_velocidade.

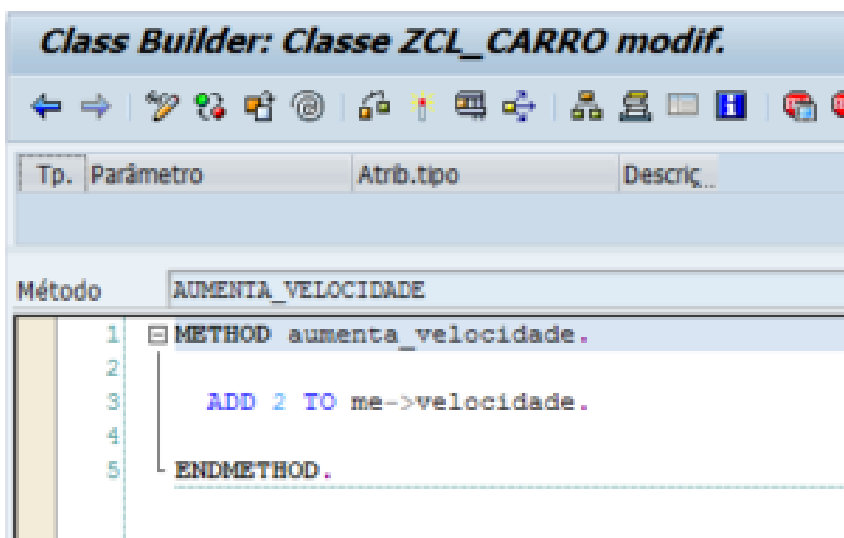
Regra de negócio desta classe, a velocidade aumenta de 2 em 2.



Atributos



Métodos



Método Aumenta_Velocidade

Para os métodos GETs e SETs vocês podem ler a explicação no [Perdeu preiboi, agora é OO – Parte II](#).

Exemplo de Programa

```
DATA: o_carro TYPE REF TO zcl_carro,
v_velocidade TYPE i.

CREATE OBJECT o_carro.

o_carro->set_marca( 'VW' ).
o_carro->set_modelo( 'GOL' ).

o_carro->aumenta_velocidade( ).
o_carro->aumenta_velocidade( ).
o_carro->aumenta_velocidade( ).

v_velocidade = o_carro->get_velocidade( ).

WRITE v_velocidade.
```

Agora temos que desenvolver a classe para controlar o carro esporte. O carro esporte tem marca, modelo, aumenta_velocidade e os GETs e SETs, exatamente igual o carro, unica coisa diferente é o aumento da velocidade, o dobro do carro.

Classe Carro Esporte

Atributos: Modelo, Marca, Velocidade

Métodos: set_modelo, get_modelo, set_marca, get_marca, aumenta_velocidade, get_velocidade.

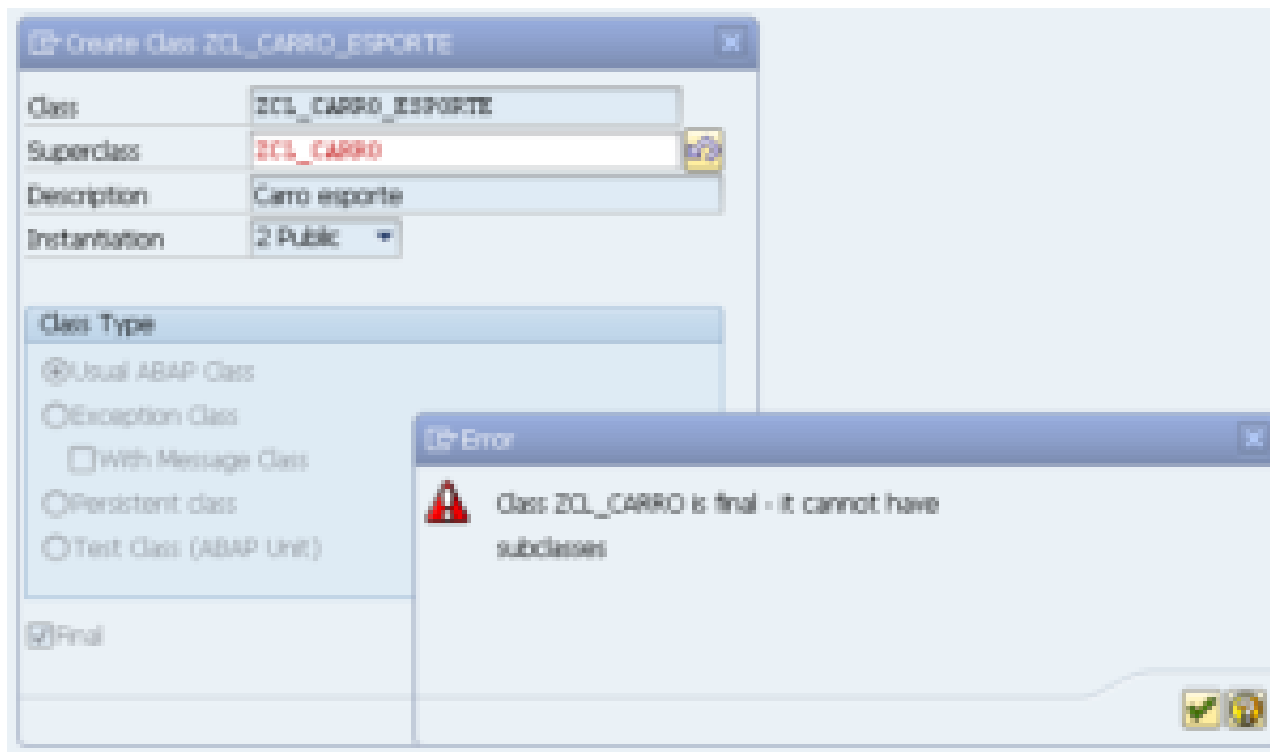
Regra de negócio desta classe, a velocidade aumenta de 4 em 4 (dobro da classe CARRO).

Sendo assim, a classe de carro esporte é igual a classe de carro, apenas o método aumenta_velocidade que sera diferente, vamos criar uma herança da classe Carro.

Para criar a herança, basta criar a classe pela SE24, como no post [Perdeu preiboi, agora é OO – Parte I](#) , so que clique no botão



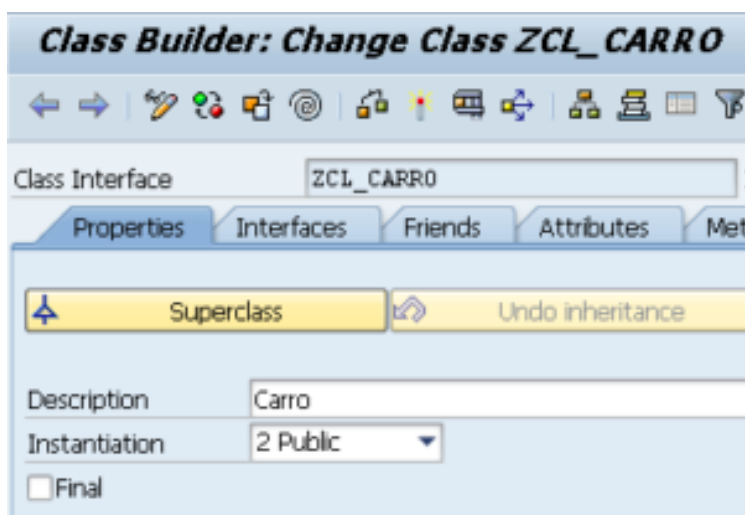
irá aparecer um campo novo, coloque o nome da classe mãe, ou seja, da classe do carro.



ERROR, NO! – Ainda não foi desta vez

Este erro ocorre quando a sua classe mãe esta marcada como Final, essa marcação serve para dizer que esta classe não permite herança.

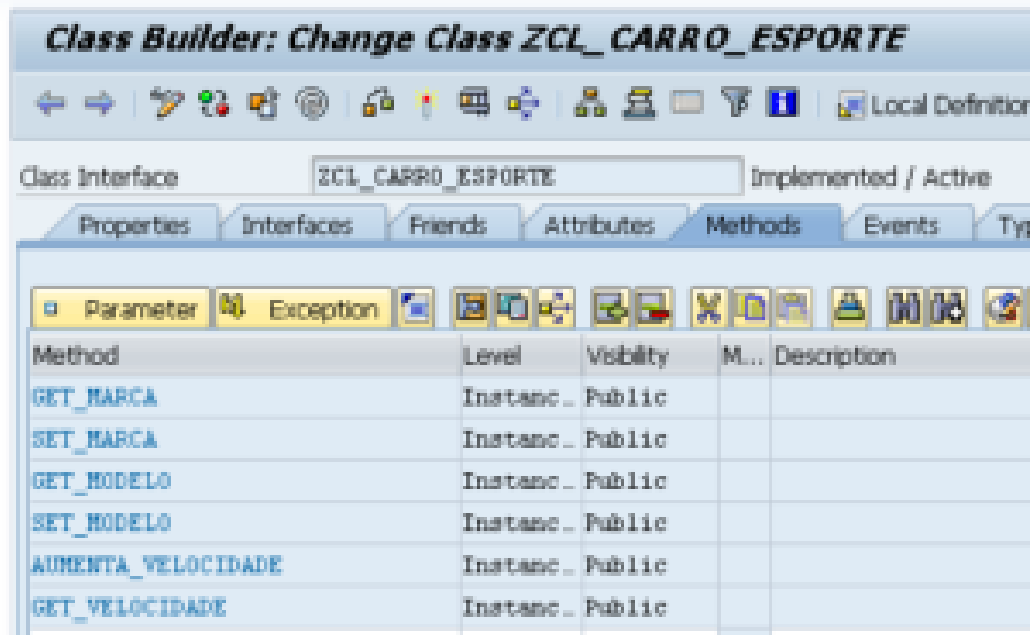
Acesse a classe mãe, tire a marcação de Final.



Classe de Carro sem a marcação de Final

Vá na SE24 como um bom Zombiezinho ABAP e crie sua classe com herança, agora vai.

Classe de Carro Esporte, coisa linda.

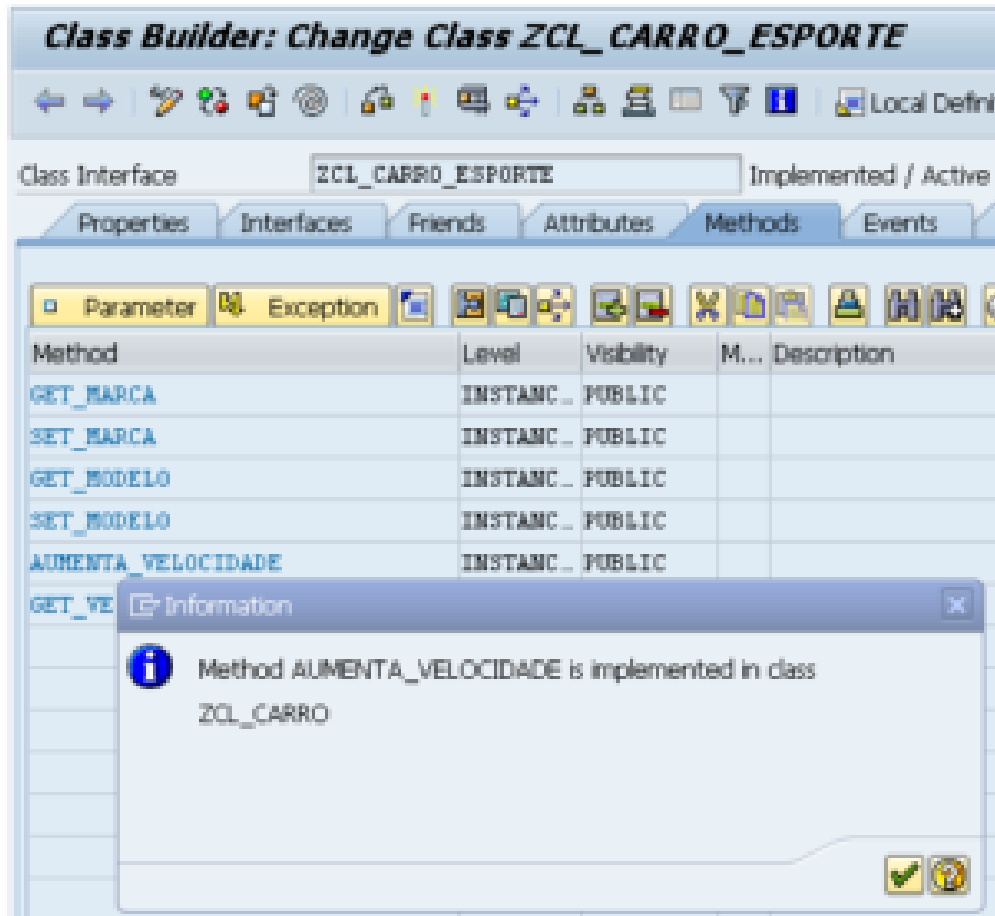


Classe de Carro Esporte

DICA Zumbistica: Os métodos e Atributos em Azul, significam que não pertencem a classe que esta em visualização, são herdados.

Pode ser que na sua visualização da classe não apareça estes métodos e atributos em azul, na tela de visualização da classe tem um checkbox chamado "Filtro", se estiver marcado, serão exibidos apenas os métodos e atributos da classe que esta sendo visualizado, a herança não é visualizada. Não muda nada na execução, apenas na exibição da classe.

Caso clique em algum método, sera exibido a mensagem dizendo que o método pertence a outra classe.



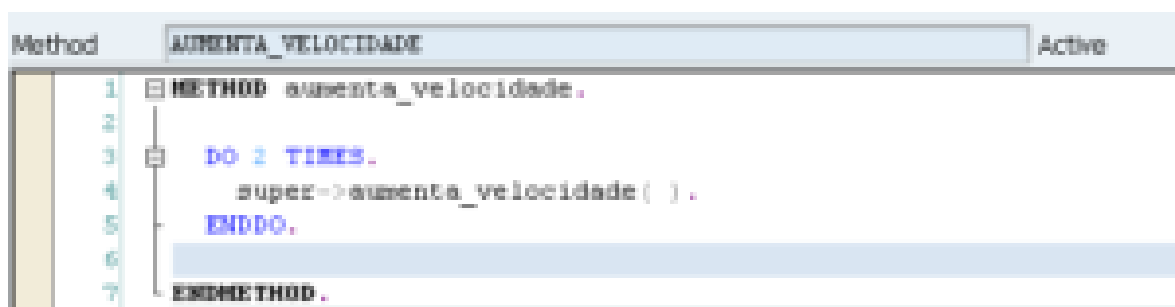
A alteração de um método para ser algo específico de uma classe é chamado de redefinição. Na classe de Carro Esporte temos o método `aumenta_velocidade` que será específico para esta classe.

Para criar uma redefinição, basta clicar sobre o método que deseja redefinir, depois, clique sobre o botão



No post [Perdeu preiboi, agora é OO – Parte II](#) é explicado o que é a referência ME, como agora estamos trabalhando com Herança temos a referência SUPER, quer dizer que vamos executar o método ou atributo da classe Mãe.

Como em nossa regra, o aumentar a velocidade, aumenta o valor da velocidade duas vezes o valor da classe de Carro, optei por fazer 2 chamadas do método `aumenta_velocidade` da classe SUPER.



Exemplo de Código

```

DATA: o_carro      TYPE REF TO zcl_carro_esporte,
      v_velocidade TYPE i.

CREATE OBJECT o_carro.

o_carro->set_marca( 'Porsche' ).
o_carro->set_modelo( '911' ).

o_carro->aumenta_velocidade( ).
o_carro->aumenta_velocidade( ).
o_carro->aumenta_velocidade( ).

v_velocidade = o_carro->get_velocidade( ).

WRITE v_velocidade.

```

Assim, os métodos GETs e SETs e os atributos foram todos herdados, apenas a regra específica para a classe de Carro Esporte.

VIVA OO, VIVA Reaproveitamento de Código.

Já que estamos empolgados e felizes.

Vamos criar a classe de caminhão, mesmos atributos e métodos, apenas a regra que a velocidade aumenta de 1 em 1.

Classe Caminhão

Atributos: Modelo, Marca, Velocidade

Métodos: set_modelo, get_modelo, set_marca, get_marca, aumenta_velocidade, get_velocidade.

Regra de negócio desta classe, a velocidade aumenta de 1 em 1.

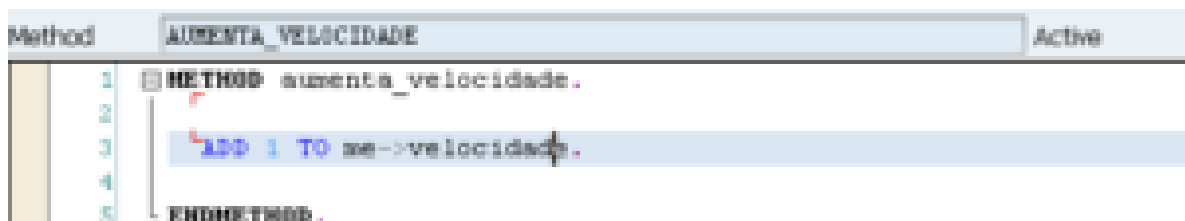
Para este exemplo tem um detalhe, a classe filha de caminhão terá que acessar o atributo VELOCIDADE da classe Carro, mas este atributo é privado, pode ler sobre o encapsulamento no post [Perdeu preiboi, agora é OO – Parte I](#). Será necessário alterar o encapsulamento do atributo VELOCIDADE para PROTEGIDO, para a classe filha poder ter acesso.

Pergunta

Como a classe de Carro Esporte conseguiu alterar a velocidade?

A Classe de Carro Esporte apenas chamou o método da Classe de Carro, não tentou acessar o atributo diretamente, até o momento, a regra para alteração do valor da VELOCIDADE estava na classe de Carro.

Redefinição do método aumenta_velocidade da classe de Caminhão



```

Method: aumenta_velocidade Active
1 METHOD aumenta_velocidade.
2
3   ADD 1 TO me->velocidade.
4
5 ENDMETHOD.

```

Como o atributo VELOCIDADE da classe Carro, esta como protegido, a classe de caminhão tem acesso direto a este atributo.

Exemplo de Código

```
DATA: o_carro      TYPE REF TO zcl_caminhao,
      v_velocidade TYPE i.

CREATE OBJECT o_carro.

o_carro->set_marca( 'Mercedes' ).
o_carro->set_modelo( '915' ).

o_carro->aumenta_velocidade( ).
o_carro->aumenta_velocidade( ).
o_carro->aumenta_velocidade( ).

v_velocidade = o_carro->get_velocidade( ).

WRITE v_velocidade.
```

Então é isso Zumbizada, bom estudos.

Abraços a todos os Zumbies.

Comentários

Lucas — 14/07/2016 16:09

Muito obrigado pela resposta Mauricio, concordo com você mas é que o sentimento que tenho tido nesses últimos dias é que não estudei direito, estou pegando as indicações de livros que o pessoal lá tem deixado e realmente me frustrei comigo mesmo, mas entendi o pq simplesmente criar uma classe na SE24 apenas não significa que sei OO.

Lucas — 13/07/2016 20:14

Primeiramente excelente o seu post.

Vou fazer basicamente a mesma pergunta que fiz no Bar8 perguntando ao contrario,

Já havia lido esse post a um tempo atrás quando estava estudando orientação a objeto no ABAP, mas comecei a acompanhar o blog do bar8 e existem alguns posts sobre orientação a objeto que eu simplesmente viajo ao tentar ler, mesmo tendo lido um livro de ABAP Object e lido seus pots e do ABAP101 nunca vi nada sobre os conceitos que o bar8 está trazendo,

Então vai a minha dúvida, para a mim no dia a dia eu preciso mesmo saber o que é modelo rico ou modelo anêmico por exemplo? você por exemplo se preocupa com isso quando desenvolve?

Estou perguntando pq estou realmente “assustado”, começo a ler uns posts e as vezes levo horas para terminar por não conhecer nada do que eles falam.

Valeu!

Mauricio Cruz — 14/07/2016 12:39

Lucas,

Tudo o que eu posso dizer é: relaxe e programe. Se você ficar preocupado em aprender tudo conceitualmente antes de fazer alguma coisa, você vai ficar maluco.

Não posso responder dizendo que é "ok" deixar de aprender partes conceituais de programação pq elas vão sim fazer você transformar-se em um programador melhor.

Porém, saber só os conceitos não vão te ajudar em absolutamente nada: você precisa sentar e programar com aquilo que você sabe. Só assim você passará por dilemas e incertezas que farão você aprender os conceitos com algum propósito embutido (no caso, o programa que você estiver fazendo).

Deixando um comentário pessoal, conforme os anos foram passando eu perdi um pouco essa "paixão" que eu tinha por conceitos extremamente aprofundados sobre coisas relacionadas a programação. Hoje eu me preocupo muito mais em estudar um conceito porque eu preciso fazer um programa que depende de eu entendê-lo, ao invés de estudar coisas randomicamente (coisa q eu já fiz muito na minha vida).

Enfim, vai lá, programa e dane-se o mundo. Precisou, estuda e seja feliz.

Abs!

Robson — 10/12/2014 13:36

Uma dúvida de quem ainda está caminhando para o entendimento de OO.

Entendo que estes foram exemplos didáticos, mas se no mundo real o método AUMENTA_VELOCIDADE da classe CARRO recebesse um parâmetro para ser utilizado na aplicação da velocidade, estaria ferindo alguma regra da POO?

Obrigado e abraços!

Mauricio Cruz — 11/12/2014 09:34

(me metendo um pouco na discussão)

O legal de OO é quando você começar a ficar se perguntando se alguma coisa deveria ou não "ser daquele jeito". Isso, por si só, demonstra que você está PENSANDO antes de fazer, algo crucial para a evolução de um bom programador.



Robson — 11/12/2014 12:08

Pois é exatamente nisso que eu travo. Quando eu fico me perguntando qual será a melhor forma de se modelar a solução, o que eu acredito ser a etapa mais importante da coisa, pois o resto é sintaxe.

Voltando ao exemplo do método de aceleração, eu não consigo concluir se o melhor seria incluir um parâmetro ou não.

Abraços!

Mauro Laranjeira — 10/12/2014 13:58

Neste exemplo cada objeto tem a sua forma de aumentar a velocidade e com isso garantir a consistência regra.

Não faz sentido um caminhão acelerar mais rápido que um carro esporte.

Não existe uma receita de bolo para POO, se para o seu negocio o aumentar velocidade receba um parâmetro, quem sou eu para te questionar rs.

Mas se vc colocar um parâmetro na classe de carro, a classes fihas também teriam este parâmetro.

Espero ter ajudado 😊

Fernando — 10/12/2014 13:52

Não....Na verdade um método pode ter quantos parâmetros desejar. Eles apenas servem para alimentar a lógica aplicada no método.

No mundo real temos:

Método para criar o registro 0150 do SPED. Ele recebe o código do parceiro SAP como parametro. Poderia receber mais coisas, se fosse necessário para criar a linha do SPED 0150. Mas apenas com o código do parceiro, a rotina do método já consegue preencher todos os campos necessários. Ele recebe(importing) como parametro o PARID e devolve(exporting) uma estrutura i_0150, por exemplo, com as informações que precisam ir para o SPED.

cegonha — 07/05/2014 23:18

entendi, digo que causa estranheza justamente pelo fato das outras serem muito parecidas, até certo ponto simples, olhando esse exemplo de abap é meio "estranho" hehe

Mauricio Cruz — 09/05/2014 09:18

Sim, eu te entendo. O pessoal da SAP está tentando diminuir essa diferença com as novas versões do ABAP, mas ainda vai demorar para isso chegar na realidade do dia-a-dia (porque depende dos clientes atualizarem as versões dos seus ERPs).

Vou fazer alguns posts com as coisas que estão mudando nas próximas semanas.

Abs!

cegonha — 09/05/2014 13:43

legal legal!

cegonha — 07/05/2014 17:44

que bagunça ein? os conceitos de herança, polimorfismo, referência não são novos.
não tenho experiência em abap mas tenho alguma em c/c#/java...
é impressão ou aí a coisa é meio estranha mesmo? hehe

Mauricio Cruz — 07/05/2014 19:10

Hey cegonha! Criar classe global como no exemplo que o Mauro postou realmente soa um pouco estranho para quem está acostumado com outras linguagens. Só de ter um dashboard maluco para criar a classe, já deve causar uma aversão.

Mas no ABAP tem (quase) tudo que as outras linguagens tem em relação a OO. A grande diferença é que a maioria das ABAPers não faz questão de utilizar OO, e continuam na forma procedural de pensar.

Se você trabalha com ABAP, aproveite a XP de outras linguagens e destrói. Só se prepare para escrever um pouco mais de código do que você está acostumado para declarar uma classe, método e etc...

Fernando Mafra — 27/04/2014 17:10

Parabéns pelos Posts Mauro. Explica de uma forma funcional o processo de aproveitamento de código com ABAP. Não sou ABAP, mas solicito muitos desenvolvimentos em Métodos em classes z, criadas a partir de uma implementação de Badl. Percebo muita dificuldade dos programadores em desenvolver novas rotinas através da criação de novos métodos na própria classe. Como fica a chamada de um método novo na classe da Badl ? Seria utilizando diretamente "me->[método novo]" ? Pois o programa, quando chama a implementação, já cria o objeto no processamento ? Existe alguma diferença para estas classes ?

Parabéns pela iniciativa

Mauricio Cruz — 28/04/2014 11:22

Fernando, tudo beleza?

Sim, se você criar um novo método na classe que implementa a interface da BADI, você pode chamá-lo com o me->[novo_método]. Não tem diferença nenhuma.

E é isso mesmo, no caso de BADIs, que cria a instância da classe é o standard, e quando o programa chegar no seu código da BADI ele já vai estar "dentro" do objeto que foi criado.

E olha, para quem não é ABAP, o seu entendimento do framework é muito bom 😊

Abraços!

Flavio Oliveira Balan — 17/02/2014 15:35

Eu fiquei com uma duvida no seu exemplo, quando criados a classe de caminhao e alteramos um metodo da classe pai, nao podemos estar prejudicando algum outro programa atrelado a essa classe e utilizando esse metodo? Como voce citou no começo do seu post " Resultado : CAOS NA TERRA".

Mauro Laranjeira — 17/02/2014 18:08

Flavio,

Ai que esta o ponto legal, nós não alteramos o método da classe pai, nos criamos uma redefinição, essa redefinição apenas sera executada para o objeto do tipo de caminhão, quem utiliza a classe de carro não sofre nenhuma alteração, ela continua da mesma forma, apenas “aproveitamos” as características e algumas ações da classe de carro.

Espero que tenha entendido.

Abs,

Mauro

Flavio Oliveira Balan — 18/02/2014 09:43

Ah simmm, entendi!

Obrigado

Aline — 09/01/2014 12:07

Olá Mauro!

Esses posts sobre OO me ajudaram demais! Sou iniciante em Abap e tinha muitas dúvidas que foram sanadas aqui.

Obrigado por dividir o conhecimento conosco.

E parabéns! Muito bom!

=D

Raphael Camargo Alberto — 28/03/2019 10:09

Minhas dúvidas também foram sanadas!! Muito bom!!!

Paulo Cesar Alexandrino — 16/11/2013 14:15

Qual editor você usa para escrever os códigos? Obrigado.

Mauro Laranjeira — 18/11/2013 12:18

Certo Paulo.

Uso a própria SE24