06

Colocando em prática

Hora de melhorar ainda mais nosso código. Ainda lembra do problema apresentado neste capítulo? Vamos recordá-lo:

Em nossa aplicação de orçamentos, diversos impostos possuem cálculos parecidos. Por exemplo,temos dois impostos hipotéticos: ICPP e IKCV. O imposto ICPP é calculado da seguinte forma: caso o valor do orçamento seja menor que 500,00, deve-se cobrar 5%; caso contrário, 7%.

Vamos adicionar o ICPP no já existente `impostos.py`, respeitando a convenção que todo imposto deve ter o método `calcula` que recebe um orcamento:

```
# # -*- coding: UTF-8 -*-
# impostos.py
# adicionado no início do arquivo
class ICPP(object):

    def calcula(self, orcamento):
        if(orcamento.valor > 500):
            return orcamento.valor * 0.07
        else:
            return orcamento.valor * 0.05

# outros impostos omitidos
```

Já o imposto IKCV, caso o valor do orçamento seja maior que 500,00 e algum item tiver valor superior a 100,00, o imposto a ser cobrado é de 10%; caso contrário 6%.

Também vamos adicioná-lo em `impostos.py`, como primeiro imposto:

```
# # -*- coding: UTF-8 -*-
# impostos.py
# adicionado no início do arquivo
class IKCV(object):

    def calcula(self, orcamento):
        if(orcamento.valor > 500 and self.__tem_item_maior_que_100_reais(orcamento)):
            return orcamento.valor * 0.10
        else:
            return orcamento.valor * 0.06

    def __tem_item_maior_que_100_reais(self, orcamento):
        for item in orcamento.obter_itens():
            if(item.valor > 100):
                return True
            return False

# outros impostos omitidos
```

Agora, nada mais justo do que rodar um teste para sabermos se tudo está funcionando. Precisamos modificar ligeiramente `calculador_de_impostos.py`:

```
# -*- coding: UTF-8 -*-
# calculador_de_impostos.py
class Calculador_de_impostos(object):

    def realiza_calculo(self, orcamento, imposto):
        valor = imposto.calcula(orcamento)
        print valor

if __name__ == '__main__':

    from orcamento import Orcamento, Item

    # adicionado os impostos no import
    from impostos import ISS, ICMS, ICPP, IKCV

    orcamento = Orcamento()
    # adicionando itens ao orçamento
    orcamento.adiciona_item(Item('ITEM 1', 50))
    orcamento.adiciona_item(Item('ITEM 2', 200))
    orcamento.adiciona_item(Item('ITEM 3', 250))

    calculador_de_impostos = Calculador_de_impostos()
    print 'ISS e ICMS'
    calculador_de_impostos.realiza_calculo(orcamento, ISS())
    calculador_de_impostos.realiza_calculo(orcamento, ICMS())

    # cálculo dos novos impostos
    print 'ICPP e IKCV'
    calculador_de_impostos.realiza_calculo(orcamento, ICPP()) # imprime 25.0
    calculador_de_impostos.realiza_calculo(orcamento, IKCV()) # imprime 30.0
```

Veja que os dois impostos, apesar de diferentes, possuem uma estrutura em comum. Ambos checam o orçamento para ver se devem cobrar a taxação máxima e, a partir daí, cobram a máxima ou a mínima. Será que podemos generalizar esse comportamento e todo imposto que tiver que aplicar um valor diferenciado para taxação máxima ou mínima? Sim, é justamente o que aprendemos neste capítulo. Resumindo a história: queremos uma mesma estrutura para ICPP e IKVC.

É ai que chegamos ao **template method**. Com base no que você aprendeu, melhore o código anterior aplicando o template method.

DICA: classes abstratas em Python seguem a seguinte estrutura:

```
# exemplo de classe abstrata

from abc import ABCMeta, abstractmethod
class Sou_uma_class_abstrata(object):

     __metaclass__ = ABCMeta

    @abstractmethod
    def sou_um_metodo_abstrato(): pass
```

Responda

INSERIR CÓDIGO		FORMATAÇÃO