1 the simplest thing

BulkItem description weight price __init__

```
class BulkItem(object):

    def __init__(self, description, weight, price):
        self.description = description
        self.weight = weight
        self.price = price
```

2 validation via property

```
[1] bulkfood.py: [2] bulkfood.py 🗱
                                                      ▼ Navegar...
 /home/luciano/prj/pycon2013/talks/encap/1/bulkfood.py
                                                                                                                          ▼ Navegar... 🔓
                                                                       /home/luciano/prj/pycon2013/talks/encap/2/bulkfood.py
  class BulkItem(object):
                                                                      class BulkItem(object):
      def init (self, description, weight, price):
                                                                           def init (self, description, weight, price):
          self.description = description
                                                                               self.description = description
          self.weight = weight
                                                                               self.weight = weight
          self.price = price
                                                                               self.price = price
                                                                           Oproperty
                                                                           def weight(self):
                                                                               return self. weight
                                                                           @weight.setter
                                                                           def weight(self, value):
                                                                               if value > 0:
                                                                                   self. weight = value
                                                                               else:
                                                                                   raise ValueError('value must be > 0')
```

2 validation via property

BulkItem description __weight price __init__ weight {prop. get} weight {prop. set}

```
class BulkItem(object):

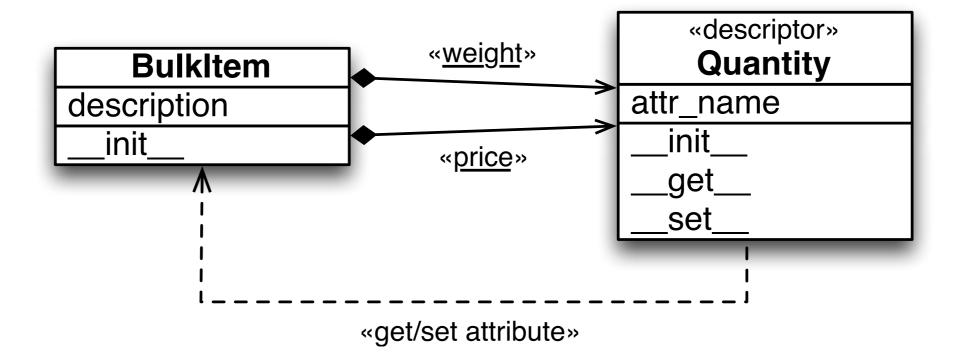
    def __init__(self, description, weight, price):
        self.description = description
        self.weight = weight
        self.price = price

    @property
    def weight(self):
        return self.__weight

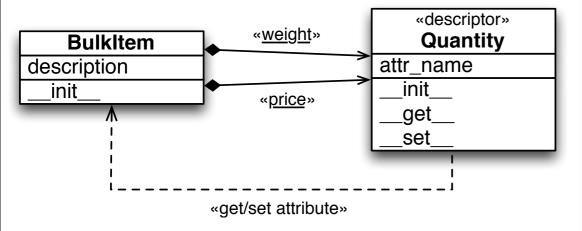
    @weight.setter
    def weight(self, value):
        if value > 0:
            self.__weight = value
        else:
            raise ValueError('value must be > 0')
```

This works, but what if **price** needs a similar treatment?

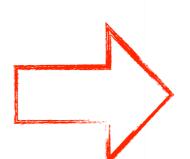
```
[2] bulkfood.py: [3] bulkfood.py *
                                                    ▼ Navegar...
 /home/luciano/prj/pycon2013/talks/encap/2/bulkfood.py
                                                                    /home/luciano/prj/pycon2013/talks/encap/3/bulkfood.py
                                                                                                                      ▼ Navegar...
                                                                  ← class Quantity(object):
 class BulkItem(object):
      def init (self, description, weight, price):
                                                                        def init (self):
                                                                            prefix = self. class . name
          self.description = description
          self.weight = weight
                                                                            key = id(self)
          self.price = price
                                                                            self.attr name = '%s %s' % (prefix, key)
                                                                            get (self, instance, owner):
      @property
      def weight(self):
                                                                            return getattr(instance, self.attr name)
          return self. weight
                                                                        def set (self, instance, value):
                                                                            if value > 0:
      @weight.setter
                                                                                setattr(instance, self.attr name, value)
      def weight(self, value):
          if value > 0:
                                                                            else:
                                                                                raise ValueError('value must be > 0')
              self. weight = value
          else:
              raise ValueError('value must be > 0')
                                                                    class BulkItem(object):
                                                                        weight = Quantity()
                                                                        price = Quantity()
                                                                        def init (self, description, weight, price):
                                                                            self.description = description
                                                                            self.weight = weight
                                                                            self.price = price
```



Descriptors enable reuse of validation logic through **composition**



```
class Quantity(object):
   def init (self):
       prefix = self. class . name
       key = id(self)
       self.attr name = '%s %s' % (prefix, key)
   def get (self, instance, owner):
       return getattr(instance, self.attr name)
   def set (self, instance, value):
       if value > 0:
           setattr(instance, self.attr name, value)
       else:
           raise ValueError('value must be > 0')
class BulkItem(object):
   weight = Quantity()
   price = Quantity()
   def init (self, description, weight, price):
       self.description = description
       self.weight = weight
       self.price = price
```



Field data is stored in **BulkItem** instance
attributes with generic
names like

Quantity_14199423

```
class Quantity(object):
   def init (self):
       prefix = self. class . name
       key = id(self)
       self.attr name = '%s %s' % (prefix, key)
   def get (self, instance, owner):
       return getattr(instance, self.attr name)
   def set (self, instance, value):
       if value > 0:
           setattr(instance, self.attr name, value)
       else:
           raise ValueError('value must be > 0')
class BulkItem(object):
   weight = Quantity()
   price = Quantity()
   def init (self, description, weight, price):
       self.description = description
       self.weight = weight
       self.price = price
```

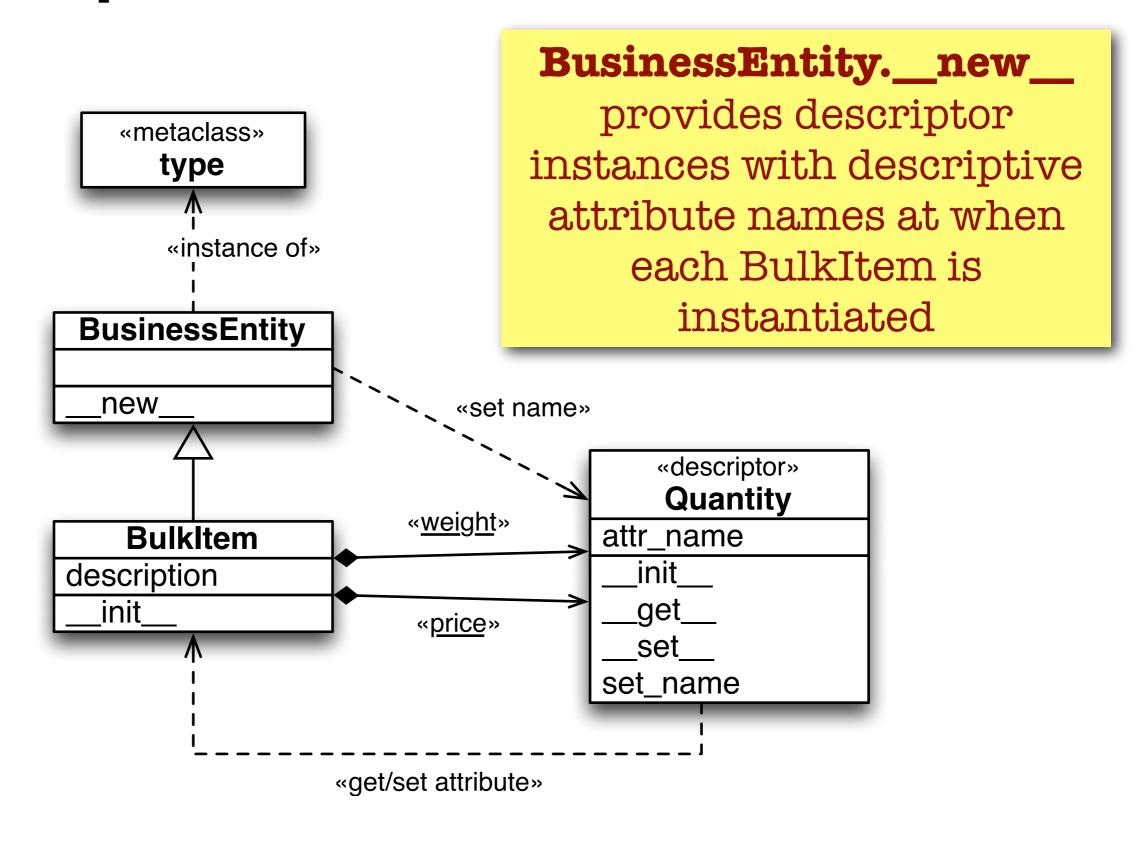
4 proper names for attrs

```
[3] bulkfood.py: [4] bulkfood.py 💥
 /home/luciano/prj/pycon2013/talks/encap/3/bulkfood.py

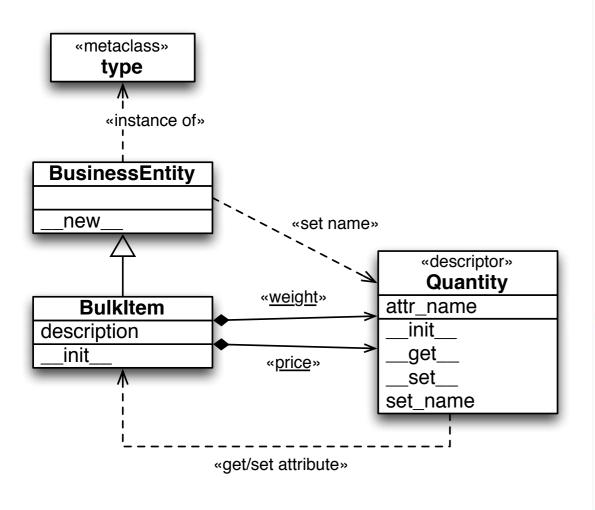
▼ Navegar...

                                                                  /home/luciano/prj/pycon2013/talks/encap/4/bulkfood.py
                                                                                                                   ▼ Navegar...
 class Quantity(object):
                                                                  class Quantity(object):
     def init (self):
                                                                      def init (self):
         prefix = self. class . name
                                                                          self.set name(self. class . name , id(self))
         key = id(self)
         self.attr_name = '%s %s' % (prefix, key)
                                                                      def set name(self, prefix, key):
                                                                          self.attr name = '%s %s' % (prefix, key)
     def get (self, instance, owner):
         return getattr(instance, self.attr name)
                                                                      def get (self, instance, owner):
                                                                          return getattr(instance, self.attr name)
     def set (self, instance, value):
         if value > 0:
                                                                      def set (self, instance, value):
                                                                          if value > 0:
             setattr(instance, self.attr name, value)
         else:
                                                                              setattr(instance, self.attr name, value)
             raise ValueError('value must be > 0')
                                                                              raise ValueError('value must be > 0')
 class BulkItem(object):
                                                                ← class BusinessEntity(object):
     weight = Quantity()
                                                                      def new (cls, *args, **kwargs):
     price = Quantity()
                                                                          for key, attr in cls. dict .items():
                                                                              if isinstance(attr, Quantity):
     def init (self, description, weight, price):
                                                                                  attr.set name(' ' + cls. name , key)
         self.description = description
                                                                          return super(BusinessEntity, cls). new (cls, *arc
         self.weight = weight
         self.price = price
                                                                  class BulkItem(BusinessEntity):
                                                                      weight = Quantity()
                                                                      price = Quantity()
                                                                      def init (self, description, weight, price):
                                                                          self.description = description
                                                                          self.weight = weight
                                                                          self.price = price
```

4 proper names for attrs



4 proper names for attrs



```
class Quantity(object):
   def init (self):
       self.set name(self. class . name , id(self))
   def set name(self, prefix, key):
        self.attr name = '%s %s' % (prefix, key)
   def get (self, instance, owner):
       return getattr(instance, self.attr name)
   def set (self, instance, value):
       if value > 0:
           setattr(instance, self.attr name, value)
       else:
           raise ValueError('value must be > 0')
class BusinessEntity(object):
   def new (cls, *args, **kwargs):
        for key, attr in cls. dict .items():
           if isinstance(attr, Quantity):
               attr.set name(' ' + cls.__name__, key)
        return super(BusinessEntity, cls). new (cls, *arg
class BulkItem(BusinessEntity):
   weight = Quantity()
   price = Quantity()
   def init (self, description, weight, price):
        self.description = description
       self.weight = weight
        self.price = price
```

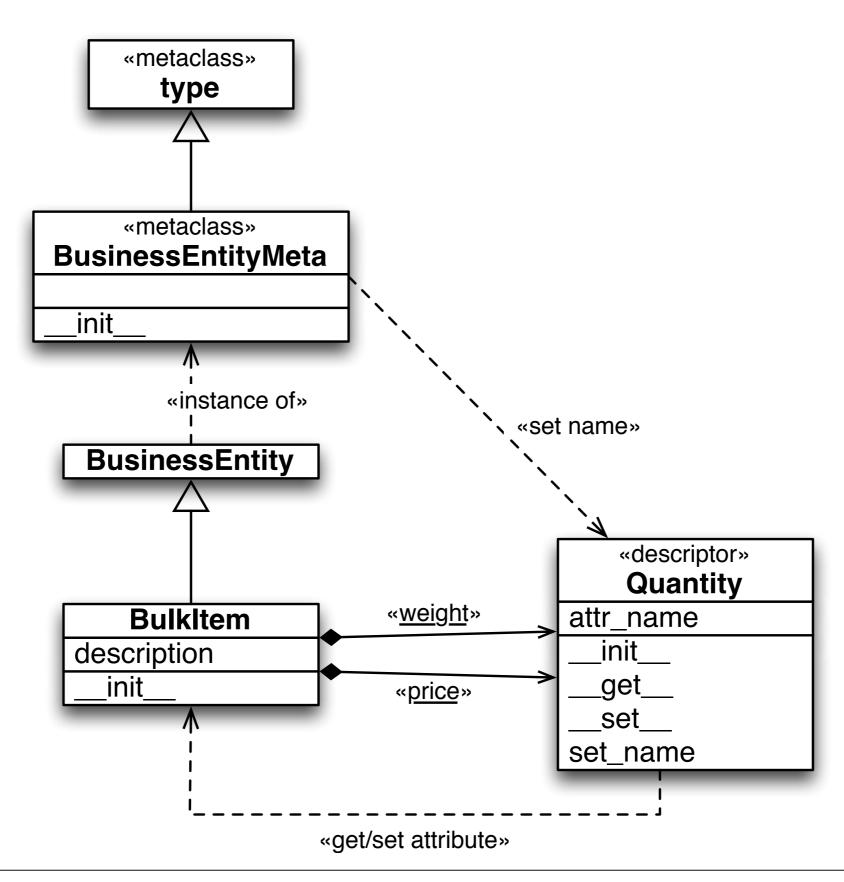
6 avoiding redundant work

```
[4] bulkfood.py: [5] bulkfood.py *
 /home/luciano/prj/pycon2013/talks/encap/4/bulkfood.py

▼ Navegar...

                                                                  /home/luciano/prj/pycon2013/talks/encap/5/bulkfood.py
                                                                                                                   ▼ Navegar... @
 class Quantity(object):
                                                                  class Quantity(object):
                                                                      def init (self):
     def init (self):
         self.set name(self. class . name , id(self))
                                                                          self.set name(self. class . name , id(self))
     def set name(self, prefix, key):
                                                                      def set name(self, prefix, key):
         self.attr_name = '%s %s' % (prefix, key)
                                                                          self.attr name = '%s %s' % (prefix, key)
     def get (self, instance, owner):
                                                                      def get (self, instance, owner):
         return getattr(instance, self.attr name)
                                                                          return getattr(instance, self.attr name)
     def set (self, instance, value):
                                                                      def set (self, instance, value):
         if value > 0:
                                                                          if value > 0:
             setattr(instance, self.attr name, value)
                                                                              setattr(instance, self.attr name, value)
                                                                              raise ValueError('value must be > 0')
             raise ValueError('value must be > 0')
 class BusinessEntity(object):
                                                            → ← class BusinessEntityMeta(type):
     def new (cls, *args, **kwargs):
                                                                      def init (mcs, name, bases, dict ):
         for key, attr in cls. dict .items():
                                                                          super(BusinessEntityMeta, mcs). init (name, bases
             if isinstance(attr, Quantity):
                                                                          for key, attr in dict .items():
                 attr.set name(' '+cls. name , key)
                                                                              if isinstance(attr, Quantity):
         return super(BusinessEntity, cls). new (cls, *arc
                                                                                  attr.set name(' '+name, key)
 class BulkItem(BusinessEntity):
                                                                  class BusinessEntity(object):
     weight = Quantity()
                                                                        metaclass = BusinessEntityMeta
     price = Quantity()
     def init (self, description, weight, price):
                                                                  class BulkItem(BusinessEntity):
         self.description = description
                                                                      weight = Quantity()
         self.weight = weight
                                                                      price = Quantity()
         self.price = price
                                                                      def init (self, description, weight, price):
                                                                          self.description = description
                                                                          self.weight = weight
                                                                          self.price = price
```

1 avoiding redundant work



avoiding redundant work

```
«metaclass»
        type
      «metaclass»
BusinessEntityMeta
 init
         «instance of»
                                      «set name»
  BusinessEntity
                                               «descriptor»
                                               Quantity
                           «weight»
      BulkItem
                                          attr_name
                                             _init___
 description
    init
                                             _get___
                            «price»
                                             set
                                          set name
                      «get/set attribute»
```

```
class Quantity(object):
    def init (self):
       self.set name(self. class . name , id(self))
    def set name(self, prefix, key):
        self.attr name = '%s %s' % (prefix, key)
   def get (self, instance, owner):
        return getattr(instance, self.attr name)
   def __set__(self, instance, value):
       if value > 0:
           setattr(instance, self.attr name, value)
        else:
           raise ValueError('value must be > 0')
class BusinessEntityMeta(type):
   def init (mcs, name, bases, dict ):
        super(BusinessEntityMeta, mcs). init (name, base
       for key, attr in dict .items():
           if isinstance(attr, Quantity):
               attr.set name(' '+name, key)
class BusinessEntity(object):
     metaclass = BusinessEntityMeta
class BulkItem(BusinessEntity):
   weight = Quantity()
   price = Quantity()
   def init (self, description, weight, price):
        self.description = description
        self.weight = weight
        self.price = price
```

② avoiding redundant work

The

BusinessEntityMeta

metaclass provides
descriptor instances
with descriptive
attribute names at
import time

```
class Quantity(object):
    def init (self):
       self.set name(self. class . name , id(self))
    def set name(self, prefix, key):
        self.attr name = '%s %s' % (prefix, key)
   def get (self, instance, owner):
        return getattr(instance, self.attr name)
   def __set__(self, instance, value):
       if value > 0:
           setattr(instance, self.attr name, value)
        else:
           raise ValueError('value must be > 0')
class BusinessEntityMeta(type):
   def init (mcs, name, bases, dict ):
        super(BusinessEntityMeta, mcs). init (name, base
       for key, attr in dict .items():
           if isinstance(attr, Quantity):
               attr.set_name('__'+name, key)
class BusinessEntity(object):
     metaclass = BusinessEntityMeta
class BulkItem(BusinessEntity):
   weight = Quantity()
   price = Quantity()
   def init (self, description, weight, price):
        self.description = description
        self.weight = weight
        self.price = price
```