

```
Smalltalk.OOMPA defineNamespace: #Designmönster
private: false
imports: '
    private Smalltalk.*
'
category: 'OOMPA-1'
```

```
OOMPA.Designmönster defineNameSpace: #Strategy
private: false
imports: '
    private Smalltalk.*
'
category: 'OOMPA-1'
```

```
OOMPA.Designmönster.Strategy.defineClass: #CounterTest
  superclass: #{XProgramming.SUnit.TestCase}
  indexedType: #none
  private: false
  instanceVariableNames: "
  classInstanceVariableNames: "
  imports: "
  category: 'OOMPA-1'
```

OOMPA.Designmönster.Strategy.CounterTest methodsFor: testing

testDown

```
| strategy counter |
strategy := CounterDown
  start: 10
  stop: 0
  step: 1.
counter := Counter counterStrategy: strategy.
counter start
```

testUp

```
| strategy counter |
strategy := CounterUp
  start: 1
  stop: 10
  step: 2.
counter := Counter counterStrategy: strategy.
counter start
```

testUpDownOutput

```
| strategy counter collection output |
strategy := CounterDown
  start: 10
  stop: 0
  step: 1.
collection := OrderedCollection new.
output := [:v | collection add: v].
counter := Counter counterStrategy: strategy output: output.
counter start.
self assert: collection asArray = #(10 9 8 7 6 5 4 3 2 1).
```

testUpWithOutput

```
| strategy counter collection output |
strategy := CounterUp
  start: 1
  stop: 10
  step: 2.
collection := OrderedCollection new.
output := [:v | collection add: v].
counter := Counter counterStrategy: strategy output: output.
counter start.
```

self assert: collection asArray = #(1 3 5 7 9).

```
OOMPA.Designmönster defineNameSpace: #Observer
private: false
imports: '
    private Smalltalk.*
'
category: 'OOMPA-1'
```

OOMPA.Designmönster.Observer defineClass: #Publisher
superclass: #{Core.Object}
indexedType: #none
private: false
instanceVariableNames: 'prints '
classInstanceVariableNames: "
imports: "
category: 'OOMPA-1'

OOMPA.Designmönster.Observer.Publisher methodsFor: initialize-release

initialize

prints := OrderedCollection new

OOMPA.Designmönster.Observer.Publisher methodsFor: accessing

addPrint: aPrint

prints add: aPrint.

self changed: #newPrint with: aPrint

addSubscriber: anObject

self addDependent: anObject

OOMPA.Designmönster.Observer.Publisher class

instanceVariableNames: "

OOMPA.Designmönster.Observer.Publisher class methodsFor: instance creation

new

^super new initialize

OOMPA.Designmönster.Observer defineClass: #Subscriber
superclass: #{Core.Object}
indexedType: #none
private: false
instanceVariableNames: 'lastPrint '
classInstanceVariableNames: "
imports: "
category: 'OOMPA-1'

OOMPA.Designmönster.Observer.Subscriber methodsFor: accessing

lastPrint

^lastPrint

OOMPA.Designmönster.Observer.Subscriber methodsFor: updating

update: anAspectSymbol with: aParameter

anAspectSymbol = #newPrint ifTrue: [lastPrint := aParameter]

```
OOMPA.Designmönster defineNameSpace: #TemplateMethod
private: false
imports: '
    private Smalltalk.*
'
category: 'OOMPA-1'
```

```
OOMPA.Designmönster defineNameSpace: #Composition
private: false
imports: '
    private Smalltalk.*
'
category: 'OOMPA-1'
```

```
OOMPA.Designmönster.Composition.defineClass: #ProductTest
  superclass: #{XProgramming.SUnit.TestCase}
  indexedType: #none
  private: false
  instanceVariableNames: ""
  classInstanceVariableNames: ""
  imports: ""
  category: 'OOMPA-1'
```

OOMPA.Designmönster.Composition.ProductTest methodsFor: testing

testCreationComposite

```
| product |
product := Composite new.
self assert: product price = Number zero
```

testCreationCompositeWithComposite

```
| price1 price2 product1 product2 composite1 price3 price4 product3 product4 composite |
price1 := 107.5.
price2 := 225.75.
product1 := SomeProduct price: price1.
product2 := SomeProduct price: price2.
composite1 := Composite new.
composite1 addPart: product1.
composite1 addPart: product2.
price3 := 1107.5.
price4 := 125.75.
product3 := SomeProduct price: price3.
product4 := SomeProduct price: price4.
composite := Composite new.
composite addPart: product3.
composite addPart: product4.
composite addPart: composite1.
self assert: composite price = (price1 + price2 + price3 + price4)
```

testCreationCompositeWithTwoProducts

```
| price1 price2 product1 product2 composite |
price1 := 107.5.
price2 := 225.75.
product1 := SomeProduct price: price1.
product2 := SomeProduct price: price2.
composite := Composite new.
composite addPart: product1.
composite addPart: product2.
self assert: composite price = (price1 + price2)
```

testCreationSomeProduct

```
| product |
product := SomeProduct new.
self assert: product price = Number zero
```

testCreationSomeProductWithConstructorPrice

```
| product price1 |  
price1 := 107.5.  
product := SomeProduct price: price1.  
self assert: product price = price1
```

testCreationSomeProductWithPrice

```
| product price1 |  
product := SomeProduct new.  
price1 := 107.50.  
product price: price1.  
self assert: product price = price1
```

```
OOMPA.Designmönster.Observer defineClass: #ObserverTest
  superclass: #{XProgramming.SUnit.TestCase}
  indexedType: #none
  private: false
  instanceVariableNames: "
  classInstanceVariableNames: "
  imports: "
  category: 'OOMPA-1'
```

OOMPA.Designmönster.Observer.ObserverTest methodsFor: testing

test1

```
| publisher sub1 print |
publisher := Publisher new.
sub1 := Subscriber new.
publisher addSubscriber: sub1.
print := 'Smalltalkbok'.
publisher addPrint: print.
self assert: sub1 lastPrint = print
```

test2

```
| publisher sub1 print sub2 |
publisher := Publisher new.
sub1 := Subscriber new.
sub2 := Subscriber new.
publisher addSubscriber: sub1.
publisher addSubscriber: sub2.
print := 'Smalltalkbok del 2'.
publisher addPrint: print.
self assert: sub1 lastPrint = print.
self assert: sub2 lastPrint = print
```

```
OOMPA.Designmönster defineNameSpace: #AbstractFactory
private: false
imports: '
    private Smalltalk.*
'
category: 'OOMPA-1'
```

```
OOMPA.Designmönster defineNameSpace: #Adapter
private: false
imports: '
    private Smalltalk.*
'
category: 'OOMPA-1'
```

```
OOMPA.Designmönster.Adapter defineClass: #AdapterTest
  superclass: #{XProgramming.SUnit.TestCase}
  indexedType: #none
  private: false
  instanceVariableNames: "
  classInstanceVariableNames: "
  imports: "
  category: 'OOMPA-1'
```

```
OOMPA.Designmönster.Adapter.AdapterTest methodsFor: testing
```

testOne

```
| client adaptee success |
client := Client new.
adaptee := Adaptee new.
adaptee setValue: 10.
client setValueHolder: adaptee.
success := false.
[client value = 10.
success := true] on: Error do: [:ex | ].
self deny: success
```

testThree

```
| client adaptee adapter |
client := Client new.
adaptee := Adaptee new.
adaptee setValue: 10.
adapter := Adapter new.
adapter setValueHolder: adaptee.
client setValueHolder: adapter.
client value: 100.
self assert: client value = 100.
self assert: adaptee getValue = client value
```

testTwo

```
| client adaptee adapter |
client := Client new.
adaptee := Adaptee new.
adaptee setValue: 100.
adapter := Adapter new.
adapter setValueHolder: adaptee.
client setValueHolder: adapter.
self assert: client value = 10
```

OOMPA.Designmönster.TemplateMethod defineClass: #SIConverter
superclass: #{Core.Object}
indexedType: #none
private: false
instanceVariableNames: "
classInstanceVariableNames: "
imports: "
category: 'OOMPA-1'

OOMPA.Designmönster.TemplateMethod.SIConverter methodsFor: constants

gallon

self subclassResponsibility

OOMPA.Designmönster.TemplateMethod.SIConverter methodsFor: accessing

pint

^self gallon / 8

OOMPA.Designmönster.TemplateMethod defineClass: #USConverter
superclass: #{OOMPA.Designmönster.TemplateMethod.SIConverter}
indexedType: #none
private: false
instanceVariableNames: "
classInstanceVariableNames: "
imports: "
category: 'OOMPA-1'

OOMPA.Designmönster.TemplateMethod.USConverter methodsFor: constants

gallon
^3.785

OOMPA.Designmönster.TemplateMethod defineClass: #ImperialConverter
superclass: #{OOMPA.Designmönster.TemplateMethod.SIConverter}
indexedType: #none
private: false
instanceVariableNames: "
classInstanceVariableNames: "
imports: "
category: 'OOMPA-1'

OOMPA.Designmönster.TemplateMethod.ImperialConverter methodsFor: constants

gallon
^4.546

OOMPA.Designmönster.Composition defineClass: #Product
superclass: #{Core.Object}
indexedType: #none
private: false
instanceVariableNames: "
classInstanceVariableNames: "
imports: "
category: 'OOMPA-1'

OOMPA.Designmönster.Composition.Product methodsFor: initialize-release

initialize

OOMPA.Designmönster.Composition.Product methodsFor: accessing

price

self subclassResponsibility

OOMPA.Designmönster.Composition.Product class

instanceVariableNames: "

OOMPA.Designmönster.Composition.Product class methodsFor: instance creation

new

self == Product ifTrue: [^self error: 'Abstract'].

^super new initialize

OOMPA.Designmönster.Composition defineClass: #Composite
superclass: #{OOMPA.Designmönster.Composition.Product}
indexedType: #none
private: false
instanceVariableNames: 'parts '
classInstanceVariableNames: "
imports: "
category: 'OOMPA-1'

OOMPA.Designmönster.Composition.Composite methodsFor: initialize-release

initialize

parts := OrderedCollection new

OOMPA.Designmönster.Composition.Composite methodsFor: accessing

addPart: aProduct

parts add: aProduct

price

^parts inject: Number zero into: [:tot :part | tot + part price]

OOMPA.Designmönster.Composition defineClass: #SomeProduct
superclass: #{OOMPA.Designmönster.Composition.Product}
indexedType: #none
private: false
instanceVariableNames: 'price '
classInstanceVariableNames: "
imports: "
category: 'OOMPA-1'

OOMPA.Designmönster.Composition.SomeProduct methodsFor: initialize-release

initialize

price := Number zero

OOMPA.Designmönster.Composition.SomeProduct methodsFor: accessing

price

^price

price: aNumber

price := aNumber

OOMPA.Designmönster.Composition.SomeProduct class

instanceVariableNames: "

OOMPA.Designmönster.Composition.SomeProduct class methodsFor: instance creation

price: aNumber

^super new price: aNumber